# Bluetooth Core Specification

***Bluetooth*® Specification**

- **Version:** v6.3
- **Version Date:** 2026-05-05
- **Prepared By:** Core Specification Working Group


## Abstract

This specification defines the technologies required to create interoperable Bluetooth devices.

## Disclaimer and Copyright Notice

**Use of this specification is your acknowledgement that you agree to and will comply with the following notices and disclaimers. You are advised to seek appropriate legal, engineering, and other professional advice regarding the use, interpretation, and effect of this specification.**

**Use of Bluetooth specifications by members of Bluetooth SIG is governed by the membership and other related agreements between Bluetooth SIG and its members, including those agreements posted on Bluetooth SIG's website located at www.bluetooth.com. Any use of this specification by a member that is not in compliance with the applicable membership and other related agreements is prohibited and, among other things, may result in (i) termination of the applicable agreements and (ii) liability for infringement of the intellectual property rights of Bluetooth SIG and its members. This specification may provide options, because, for example, some products do not implement every portion of the specification. All content within the specification, including notes, appendices, figures, tables, message sequence charts, examples, sample data, and each option identified is intended to be within the bounds of the Scope as defined in the Bluetooth Patent/Copyright License Agreement ("PCLA"). Also, the identification of options for implementing a portion of the specification is intended to provide design flexibility without establishing, for purposes of the PCLA, that any of these options is a "technically reasonable non-infringing alternative."**

**Use of this specification by anyone who is not a member of Bluetooth SIG is prohibited and is an infringement of the intellectual property rights of Bluetooth SIG and its members. The furnishing of this specification does not grant any license to any intellectual property of Bluetooth SIG or its members. THIS SPECIFICATION IS PROVIDED "AS IS" AND BLUETOOTH SIG, ITS MEMBERS AND THEIR AFFILIATES MAKE NO REPRESENTATIONS OR WARRANTIES AND DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR THAT THE CONTENT OF THIS SPECIFICATION IS FREE OF ERRORS. For the avoidance of doubt, Bluetooth SIG has not made any search or investigation as to third parties that may claim rights in or to any specifications or any intellectual property that may be required to implement any specifications and it disclaims any obligation or duty to do so.**

**TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, BLUETOOTH SIG, ITS MEMBERS AND THEIR AFFILIATES DISCLAIM ALL LIABILITY ARISING OUT OF OR RELATING TO USE OF THIS SPECIFICATION AND ANY INFORMATION CONTAINED IN THIS SPECIFICATION, INCLUDING LOST REVENUE, PROFITS, DATA OR PROGRAMS, OR BUSINESS INTERRUPTION, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, AND EVEN IF**

**BLUETOOTH SIG, ITS MEMBERS OR THEIR AFFILIATES HAVE BEEN ADVISED OF THE POSSIBILITY OF THE DAMAGES.**

**Products equipped with Bluetooth wireless technology ("Bluetooth Products") and their combination, operation, use, implementation, and distribution may be subject to regulatory controls under the laws and regulations of numerous countries that regulate products that use wireless non-licensed spectrum. Examples include airline regulations, telecommunications regulations, technology transfer controls, and health and safety regulations. You are solely responsible for complying with all applicable laws and regulations and for obtaining any and all required authorizations, permits, or licenses in connection with your use of this specification and development, manufacture, and distribution of Bluetooth Products. Nothing in this specification provides any information or assistance in connection with complying with applicable laws or regulations or obtaining required authorizations, permits, or licenses.**

**Bluetooth SIG is not required to adopt any specification or portion thereof. If this specification is not the final version adopted by Bluetooth SIG's Board of Directors, it may not be adopted. Any specification adopted by Bluetooth SIG's Board of Directors may be withdrawn, replaced, or modified at any time. Bluetooth SIG reserves the right to change or alter final specifications in accordance with its membership and operating agreements.**

**Copyright © 1999–2026. All copyrights in the Bluetooth Specifications themselves are owned by Apple Inc., Ericsson AB, Intel Corporation, Google LLC, Lenovo (Singapore) Pte. Ltd., Microsoft Corporation, Nokia Corporation, and Toshiba Corporation. The Bluetooth word mark and logos are owned by Bluetooth SIG, Inc. Other third-party brands and names are the property of their respective owners.**

## Version History

The Version History is shown in [Vol 0] Part C.

## Acknowledgments

The persons who contributed to this specification are listed in [Vol 0] Part C.

## Web Site

This specification can also be found on the official Bluetooth web site:
https://www.bluetooth.com/specifications/adopted-specifications

# Consolidated Table of Contents, Acknowledgments, & Core Configurations

**Specification of the *Bluetooth*® System**

**Volume 0**

**Covered Core Package Version:** v6.3

**Version Date:** 2026-05-05

Bluetooth SIG Proprietary

# CONSOLIDATED TABLE OF CONTENTS

*This table of contents (TOC) covers the entire specification.*

*In addition, each Part of a volume is preceded by a detailed TOC.*

# THE BLUETOOTH CORE SPECIFICATION CONSOLIDATED TABLE OF CONTENTS

In the following Consolidated Table of Contents:

- The TOC for each Volume starts at the top of a page.
- The Volume name in red is followed by the Volume number in black.
- A Volume contains one or more Parts (A, B, etc.); each Part can be viewed independently and has its own TOC.
- Red or blue text on the following pages provides hypertext links to the indicated sections.

# Consolidated Table of Contents, Acknowledgments, & Core Configurations
## *Specification Volume 0*

**Part A**
**CONSOLIDATED TABLE OF CONTENTS**

**The Bluetooth Core Specification Consolidated Table of Contents ..... 6**

**Part B**
**[THIS PART IS NO LONGER USED]**

**Part C**
**VERSION HISTORY AND ACKNOWLEDGMENTS**

**1 Version History ..... 91**
1.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations ..... 91
1.2 [Vol 1] Architecture, Change History, and Conventions ..... 92
1.3 [Vols 2, 3, 5, 6 & 7] Controllers and Host ..... 94
1.4 [Vol 4] Host Controller Interface ..... 98

**2 Acknowledgments (up to v5.1) ..... 100**
2.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations ..... 100
2.1.1 Part B: Bluetooth Compliance Requirements ..... 100
2.1.2 Part C: Version History and Acknowledgments ..... 100
2.2 [Vol 1] Architecture, Change History, and Conventions ..... 101
2.2.1 Part A: Architectural Overview ..... 101
2.2.2 Part B: Acronyms & Abbreviations ..... 107
2.2.3 Part C: Core Specification Change History ..... 108
2.2.4 Part D: Mixing of Specification Versions ..... 109
2.2.5 Part E: General Terminology and Interpretation ..... 110
2.2.6 Part F: Controller Error Codes ..... 111
2.3 [Vol 2] BR/EDR Controller ..... 115
2.3.1 Part A: Radio Physical Layer Specification ..... 115
2.3.2 Part B: Baseband Specification ..... 117
2.3.3 Part C: Link Manager Protocol ..... 123
2.3.4 Part D: [This part is no longer used] ..... 128
2.3.5 Part E: [This part is no longer used] ..... 128
2.3.6 Part F: Message Sequence Charts ..... 129
2.3.7 Part G: Sample Data ..... 132

2.3.8 Part H: Security Specification ..... 135
2.4 [Vol 3] Host ..... 137
2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification ..... 137
2.4.2 Part B: Service Discovery Protocol (SDP) ..... 143
2.4.3 Part C: Generic Access Profile ..... 144
2.4.4 Part D: Test Support ..... 150
2.4.5 Part E: AMP Manager Protocol ..... 152
2.4.6 Part F: Attribute Protocol Specification ..... 152
2.4.7 Part G: Generic Attribute Profile Specification ..... 154
2.4.8 Part H: Security Manager Specification ..... 156
2.5 [Vol 4] Host Controller Interface ..... 158
2.5.1 Parts A to D: Transport Layers ..... 158
2.5.2 Part E: Bluetooth Host Controller Interface Functional Specification ..... 160
2.6 [Vol 5] AMP Controller ..... 170
2.6.1 Part A: 802.11 PAL ..... 170
2.7 [Vol 6] Low Energy Controller ..... 171
2.7.1 Part A: Radio Physical Layer Specification ..... 171
2.7.2 Part B: Link Layer Specification ..... 174
2.7.3 Part C: Sample Data ..... 179
2.7.4 Part D: Message Sequence Charts ..... 180
2.7.5 Part E: Low Energy Security Specification ..... 183
2.7.6 Part F: Direct Test Mode ..... 185
2.8 [Vol 7] Wireless Coexistence Signaling and Interfaces ..... 188
2.8.1 Part A: MWS Coexistence Logical Signaling Specification ..... 188
2.8.2 Part B: Wireless Coexistence Interface 1 (WCI-1) Transport Specification ..... 188
2.8.3 Part C: Wireless Coexistence Interface 2 (WCI-2) Transport Specification ..... 189

**3 Acknowledgments for v5.2 ..... 190**
3.1 Acknowledgments for LE Isochronous Channels ..... 190
3.2 Acknowledgments for LE Power Control ..... 191
3.3 Acknowledgments for Enhanced Attribute Protocol ..... 192

**4 Acknowledgments for v5.3 ..... 194**
4.1 Acknowledgments for AdvDataInfo in Periodic Advertising ..... 194
4.2 Acknowledgments for Host To Controller Encryption Key Control Enhancements ..... 194
4.3 Acknowledgments for LE Enhanced Connection Update ..... 194
4.4 Acknowledgments for LE Channel Classification ..... 195

4.5 Acknowledgments for Removing Alternate MAC/PHY ..... 195

**5 Acknowledgments for v5.4 ..... 196**
5.1 Coding Scheme Selection on Advertising ..... 196
5.2 Encrypted Advertising Data ..... 196
5.3 Periodic Advertising with Responses ..... 196
5.4 LE GATT Security Levels Characteristic ..... 197

**6 Acknowledgments for v6.0 ..... 198**
6.1 Decision-Based Advertising Filtering ..... 198
6.2 Channel Sounding ..... 198
6.3 Enhancements for ISOAL ..... 200
6.4 LL Extended Feature Set ..... 200
6.5 Monitoring Advertisers ..... 200
6.6 Frame Space Update ..... 201
6.7 Core Configurations ..... 201

**7 Acknowledgments for v6.1 ..... 202**
7.1 Randomized RPA Updates ..... 202

**8 Acknowledgments for v6.2 ..... 203**
8.1 LE Test Mode Enhancements ..... 203
8.2 Shorter Connection Intervals ..... 203
8.3 Channel Sounding Amplitude-based Attack Resilience ..... 204
8.4 HCI USB LE Isochronous Support ..... 204

**9 Acknowledgments for v6.3 ..... 206**
9.1 Running Out of Bits ..... 206
9.2 Channel Sounding Inline Phase Correction Term Transfer ..... 206
9.3 Channel Sounding PHY-specific RTT Accuracy ..... 206

**Part D**
**CORE CONFIGURATIONS**

**1 Introduction ..... 209**

**2 Core Configurations ..... 210**
2.1 Core-Controller Configuration ..... 210
2.1.1 BR/EDR Core-Controller Configuration ..... 210
2.1.2 LE Core-Controller Configuration ..... 211
2.1.3 BR/EDR/LE Core-Controller Configuration ..... 211
2.1.4 [This section is no longer used] ..... 212
2.1.5 [This section is no longer used] ..... 212
2.2 Core-Host Configurations ..... 212

2.2.1 BR/EDR Core-Host Configuration ..... 212
2.2.2 LE Core-Host Configuration ..... 212
2.2.3 BR/EDR/LE Core-Host Configuration ..... 213
2.2.4 [This section is no longer used] ..... 213
2.2.5 [This section is no longer used] ..... 213
2.3 Core-Complete Configurations ..... 213
2.3.1 BR/EDR Core-Complete Configuration ..... 213
2.3.2 LE Core-Complete Configuration ..... 213
2.3.3 BR/EDR/LE Core-Complete Configuration ..... 214
2.3.4 [This section is no longer used] ..... 214
2.3.5 [This section is no longer used] ..... 214
2.4 Other Core layers ..... 214

**3 Mixing requirements from different versions ..... 215**

**4 Features and their types ..... 216**

**5 Core Specification Addenda ..... 220**

# Architecture, Change History, and Conventions
## *Specification Volume 1*

**Part A**
**ARCHITECTURE**

**1 General description ..... 230**
1.1 Overview of BR/EDR operation ..... 231
1.2 Overview of Bluetooth Low Energy operation ..... 233
1.3 [This section is no longer used] ..... 239
1.4 Nomenclature ..... 239

**2 Core system architecture ..... 246**
2.1 Core architectural blocks ..... 250
2.1.1 Host architectural blocks ..... 250
2.1.2 BR/EDR/LE Controller architectural blocks ..... 251
2.1.3 [This section is no longer used] ..... 254

**3 Transport architecture ..... 255**
3.1 Core traffic bearers ..... 255
3.1.1 Framed data traffic ..... 257
3.1.2 Unframed data traffic ..... 258
3.1.3 Reliability of traffic bearers ..... 259
3.2 Transport architecture entities ..... 261
3.2.1 BR/EDR generic packet structure ..... 261
3.2.2 LE generic packet structure ..... 263
3.2.3 LE Channel Sounding generic packet structure and signaling format ..... 267
3.3 Physical channels ..... 268
3.3.1 BR/EDR physical channels ..... 269
3.3.2 LE physical channels ..... 274
3.3.3 [This section is no longer used] ..... 282
3.4 Physical links ..... 282
3.4.1 BR/EDR links supported by the basic and adapted piconet physical channels ..... 283
3.4.2 BR/EDR links supported by the scanning physical channels ..... 284
3.4.3 LE links supported by the LE physical channels ..... 284
3.4.4 [This section is no longer used] ..... 286
3.5 Logical links and logical transports ..... 286
3.5.1 Casting ..... 288
3.5.2 Scheduling and acknowledgment scheme ..... 288

3.5.3 Class of data ..... 289
3.5.4 Logical transports ..... 290
3.5.5 Logical links ..... 297
3.6 L2CAP channels ..... 300
3.7 Isochronous Adaptation Layer (ISOAL) ..... 301
3.8 Power control ..... 301
3.8.1 Power control in BR/EDR ..... 301
3.8.2 Power control in LE ..... 301

**4 Communication topology and operation ..... 302**
4.1 Piconet topology ..... 302
4.1.1 BR/EDR topology ..... 302
4.1.2 LE topology ..... 305
4.2 Operational procedures and modes ..... 307
4.2.1 BR/EDR procedures ..... 307
4.2.2 LE procedures ..... 311
4.2.3 [This section is no longer used] ..... 316

**5 Security overview ..... 317**
5.1 Security architecture ..... 317
5.2 BR/EDR Secure Simple Pairing ..... 319
5.2.1 Security goals ..... 320
5.2.2 Passive eavesdropping protection ..... 320
5.2.3 Man-in-the-middle protection ..... 321
5.2.4 Association models ..... 321
5.3 Secure Connections Only mode ..... 324
5.4 LE security ..... 325
5.4.1 Association models ..... 325
5.4.2 Key generation ..... 325
5.4.3 Encryption ..... 325
5.4.4 [This section is no longer used] ..... 326
5.4.5 Privacy feature ..... 326
5.4.6 Encrypted Advertising Data ..... 327
5.5 [This section is no longer used] ..... 327
5.6 Key generation between BR/EDR and LE physical transports ..... 327

**6 Bluetooth application architecture ..... 329**
6.1 Bluetooth profiles ..... 329
6.2 Generic Access Profile ..... 329
6.3 Profile hierarchy ..... 330
6.4 Generic Attribute Architecture ..... 331
6.4.1 Attribute Protocol ..... 331
6.4.2 Generic Attribute Profile ..... 331

6.5 GATT-Based Profile hierarchy ..... 332
6.5.1 Service ..... 333
6.5.2 Included services ..... 334
6.5.3 Characteristic ..... 334
6.6 [This section is no longer used] ..... 334

**7 Coexistence and collocation ..... 335**
7.1 Core features supporting coexistence and collocation ..... 335
7.2 Adaptive Frequency Hopping ..... 336
7.3 Coexistence between Bluetooth Devices and Wireless LAN Devices ..... 337
7.4 Mobile Wireless Standards (MWS) coexistence ..... 337
7.5 Synchronizing Bluetooth with an external timing source ..... 340
7.6 Piconet clock adjustment ..... 341
7.7 Slot Availability Mask (SAM) ..... 342

**8 Direction finding using Bluetooth Low Energy ..... 343**
8.1 Angle of arrival (AoA) method ..... 343
8.2 Angle of departure (AoD) method ..... 344

**9 Channel Sounding using Bluetooth Low Energy ..... 347**
9.1 Channel Sounding procedure ..... 347
9.2 Distance estimation based on phase and amplitude information 348
9.3 Distance estimation based on RTT packets ..... 350
9.4 Security features ..... 351

**Part B**
**ACRONYMS & ABBREVIATIONS**

**1 List of acronyms and abbreviations ..... 354**

**Part C**
**CORE SPECIFICATION CHANGE HISTORY**

**1 Removed features ..... 368**

**2 Changes from v1.1 to v1.2 ..... 369**
2.1 New features ..... 369
2.2 Structure changes ..... 369
2.3 Deprecated features list ..... 370
2.4 Changes in wording ..... 370
2.5 Nomenclature changes ..... 370

**3 Changes from v1.2 to v2.0 + EDR ..... 371**

3.1 New features ..... 371
3.2 Deprecated features ..... 371

**4 Changes from v2.0 + EDR to v2.1 + EDR ..... 372**
4.1 New features ..... 372
4.2 Removed features ..... 372

**5 Changes from v2.1 + EDR to v3.0 + HS ..... 373**
5.1 New features ..... 373
5.2 Removed features ..... 373

**6 Changes from v3.0 + HS to v4.0 ..... 374**
6.1 New features ..... 374
6.2 Removed features ..... 374

**7 Changes from v4.0 to v4.1 ..... 375**
7.1 New features ..... 375
7.1.1 Features added in CSA 4 – integrated in v4.1 ..... 375
7.1.2 Features added in CSA 3 – integrated in v4.1 ..... 375
7.1.3 Features added in CSA 2 – integrated in v4.1 ..... 376
7.2 Removed features ..... 376

**8 Changes from v4.1 to v4.2 ..... 377**
8.1 New features ..... 377
8.2 Errata incorporated in v4.2 ..... 377

**9 Changes from v4.2 to v5.0 ..... 379**
9.1 New features ..... 379
9.1.1 Features added in CSA5 - integrated in v5.0 ..... 379
9.2 Removed features ..... 379
9.3 Privacy errata ..... 379
9.4 Errata incorporated in v5.0 ..... 380

**10 Changes from v5.0 to v5.1 ..... 384**
10.1 New features ..... 384
10.1.1 Features added in CSA6 – integrated in v5.1 ..... 384
10.2 Removed features ..... 384
10.3 Security erratum ..... 384
10.4 Errata incorporated in v5.1 ..... 384

**11 Changes from v5.1 to v5.2 ..... 387**
11.1 New features ..... 387
11.2 Security erratum ..... 387
11.3 Errata incorporated in v5.2 ..... 387

**12 Changes from v5.2 to v5.3 ..... 389**
12.1 New features ..... 389
12.2 Removed Features ..... 389
12.3 Errata incorporated in v5.3 ..... 389
12.4 Global terminology changes ..... 392

**13 Changes from v5.3 to v5.4 ..... 394**
13.1 New features ..... 394
13.2 Removed features ..... 394
13.3 Errata incorporated in v5.4 ..... 394

**14 Changes from v5.4 to v6.0 ..... 397**
14.1 New features ..... 397
14.2 Removed features ..... 397
14.3 Errata incorporated in v6.0 ..... 397

**15 Changes from v6.0 to v6.1 ..... 401**
15.1 New features ..... 401
15.2 Removed features ..... 401
15.3 Errata incorporated in v6.1 ..... 401

**16 Changes from v6.1 to v6.2 ..... 403**
16.1 New features ..... 403
16.2 Removed features ..... 403
16.3 Security errata ..... 403
16.4 Errata incorporated in v6.2 ..... 404

**17 Changes from v6.2 to v6.3 ..... 406**
17.1 New features ..... 406
17.2 Removed features ..... 406
17.3 Errata incorporated in v6.3 ..... 406

**Part D**
**[THIS PART IS NO LONGER USED]**

**Part E**
**GENERAL TERMINOLOGY AND INTERPRETATION**

**1 Language conventions ..... 412**
1.1 [This section is no longer used] ..... 413
1.2 [This section is no longer used] ..... 413
1.3 [This section is no longer used] ..... 413
1.4 [This section is no longer used] ..... 413
1.5 [This section is no longer used] ..... 413

1.6 [This section is no longer used] ..... 413
1.7 Implementation Alternatives ..... 413
1.8 Discrepancies ..... 413
1.9 Appropriate Language ..... 413

**2 General interpretation rules ..... 414**
2.1 Binary and hexadecimal values ..... 414
2.2 Bit numbers and bit fields ..... 414
2.3 Specification of bit values ..... 414
2.4 Values with restricted purposes ..... 415
2.4.1 Reserved for future use ..... 415
2.4.2 Previously used ..... 415
2.4.3 Reserved for specification development purposes ..... 416
2.5 Use of invalid values in checksums and other calculations ..... 416
2.6 Assigned number requirements ..... 416
2.7 Responding to invalid behavior ..... 416
2.8 Ranges of values ..... 418
2.9 Type Names ..... 418
2.9.1 Basic types ..... 419
2.9.2 Array types ..... 419
2.9.3 Variable length types ..... 420
2.10 Mathematical conventions ..... 420
2.11 Requirement status symbols ..... 422
2.12 Table structure ..... 422
2.13 References to HCI commands and events ..... 423

**3 Naming conventions ..... 424**
3.1 BR/EDR ..... 424
3.2 Bluetooth Low Energy ..... 424
3.2.1 Link Layer PDUs ..... 424

**4 Sample Data ..... 427**

**Part F**
**CONTROLLER ERROR CODES**

**1 Overview of error codes ..... 431**
1.1 Usage descriptions ..... 431
1.2 [This section is no longer used] ..... 431
1.3 List of error codes ..... 431

**2 Error code descriptions ..... 434**
2.1 Unknown HCI Command (0x01) ..... 434
2.2 Unknown Connection Identifier (0x02) ..... 434

2.3 Hardware Failure (0x03) ..... 434
2.4 Page Timeout (0x04) ..... 434
2.5 Authentication Failure (0x05) ..... 434
2.6 PIN or Key Missing (0x06) ..... 434
2.7 Memory Capacity Exceeded (0x07) ..... 434
2.8 Connection Timeout (0x08) ..... 435
2.9 Connection Limit Exceeded (0x09) ..... 435
2.10 Synchronous Connection Limit to a Device Exceeded (0x0A) ..... 435
2.11 Connection Already Exists (0x0B) ..... 435
2.12 Command Disallowed (0x0C) ..... 435
2.13 Rejected due to Limited Resources (0x0D) ..... 435
2.14 Rejected due to Security Reasons (0x0E) ..... 435
2.15 Rejected due to Unacceptable BD_ADDR (0x0F) ..... 436
2.16 Connection Accept Timeout Exceeded (0x10) ..... 436
2.17 Unsupported Feature or Parameter Value (0x11) ..... 436
2.18 Invalid HCI Command Parameters (0x12) ..... 436
2.19 Remote User Terminated Connection (0x13) ..... 436
2.20 Remote Device Terminated Connection due to Low Resources (0x14) ..... 437
2.21 Remote Device Terminated Connection due to Power Off (0x15) ..... 437
2.22 Connection Terminated by Local Host (0x16) ..... 437
2.23 Repeated Attempts (0x17) ..... 437
2.24 Pairing Not Allowed (0x18) ..... 437
2.25 Unknown LMP PDU (0x19) ..... 437
2.26 Unsupported Remote Feature (0x1A) ..... 437
2.27 SCO Offset Rejected (0x1B) ..... 437
2.28 SCO Interval Rejected (0x1C) ..... 438
2.29 SCO Air Mode Rejected (0x1D) ..... 438
2.30 Invalid LMP Parameters / Invalid LL Parameters (0x1E) ..... 438
2.31 Unspecified Error (0x1F) ..... 438
2.32 Unsupported LMP Parameter Value / Unsupported LL Parameter Value (0x20) ..... 438
2.33 Role Change Not Allowed (0x21) ..... 438
2.34 LMP Response Timeout / LL Response Timeout (0x22) ..... 439
2.35 LMP Error Transaction Collision / LL Procedure Collision (0x23) 439
2.36 LMP PDU Not Allowed (0x24) ..... 439
2.37 Encryption Mode Not Acceptable (0x25) ..... 439
2.38 Link Key cannot be Changed (0x26) ..... 439
2.39 Requested QoS Not Supported (0x27) ..... 439
2.40 Instant Passed (0x28) ..... 439
2.41 Pairing with Unit Key Not Supported (0x29) ..... 439
2.42 Different Transaction Collision (0x2A) ..... 440

2.43 QoS Unacceptable Parameter (0x2C) ..... 440
2.44 QoS Rejected (0x2D) ..... 440
2.45 Channel Assessment Not Supported (0x2E) ..... 440
2.46 Insufficient Security (0x2F) ..... 440
2.47 Parameter Out of Mandatory Range (0x30) ..... 440
2.48 Role Switch Pending (0x32) ..... 440
2.49 Reserved Slot Violation (0x34) ..... 440
2.50 Role Switch Failed (0x35) ..... 441
2.51 Extended Inquiry Response Too Large (0x36) ..... 441
2.52 Secure Simple Pairing Not Supported by Host (0x37) ..... 441
2.53 Host Busy - Pairing (0x38) ..... 441
2.54 Rejected due to No Suitable Channel Found (0x39) ..... 441
2.55 Controller Busy (0x3A) ..... 441
2.56 Unacceptable Connection Parameters (0x3B) ..... 441
2.57 Advertising Timeout (0x3C) ..... 442
2.58 Connection Terminated due to MIC Failure (0x3D) ..... 442
2.59 Connection Failed to be Established / Synchronization Timeout (0x3E) ..... 442
2.60 [This section is no longer used] ..... 442
2.61 Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging (0x40) ..... 442
2.62 Type0 Submap Not Defined (0x41) ..... 442
2.63 Unknown Advertising Identifier (0x42) ..... 442
2.64 Limit Reached (0x43) ..... 443
2.65 Operation Cancelled by Host (0x44) ..... 443
2.66 Packet Too Long (0x45) ..... 443
2.67 Too Late (0x46) ..... 443
2.68 Too Early (0x47) ..... 443
2.69 Insufficient Channels (0x48) ..... 443

# BR/EDR Controller
## *Specification Volume 2*

**Part A**
**RADIO PHYSICAL LAYER SPECIFICATION**

**1 Scope ..... 448**
1.1 Requirements ..... 448
1.1.1 π/4-DQPSK modulation ..... 449
1.1.2 8DPSK modulation ..... 449
1.1.3 3-slot packets ..... 449
1.1.4 5-slot packets ..... 450
1.1.5 Power control ..... 450
1.1.6 Enhanced power control ..... 450

**2 Frequency bands and channel arrangement ..... 451**

**3 Transmitter characteristics ..... 452**
3.1 Basic Rate ..... 452
3.1.1 Modulation characteristics ..... 452
3.1.2 Spurious emissions ..... 453
3.1.3 Radio frequency tolerance ..... 453
3.2 Enhanced Data Rate ..... 454
3.2.1 Modulation characteristics ..... 454
3.2.2 Spurious emissions ..... 456
3.2.3 Radio frequency tolerance ..... 457
3.2.4 Relative transmit power ..... 458

**4 Receiver characteristics ..... 459**
4.1 Basic Rate ..... 459
4.1.1 Actual sensitivity level ..... 459
4.1.2 Interference performance ..... 459
4.1.3 Out-of-band blocking ..... 460
4.1.4 Intermodulation characteristics ..... 460
4.1.5 Maximum usable level ..... 461
4.1.6 Received Signal Strength Indication ..... 461
4.1.7 Reference signal definition ..... 461
4.2 Enhanced Data Rate ..... 461
4.2.1 Actual sensitivity level ..... 461
4.2.2 BER floor performance ..... 461
4.2.3 Interference performance ..... 461
4.2.4 Maximum usable level ..... 462

4.2.5 Out-of-band and intermodulation characteristics ..... 462
4.2.6 Reference signal definition ..... 463

**5 Power management ..... 464**
5.1 Power classes ..... 464
5.2 Power control ..... 464
5.3 Enhanced power control ..... 465

**Appendix A Test conditions ..... 466**
A.1 Nominal test conditions ..... 466
A.1.1 Nominal temperature ..... 466
A.1.2 Nominal power source ..... 466
A.2 [This section is no longer used] ..... 466

**Appendix B [This Appendix is no longer used] ..... 467**

**Appendix C Modulation accuracy definition ..... 468**
C.1 Enhanced Data Rate modulation accuracy ..... 468
C.1.1 RMS DEVM ..... 470
C.1.2 Peak DEVM ..... 470

**Part B**
**BASEBAND SPECIFICATION**

**1 General description ..... 480**
1.1 Bluetooth clock ..... 481
1.2 Bluetooth Device addressing ..... 482
1.2.1 Reserved addresses ..... 483
1.3 Access codes ..... 484

**2 Physical channels ..... 485**
2.1 Physical channel definition ..... 486
2.2 Basic piconet physical channel ..... 486
2.2.1 Central and Peripheral roles ..... 486
2.2.2 Hopping characteristics ..... 487
2.2.3 Time slots ..... 487
2.2.4 Piconet clocks ..... 488
2.2.5 Transmit/receive timing ..... 488
2.3 Adapted piconet physical channel ..... 492
2.3.1 Hopping characteristics ..... 492
2.4 Page scan physical channel ..... 492
2.4.1 Clock estimate for paging ..... 493
2.4.2 Hopping characteristics ..... 493
2.4.3 Paging procedure timing ..... 493

2.4.4 Page response timing ..... 494
2.5 Inquiry scan physical channel ..... 496
2.5.1 Clock for inquiry ..... 496
2.5.2 Hopping characteristics ..... 497
2.5.3 Inquiry procedure timing ..... 497
2.5.4 Inquiry response timing ..... 497
2.6 Hop selection ..... 498
2.6.1 General selection scheme ..... 499
2.6.2 Selection kernel ..... 502
2.6.3 Adapted hop selection kernel ..... 506
2.6.4 Control word ..... 507
2.7 Synchronization scan physical channel ..... 511
2.7.1 Hopping characteristics ..... 511
2.7.2 Synchronization Train procedure timing ..... 512
2.7.3 Synchronization Scan procedure timing ..... 513

**3 Physical links ..... 514**
3.1 Link supervision for active physical links ..... 514
3.2 Link supervision for Connectionless Peripheral Broadcast physical links ..... 515
3.3 Authenticated payload timeout for active links ..... 515

**4 Logical transports ..... 516**
4.1 General ..... 516
4.2 Logical transport address (LT_ADDR) ..... 516
4.3 Synchronous logical transports ..... 517
4.4 Asynchronous logical transport ..... 517
4.5 Transmit/receive routines ..... 518
4.5.1 TX routine ..... 518
4.5.2 RX routine ..... 521
4.5.3 Flow control ..... 522
4.6 Active Peripheral broadcast transport ..... 523
4.7 [This section is no longer used] ..... 524
4.8 Connectionless Peripheral Broadcast logical transport ..... 524

**5 Logical links ..... 525**
5.1 Link Control logical link (LC) ..... 525
5.2 ACL Control logical links (ACL-C and APB-C) ..... 525
5.3 User asynchronous/isochronous logical links (ACL-U and APB-U) ..... 526
5.3.1 Pausing the ACL-U logical link ..... 526
5.4 User synchronous data logical link (SCO-S) ..... 526
5.5 User extended synchronous data logical link (eSCO-S) ..... 526

5.6 Logical link priorities ..... 526
5.7 Profile broadcast data logical link ..... 527

**6 Packets ..... 528**
6.1 General format ..... 528
6.1.1 Basic Rate ..... 528
6.1.2 Enhanced Data Rate ..... 528
6.2 Bit ordering ..... 529
6.3 Access code ..... 529
6.3.1 Access code types ..... 529
6.3.2 Preamble ..... 530
6.3.3 Sync word ..... 530
6.3.4 Trailer ..... 534
6.4 Packet header ..... 534
6.4.1 LT_ADDR ..... 535
6.4.2 TYPE ..... 535
6.4.3 FLOW ..... 535
6.4.4 ARQN ..... 535
6.4.5 SEQN ..... 536
6.4.6 HEC ..... 536
6.5 Packet types ..... 536
6.5.1 Common packet types ..... 539
6.5.2 SCO packets ..... 541
6.5.3 eSCO packets ..... 542
6.5.4 ACL packets ..... 544
6.6 Payload format ..... 546
6.6.1 Synchronous data field ..... 547
6.6.2 Asynchronous data field ..... 548
6.7 Packet summary ..... 552

**7 Bit stream processing ..... 554**
7.1 Error checking ..... 555
7.1.1 HEC generation ..... 556
7.1.2 CRC generation ..... 557
7.2 Data whitening ..... 558
7.3 Error correction ..... 559
7.4 FEC code: rate 1/3 ..... 560
7.5 FEC code: rate 2/3 ..... 560
7.6 ARQ scheme ..... 561
7.6.1 Unnumbered ARQ ..... 561
7.6.2 Retransmit filtering ..... 565
7.6.3 Flushing payloads ..... 568
7.6.4 Multi-Peripheral considerations ..... 569

7.6.5 Active Peripheral Broadcast packets ..... 569
7.7 Erroneous synchronous data reporting ..... 570
7.8 Message Integrity Check ..... 570

**8 Link Controller operation ..... 571**
8.1 Overview of states ..... 571
8.2 Standby state ..... 572
8.3 Connection establishment substates ..... 572
8.3.1 Page Scan substate ..... 572
8.3.2 Page substate ..... 574
8.3.3 Page response substates ..... 576
8.4 Device discovery substates ..... 581
8.4.1 Inquiry Scan substate ..... 582
8.4.2 Inquiry substate ..... 583
8.4.3 Inquiry Response substate ..... 584
8.5 Connection state ..... 586
8.6 Active mode ..... 587
8.6.1 Polling in the Active mode ..... 588
8.6.2 SCO ..... 589
8.6.3 eSCO ..... 590
8.6.4 Broadcast scheme ..... 593
8.6.5 Role switch ..... 593
8.6.6 Scatternet ..... 595
8.6.7 Hop sequence switching ..... 597
8.6.8 Channel classification and channel map selection ..... 600
8.6.9 Power management ..... 601
8.6.10 Piconet clock adjustment ..... 602
8.6.11 Slot Availability Mask (SAM) ..... 605
8.7 Sniff mode ..... 612
8.7.1 Sniff Transition mode ..... 614
8.7.2 Sniff subrating ..... 614
8.8 Hold mode ..... 616
8.9 [This section is no longer used] ..... 616
8.10 Connectionless Peripheral Broadcast mode ..... 616
8.10.1 Connectionless Peripheral Broadcast transmit operation ..... 617
8.10.2 Connectionless Peripheral Broadcast receive operation ..... 618
8.10.3 AFH in Connectionless Peripheral Broadcast ..... 619
8.11 Synchronization establishment substates ..... 619
8.11.1 Synchronization Scan substate ..... 619
8.11.2 Synchronization Train substate ..... 619

**9 Audio ..... 623**
9.1 LOG PCM codec ..... 623
9.2 CVSD codec ..... 623
9.3 Error handling ..... 626
9.4 General audio requirements ..... 626
9.4.1 Signal levels ..... 626
9.4.2 CVSD audio quality ..... 626

**Appendix A General audio recommendations ..... 627**
A.1 Maximum sound pressure ..... 627
A.2 [This section is no longer used] ..... 627
A.3 Audio levels for Bluetooth ..... 627
A.4 Microphone path ..... 628
A.5 Loudspeaker path ..... 628
A.6 Bluetooth voice interface ..... 628
A.7 Frequency mask ..... 629

**Appendix B Timers ..... 631**
B.1 List of timers ..... 631
B.1.1 inquiryTO ..... 631
B.1.2 pageTO ..... 631
B.1.3 extended_pageTO ..... 631
B.1.4 pagerespTO ..... 631
B.1.5 newconnectionTO ..... 631
B.1.6 supervisionTO ..... 632
B.1.7 CPB_supervisionTO ..... 632
B.1.8 synchronization_trainTO ..... 632
B.1.9 synchronization_scanTO ..... 632
B.1.10 authenticatedPayloadTO ..... 633
B.1.11 CLK_adj_dragTO ..... 633

**Appendix C Recommendations for AFH operation in Hold, Sniff, and Connectionless Peripheral Broadcast modes ..... 634**
C.1 Operation at the Central ..... 634
C.2 [This section is no longer used] ..... 635
C.3 AFH operation in Sniff mode ..... 635
C.4 AFH operation in Hold mode ..... 635
C.5 AFH operation in Connectionless Peripheral Broadcast ..... 635

**Part C**
**LINK MANAGER PROTOCOL SPECIFICATION**

**1 Introduction ..... 642**

**2 General rules ..... 643**
2.1 Message transport ..... 643
2.2 Synchronization ..... 643
2.3 Packet format ..... 644
2.4 Transactions ..... 645
2.4.1 LMP response timeout ..... 645
2.5 Error handling ..... 646
2.5.1 Transaction collision resolution ..... 647
2.6 Procedure rules ..... 647
2.7 General response messages ..... 648
2.8 LMP message constraints ..... 648

**3 Device features ..... 649**
3.1 General description ..... 649
3.2 Feature definitions ..... 649
3.3 Feature mask definition ..... 654
3.4 Link Manager interoperability policy ..... 657
3.5 Feature requirements ..... 657
3.5.1 [This section is no longer used] ..... 660
3.5.2 [This section is no longer used] ..... 660

**4 Procedure rules ..... 661**
4.1 Connection control ..... 661
4.1.1 Connection establishment ..... 661
4.1.2 Detach ..... 662
4.1.3 Power control ..... 663
4.1.4 Adaptive frequency hopping ..... 666
4.1.5 Channel classification ..... 669
4.1.6 Link supervision ..... 671
4.1.7 Channel quality driven data rate change (CQDDR) ..... 672
4.1.8 Quality of service (QoS) ..... 673
4.1.9 Paging scheme parameters ..... 674
4.1.10 Control of multi-slot packets ..... 676
4.1.11 Enhanced Data Rate ..... 677
4.1.12 Encapsulated LMP PDUs ..... 678
4.1.13 Ping ..... 679
4.1.14 Piconet clock adjustment ..... 680
4.1.15 Slot Availability Mask ..... 684
4.2 Security ..... 688
4.2.1 Authentication ..... 689
4.2.2 Pairing ..... 692
4.2.3 Change link key ..... 695
4.2.4 Change current link key type ..... 696

4.2.5 Encryption ..... 697
4.2.6 Request supported encryption key size ..... 707
4.2.7 Secure Simple Pairing ..... 707
4.3 Informational requests ..... 721
4.3.1 Timing accuracy ..... 721
4.3.2 Clock offset ..... 722
4.3.3 LMP version ..... 723
4.3.4 Supported features ..... 724
4.3.5 Name request ..... 725
4.4 Role switch ..... 726
4.4.1 Slot offset ..... 726
4.4.2 Role switch ..... 727
4.5 Modes of operation ..... 730
4.5.1 Hold mode ..... 730
4.5.2 [This section is no longer used] ..... 732
4.5.3 Sniff mode ..... 732
4.6 Logical transports ..... 736
4.6.1 SCO logical transport ..... 736
4.6.2 eSCO logical transport ..... 739
4.7 Test mode ..... 744
4.7.1 Activation and deactivation of Test mode ..... 744
4.7.2 Control of Test mode ..... 745
4.7.3 Summary of Test mode PDUs ..... 746

**5 Summary ..... 749**
5.1 PDU summary ..... 749
5.2 Parameter definitions ..... 757
5.3 LMP encapsulated ..... 768
5.4 Default values ..... 768

**Appendix A Changes to parameter names ..... 770**

**Part D**
**[THIS PART IS NO LONGER USED]**

**Part E**
**[THIS PART IS NO LONGER USED]**

**Part F**
**MESSAGE SEQUENCE CHARTS**

**1 Introduction ..... 779**
1.1 Notation ..... 779

1.2 Flow of control ..... 780
1.3 Example MSC ..... 780
1.4 Forward compatibility ..... 780

**2 Services without connection request ..... 782**
2.1 Remote Name Request ..... 782
2.2 One-time inquiry ..... 784
2.3 Periodic inquiry ..... 786

**3 ACL connection establishment and detachment ..... 789**
3.1 Connection setup ..... 790

**4 Optional activities after ACL connection establishment ..... 798**
4.1 Authentication requested ..... 798
4.2 Secure Simple Pairing message sequence charts ..... 800
4.2.1 Optional OOB information collection ..... 801
4.2.2 Enable Secure Simple Pairing and Secure Connections ..... 802
4.2.3 Connection establishment ..... 803
4.2.4 L2CAP connection request for a secure service ..... 803
4.2.5 Optional OOB information transfer ..... 804
4.2.6 Start Secure Simple Pairing ..... 804
4.2.7 IO capability exchange ..... 806
4.2.8 Public key exchange ..... 806
4.2.9 Authentication ..... 807
4.2.10 Numeric Comparison ..... 808
4.2.11 Numeric Comparison failure on initiating side ..... 809
4.2.12 Numeric Comparison failure on responding side ..... 810
4.2.13 Passkey Entry ..... 810
4.2.14 Passkey Entry failure on responding side ..... 812
4.2.15 Passkey Entry failure on initiator side ..... 813
4.2.16 Out of Band ..... 814
4.2.17 OOB failure on initiator side ..... 816
4.2.18 DHKey checks ..... 816
4.2.19 Calculate link key ..... 818
4.2.20 Enable encryption ..... 819
4.2.21 L2CAP connection response ..... 819
4.2.22 LMP ping ..... 820
4.3 Link Supervision Timeout Changed event ..... 822
4.4 Set Connection Encryption ..... 823
4.5 Change connection link key ..... 825
4.6 Change connection link key with encryption pause and resume 825
4.7 Temporary Link Key ..... 827

4.8 Read remote supported features ..... 828
4.9 Read remote extended features ..... 829
4.10 Read clock offset ..... 830
4.11 Role switch on an encrypted link using encryption pause and resume ..... 831
4.12 Refreshing encryption keys ..... 833
4.13 Read remote version information ..... 834
4.14 QoS setup ..... 835
4.15 Switch role ..... 836
4.16 [This section is no longer used] ..... 838
4.17 [This section is no longer used] ..... 838
4.18 Slot availability mask ..... 838
4.19 LMP transaction collision ..... 842

**5 Synchronous connection establishment and detachment ..... 843**
5.1 Synchronous connection setup ..... 843
5.2 Synchronous connection setup with enhanced synchronous commands ..... 850

**6 Sniff and Hold modes ..... 857**
6.1 Sniff mode ..... 857
6.2 Hold mode ..... 858
6.3 [This section is no longer used] ..... 860

**7 Buffer management, flow control ..... 861**

**8 Loopback mode ..... 863**
8.1 Local Loopback mode ..... 863
8.2 Remote Loopback mode ..... 865

**9 Connectionless Peripheral Broadcast services ..... 867**

**Part G**
**SAMPLE DATA**

**1 Encryption sample data ..... 874**
1.1 E0 encryption sample data ..... 874
1.1.1 Generating K_session from K_enc ..... 874
1.1.2 First set of sample data ..... 876
1.1.3 Second set of sample data ..... 886
1.1.4 Third set of samples ..... 896
1.1.5 Fourth set of samples ..... 906
1.2 AES-CCM encryption sample data ..... 916
1.2.1 Sample data 1 (DM1, Central → Peripheral) ..... 916

1.2.2 Sample data 2 (DM1, Central → Peripheral) ..... 917
1.2.3 Sample data 3 (DM1, Peripheral → Central) ..... 918
1.2.4 Sample data 4 (DM1, Central → Peripheral) ..... 919
1.2.5 Sample data 5 (DM1, Peripheral → Central) ..... 920
1.2.6 Sample data 6 (DH1, Central → Peripheral) ..... 921
1.2.7 Sample data 7 (DH1, Peripheral → Central) ..... 922
1.2.8 Sample data 8 (DH1, Central → Peripheral) ..... 923
1.2.9 Sample data 9 (DH1, Peripheral → Central) ..... 924
1.2.10 Sample data 10 (2-DH3, Central → Peripheral) ..... 925
1.2.11 Sample data 11 (2-DH3, Peripheral → Central) ..... 929
1.2.12 Sample data 12 (3-DH5, Central → Peripheral) ..... 933
1.2.13 Sample data 13 (3-DH5, Peripheral → Central) ..... 943
1.2.14 Sample data 14 (EV3) ..... 953

**2 Frequency hopping sample data ..... 954**
2.1 First set ..... 954
2.2 Second set ..... 960
2.3 Third set ..... 966

**3 Access code sample data ..... 973**

**4 HEC and packet header sample data ..... 976**

**5 CRC sample data ..... 977**

**6 Complete sample packets ..... 978**
6.1 Example of DH1 packet ..... 978
6.2 Example of DM1 packet ..... 980

**7 Secure Simple Pairing sample data ..... 982**
7.1 Elliptic curve sample data ..... 982
7.1.1 P-192 sample data ..... 982
7.1.2 P-256 sample data ..... 983
7.2 Hash functions sample data ..... 984
7.2.1 f1() ..... 984
7.2.2 g() ..... 986
7.2.3 f2() ..... 986
7.2.4 f3() ..... 987
7.2.5 [This section is no longer used] ..... 993
7.2.6 h4() ..... 993
7.2.7 h5() ..... 994
7.2.8 h3() ..... 994

**8 Whitening sequence sample data ..... 995**

**9 FEC sample data ..... 998**

**10 Encryption key sample data ..... 999**
10.1 Four tests of E1 ..... 999
10.2 Four tests of E21 ..... 1004
10.3 Three tests of E22 ..... 1007
10.4 Tests of E22 with Pin augmenting ..... 1009
10.5 Four tests of E3 ..... 1020

**11 Connectionless Peripheral Broadcast sample data ..... 1025**

**Part H**
**SECURITY SPECIFICATION**

**1 Security overview ..... 1030**
1.1 Pausing encryption and role switch ..... 1031
1.2 Change connection link keys ..... 1031
1.3 Periodically refreshing encryption keys ..... 1032

**2 Random number generation ..... 1033**

**3 Key management ..... 1035**
3.1 Key types ..... 1035
3.2 Key generation and initialization ..... 1037
3.2.1 Generation of initialization key, $K_{init}$ ..... 1037
3.2.2 Authentication ..... 1038
3.2.3 [This section is no longer used] ..... 1038
3.2.4 Generation of a combination key ..... 1038
3.2.5 Generating the encryption key ..... 1040
3.2.6 Point-to-multipoint configuration ..... 1040
3.2.7 Modifying the link keys ..... 1041
3.2.8 Generating a temporary link key ..... 1041

**4 Encryption (E0) ..... 1043**
4.1 Encryption key size negotiation ..... 1043
4.2 Encryption of broadcast messages ..... 1044
4.3 Encryption concept ..... 1044
4.4 Encryption algorithm ..... 1045
4.4.1 The operation of the cipher ..... 1047
4.5 LFSR initialization ..... 1048
4.6 Key stream sequence ..... 1051

**5 Authentication ..... 1052**
5.1 Repeated attempts ..... 1055

**6 The authentication and key-generating functions ..... 1056**
6.1 The authentication function $E_1$ ..... 1056
6.2 The functions $A_r$ and $A'_r$ ..... 1058
6.2.1 The round computations ..... 1059
6.2.2 The substitution boxes “e” and “l” ..... 1059
6.2.3 Key scheduling ..... 1061
6.3 $E_2$-key generation function for authentication ..... 1061
6.4 $E_3$-key generation function for encryption ..... 1063

**7 Secure Simple Pairing ..... 1065**
7.1 Phase 1: Public key exchange ..... 1066
7.2 Phase 2: Authentication stage 1 ..... 1067
7.2.1 Authentication stage 1: Numeric Comparison protocol 1067
7.2.2 Authentication stage 1: Out of Band protocol ..... 1069
7.2.3 Authentication stage 1: Passkey Entry protocol ..... 1071
7.3 Phase 3: Authentication stage 2 ..... 1073
7.4 Phase 4: Link key calculation ..... 1074
7.5 Phase 5: LMP authentication and encryption ..... 1075
7.6 Elliptic curve definition ..... 1075
7.7 Cryptographic function definitions ..... 1076
7.7.1 The Secure Simple Pairing commitment function *f1* .. 1076
7.7.2 The Secure Simple Pairing numeric verification function *g* ..... 1077
7.7.3 The Secure Simple Pairing key derivation function *f2* 1078
7.7.4 The Secure Simple Pairing check function *f3* ..... 1079
7.7.5 [This section is no longer used] ..... 1080
7.7.6 The AES encryption key generation function *h3* ..... 1080
7.7.7 The Device authentication key generation function *h4* 1081
7.7.8 The Device authentication confirmation function *h5* .. 1081

**8 [This section is no longer used] ..... 1083**

**9 AES-CCM encryption for BR/EDR ..... 1084**
9.1 Nonce formats ..... 1084
9.2 Counter mode blocks ..... 1086
9.3 Encryption blocks ..... 1087
9.4 Encryption key size reduction ..... 1088
9.5 Repeated MIC failures ..... 1088

# Host
## *Specification Volume 3*

**Part A**
**LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION**

**1 Introduction ..... 1096**
1.1 L2CAP features ..... 1096
1.2 Assumptions ..... 1099
1.3 Scope ..... 1100
1.4 Terminology ..... 1100

**2 General operation ..... 1104**
2.1 Channel identifiers ..... 1104
2.2 Operation between devices ..... 1107
2.3 Operation between layers ..... 1108
2.4 Modes of operation ..... 1109
2.5 Mapping channels to logical links ..... 1111

**3 Data packet format ..... 1112**
3.1 Connection-oriented channels in Basic L2CAP mode ..... 1112
3.2 Connectionless data channel in Basic L2CAP mode ..... 1113
3.3 Connection-oriented channel in Retransmission/Flow Control/Streaming modes ..... 1114
3.3.1 L2CAP header fields ..... 1114
3.3.2 Control field ..... 1115
3.3.3 L2CAP SDU Length field (2 octets) ..... 1119
3.3.4 Information Payload field ..... 1119
3.3.5 Frame Check Sequence (2 octets) ..... 1119
3.3.6 Invalid Frame Detection (Retransmission and Flow Control modes) ..... 1120
3.3.7 Invalid Frame Detection algorithm ..... 1121
3.4 Connection-oriented channels in LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode 1122
3.4.1 L2CAP Header fields ..... 1122
3.4.2 L2CAP SDU Length field (2 octets) ..... 1123
3.4.3 Information Payload field ..... 1123

**4 Signaling packet formats ..... 1124**
4.1 L2CAP_COMMAND_REJECT_RSP (code 0x01) ..... 1127
4.2 L2CAP_CONNECTION_REQ (code 0x02) ..... 1128
4.3 L2CAP_CONNECTION_RSP (code 0x03) ..... 1129

4.4 L2CAP_CONFIGURATION_REQ (code 0x04) ..... 1131
4.5 L2CAP_CONFIGURATION_RSP (code 0x05) ..... 1133
4.6 L2CAP_DISCONNECTION_REQ (code 0x06) ..... 1136
4.7 L2CAP_DISCONNECTION_RSP (code 0x07) ..... 1136
4.8 L2CAP_ECHO_REQ (code 0x08) ..... 1137
4.9 L2CAP_ECHO_RSP (code 0x09) ..... 1137
4.10 L2CAP_INFORMATION_REQ (code 0x0A) ..... 1138
4.11 L2CAP_INFORMATION_RSP (code 0x0B) ..... 1139
4.12 Extended Feature Mask ..... 1140
4.13 Fixed Channels Supported over BR/EDR ..... 1141
4.14 [This section is no longer used] ..... 1142
4.15 [This section is no longer used] ..... 1142
4.16 [This section is no longer used] ..... 1142
4.17 [This section is no longer used] ..... 1142
4.18 [This section is no longer used] ..... 1142
4.19 [This section is no longer used] ..... 1142
4.20 L2CAP_CONNECTION_PARAMETER_UPDATE_REQ (code 0x12) ..... 1142
4.21 L2CAP_CONNECTION_PARAMETER_UPDATE_RSP (code 0x13) ..... 1143
4.22 L2CAP_LE_CREDIT_BASED_CONNECTION_REQ (code 0x14) ..... 1144
4.23 L2CAP_LE_CREDIT_BASED_CONNECTION_RSP (code 0x15) ..... 1145
4.24 L2CAP_FLOW_CONTROL_CREDIT_IND (code 0x16) ..... 1147
4.25 L2CAP_CREDIT_BASED_CONNECTION_REQ (code 0x17) . 1148
4.26 L2CAP_CREDIT_BASED_CONNECTION_RSP (code 0x18) .. 1149
4.27 L2CAP_CREDIT_BASED_RECONFIGURE_REQ (code 0x19) 1151
4.28 L2CAP_CREDIT_BASED_RECONFIGURE_RSP (code 0x1A) 1152

**5 Configuration parameter options ..... 1153**
5.1 Maximum Transmission Unit (MTU) ..... 1153
5.2 Flush Timeout option ..... 1155
5.3 Quality of Service (QoS) option ..... 1156
5.4 Retransmission and Flow Control option ..... 1160
5.5 Frame Check Sequence (FCS) option ..... 1165
5.6 Extended Flow Specification option ..... 1166
5.7 Extended Window Size option ..... 1171

**6 State machine ..... 1173**
6.1 General rules for the state machine ..... 1173
6.1.1 CLOSED state ..... 1174
6.1.2 WAIT_CONNECT_RSP state ..... 1175

6.1.3 WAIT_CONNECT state ..... 1176
6.1.4 CONFIG state ..... 1176
6.1.5 OPEN state ..... 1181
6.1.6 WAIT_DISCONNECT state ..... 1182
6.1.7 [This section is no longer used] ..... 1182
6.1.8 [This section is no longer used] ..... 1182
6.1.9 [This section is no longer used] ..... 1182
6.1.10 [This section is no longer used] ..... 1182
6.1.11 [This section is no longer used] ..... 1182
6.1.12 [This section is no longer used] ..... 1182
6.2 Timers events ..... 1182
6.2.1 RTX ..... 1182
6.2.2 ERTX ..... 1183

**7 General procedures ..... 1186**
7.1 Configuration process ..... 1186
7.1.1 Request Path ..... 1187
7.1.2 Response Path ..... 1188
7.1.3 Lockstep Configuration process ..... 1188
7.1.4 Standard Configuration process ..... 1191
7.2 Fragmentation and recombination ..... 1193
7.2.1 Fragmentation of L2CAP PDUs ..... 1193
7.2.2 Recombination of L2CAP PDUs ..... 1194
7.3 Encapsulation of SDUs ..... 1195
7.3.1 Segmentation of L2CAP SDUs ..... 1195
7.3.2 Reassembly of L2CAP SDUs ..... 1196
7.3.3 Segmentation and fragmentation ..... 1196
7.4 Delivery of erroneous L2CAP SDUs ..... 1197
7.5 Operation with flushing on ACL-U logical links ..... 1198
7.6 Connectionless data channel ..... 1199
7.7 Operation collision resolution ..... 1201
7.8 [This section is no longer used] ..... 1201
7.9 Prioritizing data over HCI ..... 1201
7.10 Supporting Extended Flow Specification for BR/EDR and BR/EDR/LE Controllers ..... 1201
7.11 Enhanced Credit-Based Flow Control Reconfiguration ..... 1203
7.12 Operation with flushing on LE-U logical links ..... 1203

**8 Procedures for Flow Control and Retransmission ..... 1204**
8.1 Information retrieval ..... 1204
8.2 Function of PDU Types for Flow Control and Retransmission ..... 1204
8.2.1 Information frame (I-frame) ..... 1204
8.2.2 Supervisory frame (S-frame) ..... 1204

8.3 Variables and sequence numbers ..... 1205
8.3.1 Sending peer ..... 1206
8.3.2 Receiving peer ..... 1207
8.4 Retransmission mode ..... 1209
8.4.1 Transmitting frames ..... 1209
8.4.2 Receiving I-frames ..... 1211
8.4.3 I-frames pulled by the SDU reassembly function ..... 1211
8.4.4 Sending and receiving acknowledgments ..... 1211
8.4.5 Receiving REJ frames ..... 1213
8.4.6 Waiting acknowledgments ..... 1213
8.4.7 Exception conditions ..... 1213
8.5 Flow Control mode ..... 1215
8.5.1 Transmitting I-frames ..... 1215
8.5.2 Receiving I-frames ..... 1216
8.5.3 I-frames pulled by the SDU reassembly function ..... 1216
8.5.4 Sending and receiving acknowledgments ..... 1216
8.5.5 Waiting acknowledgments ..... 1217
8.5.6 Exception conditions ..... 1217
8.6 Enhanced Retransmission mode ..... 1218
8.6.1 Function of PDU types ..... 1218
8.6.2 Rules for timers ..... 1220
8.6.3 General rules for the state machine ..... 1221
8.6.4 State diagram ..... 1222
8.6.5 States tables ..... 1223
8.7 Streaming mode ..... 1247
8.7.1 Transmitting I-frames ..... 1247
8.7.2 Receiving I-frames ..... 1247
8.7.3 Exception conditions ..... 1248

**9 [This section is no longer used] ..... 1249**

**10 Procedures for Credit Based Flow Control ..... 1250**
10.1 LE Credit Based Flow Control mode ..... 1250
10.2 Enhanced Credit Based Flow Control Mode ..... 1251

**Appendix A Configuration MSCs ..... 1253**

**Appendix B Changes to signaling packet names ..... 1256**

**Part B**
**SERVICE DISCOVERY PROTOCOL (SDP) SPECIFICATION**

**1 Introduction ..... 1261**
1.1 General description ..... 1261

1.2 [This section is no longer used] ..... 1261
1.3 [This section is no longer used] ..... 1261
1.4 [This section is no longer used] ..... 1261
1.5 Conventions ..... 1261
1.5.1 Bit and byte ordering conventions ..... 1261

**2 Overview ..... 1262**
2.1 SDP Client-Server architecture ..... 1262
2.2 Service record ..... 1263
2.3 Service attribute ..... 1264
2.3.1 Attribute ID ..... 1265
2.3.2 Attribute value ..... 1265
2.4 Service class ..... 1265
2.4.1 A printer service class example ..... 1266
2.5 Searching for services ..... 1266
2.5.1 UUID ..... 1267
2.5.2 Service search patterns ..... 1267
2.6 Browsing for services ..... 1268
2.6.1 Example service browsing hierarchy ..... 1268

**3 Data representation ..... 1271**
3.1 Data element ..... 1271
3.2 Data element type descriptor ..... 1271
3.3 Data element size descriptor ..... 1272
3.4 Data element examples ..... 1273

**4 Protocol description ..... 1274**
4.1 Transfer byte order ..... 1274
4.2 Protocol Data Unit format ..... 1274
4.3 Partial responses and continuation state ..... 1275
4.4 Error handling ..... 1276
4.4.1 SDP_ERROR_RSP PDU ..... 1277
4.5 Service Search transaction ..... 1277
4.5.1 SDP_SERVICE_SEARCH_REQ PDU ..... 1278
4.5.2 SDP_SERVICE_SEARCH_RSP PDU ..... 1279
4.6 Service Attribute transaction ..... 1280
4.6.1 SDP_SERVICE_ATTR_REQ PDU ..... 1281
4.6.2 SDP_SERVICE_ATTR_RSP PDU ..... 1282
4.7 Service Search Attribute transaction ..... 1283
4.7.1 SDP_SERVICE_SEARCH_ATTR_REQ PDU ..... 1284
4.7.2 SDP_SERVICE_SEARCH_ATTR_RSP PDU ..... 1285

**5 Service attribute definitions ..... 1288**
5.1 Universal attribute definitions ..... 1288

5.1.1 ServiceRecordHandle attribute ..... 1288
5.1.2 ServiceClassIDList attribute ..... 1289
5.1.3 ServiceRecordState attribute ..... 1289
5.1.4 ServiceID attribute ..... 1289
5.1.5 ProtocolDescriptorList attribute ..... 1290
5.1.6 AdditionalProtocolDescriptorLists attribute ..... 1292
5.1.7 BrowseGroupList attribute ..... 1293
5.1.8 LanguageBaseAttributeIDList attribute ..... 1294
5.1.9 ServiceInfoTimeToLive attribute ..... 1295
5.1.10 ServiceAvailability attribute ..... 1295
5.1.11 BluetoothProfileDescriptorList attribute ..... 1296
5.1.12 DocumentationURL attribute ..... 1296
5.1.13 ClientExecutableURL attribute ..... 1296
5.1.14 IconURL attribute ..... 1297
5.1.15 ServiceName attribute ..... 1297
5.1.16 ServiceDescription attribute ..... 1298
5.1.17 ProviderName attribute ..... 1298
5.1.18 [This section is no longer used] ..... 1298
5.2 ServiceDiscoveryServer service class attribute definitions ..... 1298
5.2.1 ServiceRecordHandle attribute ..... 1298
5.2.2 ServiceClassIDList attribute ..... 1299
5.2.3 VersionNumberList attribute ..... 1299
5.2.4 ServiceDatabaseState attribute ..... 1299
5.2.5 [This section is no longer used] ..... 1300
5.3 BrowseGroupDescriptor service class attribute definitions ..... 1300
5.3.1 ServiceClassIDList attribute ..... 1300
5.3.2 GroupID attribute ..... 1300
5.3.3 [This section is no longer used] ..... 1300

**6 Security ..... 1301**

**Appendix A [This appendix is no longer used] ..... 1302**

**Appendix B Example SDP Transactions ..... 1303**
B.1 SDP example 1 – ServiceSearchRequest ..... 1303
B.2 SDP example 2 – ServiceAttributeTransaction ..... 1305
B.3 SDP example 3 – ServiceSearchAttributeTransaction ..... 1310

**Appendix C Changes to PDU names ..... 1321**

**Part C**
**GENERIC ACCESS PROFILE**

**1 Foreword ..... 1333**

1.1 Scope ..... 1333
1.2 Symbols and conventions ..... 1334
1.2.1 [This section is no longer used] ..... 1334
1.2.2 Signaling diagram conventions ..... 1334
1.2.3 Notation for timers and counters ..... 1335
1.2.4 Notation for UUIDs ..... 1335
1.3 GAP requirements ..... 1335

**2 Profile overview ..... 1336**
2.1 Profile stack ..... 1336
2.2 Profile roles ..... 1336
2.2.1 Roles when operating over BR/EDR Physical Transport ..... 1336
2.2.2 Roles when operating over an LE physical transport 1337
2.3 User requirements and scenarios ..... 1340
2.4 Profile fundamentals ..... 1340
2.5 [This section is no longer used] ..... 1341

**3 User interface aspects ..... 1342**
3.1 The user interface level ..... 1342
3.2 Representation of Bluetooth parameters ..... 1342
3.2.1 Bluetooth Device Address (BD_ADDR) ..... 1342
3.2.2 Bluetooth Device Name (the user-friendly name) ..... 1342
3.2.3 Bluetooth Passkey (Bluetooth PIN) ..... 1343
3.2.4 Class of Device ..... 1345
3.2.5 Appearance characteristic ..... 1346
3.2.6 Broadcast Code ..... 1346
3.3 Pairing ..... 1347

**4 Modes – BR/EDR physical transport ..... 1348**
4.1 Discoverability modes ..... 1348
4.1.1 Non-discoverable mode ..... 1349
4.1.2 Limited Discoverable mode ..... 1349
4.1.3 General Discoverable mode ..... 1350
4.2 Connectability modes ..... 1351
4.2.1 Non-connectable mode ..... 1351
4.2.2 Connectable mode ..... 1352
4.3 Bondable modes ..... 1353
4.3.1 Non-bondable mode ..... 1353
4.3.2 Bondable mode ..... 1353
4.4 Synchronizability modes ..... 1354
4.4.1 Non-synchronizable mode ..... 1354
4.4.2 Synchronizable mode ..... 1354

**5 Security aspects – BR/EDR physical transport ..... 1355**
5.1 Authentication ..... 1355
5.1.1 Purpose ..... 1355
5.1.2 Term on UI level ..... 1355
5.1.3 Procedure ..... 1356
5.1.4 Conditions ..... 1357
5.2 Security modes ..... 1357
5.2.1 Legacy security modes ..... 1358
5.2.2 Security mode 4 (service level enforced security) ..... 1359

**6 Idle mode procedures – BR/EDR physical transport ..... 1376**
6.1 General Inquiry ..... 1376
6.1.1 Purpose ..... 1376
6.1.2 Term on UI level ..... 1377
6.1.3 Description ..... 1377
6.1.4 Conditions ..... 1377
6.2 Limited Inquiry ..... 1377
6.2.1 Purpose ..... 1377
6.2.2 Term on UI level ..... 1378
6.2.3 Description ..... 1378
6.2.4 Conditions ..... 1378
6.3 Name Discovery ..... 1378
6.3.1 Purpose ..... 1378
6.3.2 Term on UI level ..... 1379
6.3.3 Description ..... 1379
6.3.4 Conditions ..... 1380
6.4 Device Discovery ..... 1380
6.4.1 Purpose ..... 1380
6.4.2 Term on UI level ..... 1380
6.4.3 Description ..... 1380
6.4.4 Conditions ..... 1381
6.5 Bonding ..... 1381
6.5.1 Purpose ..... 1381
6.5.2 Term on UI level ..... 1381
6.5.3 Description ..... 1382
6.5.4 Conditions ..... 1384

**7 Establishment procedures – BR/EDR physical transport ..... 1385**
7.1 Link Establishment ..... 1385
7.1.1 Purpose ..... 1385
7.1.2 Term on UI level ..... 1385
7.1.3 Description ..... 1386
7.1.4 Conditions ..... 1387

7.2 Channel Establishment ..... 1388
7.2.1 Purpose ..... 1388
7.2.2 Term on UI level ..... 1388
7.2.3 Description ..... 1388
7.2.4 Conditions ..... 1389
7.3 Connection Establishment ..... 1390
7.3.1 Purpose ..... 1390
7.3.2 Term on UI level ..... 1390
7.3.3 Description ..... 1390
7.3.4 Conditions ..... 1391
7.4 Establishment of additional connection ..... 1391
7.5 Synchronization Establishment ..... 1392
7.5.1 Purpose ..... 1392
7.5.2 Term on UI Level ..... 1392
7.5.3 Description ..... 1392
7.5.4 Conditions ..... 1392

**8 Extended inquiry response data format ..... 1394**

**9 Operational modes and procedures – LE physical transport ..... 1396**
9.1 Broadcast mode and Observation procedure ..... 1396
9.1.1 Broadcast mode ..... 1397
9.1.2 Observation procedure ..... 1397
9.2 Discovery modes and procedures ..... 1398
9.2.1 Requirements ..... 1398
9.2.2 Non-discoverable mode ..... 1398
9.2.3 Limited Discoverable mode ..... 1399
9.2.4 General Discoverable mode ..... 1400
9.2.5 Limited Discovery procedure ..... 1402
9.2.6 General Discovery procedure ..... 1403
9.2.7 Name Discovery procedure ..... 1405
9.3 Connection modes and procedures ..... 1405
9.3.1 Requirements ..... 1406
9.3.2 Non-connectable mode ..... 1407
9.3.3 Directed Connectable mode ..... 1407
9.3.4 Undirected Connectable mode ..... 1407
9.3.5 Auto Connection Establishment procedure ..... 1408
9.3.6 General Connection Establishment procedure ..... 1410
9.3.7 Selective Connection Establishment procedure ..... 1412
9.3.8 Direct Connection Establishment procedure ..... 1415
9.3.9 Connection Parameter Update procedure ..... 1416
9.3.10 Terminate Connection procedure ..... 1417
9.3.11 Connection Establishment Timing parameters ..... 1417

9.3.12 Connection interval timing parameters ..... 1418
9.3.13 Connected Isochronous Stream Central Establishment procedure ..... 1420
9.3.14 Connected Isochronous Stream Peripheral Establishment procedure ..... 1420
9.3.15 Connected Isochronous Stream Terminate procedure 1420
9.3.16 Connection Subrate procedure ..... 1421
9.3.17 Periodic Advertising Connection procedure ..... 1421
9.4 Bonding modes and procedures ..... 1421
9.4.1 Requirements ..... 1422
9.4.2 Non-bondable mode ..... 1422
9.4.3 Bondable mode ..... 1422
9.4.4 Bonding procedure ..... 1423
9.5 Periodic advertising modes and procedure ..... 1423
9.5.1 Periodic Advertising Synchronizability mode ..... 1424
9.5.2 Periodic Advertising mode ..... 1424
9.5.3 Periodic Advertising Synchronization Establishment procedure ..... 1425
9.5.4 Periodic Advertising Synchronization Transfer procedure ..... 1425
9.5.5 [This section is no longer used] ..... 1426
9.6 Isochronous Broadcast modes and procedures ..... 1426
9.6.1 Broadcast Isochronous Synchronizability mode ..... 1426
9.6.2 Broadcast Isochronous Broadcasting mode ..... 1427
9.6.3 Broadcast Isochronous Synchronization Establishment procedure ..... 1427
9.6.4 Broadcast Isochronous Channel Map Update procedure ..... 1427
9.6.5 Broadcast Isochronous Terminate procedure ..... 1427
9.7 Channel Sounding procedures ..... 1428
9.7.1 Channel Sounding initiator procedure ..... 1428
9.7.2 Channel Sounding reflector procedure ..... 1428

**10 Security aspects – LE physical transport ..... 1430**
10.1 Requirements ..... 1430
10.2 LE security modes ..... 1430
10.2.1 LE security mode 1 ..... 1431
10.2.2 [This section is no longer used] ..... 1431
10.2.3 [This section is no longer used] ..... 1431
10.2.4 Secure Connections Only mode ..... 1431
10.2.5 LE security mode 3 ..... 1431
10.3 Authentication procedure ..... 1432
10.3.1 Responding to a service request ..... 1432

10.3.2 Initiating a service request ..... 1436
10.4 [This section is no longer used] ..... 1440
10.5 Authorization procedure ..... 1440
10.6 Encryption procedure ..... 1440
10.7 Privacy feature ..... 1441
10.7.1 Privacy feature in a Peripheral ..... 1442
10.7.2 Privacy feature in a Central ..... 1443
10.7.3 Privacy feature in a Broadcaster ..... 1444
10.7.4 Privacy feature in an Observer ..... 1445
10.8 Random Device address ..... 1445
10.8.1 Static address ..... 1446
10.8.2 Private address ..... 1446
10.9 Encrypted Broadcast Isochronous Group ..... 1446
10.10 Encrypted Advertising Data procedure ..... 1447
10.11 LE Channel Sounding ..... 1447
10.11.1 Channel Sounding security ..... 1447

**11 Advertising and Scan Response data format ..... 1449**

**12 GAP Service and characteristics for GATT Server ..... 1451**
12.1 Device Name characteristic ..... 1452
12.2 Appearance characteristic ..... 1452
12.3 Peripheral Preferred Connection Parameters characteristic ..... 1453
12.4 Central Address Resolution characteristic ..... 1454
12.5 Resolvable Private Address Only characteristic ..... 1454
12.6 Encrypted Data Key Material ..... 1455
12.7 LE GATT Security Levels Characteristic ..... 1456

**13 BR/EDR/LE operation ..... 1458**
13.1 Modes, procedures, and security aspects ..... 1458
13.1.1 Discoverable mode requirements ..... 1458
13.2 Bonding for BR/EDR/LE implementations ..... 1458
13.3 Relationship between physical transports ..... 1459

**14 BR/EDR/LE security aspects ..... 1460**
14.1 Cross-transport key derivation ..... 1460
14.2 Collision handling ..... 1461
14.3 Secure Connections Only Mode ..... 1461

**15 Bluetooth Device requirements ..... 1462**
15.1 Bluetooth Device address ..... 1462
15.1.1 Bluetooth Device Address types ..... 1462
15.2 GATT Profile requirements ..... 1462
15.3 SDP requirements ..... 1462

15.4 SDP service record requirement ..... 1463

**16 Definitions ..... 1465**
16.1 General definitions ..... 1465
16.2 Connection-related definitions ..... 1465
16.3 Device-related definitions ..... 1466
16.4 Procedure-related definitions ..... 1467
16.5 Security-related definitions ..... 1467

**17 References ..... 1469**

**Appendix A Timers and Constants ..... 1470**

**Appendix B Information Flows of Related Procedures ..... 1475**
B.1 LMP – authentication ..... 1475
B.2 LMP – pairing ..... 1476
B.3 Service Discovery ..... 1476
B.4 Generating a resolvable private address ..... 1477
B.5 Resolving a resolvable private address ..... 1477

**Part D**
**TEST SUPPORT**

**1 Test methodology ..... 1480**
1.1 BR/EDR test scenarios ..... 1480
1.1.1 Test setup ..... 1480
1.1.2 Transmitter test ..... 1481
1.1.3 LoopBack test ..... 1485
1.1.4 Pause test ..... 1489
1.2 [This section is no longer used] ..... 1489
1.3 References ..... 1489

**2 [This section is no longer used] ..... 1490**

**Part E**
**[THIS PART IS NO LONGER USED]**

**Part F**
**ATTRIBUTE PROTOCOL (ATT)**

**1 Introduction ..... 1495**
1.1 Scope ..... 1495
1.2 [This section is no longer used] ..... 1495
1.3 Conventions ..... 1495

**2 Protocol overview ..... 1496**

**3 Protocol requirements ..... 1498**
3.1 Introduction ..... 1498
3.2 Basic concepts ..... 1498
3.2.1 Attribute type ..... 1498
3.2.2 Attribute handle ..... 1498
3.2.3 Attribute handle grouping ..... 1499
3.2.4 Attribute value ..... 1499
3.2.5 Attribute permissions ..... 1499
3.2.6 Control-point attributes ..... 1501
3.2.7 Protocol methods ..... 1501
3.2.8 Exchanging MTU size ..... 1502
3.2.9 Long attribute values ..... 1502
3.2.10 Atomic operations ..... 1502
3.2.11 ATT bearers ..... 1503
3.3 Attribute PDU ..... 1504
3.3.1 Attribute PDU format ..... 1505
3.3.2 Sequential protocol ..... 1505
3.3.3 Transaction ..... 1506
3.4 Attribute Protocol PDUs ..... 1507
3.4.1 Error handling ..... 1507
3.4.2 MTU exchange ..... 1509
3.4.3 Find information ..... 1511
3.4.4 Reading attributes ..... 1515
3.4.5 Writing attributes ..... 1527
3.4.6 Queued writes ..... 1530
3.4.7 Server initiated ..... 1534
3.4.8 Attribute Opcode summary ..... 1536
3.4.9 Attribute PDU response summary ..... 1538

**4 Security considerations ..... 1544**

**5 References ..... 1546**

**Appendix A Changes to PDU names ..... 1547**

**Part G**
**GENERIC ATTRIBUTE PROFILE (GATT)**

**1 Introduction ..... 1552**
1.1 Scope ..... 1552
1.2 Profile dependency ..... 1552
1.3 [This section is no longer used] ..... 1552

1.4 [This section is no longer used] ..... 1552
1.5 Conventions ..... 1552

**2 Profile overview ..... 1553**
2.1 Protocol stack ..... 1553
2.2 Configurations and roles ..... 1553
2.3 User requirements and scenarios ..... 1554
2.4 Profile fundamentals ..... 1555
2.5 Attribute Protocol ..... 1555
2.5.1 Overview ..... 1555
2.5.2 Attribute caching ..... 1556
2.5.3 Attribute grouping ..... 1562
2.5.4 UUIDs ..... 1562
2.6 GATT Profile hierarchy ..... 1562
2.6.1 Overview ..... 1562
2.6.2 Service ..... 1563
2.6.3 Included services ..... 1564
2.6.4 Characteristic ..... 1564
2.7 Configured Broadcast ..... 1564

**3 Service interoperability requirements ..... 1566**
3.1 Service definition ..... 1566
3.2 Include definition ..... 1567
3.3 Characteristic definition ..... 1567
3.3.1 Characteristic declaration ..... 1568
3.3.2 Characteristic Value declaration ..... 1570
3.3.3 Characteristic descriptor declarations ..... 1570
3.4 Summary of GATT Profile attribute types ..... 1577

**4 GATT feature requirements ..... 1578**
4.1 Overview ..... 1578
4.2 Feature support and procedure mapping ..... 1578
4.3 Server configuration ..... 1580
4.3.1 Exchange MTU ..... 1580
4.4 Primary Service Discovery ..... 1581
4.4.1 Discover All Primary Services ..... 1581
4.4.2 Discover Primary Service by Service UUID ..... 1582
4.5 Relationship Discovery ..... 1583
4.5.1 Find Included Services ..... 1584
4.6 Characteristic discovery ..... 1585
4.6.1 Discover All Characteristics of a Service ..... 1585
4.6.2 Discover Characteristics by UUID ..... 1586
4.7 Characteristic Descriptor Discovery ..... 1588

4.7.1 Discover All Characteristic Descriptors ..... 1588
4.8 Characteristic Value Read ..... 1589
4.8.1 Read Characteristic Value ..... 1589
4.8.2 Read Using Characteristic UUID ..... 1590
4.8.3 Read Long Characteristic Value ..... 1591
4.8.4 Read Multiple Characteristic Values ..... 1592
4.8.5 Read Multiple Variable Length Characteristic Values 1593
4.9 Characteristic Value Write ..... 1594
4.9.1 Write Without Response ..... 1594
4.9.2 [This section is no longer used] ..... 1594
4.9.3 Write Characteristic Value ..... 1594
4.9.4 Write Long Characteristic Value ..... 1595
4.9.5 Characteristic Value Reliable Writes ..... 1596
4.10 Characteristic Value Notification ..... 1598
4.10.1 Single Notification ..... 1599
4.10.2 Multiple Variable Length Notifications ..... 1599
4.11 Characteristic Value Indication ..... 1599
4.11.1 Indication ..... 1600
4.12 Characteristic Descriptors ..... 1600
4.12.1 Read Characteristic Descriptor ..... 1600
4.12.2 Read Long Characteristic Descriptor ..... 1601
4.12.3 Write Characteristic Descriptor ..... 1602
4.12.4 Write Long Characteristic Descriptor ..... 1603
4.13 GATT procedure mapping to ATT protocol opcodes ..... 1604
4.14 Procedure timeouts ..... 1607

**5 L2CAP interoperability requirements ..... 1608**
5.1 BR/EDR L2CAP interoperability requirements ..... 1608
5.1.1 ATT_MTU ..... 1608
5.1.2 BR/EDR channel requirements ..... 1608
5.1.3 [This section is no longer used] ..... 1609
5.2 LE L2CAP interoperability requirements ..... 1609
5.2.1 ATT_MTU ..... 1609
5.2.2 LE channel requirements ..... 1609
5.3 Enhanced ATT bearer L2CAP interoperability requirements .... 1610
5.3.1 ATT_MTU ..... 1610
5.3.2 Channel Requirements ..... 1610
5.4 L2CAP collision mitigation ..... 1610
5.5 Bearer support ..... 1611

**6 GAP interoperability requirements ..... 1612**
6.1 BR/EDR GAP interoperability requirements ..... 1612
6.1.1 Connection Establishment ..... 1612

6.2 LE GAP interoperability requirements ..... 1612
6.2.1 Connection Establishment ..... 1612
6.2.2 Profile roles ..... 1613
6.3 Disconnected events ..... 1613
6.3.1 Notifications and indications while disconnected ..... 1613

**7 Defined GATT service ..... 1614**
7.1 Service Changed ..... 1614
7.2 Client Supported Features ..... 1615
7.3 Database Hash ..... 1617
7.3.1 Database Hash calculation ..... 1618
7.4 Server Supported Features ..... 1619

**8 Security considerations ..... 1620**
8.1 Authentication requirements ..... 1620
8.2 Authorization requirements ..... 1621

**9 SDP interoperability requirements ..... 1622**

**10 References ..... 1624**

**Appendix A Example ATT Server contents ..... 1625**

**Appendix B Example Database Hash ..... 1628**

**Part H**
**SECURITY MANAGER SPECIFICATION**

**1 Introduction ..... 1635**
1.1 Scope ..... 1635
1.2 Conventions ..... 1635
1.2.1 Bit and byte ordering conventions ..... 1635
1.2.2 Random numbers ..... 1636
1.3 Requirements ..... 1636

**2 Security Manager ..... 1637**
2.1 Introduction ..... 1637
2.2 Cryptographic toolbox ..... 1639
2.2.1 Security function *e* ..... 1640
2.2.2 Random address hash function *ah* ..... 1640
2.2.3 Confirm value generation function *c1* for LE legacy pairing ..... 1641
2.2.4 Key generation function *s1* for LE legacy pairing ..... 1642
2.2.5 Function AES-CMAC ..... 1643

2.2.6 LE Secure Connections confirm value generation function *f4* ..... 1643
2.2.7 LE Secure Connections key generation function *f5* ... 1644
2.2.8 LE Secure Connections check value generation function *f6* ..... 1646
2.2.9 LE Secure Connections numeric comparison value generation function *g2* ..... 1647
2.2.10 Link key conversion function *h6* ..... 1648
2.2.11 Link key conversion function *h7* ..... 1648
2.3 Pairing methods ..... 1649
2.3.1 Security Properties ..... 1650
2.3.2 IO capabilities ..... 1651
2.3.3 OOB authentication data ..... 1652
2.3.4 Encryption key size ..... 1652
2.3.5 Pairing algorithms ..... 1653
2.3.6 Repeated attempts ..... 1669
2.4 Security in Bluetooth Low Energy ..... 1669
2.4.1 Definition of keys and values ..... 1669
2.4.2 Generation of distributed keys ..... 1670
2.4.3 Distribution of keys ..... 1672
2.4.4 Encrypted session setup ..... 1673
2.4.5 [This section is no longer used] ..... 1674
2.4.6 Peripheral Security Request ..... 1674

**3 Security Manager Protocol ..... 1677**
3.1 Introduction ..... 1677
3.2 Security Manager Channel over L2CAP ..... 1677
3.3 Command format ..... 1677
3.4 SMP timeout ..... 1678
3.5 Pairing methods ..... 1679
3.5.1 Pairing Request ..... 1679
3.5.2 Pairing Response ..... 1682
3.5.3 Pairing Confirm ..... 1684
3.5.4 Pairing Random ..... 1685
3.5.5 Pairing Failed ..... 1686
3.5.6 Pairing Public Key ..... 1687
3.5.7 Pairing DHKey Check ..... 1688
3.5.8 Keypress Notification ..... 1689
3.6 Security in Bluetooth Low Energy ..... 1689
3.6.1 Key distribution and generation ..... 1689
3.6.2 Encryption Information ..... 1692
3.6.3 Central Identification ..... 1693
3.6.4 Identity Information ..... 1694

3.6.5 Identity Address Information ..... 1694
3.6.6 [This section is no longer used] ..... 1695
3.6.7 Security Request ..... 1695

**4 References ..... 1697**

**Appendix A EDIV and Rand Generation ..... 1698**
A.1 EDIV masking ..... 1698
A.1.1 DIV mask generation function *dm* ..... 1698
A.1.2 EDIV generation ..... 1699
A.1.3 DIV recovery ..... 1699

**Appendix B Key Management ..... 1700**
B.1 Database lookup ..... 1700
B.2 Key hierarchy ..... 1700
B.2.1 Diversifying function *d1* ..... 1701
B.2.2 Generating keys from ER ..... 1702
B.2.3 Generating keys from IR ..... 1702

**Appendix C Message sequence charts ..... 1704**
C.1 Phase 1: Pairing feature exchange ..... 1704
C.1.1 Peripheral security request – Central requests pairing 1705
C.2 Phase 2: Authenticating and encrypting ..... 1705
C.2.1 LE legacy pairing ..... 1705
C.2.2 LE Secure Connections ..... 1708
C.3 Phase 3: Transport specific key distribution ..... 1722
C.4 Security re-established using previously distributed LTK ..... 1722
C.4.1 Central initiated security - Central initiated Link Layer encryption ..... 1722
C.4.2 Peripheral security request - Central initiated Link Layer encryption ..... 1722
C.5 Failure conditions ..... 1723
C.5.1 Pairing not supported by Peripheral ..... 1723
C.5.2 Central rejects pairing because of key size ..... 1723
C.5.3 Peripheral rejects pairing because of key size ..... 1724
C.5.4 Passkey Entry failure on Central ..... 1725
C.5.5 Passkey Entry failure on Peripheral ..... 1726
C.5.6 Peripheral rejects Central's confirm value ..... 1726
C.5.7 Central rejects Peripheral's confirm value ..... 1727
C.6 Cross-transport key derivation ..... 1728
C.6.1 LE derivation from BR/EDR ..... 1728
C.6.2 BR/EDR derivation from LE ..... 1729

**Appendix D Sample data ..... 1731**

D.1 AES-CMAC RFC4493 test vectors ..... 1731
D.1.1 Example 1: Len = 0 ..... 1731
D.1.2 Example 2: Len = 16 ..... 1731
D.1.3 Example 3: Len = 40 ..... 1731
D.1.4 Example 4: Len = 64 ..... 1731
D.2 *f4* LE SC confirm value generation function ..... 1731
D.3 *f5* LE SC key generation function ..... 1732
D.4 *f6* LE SC check value generation function ..... 1732
D.5 *g2* LE SC numeric comparison generation function ..... 1733
D.6 *h6* LE SC link key conversion function ..... 1733
D.7 *ah* random address hash functions ..... 1733
D.8 *h7* LE SC link key conversion function ..... 1733
D.9 LTK to link key conversion using CT2=1 ..... 1733
D.10 LTK to link key conversion using CT2=0 ..... 1733
D.11 Link key to LTK conversion using CT2=1 ..... 1734
D.12 Link key to LTK conversion using CT2=0 ..... 1734

# Host Controller Interface
## *Specification Volume 4*

**Part A**
**UART TRANSPORT LAYER**

**1 General ..... 1738**

**2 Protocol ..... 1739**

**3 RS232 settings ..... 1740**

**4 Error recovery ..... 1741**

**Part B**
**USB TRANSPORT LAYER**

**1 Overview ..... 1744**
1.1 Requirements ..... 1745

**2 USB endpoint expectations ..... 1747**
2.1 Descriptor overview ..... 1747
2.1.1 Controller descriptors ..... 1747
2.1.2 [This section is no longer used] ..... 1753
2.2 Control endpoint expectations ..... 1753
2.2.1 Single function Controller ..... 1753
2.2.2 Controller function in a composite device ..... 1753
2.2.3 [This section is no longer used] ..... 1754
2.3 Bulk endpoints expectations ..... 1754
2.4 Interrupt endpoint expectations ..... 1754
2.5 Isochronous endpoints expectations ..... 1754

**3 Class code ..... 1756**
3.1 Bluetooth codes ..... 1756
3.1.1 [This section is no longer used] ..... 1756
3.1.2 [This section is no longer used] ..... 1756
3.1.3 [This section is no longer used] ..... 1756
3.1.4 [This section is no longer used] ..... 1756

**4 Device firmware upgrade ..... 1757**

**5 Limitations ..... 1758**
5.1 Power specific limitations ..... 1758

5.2 Other limitations ..... 1758

**6 Bluetooth Composite Device implementation ..... 1759**
6.1 Configurations ..... 1759
6.2 Using USB Interface Association Descriptors for a Controller function ..... 1759
6.3 [This section is no longer used] ..... 1760

**7 USB Bulk Serialization mode ..... 1761**

**8 References ..... 1762**

**Part C**
**SECURE DIGITAL (SD) TRANSPORT LAYER**

**1 Introduction ..... 1765**

**2 Goals ..... 1766**
2.1 Hardware goals ..... 1766
2.2 Software goals ..... 1766
2.3 Configuration goals ..... 1766
2.4 Configuration for multiple Controllers ..... 1767

**3 Physical interface documents ..... 1768**

**4 Communication ..... 1769**
4.1 Overview ..... 1769

**Appendix A Acronyms and Abbreviations ..... 1770**

**Appendix B Related Documents ..... 1771**

**Appendix C Tests ..... 1772**
C.1 Test suite structure ..... 1772

**Part D**
**THREE-WIRE UART TRANSPORT LAYER**

**1 General ..... 1776**

**2 Overview ..... 1777**

**3 Slip layer ..... 1778**
3.1 Encoding a packet ..... 1778
3.2 Decoding a packet ..... 1778

**4 Packet header ..... 1780**
4.1 Sequence Number ..... 1780
4.2 Acknowledge Number ..... 1780
4.3 Data Integrity Check Present ..... 1781
4.4 Reliable Packet ..... 1781
4.5 Packet Type ..... 1781
4.6 Payload Length ..... 1782
4.7 Packet Header Checksum ..... 1782

**5 Data Integrity Check ..... 1783**
5.1 16-bit CCITT-CRC ..... 1783

**6 Reliable packets ..... 1784**
6.1 Header Checksum error ..... 1784
6.2 Slip Payload Length error ..... 1784
6.3 Data Integrity Check error ..... 1784
6.4 Out Of Sequence Packet error ..... 1784
6.5 Acknowledgment ..... 1785
6.6 Resending packets ..... 1785
6.7 Example reliable packet flow ..... 1785

**7 Unreliable packets ..... 1788**
7.1 Unreliable packet header ..... 1788
7.2 Unreliable packet error ..... 1788

**8 Link Establishment ..... 1789**
8.1 Uninitialized state ..... 1789
8.2 Initialized state ..... 1790
8.3 Active state ..... 1790
8.4 Sync message ..... 1790
8.5 Sync Response message ..... 1791
8.6 Config message ..... 1791
8.7 Config Response message ..... 1792
8.8 Configuration Field ..... 1792
8.8.1 Configuration messages ..... 1793
8.8.2 Sliding window size ..... 1793
8.8.3 Level of Data Integrity Check ..... 1793
8.8.4 Out of Frame Software Flow Control ..... 1794
8.8.5 Version Number ..... 1794

**9 Low power ..... 1795**
9.1 Wakeup message ..... 1795
9.2 Woken message ..... 1795
9.3 Sleep message ..... 1796

**10 Out of Frame Control ..... 1797**
10.1 Software Flow Control ..... 1797

**11 Hardware configuration ..... 1798**
11.1 Wires ..... 1798
11.1.1 Transmit & receive ..... 1798
11.1.2 Ground ..... 1798
11.2 Hardware flow ..... 1798
11.2.1 RTS & CTS ..... 1798

**12 Recommended parameters ..... 1799**
12.1 Timing parameters ..... 1799
12.1.1 Acknowledgment of packets ..... 1799
12.1.2 Resending reliable packets ..... 1799

**13 References ..... 1800**

**Part E**
**HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION**

**1 Introduction ..... 1817**
1.1 Lower Layers of the Bluetooth software stack ..... 1818
1.2 Cross-version issues ..... 1819

**2 Overview of Host Controller transport layer ..... 1821**
2.1 [This section is no longer used] ..... 1821

**3 Overview of commands and events ..... 1822**
3.1 LE Controller requirements ..... 1869
3.1.1 Legacy and extended advertising ..... 1869
3.2 Underlying Support ..... 1870
3.3 Feature Exchange ..... 1871

**4 HCI flow control ..... 1872**
4.1 Host to Controller data flow control ..... 1872
4.1.1 Packet-based data flow control ..... 1873
4.1.2 Data-block-based data flow control ..... 1874
4.2 Controller to Host data flow control ..... 1875
4.3 Disconnection behavior ..... 1876
4.4 Command flow control ..... 1876
4.5 Command error handling ..... 1878
4.5.1 Generic error handling ..... 1878
4.5.2 Error handling specific to a command ..... 1879
4.6 LMP transaction and LL procedure collisions ..... 1880

4.7 LE Host and Controller synchronization ..... 1880
4.8 Versioned events ..... 1881

**5 HCI data formats ..... 1882**
5.1 Correctness ..... 1882
5.2 Data and parameter formats ..... 1882
5.3 IDs and Handles ..... 1883
5.3.1 Controller handles ..... 1883
5.3.2 [This section is no longer used] ..... 1885
5.4 Exchange of HCI-specific information ..... 1885
5.4.1 HCI Command packet ..... 1885
5.4.2 HCI ACL Data packets ..... 1887
5.4.3 HCI Synchronous Data packets ..... 1889
5.4.4 HCI Event packet ..... 1890
5.4.5 HCI ISO Data packets ..... 1892
5.5 Ignored parameters ..... 1894

**6 HCI configuration parameters ..... 1895**
6.1 Scan Enable ..... 1895
6.2 Inquiry Scan Interval ..... 1895
6.3 Inquiry Scan Window ..... 1896
6.4 Inquiry Scan Type ..... 1896
6.5 Inquiry mode ..... 1896
6.6 Page Timeout ..... 1897
6.7 Connection Accept Timeout ..... 1897
6.8 Page Scan Interval ..... 1898
6.9 Page Scan Window ..... 1898
6.10 [This section is no longer used] ..... 1898
6.11 Page Scan Type ..... 1898
6.12 Voice Setting ..... 1899
6.13 PIN Type ..... 1900
6.14 Link key ..... 1900
6.15 Failed Contact Counter ..... 1900
6.16 Authentication Enable ..... 1901
6.17 Hold Mode Activity ..... 1901
6.18 Link Policy Settings ..... 1902
6.19 Flush Timeout ..... 1902
6.20 Num Broadcast Retransmissions ..... 1903
6.21 Link Supervision Timeout ..... 1903
6.22 Synchronous Flow Control Enable ..... 1904
6.23 Local Name ..... 1904
6.24 Extended Inquiry response ..... 1904
6.25 Erroneous Data Reporting ..... 1905

6.26 Class of Device ..... 1905
6.27 Supported commands ..... 1905
6.28 [This section is no longer used] ..... 1918
6.29 [This section is no longer used] ..... 1918
6.30 [This section is no longer used] ..... 1918
6.31 [This section is no longer used] ..... 1918
6.32 [This section is no longer used] ..... 1918
6.33 Flow Control mode ..... 1918
6.34 LE Supported Host ..... 1918
6.35 [This section is no longer used] ..... 1919
6.36 Sync Train Interval ..... 1919
6.37 Sync Train Timeout ..... 1919
6.38 Service Data ..... 1919
6.39 Secure Connections Host Support ..... 1920
6.40 Authenticated Payload Timeout ..... 1920
6.41 Extended Page Timeout ..... 1920
6.42 Extended Inquiry Length ..... 1921

**7 HCI commands and events ..... 1922**
7.1 Link Control commands ..... 1922
7.1.1 Inquiry command ..... 1922
7.1.2 Inquiry Cancel command ..... 1925
7.1.3 Periodic Inquiry Mode command ..... 1926
7.1.4 Exit Periodic Inquiry Mode command ..... 1929
7.1.5 Create Connection command ..... 1930
7.1.6 Disconnect command ..... 1933
7.1.7 Create Connection Cancel command ..... 1935
7.1.8 Accept Connection Request command ..... 1937
7.1.9 Reject Connection Request command ..... 1939
7.1.10 Link Key Request Reply command ..... 1940
7.1.11 Link Key Request Negative Reply command ..... 1942
7.1.12 PIN Code Request Reply command ..... 1944
7.1.13 PIN Code Request Negative Reply command ..... 1946
7.1.14 Change Connection Packet Type command ..... 1948
7.1.15 Authentication Requested command ..... 1950
7.1.16 Set Connection Encryption command ..... 1952
7.1.17 Change Connection Link Key command ..... 1954
7.1.18 Link Key Selection command ..... 1955
7.1.19 Remote Name Request command ..... 1957
7.1.20 Remote Name Request Cancel command ..... 1959
7.1.21 Read Remote Supported Features command ..... 1961
7.1.22 Read Remote Extended Features command ..... 1962
7.1.23 Read Remote Version Information command ..... 1964

7.1.24 Read Clock Offset command ..... 1965
7.1.25 Read LMP Handle command ..... 1966
7.1.26 Setup Synchronous Connection command ..... 1968
7.1.27 Accept Synchronous Connection Request command 1972
7.1.28 Reject Synchronous Connection Request command 1975
7.1.29 IO Capability Request Reply command ..... 1976
7.1.30 User Confirmation Request Reply command ..... 1979
7.1.31 User Confirmation Request Negative Reply command ..... 1980
7.1.32 User Passkey Request Reply command ..... 1981
7.1.33 User Passkey Request Negative Reply command .... 1982
7.1.34 Remote OOB Data Request Reply command ..... 1983
7.1.35 Remote OOB Data Request Negative Reply command ..... 1985
7.1.36 IO Capability Request Negative Reply command ..... 1986
7.1.37 [This section is no longer used] ..... 1988
7.1.38 [This section is no longer used] ..... 1988
7.1.39 [This section is no longer used] ..... 1988
7.1.40 [This section is no longer used] ..... 1988
7.1.41 [This section is no longer used] ..... 1988
7.1.42 [This section is no longer used] ..... 1988
7.1.43 [This section is no longer used] ..... 1988
7.1.44 [This section is no longer used] ..... 1988
7.1.45 Enhanced Setup Synchronous Connection command 1989
7.1.46 Enhanced Accept Synchronous Connection Request command ..... 1999
7.1.47 Truncated Page command ..... 2006
7.1.48 Truncated Page Cancel command ..... 2008
7.1.49 Set Connectionless Peripheral Broadcast command 2010
7.1.50 Set Connectionless Peripheral Broadcast Receive command ..... 2014
7.1.51 Start Synchronization Train command ..... 2018
7.1.52 Receive Synchronization Train command ..... 2019
7.1.53 Remote OOB Extended Data Request Reply command ..... 2021
7.2 Link Policy commands ..... 2023
7.2.1 Hold Mode command ..... 2023
7.2.2 Sniff Mode command ..... 2026
7.2.3 Exit Sniff Mode command ..... 2029
7.2.4 [This section is no longer used] ..... 2030
7.2.5 [This section is no longer used] ..... 2030
7.2.6 QoS Setup command ..... 2031
7.2.7 Role Discovery command ..... 2034

7.2.8 Switch Role command ..... 2036
7.2.9 Read Link Policy Settings command ..... 2038
7.2.10 Write Link Policy Settings command ..... 2040
7.2.11 Read Default Link Policy Settings command ..... 2042
7.2.12 Write Default Link Policy Settings command ..... 2043
7.2.13 Flow Specification command ..... 2044
7.2.14 Sniff Subrating command ..... 2047
7.3 Controller & Baseband commands ..... 2050
7.3.1 Set Event Mask command ..... 2050
7.3.2 Reset command ..... 2053
7.3.3 Set Event Filter command ..... 2054
7.3.4 Flush command ..... 2061
7.3.5 Read PIN Type command ..... 2063
7.3.6 Write PIN Type command ..... 2064
7.3.7 [This section is no longer used] ..... 2065
7.3.8 Read Stored Link Key command ..... 2066
7.3.9 Write Stored Link Key command ..... 2068
7.3.10 Delete Stored Link Key command ..... 2070
7.3.11 Write Local Name command ..... 2072
7.3.12 Read Local Name command ..... 2073
7.3.13 Read Connection Accept Timeout command ..... 2074
7.3.14 Write Connection Accept Timeout command ..... 2075
7.3.15 Read Page Timeout command ..... 2076
7.3.16 Write Page Timeout command ..... 2077
7.3.17 Read Scan Enable command ..... 2078
7.3.18 Write Scan Enable command ..... 2079
7.3.19 Read Page Scan Activity command ..... 2080
7.3.20 Write Page Scan Activity command ..... 2082
7.3.21 Read Inquiry Scan Activity command ..... 2083
7.3.22 Write Inquiry Scan Activity command ..... 2085
7.3.23 Read Authentication Enable command ..... 2086
7.3.24 Write Authentication Enable command ..... 2087
7.3.25 Read Class of Device command ..... 2088
7.3.26 Write Class of Device command ..... 2089
7.3.27 Read Voice Setting command ..... 2090
7.3.28 Write Voice Setting command ..... 2091
7.3.29 Read Automatic Flush Timeout command ..... 2092
7.3.30 Write Automatic Flush Timeout command ..... 2094
7.3.31 Read Num Broadcast Retransmissions command ..... 2096
7.3.32 Write Num Broadcast Retransmissions command ..... 2097
7.3.33 Read Hold Mode Activity command ..... 2098
7.3.34 Write Hold Mode Activity command ..... 2099
7.3.35 Read Transmit Power Level command ..... 2100

7.3.36 Read Synchronous Flow Control Enable command .. 2102
7.3.37 Write Synchronous Flow Control Enable command .. 2103
7.3.38 Set Controller To Host Flow Control command ..... 2104
7.3.39 Host Buffer Size command ..... 2106
7.3.40 Host Number Of Completed Packets command ..... 2109
7.3.41 Read Link Supervision Timeout command ..... 2111
7.3.42 Write Link Supervision Timeout command ..... 2113
7.3.43 Read Number Of Supported IAC command ..... 2115
7.3.44 Read Current IAC LAP command ..... 2116
7.3.45 Write Current IAC LAP command ..... 2118
7.3.46 Set AFH Host Channel Classification command ..... 2120
7.3.47 Read Inquiry Scan Type command ..... 2122
7.3.48 Write Inquiry Scan Type command ..... 2123
7.3.49 Read Inquiry Mode command ..... 2124
7.3.50 Write Inquiry Mode command ..... 2125
7.3.51 Read Page Scan Type command ..... 2126
7.3.52 Write Page Scan Type command ..... 2127
7.3.53 Read AFH Channel Assessment Mode command .... 2128
7.3.54 Write AFH Channel Assessment Mode command .... 2129
7.3.55 Read Extended Inquiry Response command ..... 2131
7.3.56 Write Extended Inquiry Response command ..... 2132
7.3.57 Refresh Encryption Key command ..... 2134
7.3.58 Read Simple Pairing Mode command ..... 2135
7.3.59 Write Simple Pairing Mode command ..... 2136
7.3.60 Read Local OOB Data command ..... 2137
7.3.61 Read Inquiry Response Transmit Power Level command ..... 2139
7.3.62 Write Inquiry Transmit Power Level command ..... 2140
7.3.63 Send Keypress Notification command ..... 2141
7.3.64 Read Default Erroneous Data Reporting command .. 2143
7.3.65 Write Default Erroneous Data Reporting command .. 2144
7.3.66 Enhanced Flush command ..... 2145
7.3.67 [This section is no longer used] ..... 2147
7.3.68 [This section is no longer used] ..... 2147
7.3.69 Set Event Mask Page 2 command ..... 2148
7.3.70 [This section is no longer used] ..... 2150
7.3.71 [This section is no longer used] ..... 2150
7.3.72 Read Flow Control Mode command ..... 2151
7.3.73 Write Flow Control Mode command ..... 2152
7.3.74 Read Enhanced Transmit Power Level command .... 2153
7.3.75 [This section is no longer used] ..... 2155
7.3.76 [This section is no longer used] ..... 2155
7.3.77 [This section is no longer used] ..... 2155

7.3.78 Read LE Host Support command ..... 2156
7.3.79 Write LE Host Support command ..... 2157
7.3.80 Set MWS Channel Parameters command ..... 2158
7.3.81 Set External Frame Configuration command ..... 2160
7.3.82 Set MWS Signaling command ..... 2163
7.3.83 Set MWS Transport Layer command ..... 2168
7.3.84 Set MWS Scan Frequency Table command ..... 2169
7.3.85 Set MWS_PATTERN Configuration command ..... 2171
7.3.86 Set Reserved LT_ADDR command ..... 2174
7.3.87 Delete Reserved LT_ADDR command ..... 2176
7.3.88 Set Connectionless Peripheral Broadcast Data command ..... 2178
7.3.89 Read Synchronization Train Parameters command .. 2181
7.3.90 Write Synchronization Train Parameters command .. 2183
7.3.91 Read Secure Connections Host Support command .. 2185
7.3.92 Write Secure Connections Host Support command .. 2186
7.3.93 Read Authenticated Payload Timeout command ..... 2188
7.3.94 Write Authenticated Payload Timeout command ..... 2190
7.3.95 Read Local OOB Extended Data command ..... 2192
7.3.96 Read Extended Page Timeout command ..... 2194
7.3.97 Write Extended Page Timeout command ..... 2195
7.3.98 Read Extended Inquiry Length command ..... 2196
7.3.99 Write Extended Inquiry Length command ..... 2197
7.3.100 Set Ecosystem Base Interval command ..... 2198
7.3.101 Configure Data Path command ..... 2200
7.3.102 Set Min Encryption Key Size command ..... 2202
7.4 Informational parameters ..... 2204
7.4.1 Read Local Version Information command ..... 2204
7.4.2 Read Local Supported Commands command ..... 2206
7.4.3 Read Local Supported Features command ..... 2208
7.4.4 Read Local Extended Features command ..... 2209
7.4.5 Read Buffer Size command ..... 2211
7.4.6 Read BD_ADDR command ..... 2214
7.4.7 Read Data Block Size command ..... 2215
7.4.8 Read Local Supported Codecs command ..... 2217
7.4.9 Read Local Simple Pairing Options command ..... 2220
7.4.10 Read Local Supported Codec Capabilities command 2222
7.4.11 Read Local Supported Controller Delay command ... 2224
7.5 Status parameters ..... 2227
7.5.1 Read Failed Contact Counter command ..... 2227
7.5.2 Reset Failed Contact Counter command ..... 2229
7.5.3 Read Link Quality command ..... 2230
7.5.4 Read RSSI command ..... 2232

7.5.5 Read AFH Channel Map command ..... 2234
7.5.6 Read Clock command ..... 2236
7.5.7 Read Encryption Key Size command ..... 2238
7.5.8 [This section is no longer used] ..... 2240
7.5.9 [This section is no longer used] ..... 2240
7.5.10 [This section is no longer used] ..... 2240
7.5.11 Get MWS Transport Layer Configuration command .. 2241
7.5.12 Set Triggered Clock Capture command ..... 2244
7.6 Testing commands ..... 2247
7.6.1 Read Loopback Mode command ..... 2247
7.6.2 Write Loopback Mode command ..... 2249
7.6.3 Enable Implementation Under Test Mode command . 2252
7.6.4 Write Simple Pairing Debug Mode command ..... 2253
7.6.5 [This section is no longer used] ..... 2255
7.6.6 [This section is no longer used] ..... 2255
7.6.7 [This section is no longer used] ..... 2255
7.6.8 Write Secure Connections Test Mode command ..... 2256
7.7 Events ..... 2260
7.7.1 Inquiry Complete event ..... 2260
7.7.2 Inquiry Result event ..... 2261
7.7.3 Connection Complete event ..... 2263
7.7.4 Connection Request event ..... 2265
7.7.5 Disconnection Complete event ..... 2267
7.7.6 Authentication Complete event ..... 2269
7.7.7 Remote Name Request Complete event ..... 2270
7.7.8 Encryption Change event ..... 2271
7.7.9 Change Connection Link Key Complete event ..... 2273
7.7.10 Link Key Type Changed event ..... 2274
7.7.11 Read Remote Supported Features Complete event . 2276
7.7.12 Read Remote Version Information Complete event .. 2277
7.7.13 QoS Setup Complete event ..... 2279
7.7.14 Command Complete event ..... 2281
7.7.15 Command Status event ..... 2283
7.7.16 Hardware Error event ..... 2285
7.7.17 Flush Occurred event ..... 2286
7.7.18 Role Change event ..... 2287
7.7.19 Number Of Completed Packets event ..... 2288
7.7.20 Mode Change event ..... 2290
7.7.21 Return Link Keys event ..... 2292
7.7.22 PIN Code Request event ..... 2293
7.7.23 Link Key Request event ..... 2294
7.7.24 Link Key Notification event ..... 2295
7.7.25 Loopback Command event ..... 2297

7.7.26 Data Buffer Overflow event ..... 2298
7.7.27 Max Slots Change event ..... 2299
7.7.28 Read Clock Offset Complete event ..... 2300
7.7.29 Connection Packet Type Changed event ..... 2301
7.7.30 QoS Violation event ..... 2303
7.7.31 Page Scan Repetition Mode Change event ..... 2304
7.7.32 Flow Specification Complete event ..... 2305
7.7.33 Inquiry Result with RSSI event ..... 2308
7.7.34 Read Remote Extended Features Complete event ... 2310
7.7.35 Synchronous Connection Complete event ..... 2312
7.7.36 Synchronous Connection Changed event ..... 2315
7.7.37 Sniff Subrating event ..... 2317
7.7.38 Extended Inquiry Result event ..... 2319
7.7.39 Encryption Key Refresh Complete event ..... 2322
7.7.40 IO Capability Request event ..... 2323
7.7.41 IO Capability Response event ..... 2324
7.7.42 User Confirmation Request event ..... 2326
7.7.43 User Passkey Request event ..... 2327
7.7.44 Remote OOB Data Request event ..... 2328
7.7.45 Simple Pairing Complete event ..... 2329
7.7.46 Link Supervision Timeout Changed event ..... 2330
7.7.47 Enhanced Flush Complete event ..... 2331
7.7.48 User Passkey Notification event ..... 2332
7.7.49 Keypress Notification event ..... 2333
7.7.50 Remote Host Supported Features Notification event 2334
7.7.51 [This section is no longer used] ..... 2335
7.7.52 [This section is no longer used] ..... 2335
7.7.53 [This section is no longer used] ..... 2335
7.7.54 [This section is no longer used] ..... 2335
7.7.55 [This section is no longer used] ..... 2335
7.7.56 [This section is no longer used] ..... 2335
7.7.57 [This section is no longer used] ..... 2335
7.7.58 [This section is no longer used] ..... 2335
7.7.59 Number Of Completed Data Blocks event ..... 2336
7.7.60 [This section is no longer used] ..... 2338
7.7.61 [This section is no longer used] ..... 2338
7.7.62 [This section is no longer used] ..... 2338
7.7.63 [This section is no longer used] ..... 2338
7.7.64 [This section is no longer used] ..... 2338
7.7.65 LE Meta event ..... 2339
7.7.66 Triggered Clock Capture event ..... 2489
7.7.67 Synchronization Train Complete event ..... 2491
7.7.68 Synchronization Train Received event ..... 2492

7.7.69 Connectionless Peripheral Broadcast Receive event 2494
7.7.70 Connectionless Peripheral Broadcast Timeout event 2496
7.7.71 Truncated Page Complete event ..... 2497
7.7.72 Peripheral Page Response Timeout event ..... 2498
7.7.73 Connectionless Peripheral Broadcast Channel Map Change event ..... 2499
7.7.74 Inquiry Response Notification event ..... 2500
7.7.75 Authenticated Payload Timeout Expired event ..... 2501
7.7.76 SAM Status Change event ..... 2502
7.8 LE Controller commands ..... 2504
7.8.1 LE Set Event Mask command ..... 2504
7.8.2 LE Read Buffer Size command ..... 2508
7.8.3 LE Read Local Supported Features Page 0 command ..... 2511
7.8.4 LE Set Random Address command ..... 2512
7.8.5 LE Set Advertising Parameters command ..... 2514
7.8.6 LE Read Advertising Physical Channel Tx Power command ..... 2518
7.8.7 LE Set Advertising Data command ..... 2519
7.8.8 LE Set Scan Response Data command ..... 2521
7.8.9 LE Set Advertising Enable command ..... 2523
7.8.10 LE Set Scan Parameters command ..... 2525
7.8.11 LE Set Scan Enable command ..... 2528
7.8.12 LE Create Connection command ..... 2530
7.8.13 LE Create Connection Cancel command ..... 2536
7.8.14 LE Read Filter Accept List Size command ..... 2537
7.8.15 LE Clear Filter Accept List command ..... 2538
7.8.16 LE Add Device To Filter Accept List command ..... 2539
7.8.17 LE Remove Device From Filter Accept List command 2541
7.8.18 LE Connection Update command ..... 2543
7.8.19 LE Set Host Channel Classification command ..... 2546
7.8.20 LE Read Channel Map command ..... 2547
7.8.21 LE Read Remote Features Page 0 command ..... 2549
7.8.22 LE Encrypt command ..... 2550
7.8.23 LE Rand command ..... 2552
7.8.24 LE Enable Encryption command ..... 2553
7.8.25 LE Long Term Key Request Reply command ..... 2555
7.8.26 LE Long Term Key Request Negative Reply command ..... 2557
7.8.27 LE Read Supported States command ..... 2558
7.8.28 LE Receiver Test command ..... 2562
7.8.29 LE Transmitter Test command ..... 2566
7.8.30 LE Test End command ..... 2571

7.8.31 LE Remote Connection Parameter Request Reply command ..... 2572
7.8.32 LE Remote Connection Parameter Request Negative Reply command ..... 2576
7.8.33 LE Set Data Length command ..... 2578
7.8.34 LE Read Suggested Default Data Length command . 2580
7.8.35 LE Write Suggested Default Data Length command . 2582
7.8.36 LE Read Local P-256 Public Key command ..... 2583
7.8.37 LE Generate DHKey command ..... 2584
7.8.38 LE Add Device To Resolving List command ..... 2586
7.8.39 LE Remove Device From Resolving List command .. 2589
7.8.40 LE Clear Resolving List command ..... 2591
7.8.41 LE Read Resolving List Size command ..... 2592
7.8.42 LE Read Peer Resolvable Address command ..... 2593
7.8.43 LE Read Local Resolvable Address command ..... 2595
7.8.44 LE Set Address Resolution Enable command ..... 2597
7.8.45 LE Set Resolvable Private Address Timeout command ..... 2599
7.8.46 LE Read Maximum Data Length command ..... 2601
7.8.47 LE Read PHY command ..... 2603
7.8.48 LE Set Default PHY command ..... 2605
7.8.49 LE Set PHY command ..... 2607
7.8.50 [This section is no longer used] ..... 2610
7.8.51 [This section is no longer used] ..... 2610
7.8.52 LE Set Advertising Set Random Address command . 2611
7.8.53 LE Set Extended Advertising Parameters command 2613
7.8.54 LE Set Extended Advertising Data command ..... 2622
7.8.55 LE Set Extended Scan Response Data command .... 2626
7.8.56 LE Set Extended Advertising Enable command ..... 2629
7.8.57 LE Read Maximum Advertising Data Length command ..... 2634
7.8.58 LE Read Number of Supported Advertising Sets command ..... 2635
7.8.59 LE Remove Advertising Set command ..... 2636
7.8.60 LE Clear Advertising Sets command ..... 2637
7.8.61 LE Set Periodic Advertising Parameters command ... 2638
7.8.62 LE Set Periodic Advertising Data command ..... 2643
7.8.63 LE Set Periodic Advertising Enable command ..... 2646
7.8.64 LE Set Extended Scan Parameters command ..... 2648
7.8.65 LE Set Extended Scan Enable command ..... 2652
7.8.66 LE Extended Create Connection command ..... 2656
7.8.67 LE Periodic Advertising Create Sync command ..... 2665

7.8.68 LE Periodic Advertising Create Sync Cancel command ..... 2670
7.8.69 LE Periodic Advertising Terminate Sync command ... 2671
7.8.70 LE Add Device To Periodic Advertiser List command 2672
7.8.71 LE Remove Device From Periodic Advertiser List command ..... 2674
7.8.72 LE Clear Periodic Advertiser List command ..... 2676
7.8.73 LE Read Periodic Advertiser List Size command ..... 2677
7.8.74 LE Read Transmit Power command ..... 2678
7.8.75 LE Read RF Path Compensation command ..... 2679
7.8.76 LE Write RF Path Compensation command ..... 2680
7.8.77 LE Set Privacy Mode command ..... 2682
7.8.78 [This section is no longer used] ..... 2684
7.8.79 [This section is no longer used] ..... 2684
7.8.80 LE Set Connectionless CTE Transmit Parameters command ..... 2685
7.8.81 LE Set Connectionless CTE Transmit Enable command ..... 2688
7.8.82 LE Set Connectionless IQ Sampling Enable command ..... 2690
7.8.83 LE Set Connection CTE Receive Parameters command ..... 2693
7.8.84 LE Set Connection CTE Transmit Parameters command ..... 2696
7.8.85 LE Connection CTE Request Enable command ..... 2699
7.8.86 LE Connection CTE Response Enable command ..... 2702
7.8.87 LE Read Antenna Information command ..... 2704
7.8.88 LE Set Periodic Advertising Receive Enable command ..... 2706
7.8.89 LE Periodic Advertising Sync Transfer command ..... 2708
7.8.90 LE Periodic Advertising Set Info Transfer command . 2710
7.8.91 LE Set Periodic Advertising Sync Transfer Parameters command ..... 2712
7.8.92 LE Set Default Periodic Advertising Sync Transfer Parameters command ..... 2716
7.8.93 [This section is no longer used] ..... 2719
7.8.94 LE Modify Sleep Clock Accuracy command ..... 2720
7.8.95 [This section is no longer used] ..... 2722
7.8.96 LE Read ISO TX Sync command ..... 2723
7.8.97 LE Set CIG Parameters command ..... 2725
7.8.98 LE Set CIG Parameters Test command ..... 2733
7.8.99 LE Create CIS command ..... 2742
7.8.100 LE Remove CIG command ..... 2745

7.8.101 LE Accept CIS Request command ..... 2747
7.8.102 LE Reject CIS Request command ..... 2749
7.8.103 LE Create BIG command ..... 2751
7.8.104 LE Create BIG Test command ..... 2756
7.8.105 LE Terminate BIG command ..... 2762
7.8.106 LE BIG Create Sync command ..... 2764
7.8.107 LE BIG Terminate Sync command ..... 2768
7.8.108 LE Request Peer SCA command ..... 2770
7.8.109 LE Setup ISO Data Path command ..... 2772
7.8.110 LE Remove ISO Data Path command ..... 2776
7.8.111 LE ISO Transmit Test command ..... 2778
7.8.112 LE ISO Receive Test command ..... 2780
7.8.113 LE ISO Read Test Counters command ..... 2782
7.8.114 LE ISO Test End command ..... 2784
7.8.115 LE Set Host Feature command ..... 2786
7.8.116 LE Read ISO Link Quality command ..... 2788
7.8.117 LE Enhanced Read Transmit Power Level command 2791
7.8.118 LE Read Remote Transmit Power Level command ... 2794
7.8.119 LE Set Path Loss Reporting Parameters command .. 2796
7.8.120 LE Set Path Loss Reporting Enable command ..... 2799
7.8.121 LE Set Transmit Power Reporting Enable command 2801
7.8.122 LE Set Data Related Address Changes command ... 2803
7.8.123 LE Set Default Subrate command ..... 2805
7.8.124 LE Subrate Request command ..... 2808
7.8.125 LE Set Periodic Advertising Subevent Data command 2812
7.8.126 LE Set Periodic Advertising Response Data command ..... 2816
7.8.127 LE Set Periodic Sync Subevent command ..... 2819
7.8.128 LE Read All Local Supported Features command .... 2821
7.8.129 LE Read All Remote Features command ..... 2822
7.8.130 LE CS Read Local Supported Capabilities command 2824
7.8.131 LE CS Read Remote Supported Capabilities command ..... 2835
7.8.132 LE CS Write Cached Remote Supported Capabilities command ..... 2837
7.8.133 LE CS Security Enable command ..... 2849
7.8.134 LE CS Set Default Settings command ..... 2850
7.8.135 LE CS Read Remote FAE Table command ..... 2853
7.8.136 LE CS Write Cached Remote FAE Table command .. 2855
7.8.137 LE CS Create Config command ..... 2857
7.8.138 LE CS Remove Config command ..... 2864
7.8.139 LE CS Set Channel Classification command ..... 2866
7.8.140 LE CS Set Procedure Parameters command ..... 2868

7.8.141 LE CS Procedure Enable command ..... 2874
7.8.142 LE CS Test command ..... 2876
7.8.143 LE CS Test End command ..... 2892
7.8.144 LE Set Decision Data command ..... 2893
7.8.145 LE Set Decision Instructions command ..... 2895
7.8.146 LE Add Device To Monitored Advertisers List command ..... 2904
7.8.147 LE Remove Device From Monitored Advertisers List command ..... 2907
7.8.148 LE Clear Monitored Advertisers List command ..... 2909
7.8.149 LE Enable Monitoring Advertisers command ..... 2910
7.8.150 LE Read Monitored Advertisers List Size command . 2912
7.8.151 LE Frame Space Update command ..... 2913
7.8.152 LE Enable UTP OTA mode ..... 2916
7.8.153 LE UTP Send command ..... 2917
7.8.154 LE Connection Rate Request command ..... 2918
7.8.155 LE Set Default Rate Parameters command ..... 2923
7.8.156 LE Read Minimum Supported Connection Interval command ..... 2927
7.8.157 LE CS Set Security Requirements ..... 2930
7.8.158 CS Set Default Security Requirements ..... 2932

**Appendix A [This Appendix is no longer used] ..... 2934**

**Appendix B Removed commands and events ..... 2935**

## [This Volume Is No Longer Used]
## *Specification Volume 5*

# Low Energy Controller
## *Specification Volume 6*

**Part A**
**RADIO PHYSICAL LAYER SPECIFICATION**

**1 Scope ..... 2942**

**2 Frequency bands and channel arrangement ..... 2944**

**3 Transmitter characteristics ..... 2945**
3.1 Modulation characteristics ..... 2946
3.1.1 Stable modulation index ..... 2947
3.1.2 Modulation characteristics of Channel Sounding steps ..... 2947
3.1.3 SNR control for Channel Sounding steps ..... 2947
3.2 Spurious emissions ..... 2948
3.2.1 Modulation spectrum ..... 2948
3.2.2 In-band spurious emission ..... 2948
3.2.3 Out-of-band spurious emission ..... 2949
3.3 Radio frequency tolerance ..... 2949
3.4 Stable phase ..... 2949
3.5 Frequency measurement and generation in Channel Sounding 2950
3.5.1 Fractional frequency offset ..... 2951
3.5.2 Expected transmitted frequencies ..... 2952

**4 Receiver characteristics ..... 2953**
4.1 Actual sensitivity level ..... 2953
4.2 Interference performance ..... 2953
4.3 Out-of-band blocking ..... 2955
4.4 Intermodulation characteristics ..... 2956
4.5 Maximum usable level ..... 2956
4.6 Reference signal definition ..... 2956
4.7 Stable modulation index ..... 2957
4.8 Received Signal Strength Indication ..... 2957

**5 Antenna switching ..... 2958**
5.1 Antenna Switching for AoA/AoD ..... 2958
5.2 Receiver characteristics for AoA/AoD ..... 2959
5.2.1 Definitions ..... 2959
5.2.2 Requirements ..... 2960
5.2.3 Test switching pattern ..... 2960
5.3 Antenna Switching for Channel Sounding ..... 2960

**6 Phase measurements ..... 2962**
6.1 Reference receiver for phase-based ranging ..... 2962
6.2 Phase measurement accuracy ..... 2962
6.3 Frequency actuation error correction ..... 2963
6.4 Phase measurement timing ..... 2964
6.5 Inline PCT Transfer ..... 2965

**Appendix A Test Conditions ..... 2966**
A.1 Normal operating conditions ..... 2966
A.1.1 Normal temperature and air humidity ..... 2966
A.1.2 Nominal supply voltage ..... 2966

**Appendix B Example test equipment setup for Channel Sounding receiver and transmitter ..... 2967**

**Part B**
**LINK LAYER SPECIFICATION**

**1 General description ..... 2979**
1.1 Link Layer states ..... 2979
1.1.1 Permitted state and role combinations ..... 2981
1.1.2 Devices supporting only some states ..... 2982
1.2 Bit ordering ..... 2982
1.3 Device address ..... 2983
1.3.1 Public device address ..... 2983
1.3.2 Random device address ..... 2983
1.4 Physical channel ..... 2987
1.4.1 Physical channel indices ..... 2987

**2 Air interface packets ..... 2989**
2.1 Packet format for the LE Uncoded PHYs ..... 2989
2.1.1 Preamble ..... 2989
2.1.2 Access Address ..... 2990
2.1.3 PDU ..... 2992
2.1.4 CRC ..... 2992
2.1.5 Constant Tone Extension ..... 2993
2.2 Packet format for the LE Coded PHY ..... 2993
2.2.1 Preamble ..... 2994
2.2.2 Access Address ..... 2994
2.2.3 Coding Indicator ..... 2994
2.2.4 PDU ..... 2994
2.2.5 CRC ..... 2995
2.2.6 TERM1 and TERM2 ..... 2995

2.3 Advertising physical channel PDU ..... 2995
2.3.1 Advertising PDUs ..... 2997
2.3.2 Scanning PDUs ..... 3006
2.3.3 Initiating PDUs ..... 3008
2.3.4 Common Extended Advertising Payload Format ..... 3011
2.4 Data Physical Channel PDU ..... 3019
2.4.1 LL Data PDU ..... 3021
2.4.2 LL Control PDU ..... 3021
2.5 Constant Tone Extension and IQ sampling ..... 3071
2.5.1 Constant Tone Extension structure and types ..... 3071
2.5.2 CTEInfo field ..... 3072
2.5.3 Transmitting Constant Tone Extensions ..... 3073
2.5.4 IQ sampling ..... 3073
2.6 Isochronous Physical Channel PDU ..... 3074
2.6.1 Connected Isochronous PDU ..... 3075
2.6.2 Broadcast Isochronous PDU ..... 3076
2.6.3 BIG Control PDU ..... 3077

**3 Bit stream processing ..... 3080**
3.1 Error checking ..... 3080
3.1.1 CRC generation ..... 3080
3.2 Data whitening ..... 3081
3.3 Coding ..... 3082
3.3.1 Forward Error Correction encoder ..... 3083
3.3.2 Pattern mapper ..... 3083

**4 Air Interface protocol ..... 3084**
4.1 Frame Space ..... 3084
4.1.1 Inter Frame Space ..... 3084
4.1.2 Minimum AUX Frame Space ..... 3084
4.1.3 Minimum Isochronous Channel Subevent Space ..... 3085
4.1.4 Minimum Channel Sounding subevent space ..... 3086
4.1.5 Minimum Connection Event Spacing ..... 3087
4.2 Timing requirements ..... 3087
4.2.1 Active clock accuracy ..... 3087
4.2.2 Sleep clock accuracy ..... 3088
4.2.3 Range delay ..... 3089
4.2.4 Window widening ..... 3089
4.3 Link Layer device filtering ..... 3093
4.3.1 Filter Accept List ..... 3094
4.3.2 Advertising filter policy ..... 3094
4.3.3 Scanning filter policy ..... 3094
4.3.4 Initiator filter policy ..... 3096

4.3.5 Periodic sync establishment filter policy ..... 3096
4.4 Non-connected states ..... 3097
4.4.1 Standby state ..... 3097
4.4.2 Advertising state ..... 3097
4.4.3 Scanning state ..... 3127
4.4.4 Initiating state ..... 3133
4.4.5 Synchronization state ..... 3135
4.4.6 Isochronous Broadcasting state ..... 3138
4.5 Connection state ..... 3150
4.5.1 Connection events ..... 3151
4.5.2 Supervision timeout ..... 3156
4.5.3 Connection setup - connection event transmit window 3157
4.5.4 Connection setup – Central Role ..... 3158
4.5.5 Connection setup – Peripheral Role ..... 3159
4.5.6 Closing connection events ..... 3161
4.5.7 Sleep clock accuracy ..... 3161
4.5.8 General-purpose channel group index selection ..... 3162
4.5.9 Acknowledgment and flow control ..... 3171
4.5.10 Data PDU length management ..... 3173
4.5.11 Control PDU length management ..... 3177
4.5.12 Connection termination and loss ..... 3177
4.5.13 Connected Isochronous Stream (CIS) ..... 3178
4.5.14 Connected Isochronous Group (CIG) ..... 3184
4.5.15 Power level management ..... 3188
4.5.16 Path loss monitoring ..... 3189
4.5.17 ACL data Host transport ..... 3191
4.5.18 Channel Sounding ..... 3191
4.6 Feature support ..... 3196
4.6.1 LE Encryption ..... 3199
4.6.2 Connection Parameters Request procedure ..... 3199
4.6.3 Extended Reject Indication ..... 3200
4.6.4 Peripheral-initiated Features Exchange ..... 3200
4.6.5 LE Ping ..... 3200
4.6.6 LE Data Packet Length Extension ..... 3200
4.6.7 LL Privacy ..... 3200
4.6.8 Extended Scanning Filter Policies ..... 3201
4.6.9 Multiple PHYs ..... 3201
4.6.10 Stable Modulation Index - Transmitter ..... 3201
4.6.11 Stable Modulation Index - Receiver ..... 3202
4.6.12 LE Extended Advertising ..... 3202
4.6.13 LE Periodic Advertising ..... 3203
4.6.14 Channel Selection Algorithm #2 ..... 3203
4.6.15 Minimum Number of Used Channels procedure ..... 3203

4.6.16 Connection CTE Request ..... 3203
4.6.17 Connection CTE Response ..... 3204
4.6.18 Connectionless CTE Transmitter ..... 3204
4.6.19 Connectionless CTE Receiver ..... 3204
4.6.20 Antenna Switching During CTE Transmission (AoD) 3204
4.6.21 Antenna Switching During CTE Reception (AoA) ..... 3205
4.6.22 Receiving Constant Tone Extensions ..... 3205
4.6.23 Periodic Advertising Sync Transfer - Sender ..... 3205
4.6.24 Periodic Advertising Sync Transfer - Recipient ..... 3205
4.6.25 Sleep Clock Accuracy Updates ..... 3205
4.6.26 Remote Public Key Validation ..... 3206
4.6.27 Connected Isochronous Stream - Central and Connected Isochronous Stream - Peripheral ..... 3206
4.6.28 Isochronous Broadcaster ..... 3206
4.6.29 Synchronized Receiver ..... 3207
4.6.30 [This section is no longer used] ..... 3207
4.6.31 LE Power Control Request ..... 3207
4.6.32 LE Path Loss Monitoring ..... 3207
4.6.33 Host-set feature bits ..... 3207
4.6.34 Periodic Advertising ADI Support ..... 3208
4.6.35 Connection Subrating ..... 3209
4.6.36 Channel Classification ..... 3209
4.6.37 Advertising Coding Selection ..... 3209
4.6.38 Periodic Advertising with Responses - Advertiser ..... 3210
4.6.39 Periodic Advertising with Responses - Scanner ..... 3210
4.6.40 LL Extended Feature Set ..... 3210
4.6.41 Channel Sounding ..... 3210
4.6.42 Channel Sounding Tone Quality Indication ..... 3212
4.6.43 Decision-Based Advertising Filtering ..... 3212
4.6.44 ISOAL Unsegmented Framed Mode ..... 3212
4.6.45 Monitoring Advertisers ..... 3212
4.6.46 Frame Space Update ..... 3212
4.6.47 UTP OTA mode ..... 3213
4.6.48 UTP HCI mode ..... 3213
4.6.49 LL_OTA_UTP_IND maximum length ..... 3213
4.6.50 Shorter Connection Intervals ..... 3213
4.6.51 LE Flushable ACL Data ..... 3214
4.6.52 Channel Sounding Enhancement #1 ..... 3214
4.7 Resolving List ..... 3214

**5 Link Layer control ..... 3216**
5.1 Link Layer ACL control procedures ..... 3216
5.1.1 Connection Update procedure ..... 3216

5.1.2 Channel Map Update procedure ..... 3219
5.1.3 Encryption procedure ..... 3220
5.1.4 Feature Exchange procedure ..... 3224
5.1.5 Version Exchange procedure ..... 3226
5.1.6 ACL Termination procedure ..... 3226
5.1.7 Connection Parameters Request procedure ..... 3227
5.1.8 LE Ping procedure ..... 3238
5.1.9 Data Length Update procedure ..... 3238
5.1.10 PHY Update procedure ..... 3239
5.1.11 Minimum Number Of Used Channels procedure ..... 3242
5.1.12 Constant Tone Extension Request procedure ..... 3243
5.1.13 Periodic Advertising Sync Transfer procedure ..... 3244
5.1.14 Sleep Clock Accuracy Update procedure ..... 3247
5.1.15 Connected Isochronous Stream Creation procedure 3248
5.1.16 Connected Isochronous Stream Termination procedure ..... 3250
5.1.17 Power Control Request procedure ..... 3250
5.1.18 Power Change Indication procedure ..... 3252
5.1.19 Connection Subrate Update procedure ..... 3253
5.1.20 Connection Subrate Request procedure ..... 3255
5.1.21 Channel Classification Enable procedure ..... 3256
5.1.22 Channel Classification Reporting procedure ..... 3256
5.1.23 Channel Sounding Security Start procedure ..... 3257
5.1.24 Channel Sounding Capabilities Exchange procedure 3259
5.1.25 Channel Sounding Configuration procedure ..... 3259
5.1.26 Channel Sounding Start procedure ..... 3262
5.1.27 Channel Sounding Procedure Repeat Termination procedure ..... 3268
5.1.28 Channel Sounding Channel Map Update procedure . 3269
5.1.29 Channel Sounding Mode-0 FAE Table Request procedure ..... 3270
5.1.30 Frame Space Update procedure ..... 3271
5.1.31 UTP OTA mode procedure ..... 3273
5.1.32 Connection Rate Update procedure ..... 3274
5.1.33 Connection Rate Request procedure ..... 3276
5.2 Procedure response timeout ..... 3277
5.3 Procedure collisions ..... 3278
5.4 LE Authenticated Payload Timeout ..... 3279
5.5 Procedures with Instants ..... 3279
5.5.1 ACL control procedures ..... 3280
5.5.2 BIG control procedures ..... 3281
5.6 BIG control procedures ..... 3281
5.6.1 BIG Channel Map Update procedure ..... 3281

5.6.2 BIG Termination procedure ..... 3281

**6 Privacy ..... 3283**
6.1 Resolvable Private address generation interval ..... 3283
6.2 Privacy in the Advertising state ..... 3283
6.2.1 Connectable and scannable undirected event type ... 3283
6.2.2 Connectable directed event type ..... 3284
6.2.3 Non-connectable and non-scannable undirected and scannable undirected event types ..... 3285
6.2.4 Connectable undirected event type ..... 3286
6.2.5 Non-connectable and non-scannable directed and scannable directed event types ..... 3286
6.3 Privacy in the Scanning state ..... 3287
6.4 Privacy in the Initiating state ..... 3288
6.5 Privacy of the device ..... 3289
6.6 Privacy in the Synchronization State ..... 3289
6.6.1 Periodic advertising trains ..... 3289

**7 ISO Test Mode ..... 3291**
7.1 ISO Transmit test mode ..... 3291
7.2 ISO Receive test mode ..... 3292

**8 References ..... 3294**

**Part C**
**SAMPLE DATA**

**1 Encryption sample data ..... 3297**
1.1 Encrypt Command ..... 3300
1.2 Derivation of the MIC and encrypted data ..... 3300

**2 LE Coded PHY sample data ..... 3304**
2.1 Reference information packet ..... 3304
2.2 Forward Error Correction encoder ..... 3304
2.3 Transmitted symbols (S=2) ..... 3305
2.4 Transmitted symbols (S=8) ..... 3306

**3 LE Channel Selection algorithm #2 sample data ..... 3308**
3.1 Sample data 1 (37 used channels) ..... 3308
3.2 Sample data 2 (9 used channels) ..... 3308
3.3 Sample data 3 (3 used channels) ..... 3309

**4 Complete packets ..... 3311**
4.1 Whitening sequences ..... 3311

4.2 Advertising Physical channel PDUs ..... 3312
4.2.1 Legacy advertising PDUs ..... 3312
4.2.2 Extended advertising PDUs ..... 3313
4.3 Data channel PDUs ..... 3314
4.3.1 LL data PDUs ..... 3314
4.3.2 LL control PDUs ..... 3315

**5 Access Address generation for BISes ..... 3316**

**6 Group Session Key derivation for BIG ..... 3318**

**7 Deterministic Random Bit Generator sample data ..... 3319**

**8 Channel Sounding procedure sample data ..... 3325**
8.1 Channel Selection Algorithm #3b ..... 3325
8.1.1 Set 1 ..... 3325
8.1.2 Set 2 ..... 3347
8.1.3 Set 3 ..... 3394
8.1.4 Set 4 ..... 3437
8.2 Channel Selection Algorithm #3c ..... 3508
8.2.1 Set 1 ..... 3508
8.2.2 Set 2 ..... 3530
8.2.3 Set 3 ..... 3552

**Part D**
**MESSAGE SEQUENCE CHARTS**

**1 Introduction ..... 3590**
1.1 Notation ..... 3590
1.2 Control flow ..... 3590
1.3 Example MSC ..... 3591
1.4 Forward compatibility ..... 3591

**2 Standby state ..... 3593**
2.1 Initial setup ..... 3593
2.2 Random Device address ..... 3595
2.3 Filter Accept List ..... 3595
2.4 Adding IRK to resolving list ..... 3596
2.5 Default data length ..... 3596
2.6 Periodic Advertiser List ..... 3597

**3 Advertising state ..... 3598**
3.1 Undirected advertising ..... 3598
3.2 Directed advertising ..... 3599

3.3 Advertising using ADV_EXT_IND ..... 3601
3.4 Scan request notifications ..... 3602
3.5 Advertising duration ended ..... 3603
3.6 Periodic advertising ..... 3604
3.7 Connectionless Constant Tone Extension transmission ..... 3605
3.8 Isochronous Broadcasting State ..... 3606
3.8.1 Create a Broadcast Isochronous Group ..... 3606
3.8.2 Terminate a Broadcast Isochronous Group ..... 3607
3.9 Periodic advertising with responses (PAwR) ..... 3607
3.10 Transmitting PAwR subevents ..... 3608
3.11 Using a response slot in PAwR ..... 3608
3.12 Connecting from PAwR ..... 3610
3.13 Failed connection attempts from PAwR ..... 3610

**4 Scanning state ..... 3612**
4.1 Passive scanning ..... 3612
4.2 Active scanning ..... 3613
4.3 Passive scanning for directed advertisements with Privacy ..... 3614
4.4 Active scanning with Privacy ..... 3615
4.5 Active scanning with Privacy and Controller based resolvable private address generation ..... 3616
4.6 Active scanning on the secondary advertising Physical channel 3617
4.7 Scan timeout ..... 3618
4.8 Scanning for periodic advertisements ..... 3619
4.9 Cancel scanning for periodic advertisements ..... 3620
4.10 Periodic advertising synchronization timeout ..... 3621
4.11 Terminate reception of periodic advertising ..... 3622
4.12 Connectionless Constant Tone Extension reception ..... 3623
4.13 Synchronization with separate enable of reports ..... 3624

**5 Initiating state ..... 3625**
5.1 Initiating a connection ..... 3625
5.2 Canceling an initiation ..... 3626
5.3 Initiating a connection using undirected advertising with Privacy 3627
5.4 Initiating a connection using directed advertising with Privacy . 3628
5.5 Initiating a connection that fails to establish ..... 3629
5.6 Initiating a connection on the secondary advertising physical channel ..... 3630
5.7 Initiating a Channel Selection algorithm #2 connection ..... 3631
5.8 Initiating a connection using an advertising set ..... 3632

**6 Connection state ..... 3633**
6.1 Sending data ..... 3633

6.2 Connection update ..... 3634
6.3 Channel map update ..... 3634
6.4 Features exchange ..... 3635
6.5 Version exchange ..... 3639
6.6 Start encryption ..... 3641
6.7 Start encryption without Long Term Key ..... 3642
6.8 Start encryption with event masked ..... 3643
6.9 Start encryption without Peripheral supporting encryption ..... 3644
6.10 Restart encryption ..... 3645
6.11 Disconnect ..... 3646
6.12 Connection parameters request ..... 3647
6.13 LE Ping ..... 3652
6.14 Data length update ..... 3654
6.15 PHY update ..... 3655
6.16 Minimum number of used channels request ..... 3660
6.17 LL procedure collision ..... 3661
6.18 Constant Tone Extension Request ..... 3662
6.19 Connected Isochronous Group Setup ..... 3664
6.20 Host Rejects Connected Isochronous Stream ..... 3666
6.21 Link Layer Rejects Connected Isochronous Stream ..... 3668
6.22 Link Layer Rejects Connected Isochronous Stream ..... 3670
6.23 Host A Terminates Connected Isochronous Stream ..... 3670
6.24 ACL disconnected ..... 3672
6.25 Host A Removes Connected Isochronous Group ..... 3673
6.26 Request Sleep Clock Accuracy ..... 3675
6.27 Power Control ..... 3676
6.28 Data path setup for a music stream over a CIS ..... 3683
6.29 Data path setup for bi-directional voice over a CIS ..... 3685
6.30 [This section is no longer used] ..... 3686
6.31 Modifying the subrate of a connection ..... 3686
6.32 Channel Classification Enable ..... 3688
6.33 Channel Classification Reporting ..... 3689
6.34 Channel Sounding setup phase ..... 3690
6.35 Channel Sounding started by Central in initiator role ..... 3692
6.36 Channel Sounding started by Peripheral in reflector role ..... 3694
6.37 Channel Sounding started by Central, rejected by Peripheral ..... 3696
6.38 Channel Sounding configuration removal during an active CS procedure ..... 3696
6.39 Frame Space Update ..... 3697
6.40 Modifying the Connection Rate ..... 3699

**7 Periodic advertising sync transfer ..... 3702**
7.1 Transfer by scanner, reports initially disabled ..... 3702

7.2 Transfer by scanner, reports initially enabled ..... 3703
7.3 Transfer by the advertiser ..... 3704

**8 Synchronization state ..... 3705**
8.1 Synchronizing with a Broadcast Isochronous Group ..... 3705
8.2 Terminate Synchronization with a BIG ..... 3706
8.3 New Channel Map for Broadcast Isochronous Group ..... 3706
8.4 Lost Synchronization with a Broadcast Isochronous Group ..... 3707
8.5 Data path setup for a BIS ..... 3707

**Part E**
**LOW ENERGY LINK LAYER SECURITY**

**1 Encryption and authentication overview ..... 3711**
1.1 Cryptographic Toolbox ..... 3712
1.1.1 Group Session Key Derivation Function h8 ..... 3712
1.1.2 Derivation of Group Session Key ..... 3713

**2 CCM ..... 3714**
2.1 CCM nonce ..... 3714
2.2 Counter mode blocks ..... 3716
2.3 Encryption blocks ..... 3717
2.4 Session Keys ..... 3717

**3 Deterministic Random Bit Generator ..... 3719**
3.1 Cryptographic toolbox ..... 3720
3.1.1 Octet and bit ordering ..... 3721
3.1.2 DRBG chain function f7 ..... 3722
3.1.3 DRBG derivation function f8 ..... 3723
3.1.4 DRBG update function f9 ..... 3725
3.1.5 DRBG instantiation function h9 ..... 3727
3.1.6 Random bit generation function CS_DRBG ..... 3728
3.1.7 DRBG backtracking resistance ..... 3728
3.2 DRBG based on a block cipher ..... 3729
3.2.1 DRBG nonce V ..... 3729
3.2.2 DRBG nonce increment function ..... 3730

**4 References ..... 3733**

**Part F**
**RFPHY TEST MODES**

**1 Introduction ..... 3737**

**2 Low Energy test scenarios ..... 3739**
2.1 Test sequences ..... 3739
2.2 Message sequence charts ..... 3740
2.3 Channel Sounding test commands ..... 3741
2.4 Channel Sounding message sequence charts ..... 3742

**3 UART Test Interface ..... 3748**
3.1 UART Interface characteristics ..... 3748
3.2 UART functional description ..... 3748
3.3 Commands and events ..... 3749
3.3.1 Command and event behavior ..... 3749
3.3.2 Commands ..... 3749
3.4 Events ..... 3753
3.4.1 LE_Test_Status event ..... 3753
3.4.2 LE_Packet_Report event ..... 3755
3.5 Timing - command and event ..... 3756

**4 LE Test packet definition ..... 3757**
4.1 LE Test packets format ..... 3757
4.1.1 Whitening ..... 3758
4.1.2 Preamble and synchronization word ..... 3758
4.1.3 CRC ..... 3758
4.1.4 LE Test packet PDU ..... 3758
4.1.5 LE Test packet payload description ..... 3760
4.1.6 LE Test packet interval ..... 3761
4.1.7 Constant Tone Extension ..... 3761
4.1.8 LE Channel Sounding Test packet trailer ..... 3762
4.1.9 LE Channel Sounding Test packet payload ..... 3762
4.1.10 LE Channel Sounding Test exchanges ..... 3762

**5 Unified Test Protocol ..... 3763**
5.1 Introduction ..... 3763
5.2 Message format ..... 3765
5.3 Configuration messages ..... 3766
5.3.1 UTP_Set_RF_Channel ..... 3767
5.3.2 UTP_Set_Packet_Payload ..... 3767
5.3.3 UTP_Set_Payload_Length ..... 3768
5.3.4 UTP_Set_PHY ..... 3768
5.3.5 UTP_Set_Modulation_Index ..... 3769
5.3.6 UTP_Set_CTE_Length ..... 3769
5.3.7 UTP_Set_CTE_Type ..... 3770
5.3.8 UTP_Set_CTE_Slot_Durations ..... 3770
5.3.9 UTP_Set_CTE_Antenna_IDs ..... 3770

5.3.10 UTP_Set_Packet_Count ..... 3771
5.3.11 UTP_Set_Tx_Power_Level ..... 3771
5.3.12 UTP_Set_OTA_Exclusion_Period ..... 3771
5.3.13 UTP_Set_Vendor_Specific_Data ..... 3772
5.4 Control messages sent by the Upper Tester ..... 3772
5.4.1 UTP_Query_Supported_Features ..... 3773
5.4.2 UTP_Reset ..... 3773
5.4.3 UTP_Start_Test ..... 3773
5.4.4 UTP_Stop_Test ..... 3774
5.4.5 UTP_Vendor_Specific_Control ..... 3774
5.5 Control messages sent by the IUT ..... 3775
5.5.1 UTP_Accept ..... 3775
5.5.2 UTP_Reject ..... 3775
5.5.3 UTP_Reset_Accept ..... 3776
5.5.4 UTP_Test_Ended ..... 3776
5.6 Report messages ..... 3776
5.6.1 UTP_Report_Supported_Features ..... 3777
5.6.2 UTP_Report_IQ_Samples ..... 3777
5.6.3 UTP_Report_Receiver_Quality_Counters ..... 3779
5.6.4 UTP_Report_Vendor_Specific_Data ..... 3780

**6 UTP transport ..... 3782**
6.1 Introduction ..... 3782
6.2 2-wire UART ..... 3782
6.3 HCI ..... 3782
6.4 OTA ..... 3782
6.4.1 Introduction ..... 3782
6.4.2 Connection interval requirements ..... 3783
6.4.3 Transmitter test ..... 3784
6.4.4 Receiver test ..... 3785

**Part G**
**ISOCHRONOUS ADAPTATION LAYER**

**1 Introduction ..... 3789**
1.1 Terminology ..... 3789

**2 ISOAL features ..... 3790**
2.1 Unframed PDU ..... 3792
2.2 Framed PDU ..... 3793

**3 Time_Stamp and Time_Offset ..... 3797**
3.1 Time_Offset in framed PDUs ..... 3797

3.2 SDU synchronization reference ..... 3799
3.2.1 SDU synchronization reference using framed PDUs . 3799
3.2.2 SDU synchronization reference using unframed PDUs ..... 3801
3.3 Time Stamp for SDU ..... 3804

**4 SDU Recombination and Reassembly ..... 3806**

**Part H**
**CHANNEL SOUNDING**

**1 Channel Sounding physical channels ..... 3811**

**2 Packet formats for Channel Sounding ..... 3812**
2.1 Preamble ..... 3812
2.2 Channel Sounding Access Address ..... 3813
2.2.1 Channel Sounding Access Address selection rules .. 3814
2.2.2 Channel Sounding Access Address checking ..... 3814
2.3 Trailer ..... 3815
2.4 Sounding sequence ..... 3815
2.5 Random sequence ..... 3816
2.6 Channel Sounding extended packet formats ..... 3817

**3 Channel Sounding bit stream processing ..... 3819**
3.1 Measuring RTT ..... 3819
3.1.1 Reference receiver for round-trip time measurements 3820
3.1.2 ToD and ToA reporting accuracy ..... 3821
3.2 Timing estimate based on an Access Address ..... 3822
3.2.1 Timestamps using a native clock ..... 3822
3.2.2 Timing estimate based on a pseudo-noise bit sequence ..... 3823
3.3 Fractional timing estimate based on a sounding sequence ..... 3823
3.3.1 Phase-based PCT estimate based on a sounding sequence ..... 3824
3.4 Fractional timing estimate based on a random sequence ..... 3829
3.5 Attack detection requirements ..... 3829
3.5.1 Normalized attack detector metric ..... 3829
3.5.2 Reference signal modulated with h=0.5 GFSK ..... 3831
3.5.3 Early commit attacks with phase based detection ..... 3832
3.5.4 Amplitude-based attacks with DFT detection ..... 3840

**4 Channel Sounding interface protocol ..... 3853**
4.1 Channel selection Algorithm #3 ..... 3853
4.1.1 Conventions ..... 3853

4.1.2 Channel index shuffling function cr1 ..... 3853
4.1.3 Channel selection Algorithm #3a for mode-0 steps ... 3854
4.1.4 Channel index selection for non-mode-0 steps ..... 3855
4.2 Channel Sounding channel indices ..... 3866
4.3 Channel Sounding steps ..... 3867
4.3.1 Channel Sounding step mode-0 ..... 3868
4.3.2 Channel Sounding step mode-1 ..... 3870
4.3.3 Channel Sounding step mode-2 ..... 3871
4.3.4 Channel Sounding step mode-3 ..... 3873
4.4 Channel Sounding subevent and mode sequencing ..... 3876
4.4.1 Tone extension slots ..... 3876
4.4.2 CS subevent structure ..... 3878
4.4.3 Sub_Mode insertion ..... 3879
4.4.4 Main_mode repetition ..... 3879
4.4.5 Channel Sounding procedure and procedure repeat desynchronization ..... 3880
4.5 Timing of steps ..... 3881
4.6 Phase measurements during T_PM ..... 3882
4.7 Phase measurements with antenna switching ..... 3884
4.7.1 Antenna switching in the 1:1 configuration ..... 3884
4.7.2 Antenna switching in the N_AP:1 configuration ..... 3884
4.7.3 Antenna switching in the 1:N_AP configuration ..... 3886
4.7.4 Antenna switching in the 2:2 (N_AP = 4) configuration 3887
4.7.5 Antenna path permutations ..... 3889
4.8 Channel Sounding random bit generation ..... 3892
4.8.1 Channel Sounding random number generation function hr1 ..... 3893

# Wireless Coexistence Signaling and Interfaces
## *Specification Volume 7*

**Part A**
**MWS COEXISTENCE LOGICAL SIGNALING SPECIFICATION**

**1 Introduction ..... 3898**

**2 Logical interface ..... 3899**
2.1 Coexistence signals ..... 3899
2.1.1 FRAME_SYNC ..... 3899
2.1.2 MWS_RX ..... 3900
2.1.3 BLUETOOTH_RX_PRI ..... 3900
2.1.4 BLUETOOTH_TX_ON ..... 3901
2.1.5 MWS_PATTERN ..... 3901
2.1.6 MWS_TX ..... 3901
2.1.7 802_TX_ON ..... 3901
2.1.8 802_RX_PRI ..... 3902
2.1.9 MWS_INACTIVITY_DURATION ..... 3902
2.1.10 MWS_SCAN_FREQUENCY ..... 3902
2.2 Tolerances for offsets and jitter ..... 3902

**Part B**
**WIRELESS COEXISTENCE INTERFACE 1 (WCI-1) TRANSPORT SPECIFICATION**

**1 Introduction ..... 3908**

**2 Physical layer ..... 3909**
2.1 Physical signal specifications ..... 3910

**3 Transport layer ..... 3912**
3.1 Message types ..... 3912
3.1.1 Real-time Signal message (Type 0) ..... 3913
3.1.2 Transport Control message (Type 1) ..... 3913
3.1.3 Transparent Data message (Type 2) ..... 3914
3.1.4 MWS Inactivity Duration message (Type 3) ..... 3914
3.1.5 MWS Scan Frequency message (Type 4) ..... 3915

**Part C**
**WIRELESS COEXISTENCE INTERFACE 2 (WCI-2) TRANSPORT SPECIFICATION**

**1 Introduction ..... 3918**

**2 Physical layer ..... 3919**

**3 Transport layer ..... 3920**

3.1 Message types ..... 3920

3.1.1 Real-time Signal message (Type 0) ..... 3921

3.1.2 Transport Control message (Type 1) ..... 3921

3.1.3 Transparent Data message (Type 2) ..... 3922

3.1.4 MWS Inactivity Duration message (Type 3) ..... 3922

3.1.5 MWS Scan Frequency message (Type 4) ..... 3923

# Consolidated Table Of Contents, Acknowledgments, & Core Configurations Part B

# [THIS PART IS NO LONGER USED]

*Core configurations are located in Part D. Compliance requirements are found in the Qualification Program Reference Document (https://www.bluetooth.com/download/qprd-document/).*

Consolidated Table Of Contents, Acknowledgments, & Core Configurations
Part C

# VERSION HISTORY AND ACKNOWLEDGMENTS

# CONTENTS

**1 Version History ..... 91**
1.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations ..... 91
1.2 [Vol 1] Architecture, Change History, and Conventions ..... 92
1.3 [Vols 2, 3, 5, 6 & 7] Controllers and Host ..... 94
1.4 [Vol 4] Host Controller Interface ..... 98

**2 Acknowledgments (up to v5.1) ..... 100**
2.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations ..... 100
2.1.1 Part B: Bluetooth Compliance Requirements ..... 100
2.1.2 Part C: Version History and Acknowledgments ..... 100
2.2 [Vol 1] Architecture, Change History, and Conventions ..... 101
2.2.1 Part A: Architectural Overview ..... 101
2.2.2 Part B: Acronyms & Abbreviations ..... 107
2.2.3 Part C: Core Specification Change History ..... 108
2.2.4 Part D: Mixing of Specification Versions ..... 109
2.2.5 Part E: General Terminology and Interpretation ..... 110
2.2.6 Part F: Controller Error Codes ..... 111
2.3 [Vol 2] BR/EDR Controller ..... 115
2.3.1 Part A: Radio Physical Layer Specification ..... 115
2.3.2 Part B: Baseband Specification ..... 117
2.3.3 Part C: Link Manager Protocol ..... 123
2.3.4 Part D: [This part is no longer used] ..... 128
2.3.5 Part E: [This part is no longer used] ..... 128
2.3.6 Part F: Message Sequence Charts ..... 129
2.3.7 Part G: Sample Data ..... 132
2.3.8 Part H: Security Specification ..... 135
2.4 [Vol 3] Host ..... 137
2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification ..... 137
2.4.2 Part B: Service Discovery Protocol (SDP) ..... 143
2.4.3 Part C: Generic Access Profile ..... 144
2.4.4 Part D: Test Support ..... 150
2.4.5 Part E: AMP Manager Protocol ..... 152
2.4.6 Part F: Attribute Protocol Specification ..... 152
2.4.7 Part G: Generic Attribute Profile Specification ..... 154
2.4.8 Part H: Security Manager Specification ..... 156
2.5 [Vol 4] Host Controller Interface ..... 158

2.5.1 Parts A to D: Transport Layers ..... 158
2.5.2 Part E: Bluetooth Host Controller Interface Functional Specification ..... 160
2.6 [Vol 5] AMP Controller ..... 170
2.6.1 Part A: 802.11 PAL ..... 170
2.7 [Vol 6] Low Energy Controller ..... 171
2.7.1 Part A: Radio Physical Layer Specification ..... 171
2.7.2 Part B: Link Layer Specification ..... 174
2.7.3 Part C: Sample Data ..... 179
2.7.4 Part D: Message Sequence Charts ..... 180
2.7.5 Part E: Low Energy Security Specification ..... 183
2.7.6 Part F: Direct Test Mode ..... 185
2.8 [Vol 7] Wireless Coexistence Signaling and Interfaces ..... 188
2.8.1 Part A: MWS Coexistence Logical Signaling Specification ..... 188
2.8.2 Part B: Wireless Coexistence Interface 1 (WCI-1) Transport Specification ..... 188
2.8.3 Part C: Wireless Coexistence Interface 2 (WCI-2) Transport Specification ..... 189

**3 Acknowledgments for v5.2 ..... 190**
3.1 Acknowledgments for LE Isochronous Channels ..... 190
3.2 Acknowledgments for LE Power Control ..... 191
3.3 Acknowledgments for Enhanced Attribute Protocol ..... 192

**4 Acknowledgments for v5.3 ..... 194**
4.1 Acknowledgments for AdvDataInfo in Periodic Advertising ..... 194
4.2 Acknowledgments for Host To Controller Encryption Key Control Enhancements ..... 194
4.3 Acknowledgments for LE Enhanced Connection Update ..... 194
4.4 Acknowledgments for LE Channel Classification ..... 195
4.5 Acknowledgments for Removing Alternate MAC/PHY ..... 195

**5 Acknowledgments for v5.4 ..... 196**
5.1 Coding Scheme Selection on Advertising ..... 196
5.2 Encrypted Advertising Data ..... 196
5.3 Periodic Advertising with Responses ..... 196
5.4 LE GATT Security Levels Characteristic ..... 197

**6 Acknowledgments for v6.0 ..... 198**
6.1 Decision-Based Advertising Filtering ..... 198
6.2 Channel Sounding ..... 198
6.3 Enhancements for ISOAL ..... 200
6.4 LL Extended Feature Set ..... 200

6.5 Monitoring Advertisers ..... 200
6.6 Frame Space Update ..... 201
6.7 Core Configurations ..... 201

**7 Acknowledgments for v6.1 ..... 202**
7.1 Randomized RPA Updates ..... 202

**8 Acknowledgments for v6.2 ..... 203**
8.1 LE Test Mode Enhancements ..... 203
8.2 Shorter Connection Intervals ..... 203
8.3 Channel Sounding Amplitude-based Attack Resilience ..... 204
8.4 HCI USB LE Isochronous Support ..... 204

**9 Acknowledgments for v6.3 ..... 206**
9.1 Running Out of Bits ..... 206
9.2 Channel Sounding Inline Phase Correction Term Transfer ..... 206
9.3 Channel Sounding PHY-specific RTT Accuracy ..... 206

# 1 VERSION HISTORY

Beginning with v1.2 of the Core System Package the core Bluetooth specification documents, protocols and profiles were transferred to a new partitioning comprising volumes and individual profile specifications are each contained in an individual document instead of the two volumes (Core and Profiles) used in v1.1.

For more detailed information about changes between versions published before v1.2, see Appendix I “Revision History” in v1.1.

## 1.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations

| Rev | Date | Comments |
|---|---|---|
| 6.3 | 2026-05-05 | Incorporated errata; see [Vol 1] Part C, Section 17.3.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.2 | 2025-11-03 | Incorporated errata; see [Vol 1] Part C, Section 16.4.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.1 | 2025-05-05 | Incorporated errata; see [Vol 1] Part C, Section 15.3.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.0 | 2024-08-27 | Incorporated errata; see [Vol 1] Part C, Section 14.3. Replaced the Compliance Requirements in Part B with Core Configurations (see [Vol 0] Part D).<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.4 | 2023-01-31 | Incorporated errata; see [Vol 1] Part C, Section 13.3. Updated the requirements for the Basic Rate Core Configuration, Low Energy Core Configuration, and Basic Rate and Low Energy Combined Core Configuration.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.3 | 2021-07-13 | Incorporated errata; see [Vol 1] Part C, Section 12.3. Updated the definition of the Bluetooth Host and Controller Subsystem Products. Updated the requirements for the Low Energy Core Configuration and the Basic Rate and Low Energy Combined Core Configuration. Removed the High Speed Core Configuration.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.2 | 2019-12-31 | Incorporated errata; see [Vol 1] Part C, Section 11.3. Removed references to deprecated and withdrawn versions of the Core Specification. Renamed Volumes. Updated the definition of the Bluetooth Component Product type.<br>Adopted by the Bluetooth SIG Board of Directors. |

| Rev | Date | Comments |
|---|---|---|
| 5.1 | 2019-01-21 | Incorporated errata; see [Vol 1] Part C, Section 10.4. Incorporated CSA6: see [Vol 0] Part B, Section 3.1.2.3 and [Vol 0] Part B, Section 3.1.3 (deleted by erratum 25429). |
| 5.0 | 2016-12-06 | Incorporated errata |
| 4.2 | 2014-12-02 | Incorporated errata |
| 4.1 | 2013-12-03 | Clarifications for allowable complementary subsystems. |
| 4.0 | 2010-06-30 | Updated to support Low Energy, ATT, and GATT for BR/EDR, and to enable High Speed Controller Subsystems. |
| 3.0 + HS | 2009-04-21 | Updated to include support for the Alternative MAC/PHY feature and High Speed Core Configuration. |
| v2.1 + EDR | 2007-07-26 | No content changes. Updates to the Table of Contents. |
| v2.0 + EDR | 2004-10-15 | This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which required changes to Volume 0, Part A, Consolidated Table of Contents. |
| v1.2 | 2003-11-05 | This Part was moved from the Core volume. No content changes have been made to this document since v1.1. |

## 1.2 [Vol 1] Architecture, Change History, and Conventions

| Rev | Date | Comments |
|---|---|---|
| 6.3 | 2026-05-05 | Incorporated errata; see [Vol 1] Part C, Section 17.3. Data signing feature was deprecated and removed.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.2 | 2025-11-03 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 16.4.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.1 | 2025-05-05 | Updates to language conventions. Incorporated errata; see [Vol 1] Part C, Section 15.3.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.0 | 2024-08-27 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 14.3. Vol 1, Part D merged into Vol 0, Part D.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.4 | 2023-01-31 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 13.3.<br>Adopted by the Bluetooth SIG Board of Directors. |

| Rev | Date | Comments |
|---|---|---|
| 5.3 | 2021-07-13 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 12.3. Updated terminology to remove inappropriate language. Alternate MAC/PHY feature was deprecated and removed.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.2 | 2019-12-31 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 11.3. Vol 2, Part D moved to Vol 1, Part F. Removed references to deprecated and withdrawn versions of the Core Specification. Added definition of Resolving List.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.1 | 2019-01-21 | Updated to include architectural aspects of new features added. Incorporated errata; see [Vol 1] Part C, Section 10.4. Incorporated CSA6; see [Vol 1] Part A, Section 6.6 and [Vol 0] Part D, Table 5.1 (this was formerly [Vol 1] Part D, Table 1.3 but was moved by erratum 25429). |
| 5.0 | 2016-12-06 | Updated to include architectural aspects of new features added. Incorporated errata. |
| 4.2 | 2014-12-02 | Updated to include architectural aspects of new features added. Incorporated errata. |
| 4.1 | 2013-12-03 | Updated to include architectural aspects of new features added to CSA2, CSA3, CSA4 and v4.1. |
| 4.0 | 2010-06-30 | Updated to support Low Energy, ATT, and GATT over BR/EDR. |
| 3.0 + HS | 2009-04-21 | Updated to integrate the Alternate MAC/PHY and Unicast Connectionless Data features. |
| v2.1 + EDR | 2007-07-26 | Added definitions for new features: Encryption Pause and Resume, Erroneous Data reporting, Extended Inquiry Response, Link Supervision Timeout Event, Packet Boundary Flag, Secure Simple Pairing, Sniff Subrating. |
| v2.0 + EDR | 2004-10-15 | This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which incorporates changes to Volume 1, Part B, Acronyms and Abbreviations. |
| v1.2 | 2003-11-05 | New volume with informational content. This volume will always be updated in parallel with the Core System volumes. |

## 1.3 [Vols 2, 3, 5, 6 & 7] Controllers and Host

| Rev | Date | Comments |
|---|---|---|
| 6.3 | 2026-05-05 | • New features added in 6.3; see [Vol 1] Part C, Section 17.1.<br>Incorporated errata; see [Vol 1] Part C, Section 17.3.<br>• Data signing feature was deprecated and removed.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.2 | 2025-11-03 | • New features added in 6.2; see [Vol 1] Part C, Section 16.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 16.4.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.1 | 2025-05-05 | • Incorporated errata; see [Vol 1] Part C, Section 15.3.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.0 | 2024-08-27 | • New features added in 6.0; see [Vol 1] Part C, Section 14.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 14.3.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.4 | 2023-01-31 | • New features added in 5.4; see [Vol 1] Part C, Section 13.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 13.3.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.3 | 2021-07-13 | • New features added in 5.3; see [Vol 1] Part C, Section 12.1<br>• Incorporated errata; see [Vol 1] Part C, Section 12.3<br>• Alternate MAC/PHY feature was deprecated and removed.<br>• Updated terminology to remove inappropriate language.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.2 | 2019-12-31 | • New features added in 5.2: see [Vol 1] Part C, Section 11.1<br>• Incorporated erratum 11838: Encryption Key Size Updates; see [Vol 1] Part C, Section 11.2<br>• Incorporated other errata; see [Vol 1] Part C, Section 11.3<br>• Vol 2, Part D moved to Vol 1, Part F<br>• Vol 2, Part E moved to Vol 4, Part E<br>• Removed references to deprecated and withdrawn versions of the Core Specification<br><br>Adopted by the Bluetooth SIG Board of Directors. |

| Rev | Date | Comments |
|---|---|---|
| 5.1 | 2019-01-21 | • New features added in v5.1: see [Vol 1] Part C, Section 10.1.<br>• Unit keys were deprecated and removed; see [Vol 1] Part C, Section 10.2<br>• Incorporated erratum 10734: Pairing Updates; see [Vol 1] Part C, Section 10.3<br>• Incorporated other errata; see [Vol 1] Part C, Section 10.4. |
| 5.0 | 2016-12-06 | • New features added in 5.0:<br>– CSA 5 features (Higher Output Power)<br>– Slot Availability Mask (SAM)<br>– 2 Msym/s PHY for LE<br>– LE Long Range<br>– High Duty Cycle Non-Connectable Advertising<br>– LE Advertising Extensions<br>– LE Channel Selection Algorithm #2<br>• Park State was deprecated and removed<br>• Errata for v2.0 + EDR, v2.1 + EDR, v3.0 + HS + 4.0 + 4.1 + 4.2 (ESR09, ESR10 and ESR11).<br>See also [Vol 1] Part C, Section 9.4. |
| 4.2 | 2014-12-02 | • New features added in 4.2:<br>– LE Data Packet Length Extension<br>– LE Secure Connections<br>– Link Layer Privacy<br>– Link Layer Extended Scanner Filter Policies<br>• Errata for v2.0 + EDR, v2.1 + EDR, v3.0 + HS + 4.0 + 4.1 (ESR08). See also [Vol 1] Part C, Section 8.2. |

| Rev | Date | Comments |
|---|---|---|
| 4.1 | 2013-12-03 | • New features added and changes made in 4.1:<br>– CSA 2 features<br>– CSA 3 features<br>– CSA 4 features<br>– Secure Connections<br>– Train Nudging & Generalized Interlaced Scan<br>– Low Duty Cycle Directed Advertising<br>– 32-bit UUID Support in LE<br>– LE Dual Mode Topology<br>– Piconet Clock Adjustment<br>– Removal of At Least One New Feature<br>– LE L2CAP Connection-Oriented Channel Support<br>– LE Privacy v1.1<br>– LE Link Layer Topology<br>– LE Ping<br>• Errata for v2.0 + EDR, v2.1 + EDR, v3.0 + HS + 4.0 (ESR05, ESR06 and ESR07) |
| 4.0 | 2010-06-30 | • New features added in 4.0:<br>– Low Energy<br>• Errata for v2.0 + EDR, v2.1 + EDR, v3.0 + HS |
| 3.0 + HS | 2009-04-21 | • New features added in 3.0 + HS:<br>– AMP Manager Protocol (A2MP)<br>– Enhancements to L2CAP for AMP<br>– Enhancements to HCI for AMP<br>– Enhancements to Security for AMP<br>– 802.11 Protocol Adaptation Layer<br>– Enhanced Power Control<br>– Unicast Connectionless Data<br>– HCI Read Encryption Key Size command<br>– Generic Test Methodology for AMP<br>– Enhanced USB and SDIO HCI Transports<br>• Errata for v 2.0 + EDR and v2.1 + EDR |

| Rev | Date | Comments |
|---|---|---|
| v2.1 + EDR | 2007-07-26 | • New features added in 2.1 + EDR:<br>– Encryption Pause and Resume<br>– Erroneous Data Reporting<br>– Extended Inquiry Response<br>– Link Supervision Timeout Changed Event<br>– Non-Flushable Packet Boundary Flag<br>– Secure Simple Pairing<br>– Sniff Subrating<br>– Security Mode 4<br>• Updates to IEEE language in Volume 2, Part H, Security<br>• Errata for v2.0 + EDR |
| v2.0 + EDR | 2004-08-01 | This version of the specification is intended to be a separate Bluetooth Specification. This version was created by adding EDR and the errata. |
| v1.2 | 2003-11-05 | New features added in v1.2:<br>• Architectural overview<br>• Faster connection<br>• Adaptive frequency hopping<br>• Extended SCO links<br>• Enhanced error detection and flow control<br>• Enhanced synchronization capability<br>• Enhanced flow specification<br><br>The Core System Package now comprises two volumes and the text has gone through a radical change both in terms of structure and nomenclature. The language is also more precise and is adapted to meet the IEEE standard.<br>The following parts are moved from the Core System Package to other volumes or were deprecated:<br>RFCOMM [vol 7], Object Exchange (IrDA Interoperability) [vol 8], TCS [vol 9], Interoperability Requirements for Bluetooth as a WAP Bearer [vol 6], HCI USB Transport Layer [vol 4], HCI RS232 Transport Layer [vol 4], HCI UART Transport Layer [vol 4], Bluetooth Compliance Requirements [vol 0], Optional Paging Schemes [deprecated] |
| 1.1 | 2001-02-22 | The specification was updated with Errata items previously published on the web site. The Bluetooth Assigned Numbers appendix was lifted out from the specification to allow continuous maintenance on the web site. |

| Rev | Date | Comments |
|---|---|---|
| 1.0B | 1999-12-01 | The specification was updated with Errata items previously published on the web site and was revised from a linguistic point of view.<br>The following parts were added:<br>Interoperability Requirements for Bluetooth as a WAP Bearer, Test Control Interface, Sample Data (appendix), Bluetooth Audio (appendix), Baseband Timers (appendix) and Optional Paging Scheme (appendix) |
| 1.0a | 1999-07-26 | The first version of the Bluetooth Specification published on the public web site.<br>Added part: Bluetooth Compliance Requirements. |
| 1.0 draft | 1999-07-05 | The following parts were added:<br>Service Discovery Protocol (SDP), Telephony Control Specification (TCS), Bluetooth Assigned Numbers (appendix) and Message Sequence Charts (appendix) |
| 0.9 | 1999-04-30 | The following parts were added:<br>IrDA Interoperability, HCI RS232 Transport Layer, HCI UART Transport Layer and Test Mode |
| 0.8 | 1999-01-21 | The following parts were added:<br>Radio Specification, L2CAP, RFCOMM, HCI & HCI USB Transport Layer |
| 0.7 | 1998-10-19 | This first version only included Baseband and Link Manager Protocol |

## 1.4 [Vol 4] Host Controller Interface

| Rev | Date | Comments |
|---|---|---|
| 6.3 | 2026-05-05 | • New features added in 6.3; see [Vol 1] Part C, Section 17.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 17.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.2 | 2025-11-03 | • New features added in 6.2; see [Vol 1] Part C, Section 16.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 16.4.<br>Adopted by the Bluetooth SIG Board of Directors. |
| 6.1 | 2025-05-05 | • New feature added in 6.1; see [Vol 1] Part C, Section 15.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 15.3.<br>Adopted by the Bluetooth SIG Board of Directors. |

| Rev | Date | Comments |
|---|---|---|
| 6.0 | 2024-08-27 | • New features added in 6.0; see [Vol 1] Part C, Section 14.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 14.3.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.4 | 2023-01-31 | • New features added in 5.4; see [Vol 1] Part C, Section 13.1.<br>• Incorporated errata; see [Vol 1] Part C, Section 13.3.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.3 | 2021-07-13 | • New features added in 5.3; see [Vol 1] Part C, Section 12.1<br>• Incorporated errata; see [Vol 1] Part C, Section 12.3<br>• Alternate MAC/PHY feature was deprecated and removed.<br>• Updated terminology to remove inappropriate language.<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.2 | 2019-12-31 | • New features added in 5.2: see [Vol 1] Part C, Section 11.1<br>• Incorporated errata; see [Vol 1] Part C, Section 11.3<br>• Vol 2, Part E moved to Vol 4, Part E<br>• Added requirements for supporting HCI commands and events for BR/EDR and AMP<br>• Removed references to deprecated and withdrawn versions of the Core Specification<br><br>Adopted by the Bluetooth SIG Board of Directors. |
| 5.1 | 2019-01-21 | Incorporated errata; see [Vol 1] Part C, Section 10.4. |
| 5.0 | 2016-12-06 | Incorporated errata |
| 4.2 | 2014-12-02 | Incorporated errata |
| 4.1 | 2013-12-03 | Incorporated errata |
| 4.0 | 2010-06-30 | Incorporated errata |
| 3.0 + HS | 2009-04-21 | Updated the USB and SDIO HCI Transport protocols for high speed, support composite BR/EDR + AMP devices, and to include errata. |
| v2.1 + EDR | 2007-07-26 | Added this volume to the specification |

# 2 ACKNOWLEDGMENTS (UP TO V5.1)

## 2.1 [Vol 0] Consolidated TOC, Acknowledgments, & Core Configurations

### 2.1.1 Part B: Bluetooth Compliance Requirements

| BQRB (Editor) | |
|---|---|
| Kevin Marquess | Broadcom |
| Wayne Park | 3Com Corporation |
| Lawrence Jones | ComBit, Inc. |
| Magnus Sommansson | CSR |
| Gary Robinson | IBM Corporation |
| Georges Seuron | IBM Corporation |
| Rick Jessop | Intel Corporation |
| John Webb | Intel Corporation |
| Bruce Littlefield | Lucent Technologies, Inc. |
| Ellick Sung | Microsoft |
| Brian A. Redding | Motorola, Inc. |
| Waldemar Hontscha | Nokia Corporation |
| Petri Morko | Nokia Corporation |
| Joel Linksy | Qualcomm |
| Brian A. Redding | Qualcomm |
| Hans Andersson | ST Ericsson |
| Magnus Hansson | Telefonaktiebolaget LM Ericsson |
| Magnus Sommansson | Telefonaktiebolaget LM Ericsson |
| Göran Svennarp | Telefonaktiebolaget LM Ericsson |
| Warren Allen | Toshiba Corporation |
| John Shi | Toshiba Corporation |

### 2.1.2 Part C: Version History and Acknowledgments

| Steve Davies | Nokia |
|---|---|

## 2.2 [Vol 1] Architecture, Change History, and Conventions

### 2.2.1 Part A: Architectural Overview

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Julien Gros | Dialog Semiconductor B.V. |
| Pontus Arvidson | Ericsson AB |
| Nick Hunn | GN Hearing A/S |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Frank Berntsen | Nordic Semiconductor ASA |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Technologies, Inc. |

| | |
|---|---|
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Vivien-Thom Leng | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Szymon Slupik | Silvair, Inc. |
| Jeff Solum | Starkey Hearing Technologies |
| Latifa Ali | Synopsys, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Iman Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |
| Szymon Janc | Tieto Poland Sp.z.o.o. |
| Kanji Kerai | Widex A/S |
| Michael Ungstrup | Widex A/S |

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steve Hall | Broadcom |
| Angel Polo | Broadcom |
| Knut Odman | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Dishant Srivastava | CSR |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Harish Balasubramaniam | Intel |
| Johan Hedberg | Intel |
| Marcel Holtmann | Intel |
| Tim Wei | IVT Wireless |
| Josselin de la Broise | Marvell |

| | |
|---|---|
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |
| L.C. Ko | MediaTek |
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor |
| Sam Geeraerts | NXP |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Amre El-Hoiydi | Phonak Communications AG |
| Till Schmalmack | Phonak Communications AG |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Michael Ungstrup | Widex A/S |

*Version 1.2 to 4.2*

| | |
|---|---|
| John Padgette | Accenture |
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |

| | |
|---|---|
| Sriram Hariharan | Apple |
| Joakim Linde | Apple |
| Ayse Findikli | Atheros |
| Prasanna Desai | Broadcom |
| Shawn Ding | Broadcom |
| Leonid Eidelman | Broadcom |
| Steven Hall | Broadcom |
| Farooq Hameed | Broadcom Corporation |
| Robert Hulvey | Broadcom |
| Ash Kapur | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Jennifer Bray | CSR |
| Chris Church | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Tim Howes | CSR |
| Ian Jones | CSR |
| Simon Kingston | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Martin van der Zee | Ericsson |
| Leif Wilhelmsson | Ericsson |
| David Suvak | iAnywhere |

| | |
|---|---|
| Mattias Edlund | Infineon |
| Selim Aissi | Intel |
| Harish Balasubramaniam | Intel |
| Penny Chen | Intel |
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |
| Chris Hansen | Intel |
| Marcel Holtmann | Intel |
| Robert Hughes | Intel |
| Sharon Yang | Intel |
| Yao Wang | IVT |
| Josselin De La Broise | Marvell |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| L. C. Ko | MediaTek |
| Huanchun Ye | MediaTek |
| Anindya Bakshi | Mindtree |
| Shwetha Madadik | Mindtree |
| Krishna Singala | Mindtree |
| Greg Muchnik | Motorola |
| Brian A. Redding | Motorola |
| Lily Chen | NIST |
| N Asokan | Nokia |
| Steve Davies | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |

| | |
|---|---|
| Thomas Müller | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Jürgen Schnitzler | Nokia |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Tsuyoshi Okada | Panasonic Corporation |
| Niclas Granqvist | Polar |
| Terry Bourk | Qualcomm |
| Olaf Hirsch | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Leonard Ott | Socket Mobile |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Guido Bertoni | ST-Ericsson |
| Jorgen van Parijs | ST-Ericsson |
| Yves Wernaers | ST-Ericsson |
| Dominique Everaere | ST-NXP Wireless |
| Ed Callaway | Sunrise Micro Devices |
| Tim Howes | Symbian |
| Michael Hasling | Tality |
| Alon Cheifetz | Texas Instruments |
| Amihai Kidron | Texas Instruments |
| Alon Paycher | Texas Instruments |

| | |
|---|---|
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| John Mersh | TTPCom |
| Jason Hillyard | Wicentric |
| Rod Kimmell | X6D, Inc. |

### 2.2.2 Part B: Acronyms & Abbreviations

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |

| | |
|---|---|
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Iman Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |

*Previous versions*

| | |
|---|---|
| Dan Sönnerstam | Pyramid Communication AB |
| Steve Koester | Bluetooth SIG, Inc. |
| Steve Davies | Nokia |

### 2.2.3 Part C: Core Specification Change History

*Version 5.1*

| | |
|---|---|
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Previous versions*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Steven Singer | CSR |
| Paul Wright | CSR |
| David Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Rob Davies | Philips |
| Peter Hauser | Microsoft |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |

| | |
|---|---|
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Jorgen van Parijs | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| John Mersh | TTPCom |

### 2.2.4 Part D: Mixing of Specification Versions

*Version 5.1*

| | |
|---|---|
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Previous versions*

| | |
|---|---|
| David Suvak | iAnywhere |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Kevin Marquess | Broadcom |
| Joe Decuir | CSR |
| Robin Heydon | CSR |
| Magnus Sommansson | CSR |
| Steven Singer | CSR |
| Mattias Edlund | Infineon |
| Peter Hauser | Microsoft |
| Elick Sung | Microsoft |
| Waldemar Hontscha | Nokia |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Joel Linsky | Qualcomm Atheros |
| Terry Bourk | Qualcomm |

| | |
|---|---|
| Brian A. Redding | Qualcomm |
| Clive D.W. Feather | Samsung Electronics |
| Leonard Ott | Socket Communications |
| Jorgen van Parijs | ST |
| Hans Andersson | ST Ericsson |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

### 2.2.5 Part E: General Terminology and Interpretation

*Version 5.1*

| | |
|---|---|
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Previous versions*

| | |
|---|---|
| Angel Polo | Broadcom |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Tim Wei | IVT Wireless |
| L.C. Ko | MediaTek |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey |
| Florian Lefeuvre | Texas Instruments |

### 2.2.6 Part F: Controller Error Codes

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Iman Shawky | Synopsys, Inc. |

| | |
|---|---|
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |

*Version 5.0*

| | |
|---|---|
| Steven Hall | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Marcel Holtmann | Intel |
| Harish Balasubramaniam | Intel |
| Tim Wei | IVT Wireless |
| Josselin de la Broise | Marvell |
| L.C. Ko | MediaTek |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey |
| Florian Lefeuvre | Texas Instruments |

*Version 4.1*

| | |
|---|---|
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |

| | |
|---|---|
| Giriraj Goyal | CSR |
| Neil Stewart | CSR |
| Leif Wilhelmsson | Ericsson |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Sharon Yang | Intel |
| Olaf Hirsch | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics |
| Jorgen van Parijs | ST Ericsson |
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Robert Hulvey | Broadcom |
| Steven Wenham | CSR |
| Kanji Kerai | Nokia |
| Steve Davies | Nokia |
| Tim Howes | Nokia |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Geert Sonck | ST Ericsson |

*Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Kevin Hayes | Atheros |
| Stratos Chatzikyriakos | Artimi |
| Shawn Ding | Broadcom |
| Joe Decuir | CSR |
| David Suvak | iAnywhere |
| Koen Derom | NXP Semiconductors |
| Joel Linsky | Qualcomm |

| | |
|---|---|
| Krishnan Rajamani | Qualcomm |
| John Hillan | Qualcomm |
| Mayank Sharma | SiRF |
| Jason Hillyard | Staccato Communications |
| William Stoye | Staccato Communications |

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |

| | |
|---|---|
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Robin Heydon (section owner) | CSR |
| Roland Hellfajer | Infineon |
| Joel Linsky | Silicon Wave |
| John Mersh | TTPCom |

This part was earlier included in the LMP and HCI functional Specifications.

## 2.3 [Vol 2] BR/EDR Controller

### 2.3.1 Part A: Radio Physical Layer Specification

*Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Edward Harrison | Anritsu |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Mark Braun | Motorola |
| Joel Linsky | Qualcomm |
| Eran Reuveni | TI |

*Version 2.0 + EDR*

| | |
|---|---|
| Steven Hall | RF Micro Devices |
| Robert Young | CSR |
| Robert Kokke | Ericsson |
| Harald Kafemann | Nokia |
| Jukka Reunamäki | Nokia |

| | |
|---|---|
| Morton Gade | Digianswer |
| Mike Fitton | Toshiba |
| Oren Eliezer | Texas Instruments |
| Stephane Laurent-Michel | Tality |

*Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Morten Gade | Digianswer/Motorola |
| Henrik Hedlund | Ericsson |
| Stefan Agnani | Ericsson |
| Robert Kokke | Ericsson |
| Roland Hellfajer | Infineon |
| Thomas Müller | Nokia |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Steven Hall | Silicon Wave |
| Oren Eliezer | Texas Instruments |
| Mike Fitton | Toshiba |

*Previous versions*

| | |
|---|---|
| Steve Williams | 3Com Corporation |
| Todor V. Cooklov | Aware |
| Poul Hove Kristensen | Digianswer A/S |
| Kurt B. Fischer | Hyper Corporation |
| Kevin D. Marquess | Hyper Corporation |
| Troy Beukema | IBM Corporation |
| Brian Gaucher | IBM Corporation |
| Jeff Schiffer | Intel Corporation |
| James P. Gilb | Mobilian |
| Rich L. Ditch | Motorola, Inc. |
| Paul Burgess | Nokia Corporation |
| Olaf Joeressen | Nokia Corporation |

| | |
|---|---|
| Thomas Müller | Nokia Corporation |
| Arto T. Palin | Nokia Corporation |
| Steven J. Shellhammer | Symbol |
| Sven Mattisson | Telefonaktiebolaget LM Ericsson |
| Lars Nord (section owner) | Telefonaktiebolaget LM Ericsson |
| Anders Svensson | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Allen Hotari | Toshiba Corporation |

### 2.3.2 Part B: Baseband Specification

*Version 5.0*

| | |
|---|---|
| Steven Hall | Broadcom |
| Knut Odman | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Harish Balasubramaniam | Intel |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Prasanna Desai | Broadcom |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Knut Odman | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Ian Jones | CSR |

| | |
|---|---|
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Steven Wenham | CSR |
| Leif Wilhelmsson | Ericsson |
| Oren Haggai | Intel |
| Marcel Holtmann | Intel |
| Sharon Yang | Intel |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Huanchun Ye | MediaTek |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |
| Tsuyoshi Okada | Panasonic Corporation |
| Olaf Hirsch | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Jorgen van Parijs | ST Ericsson |
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Rod Kimmell | X6D, Inc. |

### *Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |

| | |
|---|---|
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Dave Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Oren Haggai | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |

| | |
|---|---|
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |

*Version 2.0 + EDR*

| | |
|---|---|
| Steven Hall | RF Micro Devices |
| Robert Young | CSR |
| Robert Kokke | Ericsson |
| Harald Kafemann | Nokia |
| Joel Linsky | RF Micro Devices |
| Terry Bourk | RF Micro Devices |
| Arto Palin | Nokia |

*Version 1.2*

| | |
|---|---|
| P G Madhavan | Agere |
| Hongbing Gan | Bandspeed, Inc. |
| Tod Sizer | Bell Labs |
| Alexander Thoukydides | CSR |
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Kim Schneider | Digianswer/Motorola |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Niels Nielsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Christian Gehrmann | Ericsson |
| Henrik Hedlund | Ericsson |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Rakesh Taori | Ericsson |
| Jaap Haartsen | Ericsson |
| Stefan Zürbes | Ericsson |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |

| | |
|---|---|
| HungKun Chen | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |
| Adrian Stephens | Mobilian Corporation |
| Jim Lansford | Mobilian Corporation |
| Eric Meihofer | Motorola |
| Arto Palin | Nokia |
| Carmen Kühl | Nokia |
| Hannu Laine | Nokia |
| Jürgen Schnitzler | Nokia |
| Päivi Ruuska | Nokia |
| Thomas Müller | Nokia |
| Antonio Salloum | Philips |
| Harmke de Groot | Philips |
| Marianne van de Casteelee | Philips |
| Rob Davies | Philips |
| Roland Matthijssen | Philips |
| Joel Linsky (section owner) | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Gary Schneider | Symbol Technologies, Inc. |
| Stephen J. Shellhammer | Symbol Technologies, Inc. |
| Michael Hasling | Tality |
| Amihai Kidron | Texas Instruments |
| Dan Michael | Texas Instruments |
| Eli Dekel | Texas Instruments |
| Jie Liang | Texas Instruments |
| Oren Eliezer | Texas Instruments |
| Tally Shina | Texas Instruments |
| Yariv Raveh | Texas Instruments |
| Anuj Batra | Texas Instruments |
| Katsuhiro Kinoshita | Toshiba |
| Toshiki Kizu | Toshiba |
| Yoshimitsu Shimojo | Toshiba |
| Charles Sturman | TTPCom |
| John Mersh | TTPCom |

| | |
|---|---|
| Sam Turner | TTPCom |
| Christoph Scholtz | University of Bonn |
| Simon Baatz | University of Bonn |

*Previous versions*

| | |
|---|---|
| Kevin D. Marquess | Hyper Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| James P. Gilb | Mobilian |
| David E. Cypher | NIST |
| Nada Golmie | NIST |
| Olaf Joeressen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Charlie Mellone | Motorola, Inc. |
| Harmke de Groot | Philips |
| Terry Bourk | Silicon Wave |
| Steven J. Shellhammer | Symbol |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Henrik Hedlund (section owner) | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Joakim Persson | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Onn Haran | Texas Instruments |
| Thomas M. Siep | Texas Instruments |
| Ayse Findikli | Zeevo, Inc. |

*Previous versions [Encryption Sample Data, appendix]*

| | |
|---|---|
| Joel Linksy | RFMD |
| Thomas Müller | Nokia Corporation |
| Thomas Sander | Nokia Corporation |
| Joakim Persson (section owner) | Telefonaktiebolaget LM Ericsson |

*Previous versions [Bluetooth Audio, appendix]*

| | |
|---|---|
| Magnus Hansson | Telefonaktiebolaget LM Ericsson |
| Fisseha Mekuria | Telefonaktiebolaget LM Ericsson |
| Mats Omrin | Telefonaktiebolaget LM Ericsson |
| Joakim Persson (section owner) | Telefonaktiebolaget LM Ericsson |

*Previous versions [Baseband Timers, appendix]*

| | |
|---|---|
| David E. Cyper | NIST |
| Jaap Haartsen (section owner) | Telefonaktiebolaget LM Ericsson |
| Joakim Persson | Telefonaktiebolaget LM Ericsson |
| Ayse Findikli | Zeevo, Inc. |

### 2.3.3 Part C: Link Manager Protocol

*Version 5.0*

| | |
|---|---|
| Steven Hall | Broadcom |
| Knut Odman | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Harish Balasubramaniam | Intel |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Version 4.1*

| | |
|---|---|
| Joakim Linde | Apple |
| Shawn Ding | Broadcom |
| Prasanna Desai | Broadcom |
| Steven Hall | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |

| | |
|---|---|
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Tim Howes | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Jonathan Tanner | CSR |
| Steven Wenham | CSR |
| Leif Wilhelmsson | Ericsson |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Sharon Yang | Intel |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Huanchun Ye | MediaTek |
| Krishna Singala | Mindtree |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |
| Tsuyoshi Okada | Panasonic Corporation |
| Niclas Granqvist | Polar |
| Olaf Hirsch | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Jorgen van Parijs | ST Ericsson |

| | |
|---|---|
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Rod Kimmell | X6D, Inc. |

### *Version 4.0*

| | |
|---|---|
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Kanji Kerai | Nokia |
| Steve Davies | Nokia |
| Tim Howes | Nokia |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Alon Paycher | Texas Instruments |

### *Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Edward Harrison | Anritsu |
| Shawn Ding | Broadcom |
| Robert Hulvey | Broadcom |
| Mark Braun | Motorola |
| Joel Linsky | Qualcomm |
| Eran Reuveni | TI |

### *Version 2.1 + EDR*

| | |
|---|---|
| P G Madhavan | Agere |
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |

| | |
|---|---|
| Paul Wright | CSR |
| Dave Suvak | iAnywhere |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Peter Hauser | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

### *Version 2.0 + EDR*

| | |
|---|---|
| John Mersh | TTPCom Ltd. |
| Joel Linsky | RF Micro Devices |
| Harald Kafemann | Nokia |
| Simon Morris | CSR |

### *Version 1.2*

| | |
|---|---|
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Simon Morris | CSR |
| Alexander Thoukydides | CSR |
| Kim Schneider | Digianswer/Motorola |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |
| Tod Sizer | Lucent Technologies |
| Adrian Stephens | Mobilian |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Carmen Kuhl | Nokia |
| Arto Palin | Nokia |
| Thomas Müller | Nokia |
| Roland Matthijssen | Philips |
| Rob Davies | Philips |
| Harmke de Groot | Philips |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Yariv Raveh | Texas Instruments |
| Tally Shina | Texas Instruments |

| | |
|---|---|
| Amihai Kidron | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| Toshiki Kizu | Toshiba |
| John Mersh (section owner) | TTPCom |
| Sam Turner | TTPCom |

*Previous versions*

| | |
|---|---|
| Kim Schneider | Digianswer A/S |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Julien Corthial | Nokia Corporation |
| Olaf Joeressen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Dong Nguyen | Nokia Corporation |
| Harmke de Groot | Philips |
| Terry Bourk | Silicon Wave |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Tobias Melin (section owner) | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Onn Haran | Texas Instruments |
| John Mersh | TTPCom |

### 2.3.4 Part D: [This part is no longer used]

Acknowledgments for the Controller Error Codes are located in Section 2.2.6.

### 2.3.5 Part E: [This part is no longer used]

Acknowledgments for the Host Controller Interface Functional Specification are located in Section 2.5.2.

### 2.3.6 Part F: Message Sequence Charts

*Version 5.0*

| | |
|---|---|
| Steven Hall | Broadcom |
| Knut Odman | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Harish Balasubramaniam | Intel |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Prasanna Desai | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Steven Wenham | CSR |
| Marcel Holtmann | Intel |
| Josselin De La Broise | Marvell |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |
| Tsuyoshi Okada | Panasonic Corporation |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |

| | |
|---|---|
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Rod Kimmell | X6D, Inc. |

### *Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Kevin Hayes | Atheros |
| Stratos Chatzikyriakos | Artimi |
| Shawn Ding | Broadcom |
| Joe Decuir | CSR |
| David Suvak | iAnywhere |
| Koen Derom | NXP Semiconductors |
| Joel Linsky | Qualcomm |
| Krishnan Rajamani | Qualcomm |
| John Hillan | Qualcomm |
| Mayank Sharma | SiRF |
| Jason Hillyard | Staccato Communications |
| William Stoye | Staccato Communications |

### *Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |

| | |
|---|---|
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Robin Heydon (section owner) | CSR |
| Simon Morris | CSR |
| Jan Åberg | Ericsson |
| Christian Gehrmann | Ericsson |

| | |
|---|---|
| Joel Linsky | Silicon Wave |
| John Mersh | TTPCom |

*Previous versions*

| | |
|---|---|
| Todor Cooklev | 3Com Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Greg Muchnik | Motorola, Inc. |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Dong Nguyen (section owner) | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |

### 2.3.7 Part G: Sample Data

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Prasanna Desai | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Steven Wenham | CSR |
| Marcel Holtmann | Intel |

| | |
|---|---|
| Josselin De La Broise | Marvell |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |
| Tsuyoshi Okada | Panasonic Corporation |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Rod Kimmell | X6D, Inc. |

### *Version 3.0 + HS*

| | |
|---|---|
| Kevin Hayes | Atheros |
| Robert Hulvey | Broadcom |
| Yao Wang | IVT |
| Joel Linsky | Qualcomm |
| John Saad | Qualcomm |

### *Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |

| | |
|---|---|
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Guido Bertoni | ST |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

### *Version 1.2*

| | |
|---|---|
| Joel Linsky | Silicon Wave |

### *Previous versions*

| | |
|---|---|
| Thomas Müller | Nokia Corporation |
| Thomas Sander | Nokia Corporation |
| Joakim Persson (section owner) | Telefonaktiebolaget LM Ericsson |

### 2.3.8 Part H: Security Specification

*Version 4.2*

| | |
|---|---|
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jason Hillyard | Wicentric |

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Prasanna Desai | Broadcom |
| Robert Hulvey | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Marcel Holtmann | Intel |
| Josselin De La Broise | Marvell |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |

| | |
|---|---|
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |

### *Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Edward Harrison | Anritsu |
| Kevin Hayes | Atheros |
| Shawn Ding | Broadcom |
| Robert Hulvey | Broadcom |
| Yao Wang | IVT |
| Mark Braun | Motorola |
| Kaisa Nyberg | Nokia |
| John Saad | Qualcomm |
| Joel Linsky | Qualcomm |
| Eran Reuveni | TI |

### *Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |

| | |
|---|---|
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| Mattias Edlund | Infineon |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

## 2.4 [Vol 3] Host

### 2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification

*Version 4.2*

| | |
|---|---|
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |

| | |
|---|---|
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jason Hillyard | Wicentric |

### *Version 4.1*

| | |
|---|---|
| Leonid Eidelman | Broadcom |
| Ash Kapur | Broadcom |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Chris Church | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Tim Howes | CSR |
| Neil Stewart | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |
| Marcel Holtmann | Intel |
| Robert Hughes | Intel |
| Yao Wang | IVT |
| Josselin De La Broise | Marvell |
| Anindya Bakshi | Mindtree |
| Shwetha Madadik | Mindtree |
| Krishna Singala | Mindtree |
| Niclas Granqvist | Polar |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |

| | |
|---|---|
| Rasmus Abildgren | Samsung Electronics |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Xavier Boniface | Alpwise |
| Alexandre Gimard | Alpwise |
| Ash Kapur | Broadcom |
| Robert Hulvey | Broadcom |
| Joe Decuir | CSR |
| Laurence Jupp | CSR |
| Magnus Sommansson | CSR |
| Robin Heydon | CSR |
| Dave Suvak | iAnywhere |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Kanji Kerai | Nokia |
| Miika Laaksonen | Nokia |
| Steve Davies | Nokia |
| Tim Howes | Nokia |
| David Lopes | Nordic Semiconductor |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Andy Estrada | Sony |
| Karl Torvmark | Texas Instruments |

*Version 3.0 + HS*

| | |
|---|---|
| Victor Zhodzishsky | Broadcom |
| Angel Polo | Broadcom |
| Ash Kapur | Broadcom |
| Robert Hulvey | Broadcom |
| Ken Steck | CSR |
| Dave Suvak | iAnywhere |

| | |
|---|---|
| Yao Wang | IVT |
| Jacques Chassot | Logitech |
| Nathan Sherman | Microsoft |
| Sandy Spinrad | Microsoft |
| Doug Clarke | Nokia |
| Kanji Kerai | Nokia |
| Ana Donezar Ibanez | Parrot |
| Joel Linksy | Qualcomm |
| Andy Estrada | Sony |
| Dominique Everaere | ST-NXP Wireless |

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Dave Suvak | iAnywhere |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |

| | |
|---|---|
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Tim Howes | Symbian |
| Jorgen van Parijs | ST |
| Amihai Kidron | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

### *Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Carsten Andersen (section owner) | CSR |
| Jennifer Bray | CSR |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Sam Ravnborg | Ericsson |
| Stefan Agnani | Ericsson |
| Steve McGowan | Intel Corporation |
| Joby Lafky | Microsoft |
| Doron Holan | Microsoft |
| Andy Glass | Microsoft |
| Brian A. Redding | Motorola |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Rob Davies | Philips |
| Terry Bourk | Silicon Wave |
| Michael Hasling | Tality |

*Previous versions*

| | |
|---|---|
| Jon Burgess | 3Com Corporation |
| Paul Moran | 3Com Corporation |
| Doug Kogan | Extended Systems |
| Kevin D. Marquess | Hyper Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Uma Gadamsetty | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Jon Inouye | Intel Corporation |
| Steve C. Lo | Intel Corporation |
| Chunrong Zhu | Intel Corporation |
| Sergey Solyanik | Microsoft Corporation |
| David E. Cypher | NIST |
| Nada Golmie | NIST |
| Thomas Busse | Nokia Corporation |
| Rauno Makinen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Petri Nykänen | Nokia Corporation |
| Peter Ollikainen | Nokia Corporation |
| Petri O. Nurminen | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Elco Nijboer | Telefonaktiebolaget LM Ericsson |
| Ingemar Nilsson | Telefonaktiebolaget LM Ericsson |
| Stefan Runesson | Telefonaktiebolaget LM Ericsson |
| Gerrit Slot | Telefonaktiebolaget LM Ericsson |
| Johan Sörensen | Telefonaktiebolaget LM Ericsson |
| Goran Svennarp | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Thomas M. Siep | Texas Instruments |
| Kinoshita Katsuhiro | Toshiba Corporation |

### 2.4.2 Part B: Service Discovery Protocol (SDP)

*Version 4.1*

| | |
|---|---|
| Joakim Linde | Apple |
| Robert Hulvey | Broadcom |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Tim Howes | CSR |
| Jonathan Tanner | CSR |
| Magnus Eriksson | Intel |
| Krishna Singala | Mindtree |
| Niclas Granqvist | Polar |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Rasmus Abildgren | Samsung Electronics |

*Version 2.1 + EDR*

| | |
|---|---|
| Robin Heydon | CSR |
| Mattias Edlund | Infineon |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Amihai Kidron | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Previous versions*

| | |
|---|---|
| Ned Plasson | 3Com Corporation |
| John Avery | Convergence |
| Jason Kronz | Convergence |
| Chatschik Bisdikian | IBM Corporation |
| Parviz Kermani | IBM Corporation |
| Brent Miller | IBM Corporation |
| Dick Osterman | IBM Corporation |

| | |
|---|---|
| Bob Pascoe | IBM Corporation |
| Jon Inouye | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Jay Eaglstun | Motorola, Inc. |
| Dale Farnsworth (section owner) | Motorola, Inc. |
| Jean-Michel Rosso | Motorola, Inc. |
| Jan Grönholm | Nokia Corporation |
| Kati Rantala | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Kazuaki Iwamura | Toshiba Corporation |

### 2.4.3 Part C: Generic Access Profile

*Version 5.1*

| | |
|---|---|
| Siegfried Lehmann | Apple Inc. |
| Shawn Ding | Broadcom Corporation |
| Julien Gros | Dialog Semiconductor B.V. |
| Nick Hunn | GN Hearing A/S |
| Jakub Pawlowski | Google Inc. |
| Luiz Von Dentz | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Frank Berntsen | Nordic Semiconductor ASA |
| Sandeep Choudhary | Nordic Semiconductor ASA |
| Scott Walsh | Plantronics Inc. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Vivien-Thom Leng | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Szymon Slupik | Silvair, Inc. |
| Jeff Solum | Starkey Hearing Technologies |
| Khaled Elsayed | Synopsys, Inc. |

| | |
|---|---|
| Latifa Ali | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Kanji Kerai | Widex A/S |
| Michael Ungstrup | Widex A/S |

*Version 5.0*

| | |
|---|---|
| Robert Hulvey | Broadcom |
| Angel Polo | Broadcom |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Tim Wei | IVT Wireless |
| L.C. Ko | MediaTek |
| Sam Geeraerts | NXP |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Amre El-Hoiydi | Phonak Communications AG |
| Till Schmalmack | Phonak Communications AG |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Laurence Richardson | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Michael Ungstrup | Widex A/S |

*Version 4.2*

| | |
|---|---|
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |

| | |
|---|---|
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Jonathan Tanner | CSR |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jason Hillyard | Wicentric |

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Joakim Linde | Apple |
| Prasanna Desai | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Angel Polo | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Tim Howes | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Krishnan Nair | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Neil Stewart | CSR |

| | |
|---|---|
| Jonathan Tanner | CSR |
| Steven Wenham | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |
| Marcel Holtmann | Intel |
| Robert D. Hughes | Intel |
| Josselin De La Broise | Marvell |
| Huanchun Ye | MediaTek |
| Alain Michaud | Microsoft |
| Krishna Singala | Mindtree |
| Lily Chen | NIST |
| Steve Davies | Nokia |
| Kaisa Nyberg | Nokia |
| David Engelien-Lopes | Nordic Semidonductor |
| Tsuyoshi Okada | Panasonic Corporation |
| Niclas Granqvist | Polar |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Abelardo Gonzales | ST |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Jason Hillyard | Wicentric |
| Rod Kimmell | X6D, Inc. |

*Version 4.0*

| | |
|---|---|
| Alexandre Gimard | Alpwise |
| Xavier Boniface | Alpwise |
| Mike Tsai | Atheros |
| John Padgette | Booz Allen Hamilton |
| Ash Kapur | Broadcom |
| Norbert Grunert | Broadcom |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Burch Seymour | Continental Automotive |
| Joe Decuir | CSR |
| Magnus Sommansson | CSR |
| Nick Hunn | CSR |
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| David Suvak | iAnywhere |
| Chiu-Mien Chi | ISSC |
| Anindya Bakshi | MindTree |
| Ashok Kelur | MindTree |
| John Barr | Motorola |
| Michael Russell | Motorola |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Kanji Kerai | Nokia |
| Miika Laaksonen | Nokia |
| Steve Davies | Nokia |
| Tim Howes | Nokia |
| David Lopes | Nordic Semiconductor |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Harsha Master | Sasken |
| Len Ott | Socket Mobile |

| | |
|---|---|
| Frédéric Virot | ST Ericsson |
| Geert Sonck | ST Ericsson |
| Pär-Gunnar Hjälmdahl | ST Ericsson |
| Aaron Atlas | Texas Instruments |

### *Version 3.0 + HS*

| | |
|---|---|
| Robert Hulvey | Broadcom |
| Ken Steck | CSR |
| Jacques Chassot | Logitech |
| Nathan Sherman | Microsoft |
| Sandy Spinrad | Microsoft |
| Kanji Kerai | Nokia |
| Joel Linksy | Qualcomm |
| Andy Estrada | Sony |

### *Version 2.1 + EDR*

| | |
|---|---|
| Robin Heydon | CSR |
| Mattias Edlund | Infineon |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Shimojo Yoshimitsu | Toshiba |

### *Version 1.2*

| | |
|---|---|
| Jennifer Bray | CSR |
| Alexander Thoukydides | CSR |
| Christian Gehrmann | Ericsson |
| Henrik Hedlund | Ericsson |
| Jan Åberg | Ericsson |
| Stefan Agnani | Ericsson |
| Thomas Müller | Nokia |
| Joel Linsky | Silicon Wave |

| | |
|---|---|
| Terry Bourk | Silicon Wave |
| Katsuhiro Kinoshita | Toshiba |

*Previous versions*

| | |
|---|---|
| Ken Morley | 3Com Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Jon Inouye | Intel Corporation |
| Brian A. Redding | Motorola, Inc. |
| David E. Cypher | NIST |
| Stephane Bouet | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Martin Roter | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Patric Lind (section owner) | Telefonaktiebolaget LM Ericsson |
| Erik Slotboom | Telefonaktiebolaget LM Ericsson |
| Johan Sörensen | Telefonaktiebolaget LM Ericsson |

### 2.4.4 Part D: Test Support

*Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Edward Harrison | Anritsu |
| Angus Robinson | Anritsu |
| Kevin Hayes | Atheros |
| Stratos Chatzikyriakos | Artimi |
| Shawn Ding | Broadcom |
| Robert Hulvey | Broadcom |
| Yao Wang | IVT |
| Joe Decuir | CSR |
| Mark Braun | Motorola |
| Joel Linsky | Qualcomm |
| Eran Reuveni | TI |

### *Version 1.2*

| | |
|---|---|
| Emilio Mira Escartis | Cetecom |
| Robin Heydon | CSR |
| Jennifer Bray | CSR |
| Stefan Agnani (section owner) | Ericsson |
| Terry Bourk | Silicon Wave |
| Joel Linsky | Silicon Wave |
| Michael Hasling | Tality |

### *Previous versions [Test Mode]*

| | |
|---|---|
| Jeffrey Schiffer | Intel Corporation |
| David E. Cypher | NIST |
| Daniel Bencak | Nokia Corporation |
| Arno Kefenbaum | Nokia Corporation |
| Thomas Müller (section owner) | Nokia Corporation |
| Roland Schmale | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Stefan Agnani | Telefonaktiebolaget LM Ericsson |
| Mårten Mattsson | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Lars Nord | Telefonaktiebolaget LM Ericsson |
| Fredrik Töörn | Telefonaktiebolaget LM Ericsson |
| John Mersh | TTPCom |
| Ayse Findikli | Zeevo, Inc. |

### *Previous versions [Test Control Interface]*

| | |
|---|---|
| Mike Feldman | Motorola, Inc. |
| Thomas Müller | Nokia Corporation |
| Stefan Agnani (section owner) | Telefonaktiebolaget LM Ericsson |
| Mårten Mattsson | Telefonaktiebolaget LM Ericsson |
| Dan Sönnerstam | Telefonaktiebolaget LM Ericsson |

### 2.4.5 Part E: AMP Manager Protocol

*Version 3.0 + HS*

| | |
|---|---|
| Kevin Hayes | Atheros |
| Robert Hulvey | Broadcom |
| Dave Suvak | iAnywhere |
| Yao Wang | IVT |
| Doug Clark | Nokia |
| Kaisa Nyberg | Nokia |
| James Steele | Nokia |
| Joel Linsky | Qualcomm |
| John Saad | Qualcomm |

### 2.4.6 Part F: Attribute Protocol Specification

*Version 5.1*

| | |
|---|---|
| Siegfried Lehmann | Apple Inc. |
| Jakub Pawlowski | Google Inc. |
| Luiz Von Dentz | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Sandeep Choudhary | Nordic Semiconductor ASA |
| Scott Walsh | Plantronics Inc. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Michael Ungstrup | Widex A/S |

*Version 4.1*

| | |
|---|---|
| Joakim Linde | Apple |
| Robert Hulvey | Broadcom |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Tim Howes | CSR |

| | |
|---|---|
| Jonathan Tanner | CSR |
| Magnus Eriksson | Intel |
| Krishna Singala | Mindtree |
| Niclas Granqvist | Polar |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Rasmus Abildgren | Samsung Electronics |

*Version 4.0*

| | |
|---|---|
| Alexandre Gimard | Alpwise |
| Xavier Boniface | Alpwise |
| Ash Kapur | Broadcom |
| Norbert Grunert | Broadcom |
| Srikanth Uppala | Broadcom |
| Burch Seymour | Continental Automotive |
| Joe Decuir | CSR |
| Magnus Sommansson | CSR |
| Nick Hunn | CSR |
| Robin Heydon | CSR |
| Morgan Lindqvist | Ericsson |
| Dave Suvak | iAnywhere |
| Chris Hansen | Intel |
| Marcel Holtmann | Intel |
| Anindya Bakshi | MindTree |
| John Barr | Motorola |
| Daidi Zhong | Nokia |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Juha Salokannel | Nokia |
| Kanji Kerai | Nokia |
| Miika Laaksonen | Nokia |
| Päivi Ruuska | Nokia |
| Steve Davies | Nokia |

| | |
|---|---|
| Tim Howes | Nokia |
| David Lopes | Nordic Semiconductor |
| Niclas Granqvist | Polar |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Harsha Master | Sasken |
| Len Ott | Socket Mobile |
| Pär-Gunnar Hjälmdahl | ST Ericsson |
| Aaron Atlas | Texas Instruments |

### 2.4.7 Part G: Generic Attribute Profile Specification

*Version 5.1*

| | |
|---|---|
| Siegfried Lehmann | Apple Inc. |
| Jakub Pawlowski | Google Inc. |
| Luiz Von Dentz | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Sandeep Choudhary | Nordic Semiconductor ASA |
| Scott Walsh | Plantronics Inc. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Michael Ungstrup | Widex A/S |

*Version 4.1*

| | |
|---|---|
| Joakim Linde | Apple |
| Robert Hulvey | Broadcom |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Tim Howes | CSR |
| Jonathan Tanner | CSR |

| | |
|---|---|
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |
| Krishna Singala | Mindtree |
| Niclas Granqvist | Polar |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Rasmus Abildgren | Samsung Electronics |

*Version 4.0*

| | |
|---|---|
| Alexandre Gimard | Alpwise |
| Xavier Boniface | Alpwise |
| Ash Kapur | Broadcom |
| Norbert Grunert | Broadcom |
| Srikanth Uppala | Broadcom |
| Mats Andersson | connectBlue |
| Burch Seymour | Continental Automotive |
| Joe Decuir | CSR |
| Laurence Jupp | CSR |
| Magnus Sommansson | CSR |
| Nick Hunn | CSR |
| Robin Heydon | CSR |
| Dave Suvak | iAnywhere |
| Marcel Holtmann | Intel |
| Anindya Bakshi | MindTree |
| Krishna Shingala | MindTree |
| Daidi Zhong | Nokia |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Juha Salokannel | Nokia |
| Kanji Kerai | Nokia |
| Miika Laaksonen | Nokia |
| Päivi Ruuska | Nokia |
| Steve Davies | Nokia |

| | |
|---|---|
| Tim Howes | Nokia |
| David Lopes | Nordic Semiconductor |
| Sebastien Mackaie-Blanchi | Nordic Semiconductor |
| Niclas Granqvist | Polar |
| Brian A. Redding | Qualcomm |
| Joel Linksy | Qualcomm |
| Terry Bourk | Qualcomm |
| Pär-Gunnar Hjälmdahl | ST Ericsson |
| Aaron Atlas | Texas Instruments |

### 2.4.8 Part H: Security Manager Specification

*Version 5.0*

| | |
|---|---|
| Johan Hedberg | Intel |
| Marcel Holtmann | Intel |
| Joel Linsky | Qualcomm Technologies, Inc. |

*Version 4.2*

| | |
|---|---|
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Jonathan Tanner | CSR |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Alexandre Gimard | Alpwise |
| Mike Tsai | Atheros |
| John Padgette | Booz Allen Hamilton |
| Chaojing Sun | Broadcom |
| Norbert Grunert | Broadcom |
| Prasanna Desai | Broadcom |
| Robert Hulvey | Broadcom |
| Hermanni Suominen | CSR |
| Laurence Jupp | CSR |
| Magnus Sommansson | CSR |
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Patrick Reinelt | Frontline |
| Chiu-Mien Chi | ISSC |
| Anindya Bakshi | MindTree |
| Ashok Kelur | MindTree |
| Michael Russell | Motorola |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Kanji Kerai | Nokia |
| Miika Laaksonen | Nokia |
| Noel Lobo | Nokia |
| Steve Davies | Nokia |
| Tim Howes | Nokia |
| David Lopes | Nordic Semiconductor |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Christian Zechlin | Sasken |
| Frédéric Virot | ST Ericsson |
| Geert Sonck | ST Ericsson |

| | |
|---|---|
| Alon Paycher | Texas Instruments |
| Helge Coward | Texas Instruments |

## 2.5 [Vol 4] Host Controller Interface

### 2.5.1 Parts A to D: Transport Layers

| | |
|---|---|
| Toru Aihara | IBM Corporation |
| Edgar Kerstan | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Patrick Kane | Motorola, Inc. |
| Uwe Gondrum | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Sven Jerlhagen | Telefonaktiebolaget LM Ericsson |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Lars Novak | Telefonaktiebolaget LM Ericsson |
| Masahiro Tada | Toshiba Corporation |
| Steve Ross | Digianswer A/S |
| Chatschik Bisdikian | IBM Corporation |
| Les Cline | Intel Corporation |
| Brad Hosler | Intel Corporation |
| John Howard | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Kosta Koeman | Intel Corporation |
| John McGrath | Intel Corporation |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Leonard Ott | Socket Communications |
| Rebecca O'Dell | Signia Technologies |
| Tsuyoshi Okada | Matsushita Electric |
| Robert Hulvey | Broadcom |

| | |
|---|---|
| Joe Decuir | CSR |
| Robin Heydon | CSR |
| Joel Linsky | Qualcomm Atheros |
| Toru Aihara | IBM Corporation |
| Edgar Kerstan | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Patrick Kane | Motorola, Inc. |
| Uwe Gondrum | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Sven Jerlhagen | Telefonaktiebolaget LM Ericsson |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Lars Novak | Telefonaktiebolaget LM Ericsson |
| Masahiro Tada | Toshiba Corporation |
| Steve Ross | Digianswer A/S |
| Chatschik Bisdikian | IBM Corporation |
| Les Cline | Intel Corporation |
| Brad Hosler | Intel Corporation |
| John Howard | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Kosta Koeman | Intel Corporation |
| John McGrath | Intel Corporation |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Leonard Ott | Socket Communications |
| Rebecca O'Dell | Signia Technologies |
| Tsuyoshi Okada | Matsushita Electric |

### 2.5.2 Part E: Bluetooth Host Controller Interface Functional Specification

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Julien Gros | Dialog Semiconductor B.V. |
| Nick Hunn | GN Hearing A/S |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Mayur Maheshwari | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Frank Berntsen | Nordic Semiconductor ASA |
| Brian A. Redding | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Brian A. Redding | Qualcomm Technologies, Inc. |

| | |
|---|---|
| Joel Linsky | Qualcomm Technologies, Inc. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Vivien-Thom Leng | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Iman Shawky | Synopsys, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Latifa Ali | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |
| Szymon Janc | Tieto Poland Sp.z.o.o. |
| Kanji Kerai | Widex A/S |
| Michael Ungstrup | Widex A/S |

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Robert Hulvey | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Huanchun Ye | Broadcom |
| Burch Seymour | Continental Automotive |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Tim Wei | IVT Wireless |
| Josselin de la Broise | Marvell |
| Yi-Ling Chao | Marvell |

| | |
|---|---|
| KC Chou | MediaTek |
| L.C. Ko | MediaTek |
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Amre El-Hoiydi | Phonak Communications AG |
| Till Schmalmack | Phonak Communications AG |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Laurence Richardson | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Michael Ungstrup | Widex A/S |

*Version 4.2*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Sriram Hariharan | Apple |

| | |
|---|---|
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Jonathan Tanner | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Ed Callaway | Sunrise Micro Devices |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Jason Hillyard | Wicentric |

*Version 4.1*

| | |
|---|---|
| John Padgette | Accenture |
| Joakim Linde | Apple |
| Prasanna Desai | Broadcom |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Farooq Hameed | Broadcom |
| Robert Hulvey | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Erik Rivard | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |

| | |
|---|---|
| Robin Heydon | CSR |
| Tim Howes | CSR |
| Ian Jones | CSR |
| Sean Mitchell | CSR |
| Ross O'Connor | CSR |
| Steven Singer | CSR |
| Dishant Srivastava | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Steven Wenham | CSR |
| Leif Wilhelmsson | Ericsson |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |
| Marcel Holtmann | Intel |
| Sharon Yang | Intel |
| Yao Wang | IVT Corporation |
| Josselin de la Broise | Marvell |
| L. C. Ko | MediaTek |
| Huanchun Ye | MediaTek |
| Rajan S Pallathu | MindTree Ltd. |
| Krishna Singala | Mindtree |
| Lily Chen | NIST |
| Kaisa Nyberg | Nokia |
| David Engelien-Lopes | Nordic Semiconductor |
| Tsuyoshi Okada | Panasonic Corporation |
| Niclas Granqvist | Polar |
| Olaf Hirsch | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Cameron McDonald | Qualcomm |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Phillippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |

| | |
|---|---|
| Kyong-Sok Seo | Samsung Electronics Co. Ltd |
| Len Ott | Socket Mobile |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Jorgen van Parijs | ST Ericsson |
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |
| Alon Paycher | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Jason Hillyard | Wicentric |
| Rod Kimmell | X6D, Inc. |

*Version 4.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Kevin Hayes | Atheros |
| Joe Decuir | CSR |
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Magnus Eriksson | Infineon |
| James Dent | Nokia |
| James Steele | Nokia |
| Jonathan Tanner | Nokia |
| Steve Davies | Nokia |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Len Ott | Socket Mobile |
| Geert Sonck | ST Ericsson |
| Aaron Atlas | Texas Instruments |
| Anthony Viscardi | Texas Instruments |

*Version 3.0 + HS*

| | |
|---|---|
| Phil Hough | Anritsu |
| Kevin Hayes | Atheros |

| | |
|---|---|
| Stratos Chatzikyriakos | Artimi |
| Angel Polo | Broadcom |
| Shawn Ding | Broadcom |
| Victor Zhodzishsky | Broadcom |
| Joe Decuir | CSR |
| David Suvak | iAnywhere |
| Dominique Everaere | ST-NXP Wireless |
| Yao Wang | IVT |
| Koen Derom | NXP Semiconductors |
| Ana Donezar Ibanez | Parrot |
| Joel Linsky | Qualcomm |
| John Hillan | Qualcomm |
| Krishnan Rajamani | Qualcomm |
| Mayank Sharma | SiRF |
| Jason Hillyard | Staccato Communications |
| William Stoye | Staccato Communications |
| Doug Clark | Symbian |

### *Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Mattias Edlund | Infineon |
| David Suvak | iAnywhere |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |

| | |
|---|---|
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Tim Howes | Symbian |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

### *Version 2.0 + EDR*

| | |
|---|---|
| Neil Stewart | Tality UK, Ltd. |
| Joel Linsky | RF Micro Devices |
| Robin Heydon | CSR |

*Version 1.2*

| | |
|---|---|
| Robin Heydon (section owner) | CSR |
| Jennifer Bray | CSR |
| Alexander Thoukydides | CSR |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Don Liechty | Extended Systems |
| Kevin Marquess | Hyper Corp |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |
| Tod Sizer | Lucent Technologies |
| Tsuyoshi Okada | Matsushita Electric Industrial Co. Ltd |
| Andy Glass | Microsoft |
| Adrian Stephens | Mobilian |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Rene Tischer | Nokia |
| Rob Davies | Philips |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Len Ott | Socket Communications |
| Randy Erman | Taiyo Yuden |
| Yoshimitsu Shimojo | Toshiba |
| Toshiki Kizu | Toshiba |
| Katsuhiro Kinoshita | Toshiba |
| Sam Turner | TTPCom |
| John Mersh | TTPCom |

*Previous versions*

| | |
|---|---|
| Todor Cooklev | 3Com Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Akihiko Mizutani | IBM Corporation |
| Les Cline | Intel Corporation |
| Bailey Cross | Intel Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Jon Inouye | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Steve Lo | Intel Corporation |
| Vijay Suthar | Intel Corporation |
| Bruce P. Kraemer | Intersil |
| Greg Muchnik | Motorola, Inc. |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Julien Courthial | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Dong Nguyen | Nokia Corporation |
| Jürgen Schnitzler | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Christian Johansson (section owner) | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Thomas M. Siep | Texas Instruments |
| Masahiro Tada | Toshiba Corporation |
| John Mersh | TTPCom |

## 2.6 [Vol 5] AMP Controller

### 2.6.1 Part A: 802.11 PAL

*Version 4.1*

| | |
|---|---|
| Kevin Hayes | Atheros |
| Raymond Hayes | Broadcom |
| Francoise Bannister | CSR |
| Nick Jackson | CSR |
| Shailendra Govardhan | Marvell |
| Joel Linsky | Qualcomm |

*Version 3.0 +HS*

| | |
|---|---|
| Angus Robinson | Anritsu |
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Stratos Chatzikyriakos | Artimi |
| Kevin Hayes | Atheros |
| Prerepa Viswanadham | Atheros |
| Chris Hansen | Broadcom |
| Raymond Hayes | Broadcom |
| Joe Decuir | CSR |
| Nick Jackson | CSR |
| David Suvak | iAnywhere |
| Ganesh Venkatesan | Intel |
| Quinton Yuan | Marvell |
| Raja Banerjea | Marvell |
| John Barr | Motorola |
| Janne Marin | Nokia |
| Mika Kasslin | Nokia |
| Joel Linsky | Qualcomm |
| Krishnan Rajamani | Qualcomm |
| Terry Bourk | Qualcomm |
| Ignacio Gimeno | ST-NXP Wireless |
| William Stoye | Staccato Communications |

| | |
|---|---|
| Amir Yassur | TI |
| Yossi Peery | TI |

## 2.7 [Vol 6] Low Energy Controller

### 2.7.1 Part A: Radio Physical Layer Specification

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |

| | |
|---|---|
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Iman Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Angel Polo | Broadcom |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Leif Wilhelmsson | Ericsson |
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Bjarne Klemmensen | Oticon A/S |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RiveraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |

| | |
|---|---|
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Micheal Ungstrup | Widex |

### *Version 4.2*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Mayank Batra | CSR |
| Robin Heydon | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Clive D.W. Feather | Samsung Electronics |
| Ed Callaway | Sunrise Micro Devices |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Jason Hillyard | Wicentric |

### *Version 4.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Robert Hulvey | Broadcom |
| Burch Seymour | Continental Automotive |
| Magnus Sommansson | CSR |
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Henrik Arfwedson | Infineon |
| Jukka Reunamaki | Nokia |
| Mika Kasslin | Nokia |
| Steve Davies | Nokia |
| Frank Karlsen | Nordic Semiconductor |
| Øyvind Vedal | Nordic Semiconductor |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |

| | |
|---|---|
| Terry Bourk | Qualcomm |
| Don Sturek | Texas Instruments |

### 2.7.2 Part B: Link Layer Specification

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Julien Gros | Dialog Semiconductor B.V. |
| Pontus Arvidson | Ericsson AB |
| Nick Hunn | GN Hearing A/S |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Frank Berntsen | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP Semiconductors |
| Brian A. Redding | Qualcomm Atheros |

| | |
|---|---|
| Joel Linsky | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Vivien-Thom Leng | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Szymon Slupik | Silvair, Inc. |
| Jeff Solum | Starkey Hearing Technologies |
| Iman Shawky | Synopsys, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Latifa Ali | Synopsys, Inc. |
| Tomas Motos Lopez | Texas Instruments Incorporated |
| Szymon Janc | Tieto Poland Sp.z.o.o. |
| Kanji Kerai | Widex A/S |
| Michael Ungstrup | Widex A/S |

### *Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Robert Hulvey | Broadcom |
| Knut Odman | Broadcom |
| Angel Polo | Broadcom |
| Huanchun Ye | Broadcom |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Leif Wilhelmsson | Ericsson |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |

| | |
|---|---|
| Tim Wei | IVT Wireless |
| Josselin de la Broise | Marvell |
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |
| L.C. Ko | MediaTek |
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP |
| Chris Deck | ON Semiconductor |
| Bjarne Klemmensen | Oticon A/S |
| Amre El-Hoiydi | Phonak Communications AG |
| Till Schmalmack | Phonak Communications AG |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Laurence Richardson | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Michael Knudsen | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Michael Ungstrup | Widex A/S |

### *Version 4.2*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Jonathan Tanner | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Ed Callaway | Sunrise Micro Devices |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Jason Hillyard | Wicentric |

### *Version 4.1*

| | |
|---|---|
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Neil Stewart | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Oren Haggai | Intel |

| | |
|---|---|
| Yao Wang | IVT Corporation |
| Rajan S Pallathu | MindTree Ltd. |
| David Lopes | Nordic Semiconductor |
| Joel Linsky | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics |
| Anthony Viscardi | Texas Instruments |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Angel Polo | Broadcom |
| Prasanna Desai | Broadcom |
| Robert Hulvey | Broadcom |
| Yuan Zhuang | Broadcom |
| Burch Seymour | Continental Automotive |
| Joe Decuir | CSR |
| Magnus Sommansson | CSR |
| Robin Heydon | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Robert Kvacek | EM Microelectronic |
| Henrik Arfwedson | Infineon |
| James Dent | Nokia |
| Jonathan Tanner | Nokia |
| Kanji Kerai | Nokia |
| Mika Kasslin | Nokia |
| Steve Davies | Nokia |
| Asbjørn Sæbø | Nordic Semiconductor |
| David Lopes | Nordic Semiconductor |
| Frank Karlsen | Nordic Semiconductor |
| Torstein Nesje | Nordic Semiconductor |
| Brian A. Redding | Qualcomm |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |

| | |
|---|---|
| Harsha Master | Sasken |
| Geert Sonck | ST Ericsson |
| Alon Paycher | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Don Sturek | Texas Instruments |
| Helge Coward | Texas Instruments |

### 2.7.3 Part C: Sample Data

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Steven Hall | Broadcom |
| Angel Polo | Broadcom |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Harish Balasubramaniam | Intel |
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |
| James Wang | MediaTek |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP |
| Bjarne Klemmensen | Oticon A/S |
| Amre El-Hoiydi | Phonak Communications AG |
| Till Schmalmack | Phonak Communications AG |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |

| | |
|---|---|
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Michael Ungstrup | Widex A/S |

*Version 4.0*

| | |
|---|---|
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Joel Linsky | Qualcomm |
| Geert Sonck | ST Ericsson |

### 2.7.4 Part D: Message Sequence Charts

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Julien Gros | Dialog Semiconductor B.V. |
| Nick Hunn | GN Hearing A/S |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |

| | |
|---|---|
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Frank Berntsen | Nordic Semiconductor ASA |
| Brian A. Redding | Qualcomm Atheros |
| Joel Linsky | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Brian A. Redding | Qualcomm Technologies, Inc. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Vivien-Thom Leng | RivieraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Iman Shawky | Synopsys, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Latifa Ali | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |
| Kanji Kerai | Widex A/S |
| Michael Ungstrup | Widex A/S |

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Angel Polo | Broadcom |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |

| | |
|---|---|
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Bjarne Klemmensen | Oticon A/S |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RiveraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Micheal Ungstrup | Widex |

*Version 4.2*

| | |
|---|---|
| Sriram Hariharan | Apple |
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Jonathan Tanner | CSR |
| Harish Balasubramaniam | Intel |
| Marcel Holtmann | Intel |
| Yao Wang | IVT Corporation |
| Frank Berntsen | Nordic Semiconductor |
| David Engelien-Lopes | Nordic Semiconductor |
| Rasmus Abildgren | Samsung Electronics |

| | |
|---|---|
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jason Hillyard | Wicentric |

*Version 4.1*

| | |
|---|---|
| Angel Polo | Broadcom |
| Mayank Batra | CSR |
| Joe Decuir | CSR |
| Giriraj Goyal | CSR |
| Robin Heydon | CSR |
| Neil Stewart | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Yao Wang | IVT Corporation |
| Rajan S Pallathu | MindTree Ltd. |
| David Lopes | Nordic Semiconductor |
| Joel Linsky | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics |
| Anthony Viscardi | Texas Instruments |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Steve Davies | Nokia |
| Joel Linsky | Qualcomm |
| Geert Sonck | ST Ericsson |

### 2.7.5 Part E: Low Energy Security Specification

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |

| | |
|---|---|
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |
| Robert Hulvey | Cypress Semiconductor Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Iman Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |

### *Version 4.2*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |

| | |
|---|---|
| Mayank Batra | CSR |
| Robin Heydon | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Clive D.W. Feather | Samsung Electronics |
| Ed Callaway | Sunrise Micro Devices |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Robin Heydon | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Mika Kasslin | Nokia |
| Joel Linsky | Qualcomm |
| Geert Sonck | ST Ericsson |
| Alon Paycher | Texas Instruments |

### 2.7.6 Part F: Direct Test Mode

*Version 5.1*

| | |
|---|---|
| Jason Hillyard | ARM Ltd |
| Stephen Coe | BlackBerry Limited |
| Angel Polo | Broadcom Corporation |
| Raja Banerjea | CSR |
| Mayank Batra | CSR |
| Ian Blair | CSR |
| Robin Heydon | CSR |
| Sabita Nahata | CSR |
| Neil Stewart | CSR |
| Jonathan Tanner | CSR |

| | |
|---|---|
| Robert Hulvey | Cypress Semiconductor Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Seung R. Yang | LG Electronics Inc. |
| Josselin de la Broise | Marvell Technology Group Ltd |
| Lichun Ko | Mediatek Inc. |
| Shwetha Mahadik | MindTree Limited |
| Juha Salokannel | Nokia Corporation |
| Jari Syrjärinne | Nokia Corporation |
| Jiao Xianjun | Nokia Corporation |
| Zhang Xin | Nokia Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Daphne Liu | Realsil Microelectronics Inc |
| Jean-Philippe Lambert | RivieraWaves |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Iman Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Tomas Motos Lopez | Texas Instruments Incorporated |

*Version 5.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Angel Polo | Broadcom |
| Raja Banerjea | CSR |
| Sandipan Kundu | CSR |
| Dishant Srivastava | CSR |
| Yi-Ling Chao | Marvell |
| KC Chou | MediaTek |

| | |
|---|---|
| James Wang | MediaTek |
| Thomas Varghese | Mindtree |
| Phil Corbishley | Nordic Semiconductor ASA |
| David Engelien-Lopes | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Bjarne Klemmensen | Oticon A/S |
| Niclas Granqvist | Polar |
| RaviKiran Gopalan | Qualcomm Atheros |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Robin Heydon | Qualcomm Technologies, Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Jean-Philippe Lambert | RiveraWaves |
| Clive D.W. Feather | Samsung Electronics Co., Ltd. |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments |
| Tomás Motos López | Texas Instruments |
| Anthony Viscardi | Texas Instruments |
| Micheal Ungstrup | Widex |

*Version 4.2*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Phil Hough | Anritsu |
| Mayank Batra | CSR |
| Robin Heydon | CSR |
| Harish Balasubramaniam | Intel |
| Magnus Eriksson | Intel |
| Marcel Holtmann | Intel |
| Clive D.W. Feather | Samsung Electronics |
| Ed Callaway | Sunrise Micro Devices |
| Joel Linsky | Qualcomm Atheros |
| Brian A. Redding | Qualcomm Atheros |
| Jean-Philippe Lambert | RivieraWaves |
| Jason Hillyard | Wicentric |

*Version 4.0*

| | |
|---|---|
| Edward Harrison | Anritsu |
| Robert Hulvey | Broadcom |
| Magnus Sommansson | CSR |
| Robin Heydon | CSR |
| Steven Wenham | CSR |
| Steve Davies | Nokia |
| Frank Karlsen | Nordic Semiconductor |
| Joel Linsky | Qualcomm |
| Helge Coward | Texas Instruments |

## 2.8 [Vol 7] Wireless Coexistence Signaling and Interfaces

### 2.8.1 Part A: MWS Coexistence Logical Signaling Specification

*Version 4.1*

| | |
|---|---|
| Knut Odman | Broadcom |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Joe Decuir | CSR |
| Leif Wilhelmsson | Ericsson |
| Sharon Yang | Intel |
| Josselin de la Broise | Marvell |
| Huanchun Ye | MediaTek |
| L. C. Ko | MediaTek |
| Joel Linsky | Qualcomm Atheros |
| Olaf Hirsch | Qualcomm Atheros |
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |

### 2.8.2 Part B: Wireless Coexistence Interface 1 (WCI-1) Transport Specification

*Version 4.1*

| | |
|---|---|
| Steven Hall | Broadcom |
| Joe Decuir | CSR |

| | |
|---|---|
| Clive D.W. Feather | CSR |
| Sharon Yang | Intel |
| Huanchun Ye | MediaTek |
| L. C. Ko | MediaTek |
| Aaron Hsieh | MediaTek |
| Joel Linsky | Qualcomm Atheros |
| Olaf Hirsch | Qualcomm Atheros |
| Alon Cheifetz | Texas Instruments |

### 2.8.3 Part C: Wireless Coexistence Interface 2 (WCI-2) Transport Specification

*Version 4.1*

| | |
|---|---|
| Knut Odman | Broadcom |
| Shawn Ding | Broadcom |
| Steven Hall | Broadcom |
| Joe Decuir | CSR |
| Clive D.W. Feather | CSR |
| Leif Wilhelmsson | Ericsson |
| Sharon Yang | Intel |
| Josselin De La Broise | Marvell |
| Huanchun Ye | MediaTek |
| Joel Linsky | Qualcomm |
| Olaf Hirsch | Qualcomm |
| Yves Wernaers | ST-Ericsson |
| Alon Cheifetz | Texas Instruments |

# 3 ACKNOWLEDGMENTS FOR V5.2

## 3.1 Acknowledgments for LE Isochronous Channels

| | |
|---|---|
| Jiawei Chen | ARM Ltd |
| Jason Hillyard | ARM Ltd |
| John Yi | ARM Ltd |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Rangineni Balasubramanyam | Cypress Semiconductor Corporation |
| Rohit Gupta | Cypress Semiconductor Corporation |
| Sachinakumar Hottigoudar | Cypress Semiconductor Corporation |
| Mohan Mysore | Cypress Semiconductor Corporation |
| Nick Hunn | GN Hearing A/S |
| Thorkild Pedersen | GN Hearing A/S |
| Marcel Vlaming | GN Hearing A/S |
| Liu Huazhang | Hisilicon Technologies Co., Ltd. |
| Harish Balasubramaniam | Intel Corporation |
| Himanshu Bhalla | Intel Corporation |
| Magnus Eriksson | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Robert D Hughes | Intel Corporation |
| Xuemei Sherry Ouyang | Intel Corporation |
| Balvinder Pal Singh | Intel Corporation |
| Luiz Von Dentz | Intel Corporation |
| L.C. Ko | MediaTek |
| Shwetha Mahadik | Mindtree Limited |
| Håkon Amundsen | Nordic Semiconductor ASA |
| Pål Håland | Nordic Semiconductor ASA |
| Eivind Sjøgren Olsen | Nordic Semiconductor ASA |
| Daniel Ryan | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP Semiconductors |
| Chris Deck | ON Semiconductor |

| | |
|---|---|
| Bjarne Klemmensen | Oticon A/S |
| John Yi | Packetcraft, Inc. |
| Erwin Weinans | Plantronics Inc. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian Redding | Qualcomm Technologies, Inc. |
| Mayank Batra | Qualcomm Technologies International Ltd. |
| Robin Heydon | Qualcomm Technologies International Ltd. |
| Jonathan Tanner | Qualcomm Technologies International, Ltd. |
| Frederic Belouin | RivieraWaves |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Ole Heftholm-Jensen | Samsung Electronics Co., Ltd. |
| Robert Bäuml | Sivantos GmbH |
| Riccardo Cavallari | Sivantos GmbH |
| Joerg Lebender | Sivantos GmbH |
| Stephan Gehring | Sonova AG |
| Andrew Estrada | Sony Corporation |
| Jeff Solum | Starkey Hearing Technologies |
| Latifa Ali | Synopsys, Inc. |
| Khaled Ali | Synopsys, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Karim Mokhtar | Synopsys, Inc. |
| Mahmoud Samy | Synopsys, Inc. |
| Ahmed Shawky | Synopsys, Inc. |
| Florian Lefeuvre | Texas Instruments Incorporated |
| Kanji Kerai | Widex A/S |
| Søren Møllskov Larsen | Widex A/S |
| Michael Ungstrup | Widex A/S |

## 3.2 Acknowledgments for LE Power Control

| | |
|---|---|
| Edward Harrison | Anritsu Corporation |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |

| | |
|---|---|
| John Penn | Cypress Semiconductor Corporation |
| Mesh Davaraj | Dialog Semiconductor B.V. |
| Harish Balasubramaniam | Intel Corporation |
| Magnus Eriksson | Intel Corporation |
| Rajan S Pallathu | Mindtree Limited |
| Sandeep Choudhary | Nordic Semiconductor ASA |
| Pål Håland | Nordic Semiconductor ASA |
| Bjarne Klemmensen | Oticon A/S |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Chris Church | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Joel Linsky | Qualcomm Technologies, Inc. |
| Brian Redding | Qualcomm Technologies, Inc. |
| Jean Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Khaled Elsayed | Synopsys, Inc. |

## 3.3 Acknowledgments for Enhanced Attribute Protocol

| | |
|---|---|
| Morteza Rahchamani | ARM Ltd |
| Rasmus Abildgren | Bose Corporation |
| Yogesh Ulhas Kamat Mhamai | Cypress Semiconductor Corporation |
| Himanshu Bhalla | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Robert D Hughes | Intel Corporation |
| Balvinder Pal Singh | Intel Corporation |
| Luiz Von Dentz | Intel Corporation |
| Shwetha Mahadik | Mindtree Limited |
| Pål Håland | Nordic Semiconductor ASA |
| Chris Deck | ON Semiconductor |
| Scott Walsh | Plantronics Inc. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Frederic Belouin | RivieraWaves |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

| | |
|---|---|
| Rasmus Abildgren | Samsung Electronics Co., Ltd. |
| Andrew Estrada | Sony Corporation |
| Masahiko Seki | Sony Corporation |
| Jeff Solum | Starkey Hearing Technologies |
| Florian Lefeuvre | Texas Instruments Incorporated |

# 4 ACKNOWLEDGMENTS FOR V5.3

## 4.1 Acknowledgments for AdvDataInfo in Periodic Advertising

| | |
|---|---|
| Shawn Ding | Broadcom Corporation |
| Bjarne Klemmensen | Demant A/S |
| Harish Balasubramaniam | Intel Corporation |
| Chris Deck | ON Semiconductor |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Kanji Kerai | Sivantos GmbH |
| Jeff Solum | Starkey Hearing Technologies |

## 4.2 Acknowledgments for Host To Controller Encryption Key Control Enhancements

| | |
|---|---|
| Harish Balasubramaniam | Intel Corporation |
| Erik Peterson | Microsoft Corporation |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

## 4.3 Acknowledgments for LE Enhanced Connection Update

| | |
|---|---|
| Rasmus Abildgren | Bose Corporation |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Per Skillermark | Ericsson AB |
| Harish Balasubramaniam | Intel Corporation |
| Magnus Eriksson | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Pål Håland | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP Semiconductors |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Jean-Phillippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

| | |
|---|---|
| Kanji Kerai | Sivantos GmbH |
| Florian Lefeuvre | Skyworks Solutions Inc. |
| Jeff Solum | Starkey Hearing Technologies |

## 4.4 Acknowledgments for LE Channel Classification

| | |
|---|---|
| Shawn Ding | Broadcom Corporation |
| Mesh Davaraj | Dialog Semiconductor B.V. |
| Harish Balasubramaniam | Intel Corporation |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Jean-Philippe Lambert | RivieraWaves |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

## 4.5 Acknowledgments for Removing Alternate MAC/PHY

| | |
|---|---|
| Angel Polo | Broadcom Corporation |
| Kanji Kerai | Dialog Semiconductor B.V. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

# 5 ACKNOWLEDGMENTS FOR V5.4

## 5.1 Coding Scheme Selection on Advertising

| | |
|---|---|
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Rob Davies | Signify Netherlands B.V. |
| Nancy Lee | Signify Netherlands B.V. |
| Luca Zappaterra | Signify Netherlands B.V. |

## 5.2 Encrypted Advertising Data

| | |
|---|---|
| Bjarne Klemmensen | Demant A/S |
| Mesh Davaraj | Dialog Semiconductor B.V. |
| Kanji Kerai | Dialog Semiconductor B.V. |
| Dong Jianli | GuangDong Oppo Mobile Telecommunications Corp., Ltd. |
| Pouria Zand | Infineon Technologies AG |
| Pål Håland | Nordic Semiconductor ASA |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Lauri Hintsala | Silicon Laboratories, Inc. |

## 5.3 Periodic Advertising with Responses

| | |
|---|---|
| Bjarne Klemmensen | Demant A/S |
| Kanji Kerai | Dialog Semiconductor B.V. |
| Fabien Duvoux | Ellisys |
| Dong Jianli | GuangDong Oppo Mobile Telecommunications Corp., Ltd. |
| Balasubramanyam Rangineni | Infineon Technologies AG |
| Pouria Zand | Infineon Technologies AG |
| Pål Håland | Nordic Semiconductor ASA |
| Mayank Batra | Qualcomm Technologies International, Ltd. |

| | |
|---|---|
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Lauri Hintsala | Silicon Laboratories, Inc. |

## 5.4 LE GATT Security Levels Characteristic

| | |
|---|---|
| Erik Moll | Koninklijke Philips N.V. |
| Rob Hulvey | Meta Platforms Inc. |

# 6 ACKNOWLEDGMENTS FOR V6.0

## 6.1 Decision-Based Advertising Filtering

| | |
|---|---|
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Electronics |
| Robert Lubas | Silvair, Inc. |
| Szymon Slupik | Silvair, Inc. |
| Danilo Blasi | STMicroelectronics |

## 6.2 Channel Sounding

| | |
|---|---|
| Daisuke Takai | AlpsAlpine |
| Hongyuan Chen | AlpsAlpine |
| Yann Ly-Gagnon | Apple |
| Angel Polo | Broadcom Corporation |
| Irene Fan | Broadcom Corporation |
| Kyle Qian | Broadcom Corporation |
| Daniel Lee | Denso |
| Kyle Golsch | Denso |
| Mike Stitt | Denso |
| Kanji Kerai | Dialog Semiconductor B.V. |
| Mesh Davaraj | Dialog Semiconductor B.V. |
| Jonathan Berner | E.G.O. Elektro-Gerätebau GmbH |
| Clement Vacheron | Ellisys |
| Fabien Duvoux | Ellisys |
| Kyle Penri-Williams | Ellisys |
| Jac Romme | IMEC |
| Claudio Rey | Infineon Technologies AG |
| James Wihardja | Infineon Technologies AG |
| Pouria Zand | Infineon Technologies AG |
| Harish Balasubramaniam | Intel Corporation |
| Ilan Sutskover | Intel Corporation |
| Arne Bestmann | Lambda:4 Entwicklungen GmbH |

| | |
|---|---|
| Rönne Reimann | Lambda:4 Entwicklungen GmbH |
| Yi-Ling Chao | Marvell Semiconductor, Inc. |
| Carsten Wulff | Nordic Semiconductor ASA |
| Daniel Ryan | Nordic Semiconductor ASA |
| Jan Müller | Nordic Semiconductor ASA |
| Johan Stridkvist | Nordic Semiconductor ASA |
| Pål Håland | Nordic Semiconductor ASA |
| Luc Revardel | NXP Semiconductors |
| Mihai-Ionut Stanciu | NXP Semiconductors |
| Olivier Jean | NXP Semiconductors |
| Kai Ren | Oppo |
| Andrew Fort | Qorvo |
| Mohit Maheshwari | Qualcomm Technologies, Inc. |
| Naveen Kumar Reddy Kanakanti | Qualcomm Technologies, Inc. |
| Ram Mohan Korukonda | Qualcomm Technologies, Inc. |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Paul Hiscock | Qualcomm Technologies International, Ltd. |
| Yi-Lin Li | Realsil |
| Peter Ford | RF Creations |
| Jean-Philippe Lambert | RivieraWaves |
| Kenton Payne | Rohde Schwarz |
| Allan Madsen | Samsung Electronics |
| Casper Madsen | Samsung Electronics |
| Clive D.W. Feather | Samsung Electronics |
| Fei Tong | Samsung Electronics |
| Jacob Sharpe | Samsung Electronics |
| Kim Schulz | Samsung Electronics |
| Trine Jensen | Samsung Electronics |
| Nancy Lee | Signify Netherlands B.V. |
| Guner Arslan | Silicon Laboratories, Inc. |
| Lauri Hintsala | Silicon Laboratories, Inc. |
| Michael Wu | Silicon Laboratories, Inc. |
| Khaled Elsayed | Synopsys, Inc. |
| Oystein Bjorndal | Texas Instruments Incorporated |

| | |
|---|---|
| Tomas Motos | Texas Instruments Incorporated |
| Josselin de la Broise | Tile |
| Joel Pature | Valeo |

## 6.3 Enhancements for ISOAL

| | |
|---|---|
| Shawn Ding | Broadcom Corporation |
| Bjarne Klemmensen | Demant A/S |
| Nick Hunn | GN Hearing A/S |
| Harish Balasubramaniam | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Sam Geeraerts | NXP Semiconductors |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Electronics |
| Riccardo Cavallari | Sivantos GmbH |
| Andrew Estrada | Sony Corporation |
| Jeff Solum | Starkey Hearing Technologies |
| Kanji Kerai | Widex A/S |

## 6.4 LL Extended Feature Set

| | |
|---|---|
| Clive D.W. Feather | Samsung Electronics |

## 6.5 Monitoring Advertisers

| | |
|---|---|
| Siegfried Lehmann | Apple Inc. |
| Nick Hunn | GN Hearing A/S |
| Marcel Holtmann | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Frank Yerrace | Microsoft Corporation |
| Rahul Ramadas | Microsoft Corporation |
| Pål Håland | Nordic Semiconductor ASA |
| Chris Deck | ON Semiconductor |
| Robin Heydon | Qualcomm Technologies International, Ltd. |
| Kim Schulz | Samsung |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

| | |
|---|---|
| Henrik Holbaek Pedersen | Samsung Electronics Co., Ltd. |
| Georg Dickmann | Sonova AG |
| Jeff Solum | Starkey Hearing Technologies |
| Soeren Moellskov Larsen | WS Audiology Denmark A/S |

## 6.6 Frame Space Update

| | |
|---|---|
| Angel Polo | Broadcom Corporation |
| Victor Zhodzishsky | Infineon Technologies AG |
| Oren Haggai | Intel Corporation |
| Robert Hulvey | Meta Platforms, Inc. |
| Carsten Wulff | Nordic Semiconductor ASA |
| Frank Berntsen | Nordic Semiconductor ASA |
| Pål Håland | Nordic Semiconductor ASA |
| Bjarne Klemmensen | Oticon A/S |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Kim Schulz | Samsung Electronics Co., Ltd. |
| Noureldin Mohamed | Synopsys, Inc. |

## 6.7 Core Configurations

| | |
|---|---|
| Alicia Courtney | Broadcom |
| Shirin Ebrahimi-Taghizadeh | Microsoft Corporation |
| Miles Smith | Nordic Semiconductor ASA |
| Magnus Sommansson | Qualcomm Technologies International, Ltd. |
| Clive D.W. Feather | Samsung Electronics |

# 7 ACKNOWLEDGMENTS FOR V6.1

## 7.1 Randomized RPA Updates

| | |
|---|---|
| Hai Shalom | Google LLC |

# 8 ACKNOWLEDGMENTS FOR V6.2

## 8.1 LE Test Mode Enhancements

| | |
|---|---|
| Edward Harrison | Anritsu |
| Angel Polo | Broadcom |
| Ravi Kiran Bamidi | Infineon Technologies |
| Pål Håland | Nordic Semiconductor ASA |
| Brevalan Ung | NXP |
| Mayank Batra | Qualcomm Technologies International, Ltd. |
| Mark Bale | Rohde & Schwarz Services Ltd |
| Brian Glover | Rohde & Schwarz Services Ltd |
| Kenton Payne | Rohde & Schwarz Services Ltd |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

## 8.2 Shorter Connection Intervals

| | |
|---|---|
| Oren Haggai | Apple Inc. |
| Siegfried Lehman | Apple Inc. |
| Gene Chang | Bluetooth SIG, Inc. |
| Shawn Ding | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Kyle Penri-Williams | Ellisys |
| Alain Michaud | Google LLC |
| Pouria Zand | Google LLC |
| Rangineni Balasubramanyam | Infineon Technologies AG |
| Victor Zhodzishsky | Infineon Technologies AG |
| Harish Balasubramaniam | Intel Corporation |
| Yash Devadiga | Intel Corporation |
| Robert Hughes | Intel Corporation |
| HJ Lee | LG Electronics Inc. |
| Xavier Boniface | Logitech International SA |
| Niclas Granqvist | Logitech International SA |
| Rob Hulvey | Meta Platforms, Inc. |

| | |
|---|---|
| Kanji Kerai | Meta Platforms, Inc |
| Shirin Ebrahimi-Taghizadeh | Microsoft Corporation |
| Erik Peterson | Microsoft Corporation |
| Rahul Ramadas | Microsoft Corporation |
| Frank Berntsen | Nordic Semiconductor ASA |
| Sam Geeraerts | NXP B.V. |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Jonathan Tanner | Qualcomm Technologies, Inc. |
| Clive D.W. Feather | Samsung Cambridge Solution Centre |
| Kim Schulz | Samsung Electronics Co., Ltd. |
| Nancy Lee | Signify Netherlands B.V. |
| Khaled Elsayed | Synopsys, Inc. |
| Sihui Wang | Telink Semiconductor |

## 8.3 Channel Sounding Amplitude-based Attack Resilience

| | |
|---|---|
| Angel Polo | Broadcom Corporation |
| Guillaume Rellier | Ceva |
| Kyle Golsch | Denso Corporation |
| Pouria Zand | Google LLC |
| Daniel Ryan | Nordic Semiconductor ASA |
| Mihai-Ionut Stanciu | Infineon Technologies AG |
| Jacob Sharpe | Samsung Cambridge Solution Centre |
| Oystein Bjorndal | Texas Instruments Incorporated |
| Tomas Motos | Texas Instruments Incorporated |

## 8.4 HCI USB LE Isochronous Support

| | |
|---|---|
| Shawn Ding | Broadcom Corporation |
| Harish Balasubramaniam | Intel Corporation |
| Luiz Von Dentz | Intel Corporation |
| Oren Haggai | Intel Corporation |
| Marcel Holtmann | Intel Corporation |
| Erik Peterson | Microsoft Corporation |
| Rahul Ramadas | Microsoft Corporation |

| | |
|---|---|
| Pål Håland | Nordic Semiconductor ASA |
| Mayank Batra | Qualcomm Technologies, Inc. |

# 9 ACKNOWLEDGMENTS FOR V6.3

## 9.1 Running Out of Bits

| | |
|---|---|
| Clive D.W. Feather | Samsung Cambridge Solution Centre |

## 9.2 Channel Sounding Inline Phase Correction Term Transfer

| | |
|---|---|
| Yann Ly-Gagnon | Apple Inc. |
| Angel Polo | Broadcom Corporation |
| Pouria Zand | Google LLC |
| Narasimha Rao Konaiah | Infineon Technologies AG |
| Mihai Stanciu | Infineon Technologies AG |
| Sheetal Bhasin | Intel Corporation |
| Daniel Ryan | Nordic Semiconductor ASA |
| Jerome Brillant | NXP B.V. Semiconductors |
| Mayank Batra | Qualcomm Technologies, Inc. |
| Jonas Christensen | Samsung Electronics Co., Ltd. |
| Kim Schulz | Samsung Electronics Co., Ltd. |
| Tomas Motos | Texas Instruments Incorporated |

## 9.3 Channel Sounding PHY-specific RTT Accuracy

| | |
|---|---|
| Irene Fan | Broadcom Corporation |
| Angel Polo | Broadcom Corporation |
| Tomas Motos | Texas Instruments Incorporated |

# Consolidated Table Of Contents, Acknowledgments, & Core Configurations Part D

# CORE CONFIGURATIONS

*This volume specifies Core Configurations for implementations based on the Bluetooth Core Specification.*

# CONTENTS

**1 Introduction ..... 209**

**2 Core Configurations ..... 210**

2.1 Core-Controller Configuration ..... 210

2.1.1 BR/EDR Core-Controller Configuration ..... 210

2.1.2 LE Core-Controller Configuration ..... 211

2.1.3 BR/EDR/LE Core-Controller Configuration ..... 211

2.1.4 [This section is no longer used] ..... 212

2.1.5 [This section is no longer used] ..... 212

2.2 Core-Host Configurations ..... 212

2.2.1 BR/EDR Core-Host Configuration ..... 212

2.2.2 LE Core-Host Configuration ..... 212

2.2.3 BR/EDR/LE Core-Host Configuration ..... 213

2.2.4 [This section is no longer used] ..... 213

2.2.5 [This section is no longer used] ..... 213

2.3 Core-Complete Configurations ..... 213

2.3.1 BR/EDR Core-Complete Configuration ..... 213

2.3.2 LE Core-Complete Configuration ..... 213

2.3.3 BR/EDR/LE Core-Complete Configuration ..... 214

2.3.4 [This section is no longer used] ..... 214

2.3.5 [This section is no longer used] ..... 214

2.4 Other Core layers ..... 214

**3 Mixing requirements from different versions ..... 215**

**4 Features and their types ..... 216**

**5 Core Specification Addenda ..... 220**

# 1 INTRODUCTION

This Part covers Core Configurations and related concepts for implementations based on the Core Specification.

A layer is one of the parts of this specification specified in the tables in Section 2. A Controller layer is any layer listed in Table 2.1 or Table 2.2. A Host layer is any layer listed in Table 2.3 or Table 2.4.

For some purposes in this Part, HCI is treated as if it were two separate layers, one only including the Upper HCI role and the other only including the Lower HCI role.

An implementation contains a complete layer if the implementation includes, for that layer, all mandatory features, all the mandatory elements of any optional feature that is included, and any features that are conditionally required. This also applies to sub-features or elements of a feature. For example, if the layer includes a requirement such as “if feature A is implemented, then feature B shall be implemented”, then any implementation of the layer that includes feature A but not all the elements of feature B is not complete.

An implementation of Bluetooth wireless technology based on the Core Specification shall include one or more complete layers of the Core Specification and may also implement one or more Bluetooth specifications that are external to the Core Specification. Such an implementation shall include all features required by dependencies between any layers included in the implementation.

Each capability of this specification shall be supported in the specified manner. This specification may provide options for design flexibility, because, for example, some products do not implement every portion of the specification. For each implementation option that is supported, it shall be supported as specified.

Using such an implementation to enable Bluetooth wireless technology end-to-end might require further integration before distribution.

# 2 CORE CONFIGURATIONS

A Core Configuration determines the transport and layers that an implementation shall include and any inter-layer dependency requirements that apply, as specified in other Parts of this specification.

Each implementation of a Core Configuration shall include the specific complete layers listed in the sub-section of Section 2 corresponding to that configuration. The implementation may also include other complete layers of the Core Specification and may also implement one or more Bluetooth specifications that are external to the Core Specification. It shall also implement all features required by dependencies between any layers included in the implementation. An implementation that includes all the layers of more than one Core Configuration is deemed to be an implementation of the Core Configuration that has the most layers and not of any other Core Configuration. For example, an implementation that includes all the layers in Table 2.1 and Table 2.2 implements the BR/EDR/LE Core-Controller Configuration and not either the BR/EDR Core-Controller Configuration or the LE Core-Controller Configuration.

Bluetooth wireless technology shall only be enabled end-to-end in an implementation of a Core-Complete Configuration.

## 2.1 Core-Controller Configuration

An implementation of a Core-Controller Configuration shall include Controller layers of this specification and the Lower HCI role of HCI and shall not include any Host layer (including the Upper HCI role of HCI) of this specification.

An implementation of a Core-Controller Configuration may also include one or more codec specifications implemented in the Controller. For example, an implementation of a Core-Controller Configuration may also support the LC3 codec implemented in the Controller.

An implementation of a Core-Controller Configuration shall not include any other layers external to the Core Specification.

### 2.1.1 BR/EDR Core-Controller Configuration

An implementation of the BR/EDR Core-Controller Configuration shall include the layers specified in Table 2.1.

| Layer | Reference | Layer requirement |
|---|---|---|
| HCI | [Vol 4] Part E | Mandatory (Lower HCI role only) |
| SEC | [Vol 2] Part H | Mandatory |
| LMP | [Vol 2] Part C | Mandatory |
| BB | [Vol 2] Part B | Mandatory |
| RF | [Vol 2] Part A | Mandatory |

*Table 2.1: BR/EDR Core-Controller Configuration layers*

### 2.1.2 LE Core-Controller Configuration

An implementation of the LE Core-Controller Configuration shall include the layers specified in Table 2.2.

| Layer | Reference | Layer requirement |
|---|---|---|
| HCI | [Vol 4] Part E | Mandatory (Lower HCI role only) |
| ISOAL | [Vol 6] Part G | C.2 |
| CS | [Vol 6] Part H | C.3 |
| LESEC | [Vol 6] Part E | C.1 |
| LL | [Vol 6] Part B | Mandatory |
| RFPHY | [Vol 6] Part A | Mandatory |

C.1: Mandatory if LE Encryption feature (see [Vol 6] Part B, Section 4.6.1) is supported, otherwise excluded.

C.2: Mandatory if the Link Layer supports any of the following features, otherwise excluded:

- Connected Isochronous Stream – Central (see [Vol 6] Part B, Section 4.6.27)
- Connected Isochronous Stream – Peripheral (see [Vol 6] Part B, Section 4.6.27)
- Isochronous Broadcaster (see [Vol 6] Part B, Section 4.6.28)
- Synchronized Receiver (see [Vol 6] Part B, Section 4.6.29)

C.3: Optional if LE Channel Sounding feature (see [Vol 6] Part B, Section 4.6.41) is supported, otherwise excluded.

*Table 2.2: LE Core-Controller Configuration layers*

### 2.1.3 BR/EDR/LE Core-Controller Configuration

An implementation of the BR/EDR/LE Core-Controller Configuration shall include the layers specified in Table 2.1 and Table 2.2.

### 2.1.4 [This section is no longer used]

### 2.1.5 [This section is no longer used]

## 2.2 Core-Host Configurations

An implementation of a Core-Host Configuration shall include Host layers of the Core Specification and the Upper HCI role of HCI and shall not include any Controller layer (including the Lower HCI role of HCI) of the Core Specification.

An implementation of a Core-Host Configuration may also include one or more Bluetooth specifications that are external to the Core Specification, other than those implemented in the Controller. For example, an implementation of a Core-Host Configuration may also support A2DP, BAS, HOGP, or LC3 implemented in the Host.

### 2.2.1 BR/EDR Core-Host Configuration

An implementation of the BR/EDR Core-Host Configuration shall include the layers specified in Table 2.3.

| Layer | Reference | Layer requirement |
|---|---|---|
| GATT | [Vol 3] Part G | C.1 |
| ATT | [Vol 3] Part F | Optional |
| GAP | [Vol 3] Part C | Mandatory |
| SDP | [Vol 3] Part B | Mandatory |
| L2CAP | [Vol 3] Part A | Mandatory |
| HCI | [Vol 4] Part E | Mandatory (Upper HCI role only) |
| C.1: Mandatory if ATT is included, otherwise excluded. | | |

*Table 2.3: BR/EDR Core-Host Configuration layers*

### 2.2.2 LE Core-Host Configuration

An implementation of the LE Core-Host Configuration shall include the layers specified in Table 2.4.

| Layer | Reference | Layer requirement |
|---|---|---|
| GATT | [Vol 3] Part G | C.1 |
| ATT | [Vol 3] Part F | C.1 |
| GAP | [Vol 3] Part C | Mandatory |
| SM | [Vol 3] Part H | C.1 |

| Layer | Reference | Layer requirement |
|---|---|---|
| L2CAP | [Vol 3] Part A | C.1 |
| HCI | [Vol 4] Part E | Mandatory (Upper HCI role only) |
| C.1: Mandatory if either the GAP Central role or the GAP Peripheral role is supported, otherwise excluded. | | |

*Table 2.4: LE Core-Host Configuration layers*

### 2.2.3 BR/EDR/LE Core-Host Configuration

An implementation of the BR/EDR/LE Core-Host Configuration shall include the layers specified in Table 2.3 and Table 2.4.

### 2.2.4 [This section is no longer used]

### 2.2.5 [This section is no longer used]

## 2.3 Core-Complete Configurations

An implementation of a Core-Complete Configuration may also include one or more Bluetooth specifications that are external to the Core Specification. For example, an implementation of a Core-Complete Configuration may also support A2DP, BAS, HOGP, or LC3.

An implementation of a Core-Complete Configuration may omit either or both roles of the HCI layer as specified in Table 2.1, Table 2.2, Table 2.3, and Table 2.4, except as stated in the next paragraph.

If an implementation of a Core-Complete Configuration is created by combining an implementation of a Core-Host Configuration with an implementation of a compatible Core-Controller Configuration, then the resulting implementation shall use HCI ([Vol 4] Part E) for communication between the Host and the Controller.

### 2.3.1 BR/EDR Core-Complete Configuration

An implementation of the BR/EDR Core-Complete Configuration shall include the layers specified in Table 2.1 and Table 2.3.

### 2.3.2 LE Core-Complete Configuration

An implementation of the LE Core-Complete Configuration shall include the layers specified in Table 2.2 and Table 2.4.

### 2.3.3 BR/EDR/LE Core-Complete Configuration

An implementation of the BR/EDR/LE Core-Complete Configuration shall include the layers specified in Table 2.1, Table 2.2, Table 2.3, and Table 2.4.

### 2.3.4 [This section is no longer used]

### 2.3.5 [This section is no longer used]

## 2.4 Other Core layers

An implementation may include any of the additional layers specified in Table 2.5.

Note: These layers are neither Host nor Controller layers and are not associated with a specific transport or Core Configuration.

| Layer | Reference | Layer requirement |
|---|---|---|
| HCI-UART | [Vol 4] Part A | Optional |
| HCI-USB | [Vol 4] Part B | Optional |
| HCI-SD | [Vol 4] Part C | Optional |
| HCI-3W | [Vol 4] Part D | Optional |
| DTM | [Vol 6] Part F | Optional |
| MWS | [Vol 7] Part A | Optional |
| WCI-1 | [Vol 7] Part B | Optional |
| WCI-2 | [Vol 7] Part C | Optional |

*Table 2.5: Other Core layers*

# 3 MIXING REQUIREMENTS FROM DIFFERENT VERSIONS

Subject to the requirements of this section, an implementation may mix layers that implement different versions of this specification.

All Controller layers (including Lower HCI) that are included shall implement a single version of this specification (the “Controller Version”).

All Host layers that are included (excluding Upper HCI) shall implement a single version of this specification (the “Host Version”), which may be a different version from the Controller Version.

If an implementation includes Upper HCI, then the version of Upper HCI implemented (the “Upper HCI Version”) shall be the same version or a higher version than the Host Version.

For the purposes of this section, a version of this specification mixed with one or more Core Specification Addenda (see Section 5) is the same as that version of this specification without any Core Specification Addenda.

# 4 FEATURES AND THEIR TYPES

Features in this specification are divided into the types listed in Table 4.1.

| Type | Description |
|---|---|
| Type 1 | Controller feature that cannot be configured/enabled by the Host via HCI |
| Type 2 | Controller feature that can be configured/enabled by the Host via HCI |
| Type 3 | Feature that exists in both the Controller and the Host and requires HCI commands/events to function |
| Type 4 | Host feature that does not involve the Controller |

*Table 4.1: Feature type definitions*

The behavior of a feature in an implementation of a Core-Complete Configuration depends on the feature type:

- A feature of type 1 that is specified in the Controller Version can function irrespective of the Host Version or Upper HCI Version.
- A feature of type 2 that is specified in the Controller Version can function if the Upper HCI Version is the same or a newer version than the Controller Version. Otherwise, the feature will be restricted to behavior (if any) that does not require the Host to configure/enable it; the available behavior will depend on the feature.
- A feature of type 3 can only function if it is specified in both the Controller Version and the Host Version.
- A feature of type 4 that is specified in the Host Version can function irrespective of the Controller Version or Upper HCI Version.

Table 4.2 lists the features added in versions 1.2 and higher of this specification, their types, and the version or addendum where the feature was first introduced.

| Feature | Version | Type |
|---|---|---|
| Basic AFH operation | 1.2 | 1 |
| Enhanced inquiry | 1.2 | 1 |
| Configuration of AFH (setting channels and enabling/disabling channel assessment) | 1.2 | 2 |
| Enhanced synchronization capability | 1.2 | 2 |
| Interlaced inquiry scan | 1.2 | 2 |
| Interlaced page scan | 1.2 | 2 |
| Broadcast encryption | 1.2 | 2 |

| Feature | Version | Type |
|---|---|---|
| Enhanced flow specification and flush time-out | 1.2 | 3 |
| Extended SCO links | 1.2 | 3 |
| Inquiry Result with RSSI | 1.2 | 3 |
| L2CAP flow and error control | 1.2 | 4 |
| 2 Mb/s EDR | 2.0 + EDR | 2 |
| 3 Mb/s EDR | 2.0 + EDR | 2 |
| 3 slot packets in EDR | 2.0 + EDR | 2 |
| 5 slot packets in EDR | 2.0 + EDR | 2 |
| 2 Mb/s eSCO | 2.0 + EDR | 2 |
| 3 Mb/s eSCO | 2.0 + EDR | 2 |
| 3 slot packets for EDR eSCO | 2.0 + EDR | 2 |
| Erroneous Data Reporting | 2.1 + EDR | 2 |
| Extended Inquiry Response | 2.1 + EDR | 3 |
| Encryption Pause and Resume | 2.1 + EDR | 1 |
| Link Supervision Timeout Changed Event | 2.1 + EDR | 2 |
| Non-Flushable Packet Boundary Flag | 2.1 + EDR | 3 |
| Sniff subrating | 2.1+ EDR | 2 |
| Secure Simple Pairing | 2.1.+ EDR | 3 |
| L2CAP Enhanced Retransmission Mode | Addendum 1/ 3.0 + HS | 4 |
| L2CAP Streaming Mode | Addendum 1/ 3.0 + HS | 4 |
| Enhanced Power Control | 3.0 + HS | 1 |
| Unicast Connectionless Data | 3.0 + HS | 4 |
| Low Energy Controller (PHY and LL) | 4.0 | 3 |
| Low Energy Host (L2CAP and Security Manager) | 4.0 | 4 |
| Attribute Protocol and Generic Attribute Profile | 4.0 | 4 |
| Appearance Data Type | Addendum 2 | 4 |
| MWS Coexistence Signaling | Addendum 3 | 2 |
| Connectionless Peripheral Broadcast | Addendum 4 | 3 |
| Unencrypted UCD | Addendum 4 | 4 |
| BR/EDR Secure Connections | 4.1 | 3 |
| Train Nudging | 4.1 | 2 |

| Feature | Version | Type |
|---|---|---|
| Generalized Interlaced Scan | 4.1 | 2 |
| Piconet Clock Adjustment | 4.1 | 2 |
| Low Duty Cycle Directed Advertising | 4.1 | 2 |
| 32-bit UUID Support in LE | 4.1 | 4 |
| LE Dual Mode Topology | 4.1 | 4 |
| LE L2CAP Connection Oriented Channel Support | 4.1 | 4 |
| LE Privacy v1.1 | 4.1 | 4 |
| LE Link Layer Topology | 4.1 | 3 |
| LE Ping | 4.1 | 2 |
| LE Data Packet Length Extension | 4.2 | 2 |
| LE Secure Connections | 4.2 | 4 |
| Link Layer Privacy | 4.2 | 3 |
| Link Layer Extended Filter Policies | 4.2 | 3 |
| Slot Availability Mask | 5.0 | 2 |
| LE 2M PHY | 5.0 | 2 |
| LE Coded PHY | 5.0 | 2 |
| High Duty Cycle Non-Connectable Advertising | 5.0 | 2 |
| LE Advertising Extensions | 5.0 | 3 |
| LE Channel Selection Algorithm #2 | 5.0 | 2 |
| LE Higher Output Power | Addendum 5 | 1 |
| Angle of Arrival/Angle of Departure | 5.1 | 2 |
| GATT Caching | 5.1 | 4 |
| Periodic Advertising Sync Transfer | 5.1 | 3 |
| Control Length Extension | 5.1 | 1 |
| Advertising Channel Index | 5.1 | 1 |
| HCI support for debug keys in LE Secure Connections | 5.1 | 2 |
| Sleep clock accuracy update mechanism | 5.1 | 2 |
| ADI field in scan response data | 5.1 | 1 |
| Interaction between QoS and Flow Specification | 5.1 | 2 |
| Host channel classification for secondary advertising | 5.1 | 2 |
| Allow the SID to appear in scan response reports | 5.1 | 1 |
| LE Isochronous Channels | 5.2 | 3 |
| Enhanced Attribute Protocol | 5.2 | 4 |

| Feature | Version | Type |
|---|---|---|
| LE Power Control | 5.2 | 2 |
| Periodic Advertising ADI support | 5.3 | 2 |
| Set Min Encryption Key Size command and Encryption Change [v2] event | 5.3 | 2 |
| Connection Subrating | 5.3 | 3 |
| Channel Classification | 5.3 | 2 |
| Advertising Coding Selection | 5.4 | 2 |
| Encrypted Advertising Data | 5.4 | 4 |
| Periodic Advertising with Responses | 5.4 | 3 |
| LE GATT Security Levels Characteristic | 5.4 | 4 |
| Channel Sounding | 6.0 | 3 |
| Decision-Based Advertising Filtering | 6.0 | 2 |
| ISOAL Unsegmented Framed Mode | 6.0 | 2 |
| Monitoring Advertisers | 6.0 | 2 |
| LE Frame Space Update | 6.0 | 2 |
| LL Extended Feature Set | 6.0 | 2 |
| Randomized RPA Updates | 6.1 | 2 |
| LE Unified Test Protocol | 6.2 | 3 |
| Shorter Connection Intervals | 6.2 | 2 |
| LE Flushable ACL Data | 6.2 | 2 |
| Channel Sounding Amplitude-based Attack Resilience | 6.2 | 3 |
| HCI USB LE Isochronous Support | 6.2 | n/a |
| Channel Sounding Inline PCT Transfer | 6.3 | 2 |
| Channel Sounding PHY-specific RTT Accuracy | 6.3 | 2 |

*Table 4.2: Features and their types*

# 5 CORE SPECIFICATION ADDENDA

A Core Specification Addendum (CSA) contains one or more parts of a single volume, one or more parts in multiple volumes, changes on one or more parts, or a mixture of parts and changes. Addenda are used to add, change, or replace a part of a volume according to the rules in Table 5.1. Each change in a CSA can affect one or more parts of the specification.

| Addendum | Volume and Part or change name | Addition/ Changes/ Replacement | Allowed Versions & Addenda | Mandatory / Optional / Conditional | Type |
|---|---|---|---|---|---|
| 1 | Volume 3, Part A | Replacement | 2.0 + EDR, 2.1 + EDR | O | 4 |
| 2 | Audio Architecture HCI changes | Change | 2.1 + EDR, 3.0 + HS, 4.0 | O | 2 |
| | Audio Architecture USB changes | Change | 2.1 + EDR, 3.0 + HS, 4.0 | O | 2 |
| | LE Limited Discovery Time Changes | Change | 4.0 | C.1 | 4 |
| | EIR and AD Data Types in GAP changes | Change | 4.0 | C.1 | 4 |
| | EIR and AD Data Types Specification | Addition | 4.0 | C.1 | 4 |
| | Volume 5, Part A | Replacement | 3.0 + HS, 4.0 | O | 3 |
| 3 | LE Errata | Change | 4.0 with CSA2 | C.2 | Multiple |
| | GAP Connection Parameters Changes | Change | 4.0 with CSA2 | C.1 | 4 |
| | GAP Authentication and Lost Bond Changes | Change | 4.0 with CSA2 | C.1 | 4 |
| | Common Profile and Services Error Code Range Changes | Change | 4.0 with CSA2 | C.1 | 4 |

| Addendum | Volume and Part or change name | Addition/ Changes/ Replacement | Allowed Versions & Addenda | Mandatory / Optional / Conditional | Type |
|---|---|---|---|---|---|
| | Private Addressing Changes | Change | 4.0 with CSA2 | C.1 | 4 |
| | Dual Mode Addressing Changes | Change | 4.0 with CSA2 | C.3 | 4 |
| | MWS Coexistence Logical Signaling Specification | Addition | 2.1 + EDR, 3.0 + HS, 4.0 with CSA2 | O | 2 |
| | MWS Coexistence HCI | Addition | 2.1 + EDR, 3.0 + HS, 4.0 with CSA2 | C.4 | 2 |
| | Wireless Coexistence Interface 1 (WCI-1) Transport Layer Specification | Addition | 2.1 + EDR, 3.0 + HS, 4.0 with CSA2 | C.4 | 2 |
| | Wireless Coexistence Interface 2 (WCI-2) Transport Layer Specification | Addition | 2.1 + EDR, 3.0 + HS, 4.0 with CSA2 | C.4 | 2 |
| 4 | Connectionless Peripheral Broadcast | Change | 3.0 + HS, 4.0 with CSA3 | O | 3 |
| | Unencrypted UCD | Change | 3.0 + HS, 4.0 with CSA3 | O | 4 |
| | Fast Advertising Interval | Change | 4.0 with CSA3 | C.1 | 4 |
| | eSCO Reserved Slot Clarification | Change | 2.1 + EDR, 3.0 + HS, 4.0 with CSA3 | O | 1 |

| Addendum | Volume and Part or change name | Addition/ Changes/ Replacement | Allowed Versions & Addenda | Mandatory / Optional / Conditional | Type |
|---|---|---|---|---|---|
| 5 | Higher Output Power | Change | 4.0 with CSA3,<br>4.0 with CSA4,<br>4.1, 4.2 | O | 1 |
| 6 | The changes made by CSA6 have all been removed from the Core Specification and so CSA6 is withdrawn. | | | | |

C.1: Mandatory if either the Host Part of the Low Energy Core Configuration or the Host Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.2: Mandatory if either the Host Part of the Low Energy Core Configuration, Controller Part of the Low Energy Core Configuration, Host Part of the Basic Rate and Low Energy Combined Core Configuration, or Controller Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.3: Mandatory if the Host Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.4: Optional if MWS Coexistence Logical Signaling is supported, otherwise Excluded.

*Table 5.1: Adopted specification versions to use with addenda*

# Architecture, Change History, and Conventions

**Specification of the *Bluetooth*® System**

**Volume 1**

**Covered Core Package Version:** v6.3

**Version Date:** 2026-05-05

Bluetooth SIG Proprietary

# Architecture, Change History, And Conventions Part A

# ARCHITECTURE

# CONTENTS

**1 General description ..... 230**
1.1 Overview of BR/EDR operation ..... 231
1.2 Overview of Bluetooth Low Energy operation ..... 233
1.3 [This section is no longer used] ..... 239
1.4 Nomenclature ..... 239

**2 Core system architecture ..... 246**
2.1 Core architectural blocks ..... 250
2.1.1 Host architectural blocks ..... 250
2.1.1.1 Channel manager ..... 250
2.1.1.2 L2CAP resource manager ..... 250
2.1.1.3 Security Manager Protocol ..... 250
2.1.1.4 Attribute Protocol ..... 251
2.1.1.5 [This section is no longer used] ..... 251
2.1.1.6 Generic Attribute Profile ..... 251
2.1.1.7 Generic Access Profile ..... 251
2.1.1.8 Service Discovery Protocol ..... 251
2.1.2 BR/EDR/LE Controller architectural blocks ..... 251
2.1.2.1 Device manager ..... 252
2.1.2.2 Link manager ..... 252
2.1.2.3 Baseband resource manager ..... 252
2.1.2.4 Link Controller ..... 253
2.1.2.5 PHY ..... 253
2.1.2.6 Isochronous Adaptation Layer ..... 253
2.1.2.7 Channel Sounding ..... 253
2.1.3 [This section is no longer used] ..... 254

**3 Transport architecture ..... 255**
3.1 Core traffic bearers ..... 255
3.1.1 Framed data traffic ..... 257
3.1.2 Unframed data traffic ..... 258
3.1.3 Reliability of traffic bearers ..... 259
3.1.3.1 BR/EDR reliability ..... 259
3.1.3.2 LE reliability ..... 260
3.1.3.3 [This section is no longer used] ..... 261
3.2 Transport architecture entities ..... 261
3.2.1 BR/EDR generic packet structure ..... 261
3.2.2 LE generic packet structure ..... 263
3.2.3 LE Channel Sounding generic packet structure and signaling format ..... 267

3.3 Physical channels ..... 268
3.3.1 BR/EDR physical channels ..... 269
3.3.1.1 Basic piconet channel ..... 269
3.3.1.2 Adapted piconet channel ..... 271
3.3.1.3 Inquiry scan channel ..... 271
3.3.1.4 Page scan channel ..... 272
3.3.1.5 Synchronization scan channel ..... 274
3.3.2 LE physical channels ..... 274
3.3.2.1 LE piconet physical channel ..... 275
3.3.2.2 Advertising physical channels ..... 277
3.3.2.3 Periodic physical channel ..... 278
3.3.2.4 LE Isochronous physical channel ..... 279
3.3.2.5 LE Channel Sounding physical channel ..... 281
3.3.3 [This section is no longer used] ..... 282
3.4 Physical links ..... 282
3.4.1 BR/EDR links supported by the basic and adapted piconet physical channels ..... 283
3.4.1.1 Active physical link ..... 283
3.4.1.2 [This section is no longer used] ..... 283
3.4.1.3 Connectionless Peripheral Broadcast physical link ..... 283
3.4.2 BR/EDR links supported by the scanning physical channels ..... 284
3.4.3 LE links supported by the LE physical channels ..... 284
3.4.3.1 Active physical link ..... 284
3.4.3.2 Advertising physical link ..... 285
3.4.3.3 Periodic physical link ..... 285
3.4.3.4 Isochronous physical links ..... 285
3.4.3.5 Channel Sounding physical link ..... 285
3.4.4 [This section is no longer used] ..... 286
3.5 Logical links and logical transports ..... 286
3.5.1 Casting ..... 288
3.5.2 Scheduling and acknowledgment scheme ..... 288
3.5.3 Class of data ..... 289
3.5.4 Logical transports ..... 290
3.5.4.1 BR/EDR asynchronous connection-oriented (ACL) ..... 290
3.5.4.2 BR/EDR synchronous connection-oriented (SCO) ..... 290
3.5.4.3 BR/EDR extended synchronous connection-oriented (eSCO) ..... 291
3.5.4.4 BR/EDR active Peripheral broadcast (APB) 291
3.5.4.5 [This section is no longer used] ..... 292

3.5.4.6 LE asynchronous connection (LE ACL) ..... 292
3.5.4.7 LE advertising broadcast (ADVB) ..... 294
3.5.4.8 Connectionless Peripheral Broadcast (CPB) ..... 294
3.5.4.9 LE periodic advertising ..... 295
3.5.4.10 Connected Isochronous Stream (CIS) ..... 296
3.5.4.11 Connected Isochronous Group (CIG) ..... 296
3.5.4.12 Broadcast Isochronous Stream (BIS) ..... 296
3.5.4.13 Broadcast Isochronous Group (BIG) ..... 297
3.5.5 Logical links ..... 297
3.5.5.1 BR/EDR logical links ..... 297
3.5.5.2 LE logical links ..... 298
3.5.5.3 [This section is no longer used] ..... 300
3.6 L2CAP channels ..... 300
3.7 Isochronous Adaptation Layer (ISOAL) ..... 301
3.8 Power control ..... 301
3.8.1 Power control in BR/EDR ..... 301
3.8.2 Power control in LE ..... 301

**4 Communication topology and operation ..... 302**
4.1 Piconet topology ..... 302
4.1.1 BR/EDR topology ..... 302
4.1.2 LE topology ..... 305
4.2 Operational procedures and modes ..... 307
4.2.1 BR/EDR procedures ..... 307
4.2.1.1 Inquiry (discovering) procedure ..... 307
4.2.1.2 Paging (connecting) procedure ..... 308
4.2.1.3 Connected mode ..... 308
4.2.1.4 Hold mode ..... 309
4.2.1.5 Sniff mode ..... 309
4.2.1.6 [This section is no longer used] ..... 309
4.2.1.7 Role switch procedure ..... 309
4.2.1.8 Enhanced Data Rate ..... 310
4.2.1.9 Connectionless Peripheral Broadcast mode 310
4.2.2 LE procedures ..... 311
4.2.2.1 Device filtering procedure ..... 311
4.2.2.2 Advertising procedure ..... 311
4.2.2.3 Scanning procedure ..... 312
4.2.2.4 Discovering procedure ..... 313
4.2.2.5 Connecting procedure ..... 313
4.2.2.6 Connected mode ..... 314
4.2.2.7 Periodic advertising procedure ..... 315

4.2.2.8 Periodic advertising synchronization procedure ..... 315
4.2.2.9 Periodic advertising synchronized mode ..... 316
4.2.2.10 Decision-based scanning ..... 316
4.2.3 [This section is no longer used] ..... 316

**5 Security overview ..... 317**
5.1 Security architecture ..... 317
5.2 BR/EDR Secure Simple Pairing ..... 319
5.2.1 Security goals ..... 320
5.2.2 Passive eavesdropping protection ..... 320
5.2.3 Man-in-the-middle protection ..... 321
5.2.4 Association models ..... 321
5.2.4.1 Numeric Comparison ..... 322
5.2.4.2 Just Works ..... 322
5.2.4.3 Out of Band ..... 322
5.2.4.4 Passkey Entry ..... 323
5.2.4.5 Association model overview ..... 324
5.3 Secure Connections Only mode ..... 324
5.4 LE security ..... 325
5.4.1 Association models ..... 325
5.4.2 Key generation ..... 325
5.4.3 Encryption ..... 325
5.4.4 [This section is no longer used] ..... 326
5.4.5 Privacy feature ..... 326
5.4.6 Encrypted Advertising Data ..... 327
5.5 [This section is no longer used] ..... 327
5.6 Key generation between BR/EDR and LE physical transports ... 327

**6 Bluetooth application architecture ..... 329**
6.1 Bluetooth profiles ..... 329
6.2 Generic Access Profile ..... 329
6.3 Profile hierarchy ..... 330
6.4 Generic Attribute Architecture ..... 331
6.4.1 Attribute Protocol ..... 331
6.4.2 Generic Attribute Profile ..... 331
6.5 GATT-Based Profile hierarchy ..... 332
6.5.1 Service ..... 333
6.5.2 Included services ..... 334
6.5.3 Characteristic ..... 334
6.6 [This section is no longer used] ..... 334

**7 Coexistence and collocation ..... 335**

7.1 Core features supporting coexistence and collocation ..... 335
7.2 Adaptive Frequency Hopping ..... 336
7.3 Coexistence between Bluetooth Devices and Wireless LAN Devices ..... 337
7.4 Mobile Wireless Standards (MWS) coexistence ..... 337
7.5 Synchronizing Bluetooth with an external timing source ..... 340
7.6 Piconet clock adjustment ..... 341
7.7 Slot Availability Mask (SAM) ..... 342

**8 Direction finding using Bluetooth Low Energy ..... 343**
8.1 Angle of arrival (AoA) method ..... 343
8.2 Angle of departure (AoD) method ..... 344

**9 Channel Sounding using Bluetooth Low Energy ..... 347**
9.1 Channel Sounding procedure ..... 347
9.2 Distance estimation based on phase and amplitude information 348
9.3 Distance estimation based on RTT packets ..... 350
9.4 Security features ..... 351

# 1 GENERAL DESCRIPTION

This Part of the specification provides an overview of the Bluetooth system architecture, communication topologies, and data transport features. The text in this Part of the specification is intended to provide an introduction to Bluetooth and does not cover every detail or every corner case. In case of discrepancy with other parts of this specification, the other text is to be relied on.

Bluetooth wireless technology is a short-range communications system intended to replace the cable(s) connecting portable and/or fixed electronic devices. The key features of Bluetooth wireless technology are robustness, low power consumption, and low cost. Many features of the specification are optional, allowing product differentiation.

There are two forms of Bluetooth wireless technology systems: Basic Rate (BR) and Low Energy (LE). Both systems include device discovery, connection establishment and connection mechanisms. The Basic Rate system includes an optional Enhanced Data Rate (EDR) extension. The Basic Rate system offers synchronous and asynchronous connections with data rates of 721.2 kb/s for Basic Rate and 2.1 Mb/s for Enhanced Data Rate. The LE system includes features designed to enable products that require lower current consumption, lower complexity and lower cost than BR/EDR. The LE system is also designed for use cases and applications with lower data rates and has lower duty cycles. The LE system includes an optional 2 Mb/s physical layer data rate and also offers isochronous data transfer in a connection-oriented and connectionless mechanism that uses the isochronous transports. The LE system also includes the optional modulation of tones used to convey information useful for distance estimation. Depending on the use case or application, one system including any optional parts may be more optimal than the other.

Devices implementing both systems can communicate with other devices implementing both systems as well as devices implementing either system. Some profiles and use cases will be supported by only one of the systems. Therefore, devices implementing both systems have the ability to support the most use cases.

The Bluetooth core system consists of a Host and one or more Controllers. A Host is a logical entity defined as all of the layers below the non-core profiles and above the Host Controller interface (HCI). A Controller is a logical entity defined as all of the layers below HCI. An implementation of the Host and Controller may contain the respective parts of the HCI.

An implementation of the Bluetooth Core has only one Controller which may be one of the following configurations:

- a BR/EDR Controller including the Radio, Baseband, Link Manager and optionally HCI.
- an LE Controller including the LE PHY, Link Layer and optionally HCI.
- a combined BR/EDR Controller portion and LE Controller portion (as identified in the previous two bullets) into a single Controller.

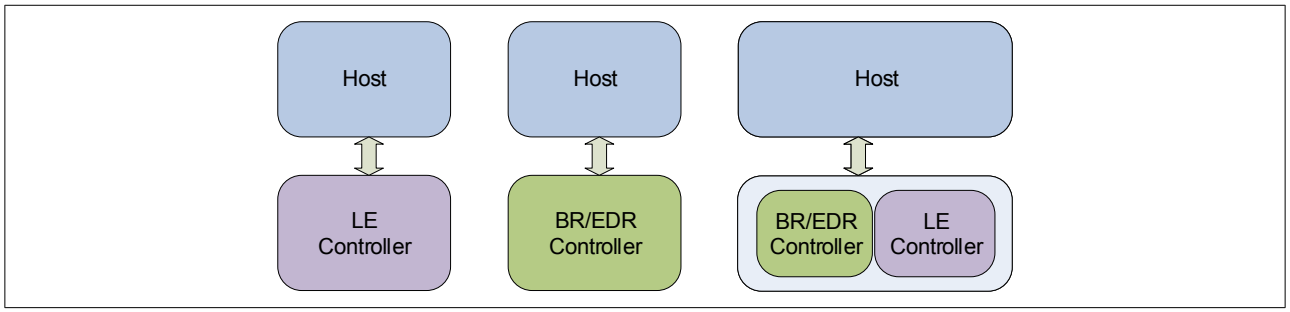



*Figure 1.1: Bluetooth Host and Controller combinations: (from left to right): LE Only Controller, BR/EDR only Controller, and BR/EDR/LE Controller*

## 1.1 Overview of BR/EDR operation

The Basic Rate / Enhanced Data Rate (BR/EDR) radio (physical layer or PHY) operates in the unlicensed ISM band at 2.4 GHz. The system employs a frequency hopping transceiver to combat interference and fading and provides many FHSS carriers. Basic Rate radio operation uses a shaped, binary frequency modulation to minimize transceiver complexity. The symbol rate is 1 megasymbol per second (Msym/s) supporting the bit rate of 1 megabit per second (Mb/s) or, with Enhanced Data Rate, a gross air bit rate of 2 Mb/s or 3 Mb/s. These modes are known as Basic Rate and Enhanced Data Rate respectively.

During typical operation a physical radio channel is shared by a group of devices that are synchronized to a common clock and frequency hopping pattern. One device provides the synchronization reference and is known as the Central. All other devices synchronized to a Central's clock and frequency hopping pattern are known as Peripherals. A group of devices synchronized in this fashion form a piconet. This is the fundamental form of communication in the Bluetooth BR/EDR wireless technology.

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by certain fields in the Bluetooth address and clock of the Central. The basic hopping pattern is a pseudo-random ordering of the 79 frequencies, separated by 1 MHz, in the ISM band. The hopping pattern can be adapted – on a per-Peripheral basis – to omit a portion of the frequencies that are used by interfering devices. The

adaptive hopping technique improves Bluetooth co-existence with static (non-hopping) ISM systems when they are co-located.

The physical channel is sub-divided into time units known as slots. Data is transmitted between Bluetooth devices in packets that are positioned in these slots. When circumstances permit, a number of consecutive slots may be allocated to a single packet. Frequency hopping may take place between the transmission or reception of packets. Bluetooth technology provides the effect of full duplex transmission through the use of a Time-Division Duplex (TDD) scheme.

Above the physical channel there is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the physical channel upwards is physical channel, physical link, logical transport, logical link and L2CAP channel. These are discussed in more detail in Section 3.3 to Section 3.6 but are introduced here to aid the understanding of the remainder of this section.

Typically within a physical channel, a physical link is formed between a Central and one or more Peripherals. Exceptions to this include Inquiry scan and Page scan physical channels, which have no associated physical link. The physical link provides bidirectional packet transport between the Central and Peripherals, except in the case of a Connectionless Peripheral Broadcast physical link. In that case, the physical link provides a unidirectional packet transport from the Central to a potentially unlimited number of Peripherals. Since a physical channel could include multiple Peripherals, there are restrictions on which devices may form a physical link. There is a physical link between each Peripheral and the Central. Physical links are not formed directly between the Peripherals in a piconet.

The physical link is used as a transport for one or more logical links that support unicast synchronous, asynchronous and isochronous traffic, and broadcast traffic. Traffic on logical links is multiplexed onto the physical link by occupying slots assigned by a scheduling function in the resource manager.

A control protocol for the baseband and physical layers is carried over logical links in addition to user data. This is the Link Manager protocol (LMP). Devices that are active in a piconet have a default asynchronous connection-oriented logical transport that is used to transport the LMP protocol signaling. For historical reasons this is known as the ACL logical transport. With the exception of Connectionless Peripheral Broadcast devices, the primary ACL logical transport is the one that is created whenever a device joins a piconet. Connectionless Peripheral Broadcast devices may join the piconet purely to listen to Connectionless Peripheral Broadcast packets. In that case, a Connectionless Peripheral Broadcast logical transport is created (also called a CPB logical transport) and no ACL logical transport is required. For all devices, additional logical transports may be created to transport synchronous data streams when required.

The Link Manager function uses LMP to control the operation of devices in the piconet and provide services to manage the lower architectural layers (radio and baseband). The LMP protocol is carried on the primary ACL and Active Peripheral Broadcast logical transports.

Above the baseband the L2CAP layer provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the default ACL logical transport. Application data submitted to the L2CAP protocol may be carried on any logical link that supports the L2CAP protocol.

## 1.2 Overview of Bluetooth Low Energy operation

Like the BR/EDR radio, the LE radio operates in the unlicensed 2.4 GHz ISM band. The LE system employs a frequency hopping transceiver to combat interference and fading and provides many FHSS carriers. LE radio operation uses a shaped, binary frequency modulation, and optional amplitude shift keying modulation to minimize transceiver complexity. LE uses terminology that differs from BR/EDR to describe supported PHYs with regards to differences in modulation, coding that may be applied, and the resulting data rates. The mandatory symbol rate is 1 megasymbol per second (Msym/s), where 1 symbol represents 1 bit therefore supporting a bit rate of 1 megabit per second (Mb/s), which is referred to as the LE 1M PHY. The 1 Msym/s symbol rate may support error correction coding, which is referred to as the LE Coded PHY. This may use either of two coding schemes: S=2, where 2 symbols represent 1 bit therefore supporting a bit rate of 500 kb/s, and S=8, where 8 symbols represent 1 bit therefore supporting a bit rate of 125 kb/s. A symbol rate of 2 Msym/s may be supported, with a bit rate of 2 Mb/s, which is referred to as the LE 2M PHY, when using a bandwidth-symbol time product (BT) of 0.5. An additional optional symbol rate at 2 Msym/s using a BT of 2.0 is also supported when used for the purpose of distance estimation, which is referred to as the LE 2M 2BT PHY. The 2 Msym/s symbol rate supports uncoded data only. LE 1M, LE 2M, and 2M 2BT are collectively referred to as the LE Uncoded PHYs. Section 3.2.2 describes this terminology in more detail.

The amplitude shift keying modulation scheme is employed to modulate tones for the purpose of gathering information for distance estimation. Section 3.2.3 describes amplitude shift keying in more detail.

LE employs two multiple access schemes: Frequency division multiple access (FDMA) and time division multiple access (TDMA). A TDMA-based polling scheme is used in which one device transmits a packet at a predetermined time and a corresponding device responds with a packet after a predetermined interval.

For the purposes of data transfer, 40 physical channels, separated by 2 MHz, are used in the FDMA scheme. Three are used as primary advertising channels and 37 are used as general-purpose channels (including as secondary advertising channels).

For the purpose of distance estimation, 72 physical channels, separated by 1 MHz, are used in the FDMA scheme.

The physical channel is sub-divided into time units known as events. Data is transmitted between LE devices in packets that are positioned in these events. The following types of events exist: Advertising, Extended Advertising, Periodic Advertising, Connection, Isochronous events (which are partitioned into BIS, BIG, CIS, and CIG events), and Channel Sounding events.

Devices that transmit advertising packets on the advertising PHY channels are referred to as advertisers. Devices that receive advertising packets on the advertising physical channels without the intention to connect to the advertising device are referred to as scanners. Transmissions on the advertising PHY channels occur in advertising events. At the start of each advertising event, the advertiser sends an advertising packet corresponding to the advertising event type. Depending on the type of advertising packet, the scanner may make a request to the advertiser on the same advertising PHY channel which may be followed by a response from the advertiser on the same advertising PHY channel. The advertising PHY channel changes on the next advertising packet sent by the advertiser in the same advertising event. The advertiser may end the advertising event at any time during the event. Each advertising packet in an advertising event uses a different advertising PHY channel. Each advertising event may use a different order for the advertising PHY channels.

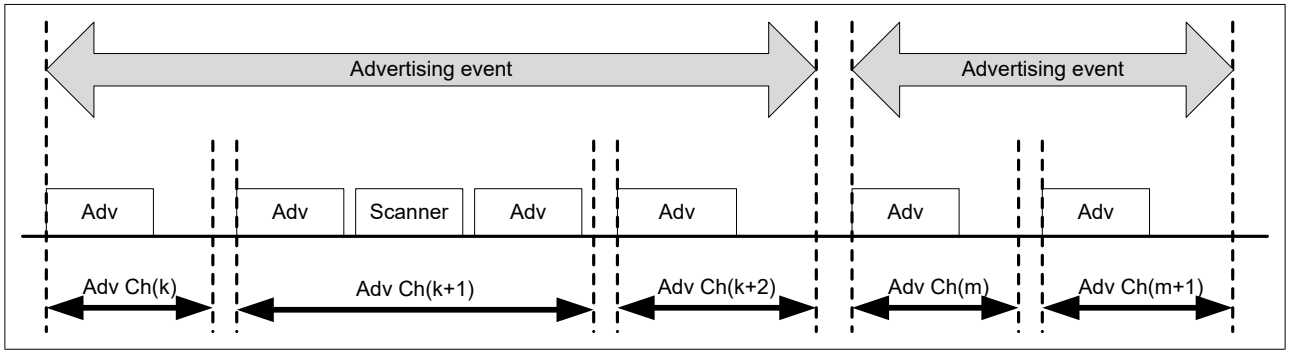



*Figure 1.2: Advertising events*

LE devices may fulfill the entire communication in the case of unidirectional or broadcast communication between two or more devices using advertising events. LE devices may also use advertising events to establish bi-directional connections with another device and to establish asynchronous or isochronous periodic broadcasts. Asynchronous periodic broadcasts may allow the advertiser to receive responses from

one or more devices. These additional activities make use of the general purpose channels in various ways.

Devices that need to form an ACL connection to another device listen for connectable advertising packets. Such devices are referred to as initiators. If the advertiser is using a connectable advertising event, an initiator may make a connection request using the same advertising PHY channel on which it received the connectable advertising packet. The advertising event is ended and connection events begin if the advertiser receives and accepts the request for a connection be initiated. Once a connection is established, the initiator becomes the Central in what is referred to as a piconet and the advertising device becomes the Peripheral. Connection events are used to send data packets between the Central and Peripherals. In connection events, channel hopping occurs at the start of each connection event. Within a connection event, the Central and Peripheral alternate sending data packets using the same data PHY channel. The Central initiates the beginning of each connection event and can end each connection event at any time.

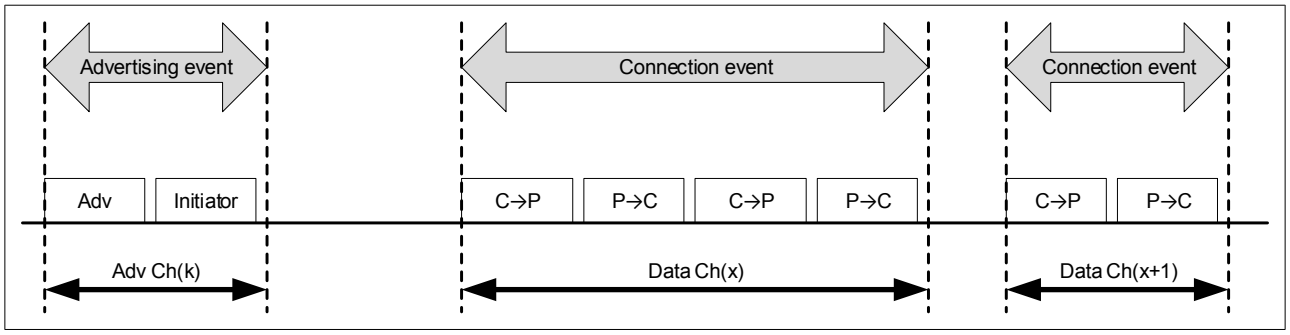



*Figure 1.3: Connection events*

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by a field contained in the connection request sent by an initiating device. The hopping pattern used on the LE data channel is a pseudo-random ordering of the 37 frequencies in the ISM band. The hopping pattern used in a Channel Sounding procedure is a pseudo-random ordering of 72 frequencies in the ISM band. The hopping pattern can be adapted to omit a portion of the frequencies that are used by interfering devices. The adaptive hopping technique improves Bluetooth co-existence with static (non-hopping) ISM systems when these systems are co-located and have access to information about the local radio environment. A Peripheral can classify frequencies as good and bad and provide that information to the Central. The Central can take this information into consideration while adapting the hopping pattern.

Using an ACL connection, a Central can establish one or more isochronous connections that use the isochronous physical channel. An isochronous connection is used to transfer isochronous data between the Central and a Peripheral by using a logical transport, which is referred to as a Connected Isochronous Stream (CIS). A CIS

consists of CIS events that occur at regular intervals (designated ISO_Interval). Every CIS event consists of one or more subevents. In each subevent, the Central transmits once and the Peripheral responds. If the Central and Peripheral have completed transferring the scheduled isochronous data in a CIS event, all remaining subevents in that event will have no radio transmissions and the event is closed. Each subevent uses a PHY channel which is determined by using the channel selection algorithm. The PHY channel that is used for a subevent is marked as ISO Ch(eventcount, subeventcount), as shown in Figure 1.4.

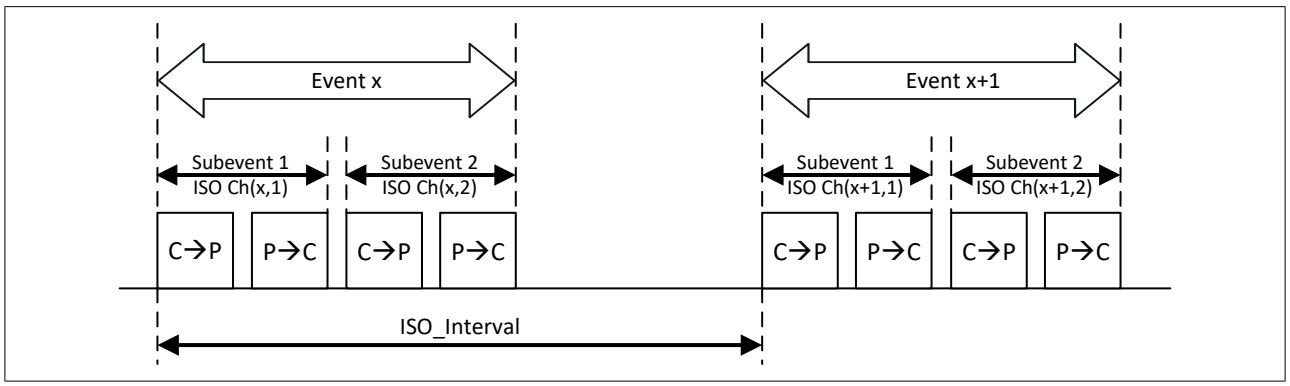



*Figure 1.4: CIS events and subevents*

A device can use an isochronous physical channel to broadcast isochronous data by using isochronous connectionless logical transports. An isochronous connectionless logical transport is referred to as a Broadcast Isochronous Stream (BIS). A BIS consists of BIS events that occur at regular intervals (designated ISO_Interval). Every BIS event consists of one or more subevents. In every subevent, a broadcasting device transmits an isochronous data packet. Each subevent uses a PHY channel that is determined using the channel selection algorithm.

A device can transmit several BISes with synchronized timing; this is referred to as a Broadcast Isochronous Group (BIG). The various BIS events together form a BIG event. The device can also use the isochronous physical channel to broadcast control information in a Control subevent, which is transmitted at the end of all subevents for a BIG, as shown in Figure 1.5.

A device that transmits BIG events also transmits periodic advertisement events that contain synchronization information of the BIG. A device that is scanning can synchronize to those periodic advertising events and receive the synchronization information. Using this synchronization information, the device can synchronize to one or more BISes in the BIG and receive the isochronous data. Figure 1.5 shows two BIG events: one with and one without a Control subevent. Each subevent uses a PHY channel marked as ISO Ch(eventcount, subeventcount), as shown in Figure 1.5.

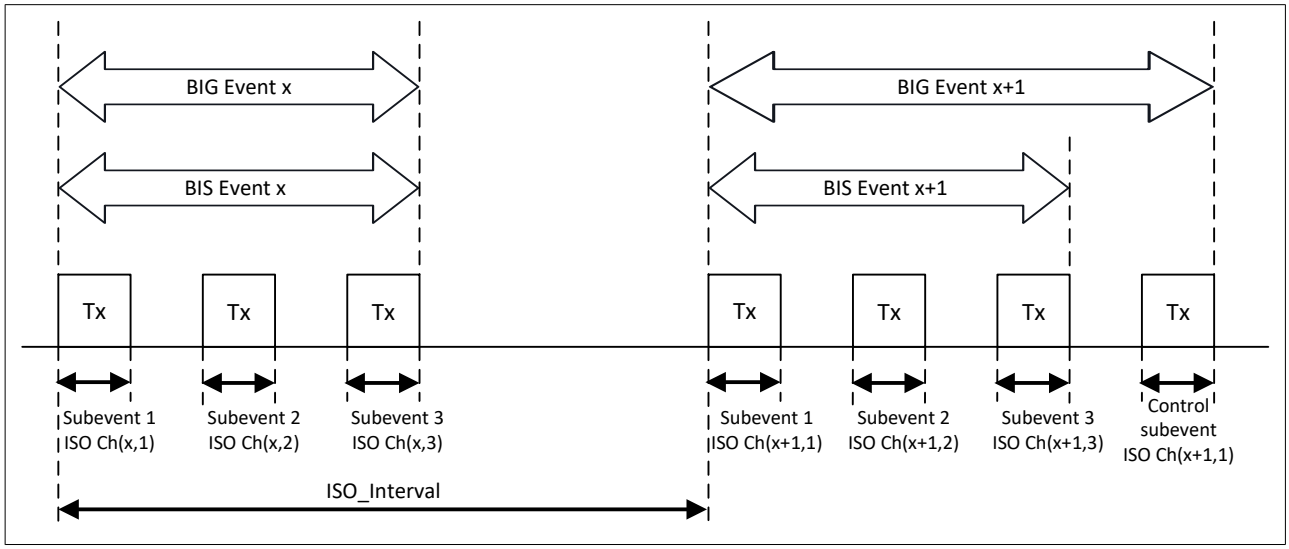



*Figure 1.5: BIG and BIS events, BIS subevents, and Control subevent*

Devices can use the LE Channel Sounding physical link to exchange information that can later be used for distance estimation calculations. A Channel Sounding procedure (and hence, the first Channel Sounding event within that procedure) is started at an offset from an ACL connection event anchor point. A Channel Sounding procedure exists only for a limited duration, and consists of Channel Sounding events, subevents, and steps. Channel Sounding events may contain one or more subevents. Channel Sounding subevents contain two or more Channel Sounding steps. Channel Sounding steps contain bilateral exchanges between the two Channel Sounding peers known as the initiator and reflector. The Channel Sounding initiator transmits first in each step, followed by one or more transmissions from the Channel Sounding reflector. These transmissions may be a packet-based GFSK-modulated exchange or a tone-based, amplitude shift keying modulated exchange, or both.

The general structure of Channel Sounding events and subevents is shown in Figure 1.6.

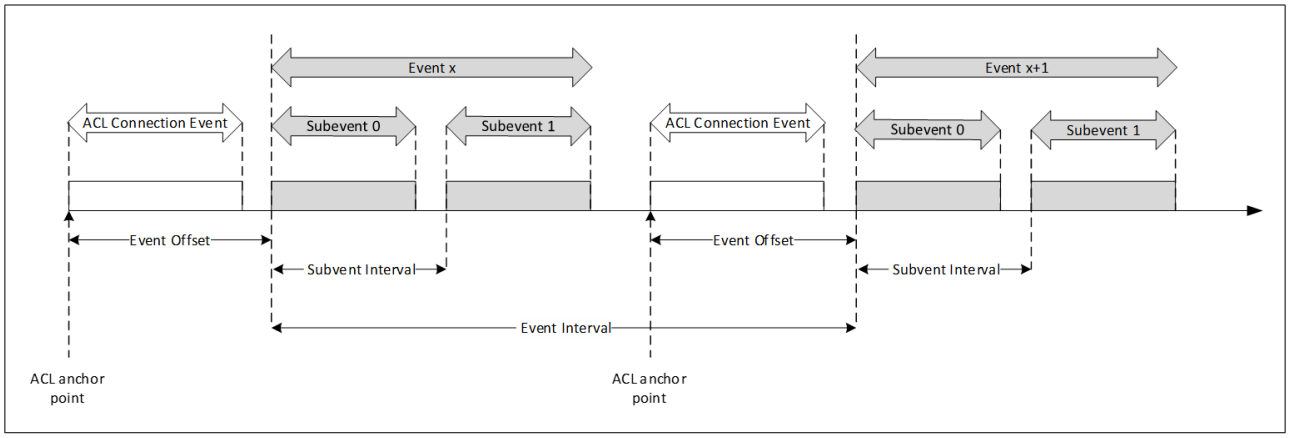



*Figure 1.6: Channel Sounding events and subevents*

Each exchange within a Channel Sounding step contains information that can be measured. These measurements can be further processed to produce a distance estimate. The step content is discussed in more detail in [Vol 6] Part H, Section 4.3. These exchanges also carry security-related information that may assist in the detection of an external attacker who is attempting to indirectly manipulate the measured results, which could in turn affect the distance estimate. The Channel Sounding step exchanges use a PHY channel that is determined using a channel selection algorithm. For Channel Sounding, this channel selection algorithm, as well as other security-related information, is seeded using a Deterministic Random Bit Generator (DRBG). The security key material for this DRBG is known only by the respective initiator and reflector devices.

Above the physical channel there are concepts of links, channels and associated control protocols. The hierarchy is physical channel, physical link, logical transport, logical link, and L2CAP channel. These are discussed in more detail in Section 3.3 to Section 3.6 but are introduced here to aid the understanding of the remainder of this section.

Within a physical channel, a physical link is formed between devices. The active physical link provides bidirectional packet transport between the Central and Peripherals. Centrals may have physical links to more than one Peripheral at a time and Peripherals may have physical links to more than one Central at a time. A device may be Central and Peripheral in different piconets at the same time. Role changes between a Central and Peripheral are not supported. The advertising and periodic physical links provide a unidirectional packet transport from the advertiser to a potentially unlimited number of scanners or initiators.

The physical link is used as a transport for one or more logical links that support asynchronous traffic. Traffic on logical links is multiplexed onto the physical link assigned by a scheduling function in the resource manager.

A control protocol for the link and physical layers is carried over logical links in addition to user data. This is the Link Layer protocol (LL). Devices that are active in a piconet

have a default LE asynchronous connection logical transport (LE ACL) that is used to transport the LL protocol signaling. The default LE ACL is the one that is created whenever a piconet is created.

The Link Layer function uses the LL protocol to control the operation of devices in the piconet and provide services to manage the lower architectural layers (PHY and LL).

Overall, a piconet consists of one ACL logical transport over the active physical link plus zero or more CIS logical transports over the isochronous physical link(s). In addition, zero or more transitory Channel Sounding procedures may exist over the Channel Sounding physical link.

Just as in BR/EDR, above the Link Layer the L2CAP layer provides a channel-based abstraction to applications and services. It carries out fragmentation and de-fragmentation of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the primary ACL logical transport.

In addition to L2CAP, LE provides two additional protocol layers that reside on top of L2CAP. The Security Manager protocol (SMP) uses a fixed L2CAP channel to implement the security functions between devices. The other is the Attribute Protocol (ATT) that provides a method to communicate small amounts of data over a fixed L2CAP channel. The Attribute Protocol is also used by devices to determine the services and capabilities of other devices. The Attribute Protocol may also be used over BR/EDR.

The LE radio provides a means for detecting the relative direction of another LE radio by using the Angle of Arrival (AoA) or Angle of Departure (AoD) method.

## 1.3 [This section is no longer used]

## 1.4 Nomenclature

Where the following terms appear in the specification they have the meaning given in Table 1.1.

| | |
|---|---|
| Active Peripheral Broadcast (APB) | The logical transport that is used to transport L2CAP user traffic and some kinds of LMP traffic to all active devices in the piconet over the BR/EDR Controller. See Section 3.5.4.4 |
| Ad Hoc Network | A network typically created in a spontaneous manner. An ad hoc network requires no formal infrastructure and is limited in temporal and spatial extent. |
| Advertiser | A Bluetooth Low Energy device that broadcasts advertising packets during advertising events on advertising channels |

| | |
|---|---|
| Advertising event | A series of between one and three advertising packets on different advertising physical channels sent by an advertiser. |
| Advertising Packet | A packet containing an advertising PDU. See [Vol 6] Part B, Section 2.3.1 |
| Angle of Arrival (AoA) | Angle of Arrival is the relative direction at which a propagating RF wave that was transmitted by a single antenna is incident on an antenna array. |
| Angle of Departure (AoD) | Angle of Departure is the relative direction from which a propagating RF wave that was transmitted using an antenna array is incident on another antenna. |
| BD_ADDR | The Bluetooth Device Address, BD_ADDR, is used to identify a Bluetooth device. |
| Bluetooth | Bluetooth is a wireless communication link, operating in the unlicensed ISM band at 2.4 GHz using a frequency hopping transceiver. It allows real-time AV and data communications between Bluetooth Hosts. The link protocol is based on time slots. |
| Bluetooth Baseband | The part of the Bluetooth system that specifies or implements the medium access and physical layer procedures to support the exchange of real-time voice, data information streams, and ad hoc networking between Bluetooth Devices. |
| Bluetooth Clock | A 28 bit clock internal to a BR/EDR Controller that ticks every 312.5 µs. The value of this clock defines the slot numbering and timing in the various physical channels. |
| Bluetooth Controller | A generic term referring to a Controller. |
| Bluetooth Device | A device that is capable of short-range wireless communications using the Bluetooth system. |
| Bluetooth Device Address | A 48 bit address used to identify each Bluetooth device. |
| BR/EDR | Bluetooth basic rate (BR) and enhanced data rate (EDR). |
| BR/EDR Controller | A term referring to the layers defined in Volume 2. |
| BR/EDR Piconet Physical Channel | A Channel that is divided into time slots in which each slot is related to an RF hop frequency. Consecutive hops normally correspond to different RF hop frequencies and occur at a standard hop rate of 1600 hops per second. These consecutive hops follow a pseudo-random hopping sequence, hopping through a 79 RF channel set, or optionally fewer channels when Adaptive Frequency Hopping (AFH) is in use. |
| BR/EDR/LE | Bluetooth basic rate (BR), enhanced data rate (EDR) and low energy (LE). |
| C-plane | Control plane |
| Channel | Either a physical channel or an L2CAP channel, depending on the context. |

| | |
|---|---|
| Channel Sounding | A Bluetooth Low Energy feature that measures and distributes information that can be used to approximate distances between devices. |
| Channel Sounding event | A group of Channel Sounding subevents that are anchored from a common LE connection event. |
| Channel Sounding procedure | A group of Channel Sounding events that are sequenced serially for the purpose of gathering information useful for estimating the distance between two devices. |
| Channel Sounding step | In Channel Sounding, an individual exchange between two devices. |
| Channel Sounding subevent | A group of Channel Sounding steps that are associated with a specific coherent timing. |
| Connect (to service) | The establishment of a connection to a service. If not already done, this also includes establishment of a physical link, logical transport, logical link and L2CAP channel. |
| Connectable device | A BR/EDR device in range that periodically listens on its page scan physical channel and will respond to a page on that channel. An LE device that is advertising using a connectable advertising event. |
| Connected devices | Two BR/EDR devices and with a physical link between them. |
| Connecting | A phase in the communication between devices when a connection between the devices is being established. (Connecting phase follows after the link establishment phase is completed.) |
| Connection | A connection between two peer applications or higher layer protocols mapped onto an L2CAP channel. |
| Connection establishment | A procedure for creating a connection mapped onto a channel. |
| Connection event | A series of one or more pairs of interleaving data packets sent between a Central and a Peripheral on the same physical channel. |
| Connectionless Peripheral Broadcast (CPB) | A feature that enables a Central to broadcast information to an unlimited number of Peripherals. |
| Connectionless Peripheral Broadcast Receiver | A Bluetooth device that receives broadcast information from a Connectionless Peripheral Broadcast Transmitter. The device is a Peripheral of the piconet. |
| Connectionless Peripheral Broadcast Transmitter | A Bluetooth device that sends Connectionless Peripheral Broadcast messages for reception by one or more Connectionless Peripheral Broadcast receivers. The device is the Central of the piconet. |
| Controller | A collective term referring to all of the layers below HCI. |
| Coverage area | The area where two Bluetooth devices can exchange messages with acceptable quality and performance. |
| Creation of a secure connection | A procedure of establishing a connection, including authentication and encryption. |

| | |
|---|---|
| Creation of a trusted relationship | A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication, or pairing, when a link key is not available. |
| Device discovery | A procedure for retrieving the Bluetooth Device Address, clock, and Class of Device from discoverable devices. |
| Discoverable device | A BR/EDR device in range that periodically listens on an inquiry scan physical channel and will respond to an inquiry on that channel. An LE device in range that is advertising with a connectable or scannable advertising event with a discoverable flag set in the advertising data. This device is in the discoverable mode. |
| Discoverable Mode | A Bluetooth device that is performing inquiry scans in BR/EDR or advertising with a discoverable or connectable advertising event with a discoverable flag set in LE. |
| Discovery procedure | A Bluetooth device that is carrying out the inquiry procedure in BR/EDR or scanning for advertisers using a discoverable or connectable advertising event with a discoverable flag set in LE. |
| HCI | The Host Controller interface (HCI) provides a command interface to the baseband Controller and link manager and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities. |
| Host | A logical entity defined as all of the layers below the non-core profiles and above the Host Controller interface (HCI); i.e., the layers specified in Volume 3. A Bluetooth Host attached to a Bluetooth Controller may communicate with other Bluetooth Hosts attached to their Controllers as well. |
| Initiator | From the perspective of an advertising bearer, a Bluetooth Low Energy device that listens on advertising physical channels for connectable advertising events to form connections.<br>From the perspective of Channel Sounding, the device that transmits first within a Channel Sounding step. |
| Inquiring device | A BR/EDR device that is carrying out the inquiry procedure. This device is performing the discovery procedure. |
| Inquiry | A procedure where a Bluetooth device transmits inquiry messages and listens for responses in order to discover the other Bluetooth devices that are within the coverage area. |
| Inquiry scan | A procedure where a Bluetooth device listens for inquiry messages received on its inquiry scan physical channel. |
| Instant | A moment in time when something happens. In particular, the (BR/EDR) slot or (LE) Connection event where a specific action will happen. |
| Interoperability | The ability of two or more devices to exchange information and to use the information that has been exchanged. |

| | |
|---|---|
| Isochronous data | Information in a stream where each information entity in the stream is bound by a time relationship to previous and successive entities. |
| Known device | A Bluetooth device for which at least the BD_ADDR is stored. |
| L2CAP | Logical Link Control and Adaptation Protocol |
| L2CAP Channel | A logical connection on L2CAP level between two devices serving a single application or higher layer protocol. |
| L2CAP Channel establishment | A procedure for establishing a logical connection on L2CAP level. |
| LE | Bluetooth Low Energy |
| LE Controller | A term referring to the layers defined in Volume 6. |
| Link | Shorthand for a logical link. |
| Link establishment | A procedure for establishing the default ACL link and hierarchy of links and channels between devices. |
| Link key | A secret key that is known by two devices and is used to authenticate the link. |
| LMP authentication | An LMP level procedure for verifying the identity of a remote device. |
| LMP pairing | A procedure that authenticates two devices and creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. |
| Logical link | The lowest architectural level used to offer independent data transport services to clients of the Bluetooth system. |
| Logical transport | Shared acknowledgment protocol and link identifiers between different logical links. |
| Name discovery | A procedure for retrieving the user-friendly name (the Bluetooth Device Name) of a connectable device. |
| Packet | Format of aggregated bits that are transmitted on a physical channel. |
| Page | The initial phase of the connection procedure where a device transmits a train of page messages until a response is received from the target device or a time-out occurs. |
| Page scan | A procedure where a device listens for page messages received on its page scan physical channel. |
| Paging device | A Bluetooth device that is carrying out the page procedure. |
| Paired device | A Bluetooth device for which a link key has been created (either before connection establishment was requested or during connecting phase). |
| Passkey | A 6-digit number used to authenticate connections when Secure Simple Pairing is used. |

| | |
|---|---|
| Periodic advertising synchronization information | The control information describing a periodic advertisement that a Bluetooth Low Energy device uses to synchronize to the advertisement it describes. |
| Physical Channel | Characterized by synchronized occupancy of a sequence of RF carriers by one or more devices. A number of physical channel types exist with characteristics defined for their different purposes. |
| Physical link | A Baseband or Link Layer level connection between two devices. |
| Physical Transport | PHY packet transmission and/or reception on an RF channel using one or more modulation schemes. |
| Piconet | A collection of devices (up to eight devices in BR/EDR, exactly two devices in LE) occupying a shared physical channel where one of the devices is the Piconet Central and the remaining devices are connected to it. |
| Piconet Central | The BR/EDR device in a piconet whose Bluetooth Clock and Bluetooth Device Address are used to define the piconet physical channel characteristics.<br>The LE device in a piconet which initiates the creation of the piconet, chooses the Access Address that identifies the piconet, and transmits first in each connection event. |
| Piconet Peripheral | Any BR/EDR device in a piconet that is not the Piconet Central, but is connected to the Piconet Central.<br>The LE device in a piconet which is not the Central but communicates with it. |
| PIN | A user-friendly number that can be used to authenticate connections to a device before pairing has taken place. |
| Profile Broadcast Data (PBD) | A logical link that carries data from a Connectionless Peripheral Broadcast Transmitter to one or more<br>Connectionless Peripheral Broadcast Receivers. |
| Pseudo-Noise Bit Sequence | A series of bits that are generated randomly. |
| Reflector | In Channel Sounding, the device that transmits second within a Channel Sounding step in response to a transmission from an initiator. |
| Resolving List | A list of records used to generate and resolve Resolvable Private Addresses. Each record contains a local Identity Resolving Key, a peer Identity Resolving Key, and a peer Identity Address. |
| Round-Trip Time | The time it takes for a packet to travel from an originating device to a responding device and back again to the originating device. |
| Scanner | A Bluetooth Low Energy device that listens for advertising events on the advertising physical channels. |
| Scatternet | Two or more piconets that have one or more devices in common. |

| | |
|---|---|
| Service discovery | Procedures for querying and browsing for services offered by or through another Bluetooth device. |
| Service Layer Protocol | A protocol that uses an L2CAP channel for transporting PDUs. |
| Silent device | A Bluetooth enabled device appears as silent to a remote device if it does not respond to inquiries made by the remote device. |
| Synchronization Scan Physical Channel | A physical channel that enables a Peripheral to receive synchronization train packets from a Central. |
| Synchronization Train | A series of packets transmitted on a set of fixed frequencies that deliver sufficient information for a receiving device to start receiving corresponding Connectionless Peripheral Broadcast packets or to recover the current piconet clock after missing a Coarse Clock Adjust. |
| Tick | (BR/EDR) the time between changes of the value of the Bluetooth Clock: 312.5 µs. |
| U-plane | User plane |
| Unknown device | A Bluetooth device for which no information (Bluetooth Device Address, link key or other) is stored. |

*Table 1.1: Nomenclature*

# 2 CORE SYSTEM ARCHITECTURE

The Bluetooth Core system consists of a Host and a Controller. A minimal implementation of the Bluetooth BR/EDR core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as one common service layer protocol; the Service Discovery Protocol (SDP) and the overall profile requirements are specified in the Generic Access Profile (GAP). A minimal implementation of a Bluetooth LE only core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as two common service layer protocols; the Security Manager (SM) and Attribute Protocol (ATT) and the overall profile requirements are specified in the Generic Attribute Profile (GATT) and Generic Access Profile (GAP). Implementations combining Bluetooth BR/EDR and LE include both of the minimal implementations described above.

A complete Bluetooth application requires a number of additional service and higher layer protocols that are defined in the Bluetooth specification, but are not described here. The core system architecture is shown in Figure 2.1.

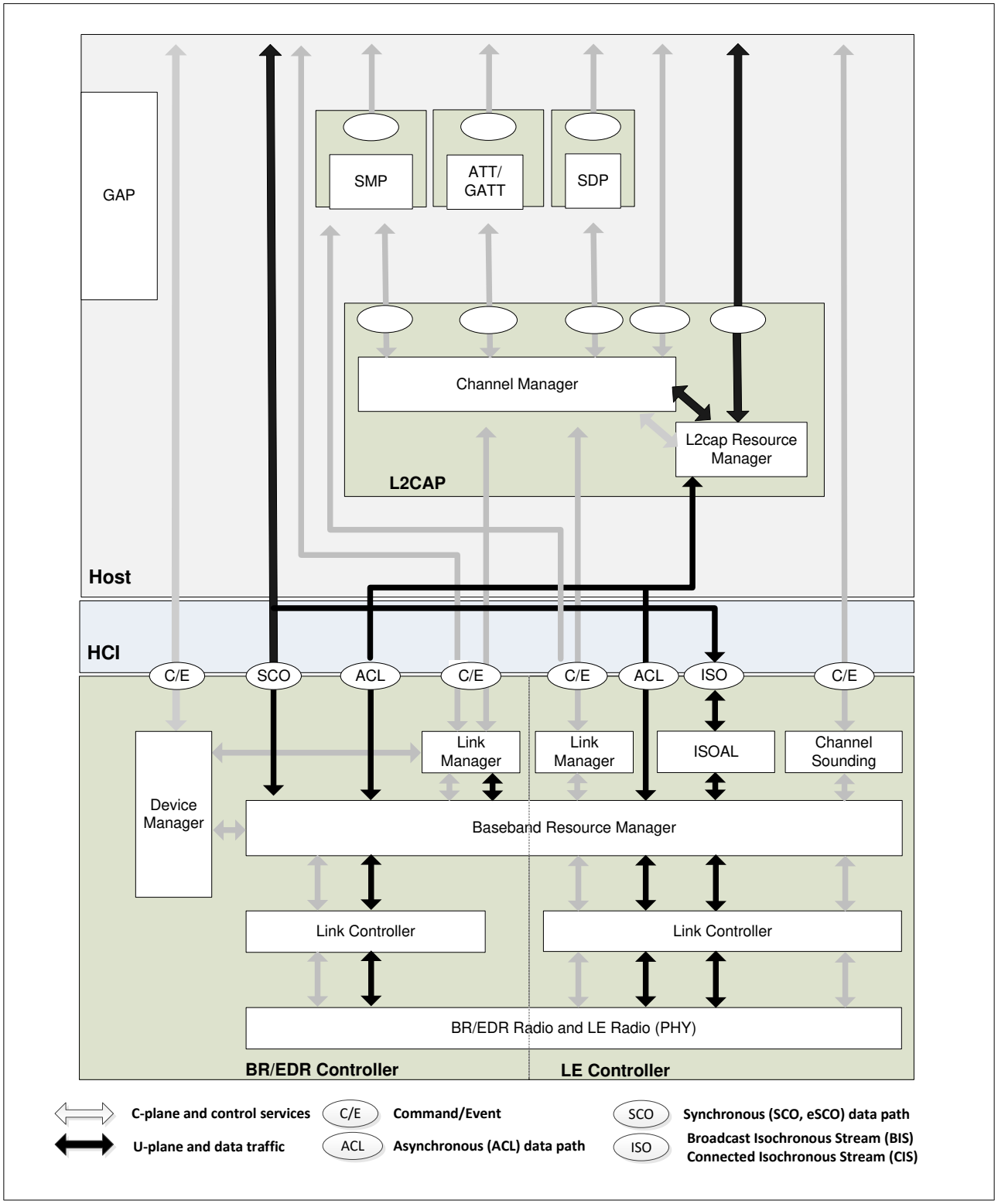



*Figure 2.1: Bluetooth core system architecture*

Figure 2.1 shows the Core blocks, each with its associated communication protocol. Link Manager, Link Controller and BR/EDR Radio blocks comprise a BR/EDR Controller. Link Manager, Link Controller and LE Radio blocks comprise an LE Controller. L2CAP, SDP and GAP blocks comprise a BR/EDR Host. L2CAP, SMP, Attribute Protocol, GAP and Generic Attribute Profile (GATT) blocks comprise an LE Host. A BR/EDR/LE Host combines the set of blocks from each respective Host. This is a common implementation involving a standard physical communication interface between the Controller and the Host. Although this interface is optional the architecture is designed to allow for its existence and characteristics. The Bluetooth specification enables interoperability between independent Bluetooth systems by defining the protocol messages exchanged between equivalent layers, and also interoperability between independent Bluetooth Controllers and Bluetooth Hosts by defining a common interface between Bluetooth Controllers and Bluetooth Hosts.

A number of functional blocks and the path of services and data between them are shown. The functional blocks shown in the diagram provide a set of conceptual entities that are used when describing the requirements of the specification; in general the Bluetooth specification does not define the details of implementations except where this is required for interoperability. Thus the functional blocks in Figure 2.1 are shown in order to aid description of the system behavior. An implementation may be different from the system shown in Figure 2.1.

Standard interactions are defined for all inter-device operation, where Bluetooth devices exchange protocol signaling according to the Bluetooth specification. The Bluetooth core system protocols are the Radio (PHY) protocol, Link Control (LC) and Link Manager (LM) protocol or Link Layer (LL) protocol, and Logical Link Control and Adaptation protocol (L2CAP), all of which are fully defined in subsequent parts of the Bluetooth specification. In addition, the Service Discovery protocol (SDP) and the Attribute Protocol (ATT) are service layer protocols that may be required by some Bluetooth applications.

The Bluetooth core system offers services through a number of service access points that are shown in the diagram as ellipses. These services consist of the basic primitives that control the Bluetooth core system. The services can be split into three types. There are device control services that modify the behavior and modes of a Bluetooth device, transport control services that create, modify and release traffic bearers (channels and links), and data services that are used to submit data for transmission over traffic bearers. It is common to consider the first two as belonging to the C-plane and the last as belonging to the U-plane.

A service interface to the Bluetooth Controller is defined such that the Controller may be considered a standard part. In this configuration the Bluetooth Controller operates the lowest four layers. The Bluetooth Host operates the L2CAP layer and other higher layers. The standard interface is called the Host Controller interface (HCI) and

its service access points are represented by the ellipses on the upper edge of the Bluetooth Controller in Figure 2.1. Implementation of this standard service interface is optional.

As the Bluetooth architecture is defined with the possibility of separate Host and Controller(s) communicating through one or more HCI transports, a number of general assumptions are made. Bluetooth Controllers are assumed to have limited data buffering capabilities in comparison with the Host. Therefore the L2CAP layer is expected to carry out some simple resource management when submitting L2CAP PDUs to the Controller for transport to a peer device. This includes segmentation of L2CAP SDUs into more manageable PDUs and then the fragmentation of PDUs into start and continuation packets of a size suitable for the Controller buffers, and management of the use of Controller buffers to ensure availability for channels with Quality of Service (QoS) commitments.

The BR/EDR Baseband and LE Link Layer provide the basic acknowledgment/repeat request (ARQ) protocol in Bluetooth. The L2CAP layer may provide further error detection and retransmission to the L2CAP PDUs. This feature is recommended for applications with requirements for a low probability of undetected errors in the user data. A window-based flow control feature may be used to manage buffer allocation in the receiving device. Both of these optional features augment the QoS performance in certain scenarios. Not all of the L2CAP capabilities are available when using the LE system.

Although these assumptions are not always required for embedded Bluetooth implementations that combine all layers in a single system, the general architectural and QoS models are defined with these assumptions in mind, in effect a lowest common denominator.

Automated conformance testing of implementations of the Bluetooth core system is required. This is achieved by allowing a test system to control the implementation through the PHY interface, test interfaces such as Direct Test Mode (DTM), Unified Test Protocol (UTP), and test commands and events over HCI which are only required for conformance testing.

The test system is composed of an Upper Tester, which controls and observes the upper service boundary of the IUT, and a Lower Tester, which controls and observes the lower service boundary of the IUT. The Lower Tester exchanges messages with the implementation under test (IUT) through the PHY interface. The Upper Tester controls the IUT through HCI, DTM, UTP, or test commands. If the targeted layer of the test is the PHY layer, then the Upper Tester may use a BR/EDR or LE connection to control the IUT over the air.

## 2.1 Core architectural blocks

This section describes the function and responsibility of each of the blocks shown in Figure 2.1. An implementation is not required to follow the architecture described above, though every implementation is still required to conform to the protocol specifications, behaviors, and other requirements specified in subsequent parts of the Bluetooth specification.

### 2.1.1 Host architectural blocks

#### 2.1.1.1 Channel manager

The channel manager is responsible for creating, managing and closing L2CAP channels for the transport of service protocols and application data streams. The channel manager uses the L2CAP protocol to interact with a channel manager on a remote (peer) device to create these L2CAP channels and connect their endpoints to the appropriate entities. The channel manager interacts with its local link manager to create new logical links (if necessary) and to configure these links to provide the required quality of service for the type of data being transported.

#### 2.1.1.2 L2CAP resource manager

The L2CAP resource manager block is responsible for managing the ordering of submission of PDU fragments to the baseband and some relative scheduling between channels to ensure that L2CAP channels with QoS commitments are not denied access to the physical channel due to Controller resource exhaustion. This is required because the architectural model does not assume that a Controller has limitless buffering, or that the HCI is a pipe of infinite bandwidth.

L2CAP Resource Managers may also carry out traffic conformance policing to check that applications are submitting L2CAP SDUs within the bounds of their negotiated QoS settings. The general Bluetooth data transport model assumes well-behaved applications, and does not define how an implementation is expected to deal with this problem.

#### 2.1.1.3 Security Manager Protocol

The Security Manager Protocol (SMP) is the peer-to-peer protocol used to generate encryption keys and identity keys. The protocol operates over a dedicated fixed L2CAP channel. The SMP block also manages storage of the encryption keys and identity keys and is responsible for generating random addresses and resolving random addresses to known device identities. The SMP block interfaces with the Controller to provide stored keys used for encryption and authentication during the encryption or pairing procedures.

This block is only used in LE systems. Similar functionality in the BR/EDR system is contained in the Link Manager block in the Controller. SMP functionality is in the Host on LE systems to reduce the implementation cost of the LE only Controllers.

#### 2.1.1.4 Attribute Protocol

The Attribute Protocol (ATT) block implements the peer-to-peer protocol between an ATT Server and an ATT Client. The ATT Client communicates with an ATT Server on a remote device over a dedicated fixed L2CAP channel. The ATT Client sends commands, requests, and confirmations to the ATT Server. The ATT Server sends responses, notifications and indications to the client. These ATT Client commands and requests provide a means to read and write values of attributes on a peer device with an ATT Server.

#### 2.1.1.5 [This section is no longer used]

#### 2.1.1.6 Generic Attribute Profile

The Generic Attribute Profile (GATT) block represents the functionality of the ATT Server and, optionally, the ATT Client. The profile describes the hierarchy of services, characteristics and attributes used in the ATT Server. The block provides interfaces for discovering, reading, writing and indicating of service characteristics and attributes. GATT is used on LE devices for LE profile service discovery.

#### 2.1.1.7 Generic Access Profile

The Generic Access Profile (GAP) block represents the base functionality common to all Bluetooth devices such as modes and access procedures used by the transports, protocols and application profiles. GAP services include device discovery, connection modes, security, authentication, association models and service discovery.

#### 2.1.1.8 Service Discovery Protocol

The Service Discovery Protocol (SDP) provides a mechanism to allow clients to search for needed services based on specific attributes of the service, including search based on class of service and browsing the entire database. SDP is used on BR/EDR devices for BR/EDR profile service discovery.

SDP focuses on discovering services available from or through Bluetooth devices. It does not define methods for accessing services once they are discovered with SDP; the access method is service-specific.

### 2.1.2 BR/EDR/LE Controller architectural blocks

In implementations where the BR/EDR and LE systems are combined, the architectural blocks may be shared between systems or each system may have their own instantiation of the block.

#### 2.1.2.1 Device manager

The device manager is the functional block in the baseband that controls the general behavior of the Bluetooth device. It is responsible for all operations of the Bluetooth system that are not directly related to data transport, such as inquiring for the presence of nearby Bluetooth devices, connecting to Bluetooth devices, or making the local Bluetooth device discoverable or connectable by other devices.

The device manager requests access to the transport medium from the baseband resource Controller in order to carry out its functions.

The device manager also controls local device behavior implied by a number of the HCI commands, such as managing the device local name, any stored link keys, and other functionality.

#### 2.1.2.2 Link manager

The link manager is responsible for the creation, modification and release of logical links (and, if required, their associated logical transports), as well as the update of parameters related to physical links between devices. The link manager achieves this by communicating with the link manager in remote Bluetooth devices using the Link Manager Protocol (LMP) in BR/EDR and the Link Layer Protocol (LL) in LE.

The LM or LL protocol allows the creation of new logical links and logical transports between devices when required, as well as the general control of link and transport attributes such as the enabling of encryption on the logical transport, the adapting of transmit power on the physical link, or the adjustment of QoS settings in BR/EDR for a logical link.

#### 2.1.2.3 Baseband resource manager

The baseband resource manager is responsible for all access to the radio medium. It has two main functions. At its heart is a scheduler that grants time on the physical channels to all of the entities that have negotiated an access contract. The other main function is to negotiate access contracts with these entities. An access contract is effectively a commitment to deliver a certain QoS that is required in order to provide a user application with an expected performance.

The access contract and scheduling function must take account of any behavior that requires use of the Controller. This includes (for example) the normal exchange of data between connected devices over logical links, and logical transports, as well as the use of the radio medium to carry out inquiries, make connections, be discoverable or connectable, or to take readings from unused carriers during the use of adaptive frequency hopping mode.

In some cases in BR/EDR systems the scheduling of a logical link results in changing a logical link to a different physical channel from the one that was previously used. This may be (for example) due to involvement in scatternet, a periodic inquiry function, or page scanning. When the physical channels are not time slot aligned, then the resource manager also accounts for the realignment time between slots on the original physical channel and slots on the new physical channel. In some cases the slots will be naturally aligned due to the same device clock being used as a reference for both physical channels.

#### 2.1.2.4 Link Controller

The Link Controller is responsible for the encoding and decoding of Bluetooth packets from the data payload and parameters related to the physical channel, logical transport and logical link.

The Link Controller carries out the link control protocol signaling in BR/EDR and Link Layer protocol in LE (in close conjunction with the scheduling function of the resource manager), which is used to communicate flow control and acknowledgment and retransmission request signals. The interpretation of these signals is a characteristic of the logical transport associated with the baseband packet. Interpretation and control of the link control signaling is normally associated with the resource manager's scheduler.

#### 2.1.2.5 PHY

The PHY block is responsible for transmitting and receiving packets of information on the physical channel. A control path between the baseband and the PHY block allows the baseband block to control the timing and frequency carrier of the PHY block. The PHY block transforms a stream of data to and from the physical channel and the baseband into required formats.

#### 2.1.2.6 Isochronous Adaptation Layer

The Isochronous Adaptation Layer (ISOAL) enables the upper layer to send or receive isochronous data to or from the Link Layer in a flexible way such that the size and interval of data packets in the upper layer can be different from the size and interval of data packets in the Link Layer. The ISOAL uses fragmentation/recombination or segmentation/reassembly operations to convert upper layer data units into lower layer data units (or the other way around).

#### 2.1.2.7 Channel Sounding

The Channel Sounding block is responsible for creation, modification, and release of Channel Sounding physical links. Channel Sounding related capabilities are first exchanged between peer Channel Sounding blocks and procedure configuration parameters are established. Security parameters are then set up. Thereafter, Channel

Sounding exchanges are coordinated via this block, which includes Channel Sounding event, subevent, and step timing. Step transmission and reception is also generated and coordinated through this block. This data is then sent to the Host. From these exchanges, data is collected that may be used by the Host for distance estimation and attack detection.

### 2.1.3 [This section is no longer used]

# 3 TRANSPORT ARCHITECTURE

The Bluetooth data transport system follows a layered architecture. This description of the Bluetooth system describes the Bluetooth core transport layers up to and including L2CAP channels. All Bluetooth operational modes follow the same generic transport architecture, which is shown in Figure 3.1.

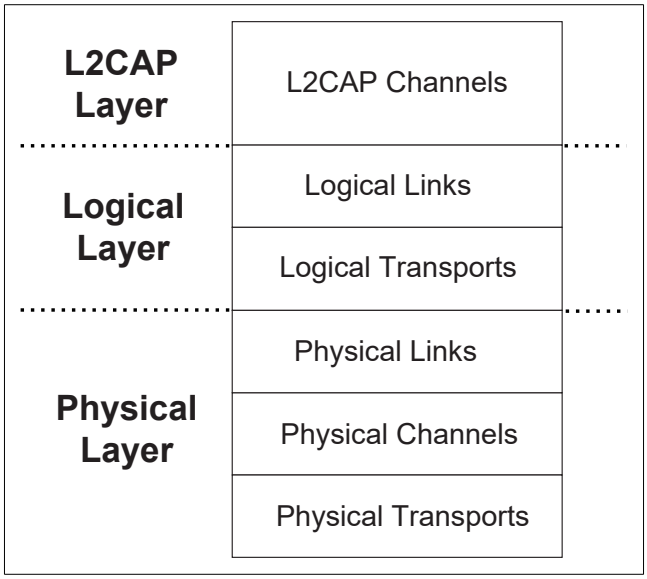



*Figure 3.1: Bluetooth generic data transport architecture*

For efficiency and legacy reasons, the Bluetooth transport architecture includes a sub-division of the logical layer, distinguishing between logical links and logical transports. This sub-division provides a general (and commonly understood) concept of a logical link that provides an independent transport between two or more devices. The logical transport sub-layer is required to describe the inter-dependence between some of the logical link types (mainly for reasons of legacy behavior).

The ACL, SCO, and eSCO connections are considered as logical transports but often behave as separate physical links. However, they are not as independent as might be desired, due to their shared use of resources such as the LT_ADDR and acknowledgment/repeat request (ARQ) scheme. Hence the architecture is incapable of representing these logical transports with a single transport layer. The additional logical transport layer goes some way towards describing this behavior.

## 3.1 Core traffic bearers

The Bluetooth core system provides a number of standard traffic bearers for the transport of service protocol and application data. These are shown in Figure 3.2 below

(for ease of representation this is shown with higher layers to the left and lower layers to the right).

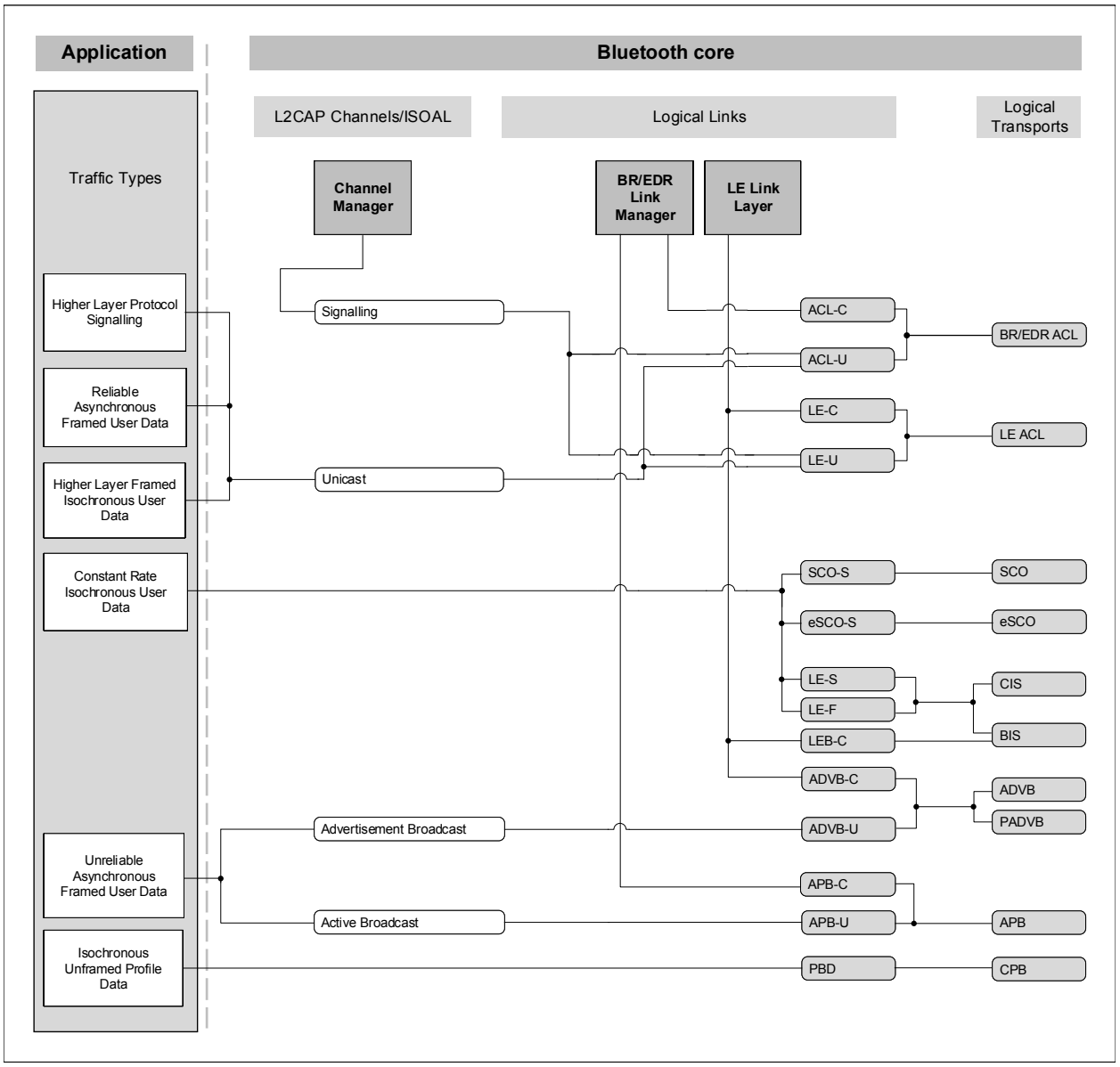



*Figure 3.2: Bluetooth traffic bearers*

The core traffic bearers that are available to applications are shown in Figure 3.2 as the shaded rounded rectangles. The architectural layers that are defined to provide these services are described in Section 2. A number of data traffic types are shown on the left of the diagram linked to the traffic bearers that are typically suitable for transporting that type of data traffic.

The logical links are named using the names of the associated logical transport and a suffix that indicates the type of data that is transported. (C for control links carrying LMP

or LL messages, U for L2CAP links carrying user data (L2CAP PDUs) and S for stream links carrying unformatted synchronous or isochronous data.) It is common for the suffix to be removed from the logical link without introducing ambiguity, thus a reference to the default ACL logical transport can be resolved to mean the ACL-C logical link in cases where the LMP protocol is being discussed, the LE-C logical link in cases where LL protocol is being discussed, or the ACL-U or LE-U logical links when the L2CAP layer is being discussed.

The mapping of application traffic types to Bluetooth core traffic bearers in Figure 3.2 is based on matching the traffic characteristics with the bearer characteristics. It is recommended to use these mappings as they provide the most natural and efficient method of transporting the data with its given characteristics.

However, an application (or an implementation of the Bluetooth core system) may choose to use a different traffic bearer, or a different mapping to achieve a similar result. For example, in a BR/EDR piconet with only one Peripheral, the Central may choose to transport L2CAP broadcasts over the ACL-U logical link rather than over the APB-U logical link. This will probably be more efficient in terms of bandwidth (if the physical channel quality is not too degraded). Use of alternative transport paths to those in Figure 3.2 is only acceptable if the characteristics of the application traffic type are preserved.

Figure 3.2 shows a number of application traffic types. These are used to classify the types of data that may be submitted to the Bluetooth core system. The original data traffic type can be different from the type that is submitted to the Bluetooth core system if an intervening process modifies it. For example, video data is generated at a constant rate but an intermediate coding process may alter this to variable rate, e.g. by MPEG4 encoding. For the purposes of the Bluetooth core system, only the characteristic of the submitted data is of interest.

### 3.1.1 Framed data traffic

The L2CAP layer services provide a frame-oriented transport for asynchronous and isochronous user data. The application submits data to this service in variable-sized frames (up to a negotiated maximum for the channel) and these frames are delivered in the same form to the corresponding application on the remote device. There is no requirement for the application to insert additional framing information into the data, although it may do so if this is required (such framing is invisible to the Bluetooth core system).

Connection-oriented L2CAP channels may be created for transport of unicast (point-to-point) data between two Bluetooth devices. Connection-oriented channels provide a context within which specific properties may be applied to data transported on the channel. For example, quality of service parameters or flow and error control modes

may be applied. Connection-oriented L2CAP channels are created using the L2CAP connection procedure.

A connectionless BR/EDR L2CAP channel exists for broadcasting data or for transport of unicast data. In the case of piconet topologies the Central is always the source of broadcast data and the Peripheral(s) are the recipients. Broadcast traffic on the connectionless L2CAP channel is uni-directional. Unicast data sent on the connectionless L2CAP channels may be uni-directional or bi-directional. Unicast data sent on the L2CAP connectionless channel provides an alternate mechanism to send data with the same level of reliability as an L2CAP connection-oriented channel operating in Basic mode but without the additional latency incurred by opening an L2CAP connection-oriented channel. LE L2CAP connectionless channels are not supported.

BR/EDR L2CAP channels have an associated QoS setting that defines constraints on the delivery of the frames of data. These QoS settings may be used to indicate (for example) that the data is isochronous, and therefore has a limited lifetime after which it becomes invalid, or that the data should be delivered within a given time period, or that the data is reliable and should be delivered without error, however long this takes.

Some L2CAP channels are fixed channels created when the ACL-U and/or LE-U logical links are established. These fixed channels have fixed channel identifiers and fixed configurations and do not permit negotiation of the configuration after they are created. These fixed channels are used for BR/EDR and LE L2CAP signaling (ACL-U or LE-U), connectionless channel (ACL-U and APB-U), Security Manager Protocol (LE-U), and Attribute Protocol (ACL-U or LE-U).

The L2CAP channel manager is responsible for arranging to transport the L2CAP channel data frames on an appropriate baseband logical link, possibly multiplexing this onto the baseband logical link with other L2CAP channels with similar characteristics.

### 3.1.2 Unframed data traffic

If the application does not require delivery of data in frames, possibly because it includes in-stream framing, or because the data is a pure stream, then it may avoid the use of L2CAP channels and make direct use of a baseband logical link.

The Bluetooth core system supports the direct transport of application data that is isochronous and of a constant rate (either bit-rate, or frame-rate for pre-framed data), using a SCO-S or eSCO-S logical link. These logical links reserve physical channel bandwidth and provide a constant rate transport locked to the piconet clock. Data is transported in fixed size packets at fixed intervals with both of these parameters negotiated during channel establishment. eSCO links provide a greater choice of bit-rates and also provide greater reliability by using limited retransmission in case of error. Enhanced Data Rate operation is supported for eSCO, but not for SCO logical

transports. SCO and eSCO logical transports do not support multiplexed logical links or any further layering within the Bluetooth core. An application may choose to layer a number of streams within the submitted SCO/eSCO stream, provided that the submitted stream is, or has the appearance of being, a constant rate stream.

The Bluetooth core system also supports the direct transport of application data using a Profile Broadcast Data (PBD) logical link. This logical link is similar to SCO-S and eSCO-S since it reserves physical channel bandwidth, provides a constant rate transport locked to the piconet clock, and transports data at fixed intervals. It does not support multiplexed logical links or any further layering within the Bluetooth core but, unlike SCO-S and eSCO-S, it supports broadcasting data from a single transmitter to many receivers.

The application chooses the most appropriate type of logical link from those available at the baseband, and creates and configures it to transport the data stream, and releases it when completed. (The application will normally also use a framed L2CAP unicast channel to transport its C-plane information to the peer application on the remote device.)

If the application data is isochronous and of a variable rate, then this may only be carried by the L2CAP unicast channel, and hence will be treated as framed data.

Unframed data traffic is not supported in the LE system.

### 3.1.3 Reliability of traffic bearers

A link or channel is characterized as reliable if the receiver is capable of detecting errors in received packets and requesting retransmission until the errors are removed. This is known as an Automatic Repeat reQuest (ARQ) scheme. Due to the error detection systems used, some residual undetected errors may still remain in the received data. The rate at which these occur depends on the details of the error detection system.

A link or channel is characterized as unreliable if the receiver is not capable of detecting errors in received packets or if it can detect errors but cannot request retransmission. In the latter case (such as with most broadcast links), the packets passed on by the receiver to higher layers may be without error but there is no guarantee that all the packets that were sent are received. Uses for unreliable links are normally dependent on techniques to improve the redundancy of the transmission, such as the use of Forward Error Connection or the repetition of data from the higher layers while the data is valid, in order to increase the probability that the receiver is able to receive at least one of the copies successfully.

#### 3.1.3.1 BR/EDR reliability

Bluetooth is a wireless communications system. In poor RF environments, this system should be considered inherently unreliable. To counteract this the system provides

levels of protection at each layer. The baseband packet header uses forward error correcting (FEC) coding to allow error correction by the receiver and a header error check (HEC) to detect errors remaining after correction. Certain Baseband packet types include FEC for the payload. Furthermore, some Baseband packet types include a cyclic redundancy error check (CRC).

On ACL logical transports the results of the error detection algorithm are used to drive a simple ARQ protocol. This provides an enhanced reliability by re-transmitting packets that do not pass the receiver's error checking algorithm. It is possible to modify this scheme to support latency-sensitive packets by discarding an unsuccessfully transmitted packet at the transmitter if the packet's useful life has expired. eSCO links use a modified version of this scheme to improve reliability by allowing a limited number of retransmissions.

The resulting reliability gained by this ARQ scheme is only as dependable as the ability of the HEC and CRC codes to detect errors. In most cases this is sufficient, however it has been shown that for the longer packet types the probability of an undetected error is too high to support typical applications, especially those with a large amount of data being transferred.

The L2CAP layer provides an additional level of error control that is designed to detect the occasional errors not detected by the baseband and request retransmission of the affected data. This provides the level of reliability required by typical Bluetooth applications. The resulting rate of residual errors is comparable to the rate in other communication systems.

The transmitter may remove packets from the transmit queue such that the receiver does not receive all the packets in the sequence. If this happens detection of the missing packets is delegated to the L2CAP layer.

Stream links have a reliability characteristic somewhere between a reliable and an unreliable link, depending on the current operating conditions.

#### 3.1.3.2 LE reliability

Like BR/EDR, in poor RF environments, the LE system should be considered inherently unreliable. To counteract this, the system provides levels of protection at each layer. The LL packet uses a 24-bit cyclic redundancy error check (CRC) to cover the contents of the packet payload. If the CRC verification fails on the packet payload, the packet is not acknowledged by the receiver and the packet gets retransmitted by the sender.

Because of the longer CRC and the shorter typical message compared with BR/EDR, it is not necessary for the L2CAP layer to provide a separate error detection and retransmission mechanism.

#### 3.1.3.3 [This section is no longer used]

## 3.2 Transport architecture entities

The Bluetooth transport architecture entities are shown in Figure 3.3 and are described from the lowest layer upwards in the subsequent sections.

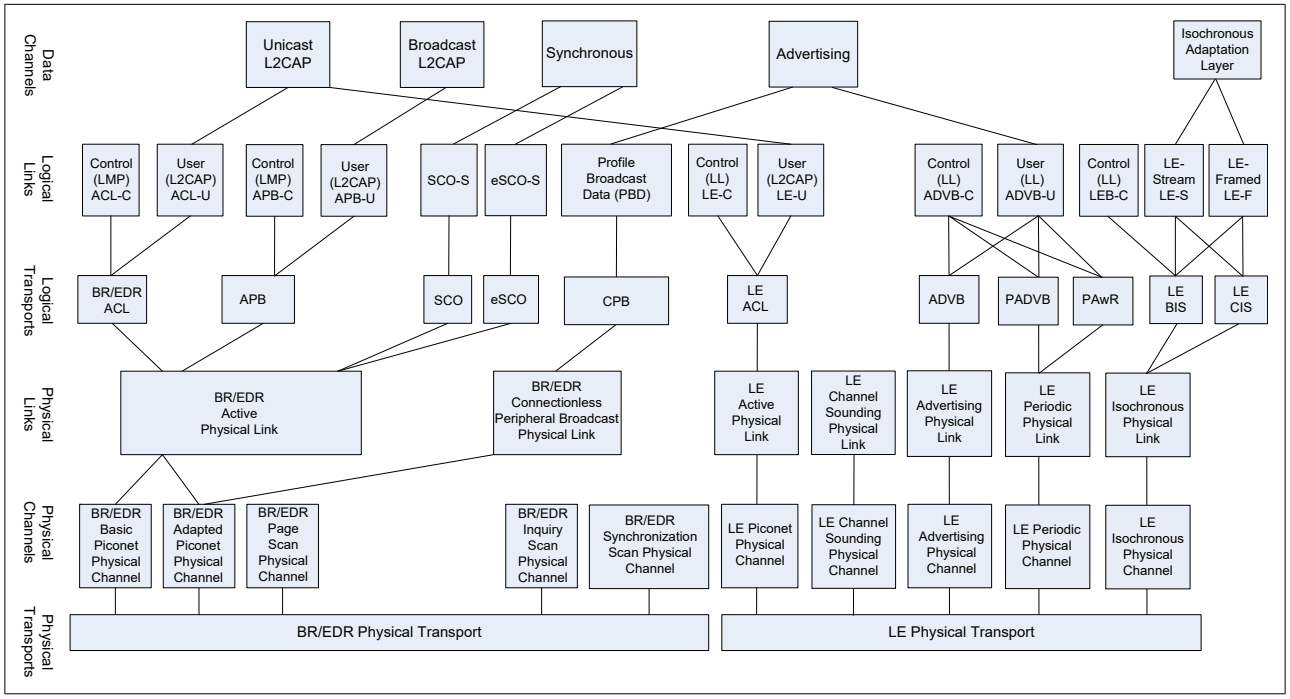



*Figure 3.3: Overview of transport architecture entities and hierarchy*

The BR/EDR Physical Transport encapsulates the BR/EDR Physical Channels. Transfers using the BR/EDR Physical Transport use the BR/EDR Generic Packet Structure. The LE Physical Transport encapsulates the LE Physical Channels. Transfers using the LE Physical Transport use the LE Generic Packet Structure for transferring user data and LL control information. The LE Physical Transport is also used for transfers using the Channel Sounding generic packet structure and signaling format.

### 3.2.1 BR/EDR generic packet structure

The generic packet structure nearly reflects the architectural layers found in the Bluetooth BR/EDR system. The BR/EDR packet structure is designed for optimal use in normal operation. It is shown in Figure 3.4.

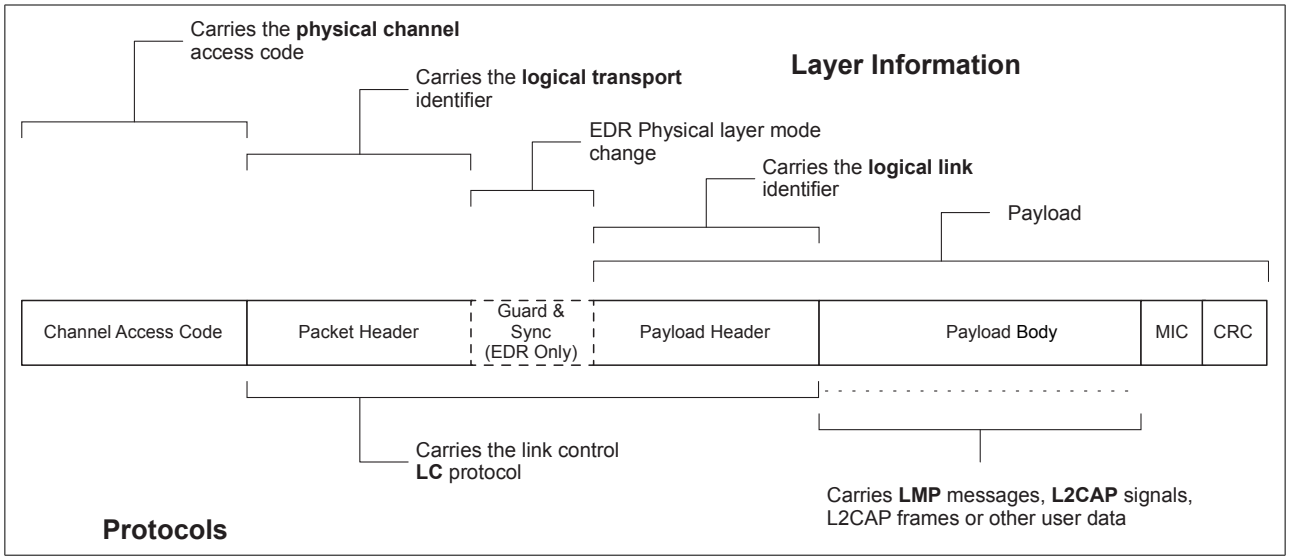



*Figure 3.4: BR/EDR packet structure*

Packets normally only include the fields that are necessary to represent the layers required by the transaction. Thus a simple inquiry request over an inquiry scan physical channel does not create or require a logical link or higher layer and therefore consists only of the channel access code (associated with the physical channel).

All packets include the channel access code. This is used to identify communications on a particular physical channel and therefore ignore packets on a different physical channel that happens to be using the same RF carrier in physical proximity.

There is no direct field within the BR/EDR packet structure that represents or contains information relating to physical links. This information is implied by the combination of the logical transport address (LT_ADDR) carried in the packet header and the channel access code (CAC).

Most BR/EDR packets include a packet header. The packet header is always present in packets transmitted on physical channels that support physical links, logical transports and logical links. The packet header carries the LT_ADDR, which is used by each receiving device to determine if the packet is addressed to the device and is used to route the packet internally.

The BR/EDR packet header also carries part of the link control (LC) protocol that is operated per logical transport (except for ACL and SCO transports that operate a shared LC protocol carried on either logical transport).

The Enhanced Data Rate (EDR) packets have a guard time and synchronization sequence before the payload. This is a field used for physical layer change of modulation scheme.

The payload header is present in all packets on logical transports that support multiple logical links. The payload header includes a logical link identifier field used for routing the payload, and a field indicating the length of the payload body. Some packet types also include a CRC at the end of the packet payload that is used to detect most errors in received packets. When AES-CCM encryption is enabled, ACL packets include a Message Integrity Check (MIC) just prior to the CRC.

EDR packets have a trailer after the CRC.

The packet payload body is used to transport the user data. The interpretation of this data is dependent on the logical transport and logical link identifiers. For ACL logical transports Link Manager protocol (LMP) messages and L2CAP signals are transported in the packet payload body, along with general user data from applications.

For SCO, eSCO, and CPB logical transports the payload body contains the user data for the logical link.

### 3.2.2 LE generic packet structure

LE radio operation is based on four PHYs and makes use of two modulation symbol rates. Table 3.1 summarizes the properties of each of the LE PHYs. Each packet transmitted uses a single PHY. Three of the PHYs are uncoded - that is, each bit maps directly to a single radio symbol in the packet - while the fourth PHY is error correction coded. There are two coding schemes: S=8 and S=2, where S is the number of symbols per bit.

| PHY | Modulation symbol rate | Modulation bandwidth-symbol time product (BT) | Coding scheme | | Data rate |
|---|---|---|---|---|---|
| | | | Access Header | Payload | |
| LE 1M | 1 Msym/s modulation | BT=0.5 | Uncoded | Uncoded | 1 Mb/s |
| LE 2M | 2 Msym/s modulation | BT=0.5 | Uncoded | Uncoded | 2 Mb/s |
| LE 2M 2BT[1] | 2 Msym/s modulation | BT=2.0 | Uncoded | Uncoded | 2 Mb/s |
| LE Coded | 1 Msym/s modulation | BT=0.5 | S=8 | S=8 | 125 kb/s |
| | | | | S=2 | 500 kb/s |

*Table 3.1: Summary of PHYs, modulation schemes, and coding schemes*

[1]LE 2M 2BT is used only for Channel Sounding.

The "Access Header" referred to in Table 3.1 includes all the bits in the packet format associated with the particular PHY prior to the start of the PDU Header but not including the preamble. The preamble is uncoded for all PHYs.

The "Payload" referred to in Table 3.1 includes all the bits in the packet format from the PDU Header to the end of the packet.

The general structure of the Link Layer Air Interface packet closely reflects the architectural layers found in the LE system. The packet structure for the LE Uncoded PHYs is designed for optimal use in normal operation and is shown in Figure 3.5.

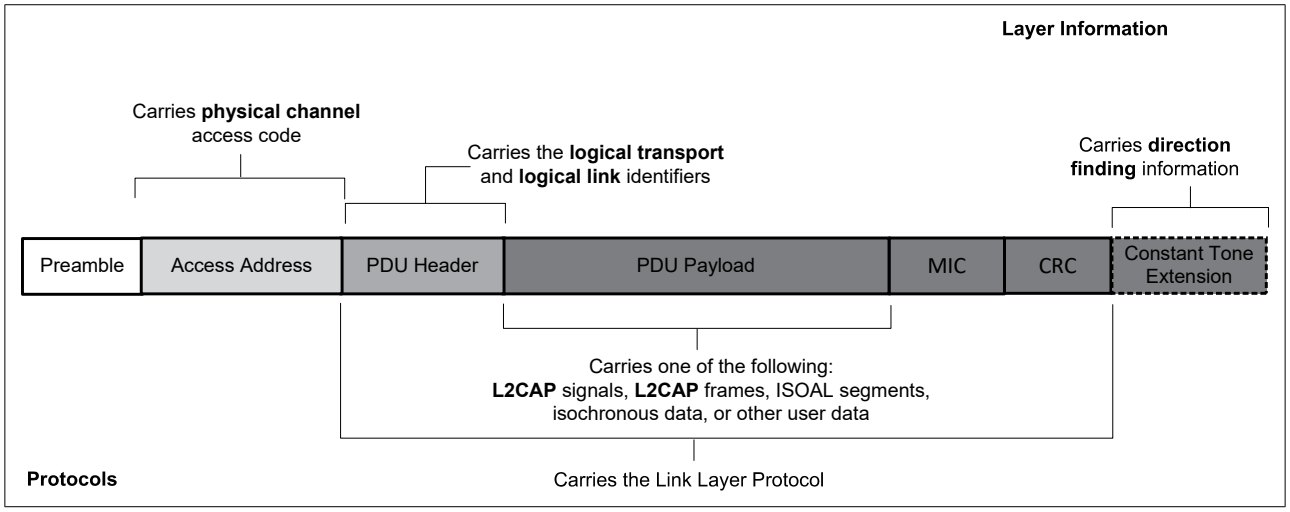



*Figure 3.5: The packet structure for the LE Uncoded PHYs*

The packet structure for the LE Coded PHY is designed for optimal use in extended range operation and is shown in Figure 3.6.

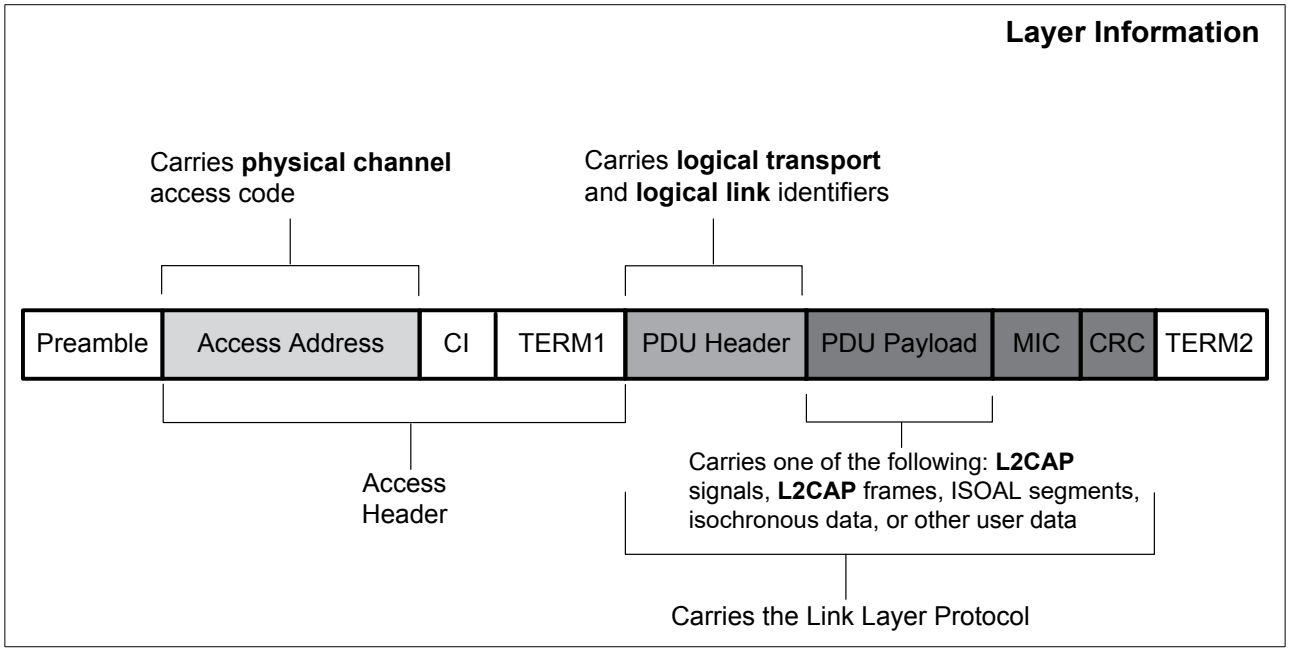



*Figure 3.6: The packet structure for the LE Coded PHY*

When using the LE Coded PHY, it is recommended to carefully consider the impact of radio-on time for power consumption and duty cycle for scheduling and coexistence over the air. The LE Coded PHY with S=8 coding (125 kb/s) represents the worst case,

when considering radio-on time and duty cycle, where each packet sent over the air will be approximately 8 times larger than LE 1M.

Table 3.2 illustrates the on-air time of advertising events with different sizes of AdvData. The first is using connectable and scannable undirected advertising events where the AdvData is sent on the primary advertising physical channel. The second is using events where the AdvData is offloaded to the secondary advertising physical channel. The usage of the primary and secondary advertising physical channels is described in Section 3.3.2.2. Numbers in parentheses are hypothetical and show cases that are not valid.

| AdvData [Bytes] | Connectable Undirected Advertising event [µs] | | Connectable Undirected Advertising event Using Offloading [µs] | |
|---|---|---|---|---|
| | LE 1M | LE Coded S=8 | LE 1M | LE Coded S=8 |
| 0 | 384 | (3,312) | 568 | 4,864 |
| 15 | 744 | (6,192) | 688 | 5,824 |
| 31 | 1,128 | (9,264) | 816 | 6,848 |
| 100 | (2,784) | (22,512) | 1,368 | 11,264 |
| 245 | (6,264) | (50,352) | 2,528 | 20,544 |

*Table 3.2: On-air time for various advertising events*

Note: The events without offloading were calculated using three ADV_IND PDUs, while the events with offloading used three ADV_EXT_IND PDUs containing only the AuxPtr and ADI fields plus one AUX_ADV_IND PDU with the AdvA and ADI fields present and holding the AdvData.

Table 3.3 illustrates, for a range of payload sizes, the difference in Link Layer Data Physical Channel PDU packet durations for connections over the LE 1M PHY and LE Coded PHY with S=8 coding. Connection duty cycle for a specific implementation may be easily calculated from this information.

| Payload [bytes] | LL Data Physical Channel PDU [µs] | |
|---|---|---|
| | LE 1M | LE Coded S=8 |
| 0 | 80 | 720 |
| 15 | 200 | 1,680 |
| 31 | 328 | 2,704 |
| 100 | 880 | 7,120 |
| 255 | 2,120 | 17,040 |

*Table 3.3: On-air time for various data physical channel packets not containing Constant Tone Extensions*

The physical link identifier is not contained in the Link Layer Air Interface packet. The physical channel identifiers are either fixed, are determined at connection setup, or are determined at periodic advertising setup. All LE packets include the Access Address. This is used to identify communications on a physical channel, and to omit or ignore packets on different physical channels that are using the same PHY channels in physical proximity. The Access Address determines whether the packet is directed to the advertising physical channel (and thus an advertising physical link) used for non-periodic advertising, the periodic physical channel used for periodic advertising, or to a piconet physical channel (and thus an active physical link to a device). The LE advertising physical channel used for non-periodic advertising uses a fixed Access Address. The LE periodic physical channel used for periodic advertising and LE piconet physical channels use a randomly generated 32-bit value as their Access Address. This provides a high number of periodic advertising trains and a high number of active devices that can be addressed in an LE periodic advertisement or an LE piconet.

All LE packets include a PDU header. The PDU header determines the type of advertisement broadcast or logical link carried over the physical channel.

For advertising physical channel PDUs, the PDU header contains the type of advertisement payload, the device address type for addresses contained in the advertisement, and the advertising physical channel PDU payload length. Most advertising physical channel PDU payloads contain the advertiser's address and advertising data. One advertising physical channel PDU payload only contains the advertiser's device address and the initiator's device address in which the advertisement is directed. Advertising physical channel PDUs with scan requests payloads contain the scanner's device address and the advertiser's device address. Advertising physical channel PDUs with scan responses contain advertiser's device address and the scan response data. Advertising physical channel PDUs with connection request payloads contain the initiator's device address, advertiser's device address and connection setup parameters.

For Data Physical Channel PDUs, the PDU header contains the Logical Link Identifier (LLID), the Next Expected Sequence Number (NESN), Sequence Number (SN), More Data (MD), CTEInfo Present (CP), payload length, and may contain CTEInfo. For Data Physical Channel PDUs that contain control commands, the Data Channel PDU payload contains a command opcode and control data that is specific to the command. There is an optional Message Integrity Check (MIC) value that is used to authenticate the data PDU. For Data Physical Channel PDUs that are data, the Data Physical Channel PDU payload contains L2CAP data.

An Isochronous Physical Channel PDU can be either a Connected Isochronous or Broadcast Isochronous PDU. A Connected Isochronous PDU contains a header and may contain an isochronous payload. The header field contains the Logical Link Identifier (LLID), Sequence Number (SN), Next Expected Sequence Number (NESN),

Close Isochronous Event (CIE), Null PDU Indicator (NPI), and the payload length. A Connected Isochronous PDU may also contain a Message Integrity Check (MIC) field.

A Broadcast Isochronous PDU contains a header and either isochronous or control data. The header field contains the Logical Link Identifier (LLID), the Control Subevent Sequence Number (CSSN), the Control Subevent Transmission Flag (CSTF), and the payload length. The Broadcast Isochronous PDU may also contain a Message Integrity Check (MIC) field.

Both advertising physical channel packets and data physical channel packets can contain a Constant Tone Extension, which can be used for determining the relative direction of a received radio signal.

### 3.2.3 LE Channel Sounding generic packet structure and signaling format

An LE Channel Sounding (CS) operation employs two types of packet structures. The first packet structure makes use of the LE 1M, LE 2M, and LE 2M 2BT PHYs described in Section 3.2.2 and uses the same modulation described for those PHYs. The second packet structure employs the carrier tone modulated with amplitude shift keying.

The first packet structure used with the LE 1M, LE 2M, and LE 2M 2BT PHYs contains a preamble, an Access Address with trailer bits, and an optional payload as shown in Figure 3.7. Unlike the packet formats described in Section 3.2.2, a PDU header is not needed, because the presence and the content type of the optional payload are pre-negotiated.

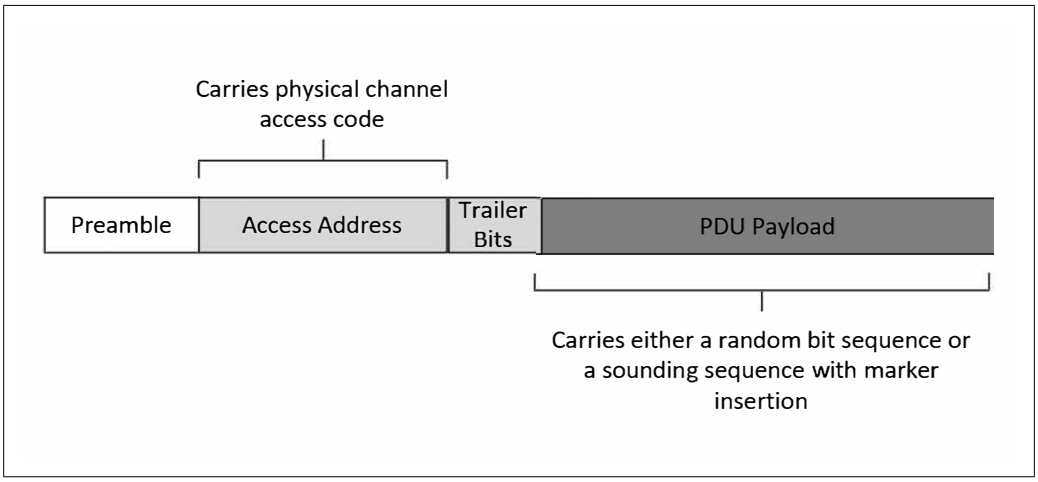



*Figure 3.7: The Channel Sounding packet structure for LE 1M and 2M PHYs*

The Preamble is the same as the preamble described for LE Uncoded PHYs in Section 3.2.2.

Similarly for CS, the Access Address is also the same as described in Section 3.2.2 but contains a bit sequence known only to the CS initiator and reflector pair. This CS Access Address is changed on every transmission.

The Trailer is a 4-bit sequence, alternating between 0 and 1 bits.

The “Payload” also follows the coding scheme rules described in Section 3.2.2. However, its content format is pre-selected and consists of either a bit sequence known only to the initiator and reflector pair, or a repeating [0 1] sounding sequence, with intermittent insertion of markers.

The second packet structure which is modulated using amplitude shift keying consists of one or more transmissions at the channel carrier frequency followed by an optional carrier transmission, over a selected antenna path. An antenna path is defined as single communication path between a specific physical antenna element on a transmitter to a specific physical antenna element on a receiver. The first set of transmissions can repeat over several antenna paths. The presence or non-presence of the final transmission extension concludes this transmission sequence. This format is shown in Figure 3.8.

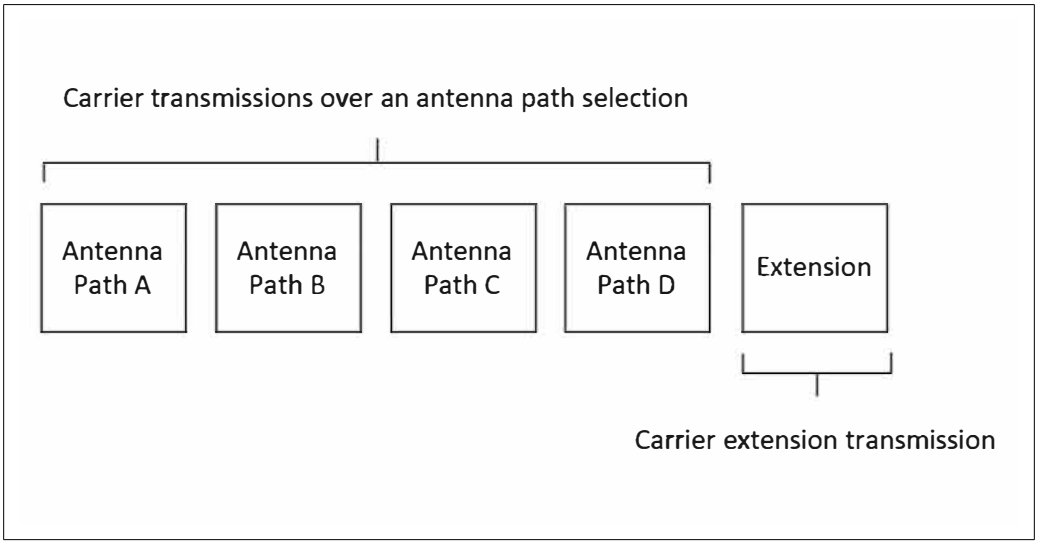



*Figure 3.8: The Channel Sounding packet structure for modulated carrier transmissions.*

## 3.3 Physical channels

A number of types of physical channel are defined. All Bluetooth physical channels are characterized by a set of PHY frequencies combined with temporal parameters and restricted by spatial considerations. For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference and for regulatory reasons.

The Bluetooth BR/EDR system and LE system differ slightly in the way they use physical channels.

### 3.3.1 BR/EDR physical channels

In the BR/EDR core system, peer devices use a shared physical channel for communication. To achieve this their transceivers need to be tuned to the same PHY frequency at the same time, and they need to be within a nominal range of each other.

Given that the number of RF carriers is limited and that many Bluetooth devices may be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This channel access code is a property of the physical channel. The access code is present at the start of every transmitted packet.

Several BR/EDR physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (the basic piconet channel and adapted piconet channel) are used for communication between connected devices and are associated with a specific piconet. Other BR/EDR physical channels are used for discovering (the inquiry scan channel) and connecting (the page scan channel) Bluetooth devices. The synchronization scan physical channel is used by devices to obtain timing and frequency information about the Connectionless Peripheral Broadcast physical link or to recover the current piconet clock.

A Bluetooth device can only use one BR/EDR physical channel at any given time. In order to support multiple concurrent operations the device uses time-division multiplexing between the channels. In this way a Bluetooth device can appear to operate simultaneously in several piconets, as well as being discoverable and connectable.

Whenever a Bluetooth device is synchronized to the timing, frequency and access code of a physical channel it is said to be ‘connected’ to this channel (whether or not it is actively involved in communications over the channel). The Bluetooth specification assumes that a device is only capable of connecting to one physical channel at any time. Advanced devices may be capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

#### 3.3.1.1 Basic piconet channel

##### 3.3.1.1.1 Overview

The basic piconet channel is used for communication between connected devices during normal operation.

#### 3.3.1.1.2 Characteristics

The basic piconet channel is characterized by a pseudo-random sequence hopping through the PHY channels. The hopping sequence is unique for the piconet and is determined by the Bluetooth Device Address of the Central. The phase in the hopping sequence is determined by the Bluetooth clock of the Central. All Bluetooth devices participating in the piconet are time- and hop-synchronized to the channel.

The channel is divided into time slots where each slot corresponds to an PHY hop frequency. Consecutive hops correspond to different PHY hop frequencies. The time slots are numbered according to the Bluetooth clock of the piconet Central. Packets are transmitted by Bluetooth devices participating in the piconet aligned to start at a slot boundary. Each packet starts with the channel access code, which is derived from the Bluetooth Device Address of the piconet Central.

On the basic piconet channel the Central controls access to the channel. The Central starts its transmission in even-numbered time slots only. Packets transmitted by the Central are aligned with the slot start and define the piconet timing. Packets transmitted by the Central may occupy up to five time slots depending on the packet type.

Each Central transmission is a packet carrying information on one of the logical transports. Peripherals may transmit on the physical channel in response. The characteristics of the response are defined by the logical transport that is addressed.

For example, on the asynchronous connection-oriented logical transport (ACL), the addressed Peripheral responds by transmitting a packet containing information for the same logical transport that is nominally aligned with the next (odd-numbered) slot start. Such a packet may occupy up to five time slots, depending on the packet type. On a broadcast logical transport no Peripherals are allowed to respond.

#### 3.3.1.1.3 Topology

A basic piconet channel may be shared by any number of Bluetooth devices, limited only by the resources available on the piconet Central. Only one device is the piconet Central, all others being piconet Peripherals. All communication is between the Central and Peripherals. There is no direct communication between Peripherals on the piconet channel.

There is, however, a limitation on the number of logical transports that can be supported within a piconet. This means that although there is no theoretical limit to the number of Bluetooth devices that share a channel there is a limit to the number of these devices that can be actively involved in exchanging data with the Central.

#### 3.3.1.1.4 Supported layers

The basic piconet channel supports a number of physical links, logical transports, logical links and L2CAP channels used for general purpose communications.

### 3.3.1.2 Adapted piconet channel

#### 3.3.1.2.1 Overview

The adapted piconet channel differs from the basic piconet channel in two ways. First, the frequency on which a Peripheral transmits is the same as the frequency used by its Central in the preceding transmission. In other words the frequency is not recomputed between Central and subsequent Peripheral packets. Second, the adapted piconet channel may be based on fewer than the full 79 frequencies. A number of frequencies may be omitted from the hopping pattern by being marked as “unused”. The remainder of the 79 frequencies are included. The two sequences are the same except that whenever the basic pseudo-random hopping sequence selects an unused frequency, it is replaced with an alternative chosen from the used set. The set of frequencies used may vary between different physical links on the same adapted piconet channel.

Because the adapted piconet channel uses the same timing and access code as the basic piconet channel, physical links on the two channels are often coincident. This provides a deliberate benefit as it allows Peripherals in either the basic piconet channel or the adapted piconet channel to adjust their synchronization to the Central.

The topology and supported layers of the adapted piconet physical channel are identical to the basic piconet physical channel with one exception: on the adapted piconet physical channel, it is possible for a single Central to transmit data to an unlimited number of Peripherals using a single CPB logical transport. In this case, however, data is only transferred from Central to Peripheral and not from Peripheral to Central.

### 3.3.1.3 Inquiry scan channel

#### 3.3.1.3.1 Overview

In order for a device to be discovered, an inquiry scan channel is used. A discoverable device listens for inquiry requests on its inquiry scan channel and then sends a response to that request. In order for a device to discover other devices, it iterates (hops) through all possible inquiry scan channel frequencies in a pseudo-random fashion, sending an inquiry request on each frequency and listening for any response.

#### 3.3.1.3.2 Characteristics

Inquiry scan channels follow a slower hopping pattern and use an access code to distinguish between occasional occupancy of the same radio frequency by two co-located devices using different physical channels.

The access code used on the inquiry scan channel is taken from a reserved set of inquiry access codes that are shared by all Bluetooth devices. One access code is used for general inquiries, and a number of additional access codes are reserved for limited inquiries. Each device has access to a number of different inquiry scan channels. As all of these channels share an identical hopping pattern, a device may concurrently occupy more than one inquiry scan channel if it is capable of concurrently correlating more than one access code.

A device using one of its inquiry scan channels remains passive on that channel until it receives an inquiry message on this channel from another Bluetooth device. This is identified by the appropriate inquiry access code. The inquiry scanning device will then follow the inquiry response procedure to return a response to the inquiring device.

In order for a device to discover other Bluetooth devices it uses the inquiry scan channel to send inquiry requests. As it has no prior knowledge of the devices to discover, it cannot know the exact characteristics of the inquiry scan channel.

The device takes advantage of the fact that inquiry scan channels have a reduced number of hop frequencies and a slower rate of hopping. The inquiring device transmits inquiry requests on each of the inquiry scan hop frequencies and listens for an inquiry response. Transmissions are done at a faster rate, allowing the inquiring device to cover all inquiry scan frequencies in a reasonably short time period.

#### 3.3.1.3.3 Topology

Inquiring and discoverable devices use a simple exchange of packets to fulfill the inquiring function. The topology formed during this transaction is a simple and transient point-to-point connection.

#### 3.3.1.3.4 Supported layers

During the exchange of packets between an inquiring and discoverable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

### 3.3.1.4 Page scan channel

#### 3.3.1.4.1 Overview

A connectable device (one that is prepared to accept connections) does so using a page scan channel. A connectable device listens for a page request on its page scan channel and, once received, enters into a sequence of exchanges with this device. In order for a device to connect to another device, it iterates (hops) through all page

scan channel frequencies in a pseudo-random fashion, sending a page request on each frequency and listening for a response.

#### 3.3.1.4.2 Characteristics

The page scan channel uses an access code derived from the scanning device's Bluetooth Device Address to identify communications on the channel. The page scan channel uses a slower hopping rate than the hop rate of the basic and adapted piconet channels. The hop selection algorithm uses the Bluetooth device clock of the scanning device as an input.

A device using its page scan channel remains passive until it receives a page request from another Bluetooth device. This is identified by the page scan channel access code. The two devices will then follow the page procedure to form a connection. Following a successful conclusion of the page procedure both devices switch to the basic piconet channel that is characterized by having the paging device as Central.

In order for a device to connect to another Bluetooth device it uses the page scan channel of the target device in order to send page requests. If the paging device does not know the phase of the target device's page scan channel it therefore does not know the current hop frequency of the target device. The paging device transmits page requests on each of the page scan hop frequencies and listens for a page response. This is done at a faster hop rate, allowing the paging device to cover all page scan frequencies in a reasonably short time period.

The paging device may have some knowledge of the target device's Bluetooth clock (indicated during a previous inquiry transaction between the two devices, or as a result of a previous involvement in a piconet with the device), in this case it is able to predict the phase of the target device's page scan channel. It may use this information to optimize the synchronization of the paging and page scanning process and speed up the formation of the connection.

#### 3.3.1.4.3 Topology

Paging and connectable devices use a simple exchange of packets to fulfill the paging function. The topology formed during this transaction is a simple and transient point-to-point connection.

#### 3.3.1.4.4 Supported layers

During the exchange of packets between a paging and connectable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

#### 3.3.1.5 Synchronization scan channel

##### 3.3.1.5.1 Overview

In order to receive packets sent on the CPB logical transport, a device must first obtain information about the timing and frequency channels of those packets. If a device misses a Coarse Clock Adjustment notification, it needs to recover the current piconet clock. The synchronization scan channel is provided for these purposes. A scanning device listens for synchronization train packets on the synchronization scan channel. Once a synchronization train packet is received, the device may stop listening for synchronization train packets because it has the timing and frequency information necessary to start receiving packets sent on the CPB logical transport or to recover the piconet clock.

##### 3.3.1.5.2 Characteristics

The synchronization scan channel uses an access code derived from the Bluetooth Device Address of the synchronization train transmitter to identify synchronization train packets on the channel. Once a synchronization train packet is received, the scanning BR/EDR Controller may start receiving packets sent on the CPB logical transport, depending on the needs of the Host and any applicable profile(s).

##### 3.3.1.5.3 Topology

The topology formed during this scan is transient and point-to-multipoint. There can be an unlimited number of scanning devices simultaneously receiving synchronization train packets from the same synchronization train transmitter.

##### 3.3.1.5.4 Supported layers

There is a one-way flow of packets from the synchronization train transmitting device to the scanning device(s). This may be considered a temporary physical link that exists only until the scanning device receives the required information. No further architectural layers are considered to be supported.

### 3.3.2 LE physical channels

In the LE core system, two Bluetooth devices use a shared physical channel for communication. To achieve this, their transceivers need to be tuned to the same PHY frequency at the same time, and they need to be within a nominal range of each other.

Given that the number of PHY channels is limited, and many Bluetooth devices can be operating independently within the same spatial and temporal area, there is a strong likelihood of two pairs of independent Bluetooth devices having their transceivers tuned to the same PHY channel, resulting in a collision. Unlike BR/EDR, where an access code is used to identify the piconet, LE uses a randomly generated Access Address to

identify a physical channel between devices. In the event that two devices happen to share the same PHY channel in the same area, the targeted device Access Address is used as a correlator to determine to which device the communication is directed.

Five LE physical channels are defined. Each is optimized and used for a different purpose. The LE piconet physical channel is used for communication between connected devices and is associated with a specific piconet. The LE advertising physical channel is used for broadcasting advertisements to LE devices. These advertisements can be used to discover, connect, or send user data to scanner or initiator devices. The periodic physical channel is used to send user data to scanner devices in periodic advertisements at a specified interval. The LE isochronous physical channel is used to transfer isochronous data between LE devices in an LE piconet or to transfer isochronous data between unconnected LE devices. The LE Channel Sounding physical channel carries security-related information, as well as content used for the purpose of measuring the time, phase, and amplitude of the communication channel between two LE devices in an LE piconet.

An LE device can only use one of these LE physical channels at any given time. In order to support multiple concurrent operations the device uses time-division multiplexing between the channels. In this way a Bluetooth device can appear to support connected devices while simultaneously sending advertising broadcasts.

Whenever an LE device is synchronized to the timing and frequency of the physical channel it is said to be connected or synchronized to this channel (whether or not it is actively involved in communications over the channel). The Bluetooth specification assumes that a device is only capable of connecting to one physical channel at a time. Advanced devices may be capable of connecting or synchronizing simultaneously to more than one physical channel, but the specification does not make this assumption.

Packets on both the LE piconet physical channel and the LE advertisement broadcast channel can contain a Constant Tone Extension that can be used for the purpose of direction finding.

#### 3.3.2.1 LE piconet physical channel

##### 3.3.2.1.1 Overview

The LE piconet physical channel is used for communication between connected LE devices during normal operation.

##### 3.3.2.1.2 Characteristics

The LE piconet physical channel is characterized by the access address, a pseudo-random sequence of PHY channels, and three additional parameters provided by the Central. The first is the channel map that indicates the set of PHY channels used in the

piconet. The second is a pseudo random number used as an index into the complete set of PHY channels. The third is the timing of the first data packet sent by the Central after the connection request.

The channel is divided into connection events where each connection event corresponds to a PHY hop channel. Consecutive connection events correspond to different PHY hop channels. The first packet sent by the Central after the connection establishment sets an anchor point for the timing of all future connection events. In a connection event the Central transmits packets to a Peripheral in the piconet and the Peripheral may respond with a packet of its own.

On the LE piconet physical channel the Central controls access to the channel. The Central starts its transmission in a connection event that occurs at regular intervals. Packets transmitted by the Central are aligned with the connection event start and define the piconet timing.

Each Central transmission contains a packet carrying information on one of the logical transports. The Peripheral can transmit on the physical channel in response.

The LE piconet physical channel is similar to the BR/EDR adapted piconet channel in that the set of PHY channels used can be modified to avoid interference. The set of used channels in the channel map is established by the Central during connection setup. While in a connection the Central can change the channel map when necessary to avoid new interferers. The Peripheral can provide channel classification information to the Central.

There are 37 LE piconet channels. The Central can reduce this number through the channel map indicating the used channels. When the hopping pattern hits an unused channel the unused channel is replaced with an alternate from the set of used channels. The LE Piconet physical channel can use any LE PHY.

#### 3.3.2.1.3 Topology

An LE piconet physical channel is shared by exactly two LE devices.

An LE device may belong to one or more piconets at a time, that is, an LE device may be a Peripheral in zero or more piconets and may also be a Central in zero or more piconets.

Only one LE piconet physical channel can exist between two LE device identities or non-resolvable private addresses.

#### 3.3.2.1.4 Supported layers

The LE piconet physical channel supports L2CAP channels used for general purpose communications.

#### 3.3.2.2 Advertising physical channels

##### 3.3.2.2.1 Overview

An LE advertising physical channel is used to set up connections between two devices or to communicate broadcast information between unconnected devices.

##### 3.3.2.2.2 Characteristics

There are two LE advertising physical channels: the primary advertising physical channel and the secondary advertising physical channel.

The primary advertising physical channel is a set of three fixed PHY channels spread evenly across the LE frequency spectrum. The number of primary advertising PHY channels can be reduced by the advertising device in order to reduce interference. The primary advertising physical channel can use either the LE 1M or LE Coded PHY.

The primary advertising physical channel is divided into advertising events where each advertising event can hop on all primary advertising PHY channels. The advertising events occur at regular intervals which are slightly modified with a random delay to aid in interference avoidance.

On the primary advertising physical channel the advertising device controls access to the physical channel. The advertising device starts its transmission in an advertising event and transmits advertising packets on one or more of the primary advertising PHY channels. Each advertising packet is sent on a different advertising PHY channel at a fixed interval. Seven types of advertising events can be used, with each advertising event type having different sized advertising packets. The PDU payloads of these advertising packets can vary in length from 6 to 37 octets.

Some advertising events sent by the advertising device permit the listening device to concurrently send scan requests or connection requests packets on the same advertising PHY channel in which the advertising packet was received. The advertising device can send a scan response packet again on the same advertising PHY channel within the same advertising event. The payload of the scan response packet can vary in length from 6 to 37 octets.

The secondary advertising physical channel is a set of 37 fixed PHY channels spread across the LE frequency spectrum. These are the same fixed LE PHY channels used by the data physical channel. The secondary advertising physical channel uses the same channel indices as the data physical channel. The payload of advertising packets used on the secondary advertising physical channel can vary in length from 0 to 255 octets. Advertising packets on the secondary advertising physical channel are not part of the advertising event but are part of the extended advertising event. These extended advertising events begin at the same time as the advertising event on the primary

advertising physical channel and conclude with the last packet on the secondary advertising physical channel.

The secondary advertising physical channel is used to offload data that would otherwise be transmitted on the primary advertising physical channel. Advertising packets on the secondary advertising physical channel ("auxiliary packets") are scheduled by the advertiser when sufficient over-the-air time is available. The advertising packet on the primary advertising physical channel contains the PHY channel and the offset to the start time of the auxiliary packet.

The secondary advertising physical channel can use any LE PHY. All advertising packets on the secondary advertising physical channel in the same extended advertising event use the same PHY, which is specified in the advertising packet on the primary advertising physical channel.

#### 3.3.2.2.3 Topology

An LE advertising physical channel can be shared by any number of LE devices. Any number of LE devices can transmit advertising packets while sharing the advertising physical channel. Any number of scanning devices can listen on the advertising physical channel. An advertising device can advertise and be connected on an LE piconet physical channel simultaneously. Scanning devices may also be connected to one or more LE piconet physical channels simultaneously.

### 3.3.2.3 Periodic physical channel

#### 3.3.2.3.1 Overview

An LE periodic physical channel is used to set up a periodic broadcast between unconnected devices.

#### 3.3.2.3.2 Characteristics

The periodic physical channel is characterized by a pseudo-random sequence of PHY channels and additional parameters provided by the advertiser. These are the channel map that indicates the set of PHY channels used in the periodic broadcast, the event counter used to determine the channel hopping sequence, the offset indicating the timing of the first periodic broadcast packet, and the interval between successive periodic broadcasts.

The channel is divided into periodic advertising events where the start of a periodic advertising event corresponds to a PHY hop channel. The start of consecutive periodic advertising events corresponds to different PHY hop channels. The first packet sent by the advertiser after the broadcast is established sets an anchor point for the timing of all future periodic advertising events.

On the periodic physical channel, the advertising device controls access to the physical channel. The advertiser starts its transmission in a periodic advertising event that occurs at regular intervals. Packets transmitted by the advertiser are aligned with the periodic advertising event and specified broadcast timing. Additional packets may also be transmitted between the periodic advertising events. The payload of packets sent by the advertiser may vary in length from 0 octets to 255 octets.

Each advertiser transmission contains a packet carrying information on the PADVB logical transports. Scanners cannot transmit on the physical channel.

There are 37 PHY channels. The advertiser can reduce this number through the channel map indicating the used channels. When the hopping pattern hits an unused channel, the unused channel is replaced with an alternate from the set of used channels. The periodic physical channel can use any PHY. All periodic advertising events use the same PHY used by the advertiser in the packet describing the characteristics of the periodic physical channel.

#### 3.3.2.3.3 Topology

An LE periodic physical channel can be shared by any number of LE devices. Any number of LE devices can transmit periodic advertising packets while sharing the same periodic physical PHY channels. Any number of scanning devices can listen on the periodic physical channel. An advertising device can advertise and be synchronized on an LE periodic physical channel simultaneously. Scanning devices may also be synchronized to one or more LE periodic physical channels simultaneously.

### 3.3.2.4 LE Isochronous physical channel

The LE isochronous physical channel can be created to transfer isochronous data between LE devices.

#### 3.3.2.4.1 Overview

The LE isochronous physical channel is used to transfer isochronous data between connected or unconnected LE devices.

#### 3.3.2.4.2 Characteristics

The LE isochronous physical channel is characterized by a pseudo-random sequence of PHY channels and by three additional parameters that are provided by a Central or a connectionless broadcaster. The first parameter is the channel map that indicates the set of PHY channels. The second parameter is a pseudo random number that is used as an index into the complete set of PHY channels. The third parameter is the timing of the first data packet. The timing of the first packet of a CIS is provided in the Link Layer message that is sent in the associated ACL connection by the Central during the

CIS establishment phase. The timing of the first packet of a BIS is referenced from a periodic advertising event associated with the BIS.

The LE isochronous physical channel is used to transfer isochronous data in isochronous events that occur at regular intervals. Each isochronous event is divided into one or more subevents. Each subevent uses a PHY channel that is selected by the channel selection algorithm.

In any subevent in an isochronous connection, the Central transmits a packet to the Peripheral and the Peripheral may respond with a packet of its own. The Central controls the access to the LE isochronous physical channel. In every CIS event, the Central starts its transmission at the start of the first subevent. Packets that are transmitted by the Central are time aligned with the start of every subevent.

A Broadcasting Isochronous transmitter transmits isochronous data packets and control packets. Any device that is synchronized to the BIS can receive these packets. The broadcasting device controls access to the LE isochronous physical channel. Within BIS events, the broadcasting device starts its transmission in the first subevent. Packets that are transmitted by the broadcasting device are aligned with the start of every subevent.

There are 37 PHY channels. The Central or the isochronous stream transmitter can reduce this number through the channel map that indicates the used channels. When the channel selection algorithm selects an unused channel, the unused channel is replaced with an alternate from the set of used channels. For CISes, the LE isochronous physical channel uses the set of PHY channels that are enabled on the LE piconet physical channel. The LE isochronous physical channel can use any LE PHY.

#### 3.3.2.4.3 Topology

The LE isochronous physical channel in a CIS can be used for one-to-one communication between the devices that are in the LE piconet. The Central may establish one or more CISes with the Peripheral in the LE piconet; that is, the LE isochronous physical channel can carry one or more CIS logical transports between a given Central and Peripheral. The LE isochronous physical channel and all the CISes it carries are terminated when the associated LE piconet physical channel is terminated. If a Central has established piconets with more than one Peripheral, it can establish LE isochronous physical channels with more than one of these Peripherals at the same time.

The LE isochronous physical channel can be used for one-to-many communication topologies of unconnected LE devices. Each LE isochronous physical channel can carry one or more BIS logical transports.

#### 3.3.2.5 LE Channel Sounding physical channel

The LE CS physical channel can be created to transfer time and phase information that can be used to generate a distance estimate.

##### 3.3.2.5.1 Overview

An LE CS physical channel is used to exchange time-interleaved modulated tones and packets that allow a pair of devices to measure the time of flight, amplitude, and phase of signals sent over the communication channel. This information can be used to estimate the distance between two LE devices.

##### 3.3.2.5.2 Characteristics

The LE CS physical channel is characterized by a pre-negotiated sequence of exchanges that make up a CS procedure. Characteristics and content of these exchanges is derived from the output of a Deterministic Random Bit Generator (DRBG). The DRBG is used to generate the content of the CS Access Address and payload content. It is also used to influence the modulated content of tone transmissions. Finally, the DRBG is used to seed the selection of the pseudo-random sequence of the PHY channels used in the channel hop sequence of a Channel Sounding procedure.

The duration of a CS procedure is determined by the time it takes for a predetermined number of CS steps to be exchanged. Additional time may be required to partition the CS steps into subevents in order to coexist with other activities using the same radio or spectrum.

CS steps are defined as a bidirectional exchange between initiator and reflector devices. These steps are further aggregated into a related timing group known as a CS subevent. These subevents are timed from an offset of the underlying LE piconet physical channel connection event anchor point. A CS event is defined as the group of all CS subevents offset from the same LE piconet physical channel connection event anchor point.

Each step exchange may include a modulated packet, a modulated tone, or both. The initiator device transmits first, followed by at least one transmission from the reflector. A modulated tone may be transmitted more than once in each direction in cases where multiple antenna paths are employed.

##### 3.3.2.5.3 Topology

An LE Channel Sounding physical channel is used for one-to-one communication between two devices that are in the same piconet, where one is in the initiator role and the other is in the reflector role. In the context of CS, an initiator is the device that starts the CS procedure, and a reflector is the device that responds. The procedure's operating parameters are exchanged in Link Layer control messages. LE CS physical

channel timing depends on the timing of the initiator device, which is followed by the reflector device. The LE Central device may assume either the initiator or reflector role. Likewise, the LE Peripheral device may assume either role.

An LE device can use only one LE Channel Sounding physical channel at a time. To support multiple concurrent Channel Sounding operations, the device uses time-division multiplexing between multiple LE Channel Sounding physical channels, which allows a device to appear to support multiple connected devices with Channel Sounding physical channels.

### 3.3.3 [This section is no longer used]

## 3.4 Physical links

A physical link represents a baseband connection between Bluetooth devices. A physical link is always associated with exactly one physical channel (although a physical channel may support more than one physical link). Within the Bluetooth system a physical link is a virtual concept that has no direct representation within the structure of a transmitted packet.

In BR/EDR the access code packet field, together with the clock and address of the Central Bluetooth device, is used to identify a physical channel. In LE, the access address and channel map, including *hopIncrement* in the case of Channel Selection Algorithm #1 or an event counter in the case of Channel Selection Algorithm #2, are used to identify a physical channel. For BR/EDR and LE, there is no subsequent part of the packet that directly identifies the physical link. Instead, the physical link may be identified by association with the logical transport, as each logical transport is only received on one physical link.

Some physical link types have properties that may be modified. An example of this is the transmit power for the link. Other physical link types have no modifiable properties. In the case of BR/EDR physical links with modifiable properties the LM protocol is used to adapt these properties. In the case of LE physical links with modifiable properties the LL protocol is used to adapt these properties. As the LM protocol (BR/EDR) or LL protocol (LE) is supported at a higher layer (by a logical link) the appropriate physical link is identified by implication from the logical link that transports the LM or LL signaling.

In the situation where a transmission is broadcast over a number of different physical links, then the transmission parameters are selected to be suitable for all of the physical links.

### 3.4.1 BR/EDR links supported by the basic and adapted piconet physical channels

The basic piconet physical channel supports a physical link which may only be active. The adapted piconet physical channel may support several physical links, including active and Connectionless Peripheral Broadcast. An active physical link is a point-to-point link between the Central and a Peripheral. A Connectionless Peripheral Broadcast physical link is a point-to-multipoint link between the Transmitter (Central) and zero or more Receivers (Peripherals). At least one physical link on the piconet physical channel is always present when a Peripheral is synchronized in the piconet.

#### 3.4.1.1 Active physical link

The physical link between a Central and a Peripheral is active if a default ACL logical transport exists between the devices. Active physical links have no direct identification of their own, but are identified by association with the default ACL logical transport ID with which there is a one-to-one correspondence.

An active physical link has the associated property of radio transmit power in each direction. Transmissions from Peripherals are always directed over the active physical link to the Central, and use the transmit power that is a property of this link in the Peripheral to Central direction. Transmissions from the Central may be directed over a single active physical link (to a specific Peripheral) or over a number of physical links (to a group of Peripherals in the piconet). In the case of point-to-point transmissions the Central uses the appropriate transmit power for the physical link in question. (In the case of point-to-multipoint transmissions the Central uses a transmit power appropriate for the set of devices addressed.)

Active physical links may be placed into Hold or Sniff mode. The effect of these modes is to modify the periods when the physical link is active and may carry traffic. Logical transports that have defined scheduling characteristics are not affected by these modes and continue according to their pre-defined scheduling behavior. The default ACL logical transport and other links with undefined scheduling characteristics are subject to the mode of the active physical link.

#### 3.4.1.2 [This section is no longer used]

#### 3.4.1.3 Connectionless Peripheral Broadcast physical link

A Connectionless Peripheral Broadcast physical link is present on a Receiver (Peripheral) when it is synchronized in the piconet where a CPB logical transport exists. On a Transmitter (Central), a Connectionless Peripheral Broadcast physical link is present when a CPB logical transport exists whether or not any Receivers are synchronized. The Connectionless Peripheral Broadcast physical link is a point-to-multipoint unidirectional link between a Transmitter and zero or more Receivers.

Connectionless Peripheral Broadcast physical links do not support power control because there is no feedback from Receivers to the Transmitter. Traffic is always directed from a single Transmitter to zero or more Receivers.

Connectionless Peripheral Broadcast packets are sent at regular intervals. The BR/EDR Controller selects an interval within a range requested by the Host.

### 3.4.2 BR/EDR links supported by the scanning physical channels

In the case of inquiry scan and page scan channels, the physical link exists for a relatively short time and cannot be controlled or modified in any way. These types of physical links are not further elaborated.

### 3.4.3 LE links supported by the LE physical channels

The LE piconet physical channels support an LE active physical link. The physical link is a point-to-point link between the Central and a Peripheral. It is always present when the Peripheral is in a connection with the Central.

The LE advertising physical channels support an LE advertising physical link. The physical link is a broadcast between the advertiser device and one or more scanner or initiator devices. It is always present when the advertiser is broadcasting advertisement events.

The LE periodic physical channels support an LE periodic physical link. The physical link is a broadcast between the advertiser device and one or more scanner devices. It is always present when the advertiser is broadcasting periodic advertising events.

The LE isochronous physical channels support LE isochronous physical links. An LE isochronous physical link can be a point-to-point link between a Central and a Peripheral or a connectionless link between a broadcast isochronous transmitter and multiple receiving devices.

#### 3.4.3.1 Active physical link

The physical link between a Central and a Peripheral is active if a default LE ACL logical transport exists between the devices. Active physical links are each associated with a separate piconet physical channel, which in turn is identified by the randomly generated Access Address used in the Link Layer packet. Each Access Address has a one-to-one relationship with the Central and the Peripheral of the active physical link.

An active physical link has the associated property of radio transmit power in each direction, which may be different in each direction. A device uses the appropriate transmit power for the physical link in question.

#### 3.4.3.2 Advertising physical link

An advertising physical link between an advertising device and an initiating device for the purposes of forming a connection (active physical link) can exist for a relatively short period of time. These advertising physical links cannot be controlled or modified in any way and these types of physical links are not further elaborated.

An advertising physical link between an advertising device and a scanning device used for periodic broadcasting of user data can exist for longer periods of time. There is no identification information about the physical link within the protocol. The relationship between the advertising and scanning device is established through the use of the Bluetooth Device Address.

#### 3.4.3.3 Periodic physical link

A periodic physical link between an advertising device and one or more scanning devices normally exists for a prolonged period of time. Periodic physical links are each associated with a separate periodic physical channel, which in turn is identified by the randomly generated Access Address used in the Link Layer packet. Each Access Address has a one-to-one relationship with the advertiser of the periodic physical link.

#### 3.4.3.4 Isochronous physical links

The isochronous physical link uses an isochronous physical channel and carries CIS and BIS logical transports.

Isochronous physical links carrying CIS(es) use the appropriate transmit power level for the physical link in question. Devices use power control on the associated ACL-C logical link to adapt the transmit power level for the physical link.

Isochronous physical links carrying BIS(es) do not support power control because there is no feedback from Observers to the Broadcaster. Traffic is always directed from a single Broadcaster to zero or more Observers.

#### 3.4.3.5 Channel Sounding physical link

Channel Sounding procedures exist within an LE Channel Sounding physical link and are characterized by their transitory existence as well as their unique timing characteristics relative to an LE active physical link. Channel Sounding procedures are based on a preexisting LE active physical link. A CS procedure’s start timing is also dependent on the timing of the corresponding LE active physical link.

A Channel Sounding physical channel supports a Channel Sounding physical link.

### 3.4.4 [This section is no longer used]

## 3.5 Logical links and logical transports

A variety of logical links are available to support different application data transport requirements. Each logical link is associated with a logical transport, which has a number of characteristics. These characteristics include flow control, acknowledgment/repeat mechanisms, sequence numbering and scheduling behavior. Logical transports are able to carry different types of logical links (depending on the type of the logical transport). In the case of some of the Bluetooth logical links these are multiplexed onto the same logical transport. Logical transports may be carried by active physical links on either the basic or the adapted piconet physical channel.

Logical transport identification and real-time (link control) signaling are carried in the packet header, and for some logical links identification is carried in the payload header. Control signaling that does not require single slot response times is carried out using the LMP protocol.

Table 3.4 lists all of the logical transport types, the supported logical link types, which type of physical links and physical channels can support them, and a brief description of the purpose of the logical transport.

| Logical transport | Links supported | Supported by | Bearer | Overview |
|---|---|---|---|---|
| Asynchronous Connection-Oriented (ACL) | Control (LMP) ACL-C<br>User (L2CAP) ACL-U | BR/EDR active physical link, BR/EDR basic or adapted piconet physical channel | BR/EDR | Reliable or time-bounded, bi-directional, point-to-point |
| Synchronous Connection-Oriented (SCO) | Stream (unframed) SCO-S | BR/EDR active physical link, BR/EDR basic or adapted piconet physical channel | BR/EDR | Bi-directional, symmetric, point-to-point, AV channels. Used for 64 kb/s constant rate data. |
| Extended Synchronous Connection-Oriented (eSCO) | Stream (unframed) eSCO-S | BR/EDR active physical link, BR/EDR basic or adapted piconet physical channel | BR/EDR | Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission. Used for constant rate data synchronized to the Central's clock. |

| Logical transport | Links supported | Supported by | Bearer | Overview |
|---|---|---|---|---|
| Active Peripheral Broadcast (APB) | Control (LMP)<br>APB-C<br>User (L2CAP)<br>APB-U | BR/EDR active physical link, basic or adapted physical channel | BR/EDR | Unreliable, uni-directional broadcast to any devices synchronized with the physical channel. Used for broadcast L2CAP groups and certain LMP messages. |
| Connectionless Peripheral Broadcast (CPB) | Profile Broadcast Data (PBD) | Connectionless Peripheral Broadcast physical link, BR/EDR adapted piconet physical channel | BR/EDR | Unreliable, unidirectional, point-to-multipoint, periodic transmissions to zero or more devices. |
| LE asynchronous connection (LE ACL | Control (LL) LE-C, User (L2CAP) LE-U | LE active physical link, LE piconet physical channel | LE | Reliable, bi-directional, point-to-point. |
| LE Advertising Broadcast (ADVB) | Control (LL) ADVB-C, User (LL) ADVB-U | LE advertising physical link, LE advertising physical channel | LE | Unreliable, uni-directional broadcast to all devices in a given area or directed to one recipient. Used to carry data and Link Layer signaling between unconnected devices. |
| LE Periodic Advertising Broadcast (PADVB) | Control (LL) ADVB-C, User (LL) ADVB-U | LE periodic physical link, LE periodic physical channel | LE | Unreliable, periodic, unidirectional broadcast to all devices in a given area. |
| Periodic Advertising with Responses (PAwR) | Control (LL) ADVB-C, User (LL) ADVB-U | LE periodic physical link, LE periodic physical channel | LE | Unreliable, periodic broadcast to all devices or a subset of devices in a given area with optional responses from the devices. |

| Logical transport | Links supported | Supported by | Bearer | Overview |
|---|---|---|---|---|
| Connected Isochronous Stream | Low Energy Stream (LE-S) and Low Energy Framed data (LE-F) | LE isochronous physical link | LE | Unidirectional or bidirectional transport in a point-to-point connection for transferring isochronous data. |
| Broadcast Isochronous Stream | Low Energy Stream (LE-S), Low Energy Framed data (LE-F) and Low Energy Broadcast Control (LEB-C) | LE isochronous physical link | LE | Unidirectional transport for broadcasting data in a point to multipoint configuration and unidirectional transport for controlling the broadcast data. |

*Table 3.4: Logical transport types*

The classification of each link type follows from a selection procedure within three categories.

### 3.5.1 Casting

The first category is that of casting. This may be either unicast or broadcast.

- Unicast links exist between exactly two endpoints. Traffic may be sent in either direction on unicast links.
- Broadcast links exist between one source device and zero or more receiver devices. Traffic is unidirectional, i e., only sent from the source devices to the receiver devices. Broadcast links are connectionless, meaning there is no procedure to create these links, and data may be sent over them at any time. Broadcast links are unreliable, and there is no guarantee that the data will be received.

### 3.5.2 Scheduling and acknowledgment scheme

The second category relates to the scheduling and acknowledgment scheme of the link, and implies the type of traffic that is supported by the link. These are synchronous, isochronous or asynchronous. There are no specific isochronous links defined, though the default ACL link can be configured to operate in this fashion.

- Synchronous links provide a method of associating the transported data with the Bluetooth piconet clock. This is achieved by reserving regular slots on the physical channel, and transmitting fixed size packets at these regular intervals. Such links are suitable for constant rate isochronous data.
- Asynchronous links provide a method for transporting data that has no time-based characteristics. The data is normally expected to be retransmitted until successfully received, and each data entity can be processed at any time after receipt, without

reference to the time of receipt of any previous or successive entity in the stream (providing the ordering of data entities is preserved).

- Isochronous links provide a method for transporting data that has time-based characteristics. The data may be retransmitted until received or expired. The data rate on the link need not be constant (this being the main difference from synchronous links).

### 3.5.3 Class of data

The final category is related to the class of data that is carried by the link. This is either control data or user data. The user data category is sub-divided into L2CAP data, stream data, and periodic broadcast data.

- Control links are only used for transporting LMP or Link Layer messages between two Controllers. These links are invisible above the baseband layer or Link Layer and cannot be directly instantiated, configured, or released by applications, other than by the use of the services, such as connection and disconnection, that have this effect implicitly. Control links are always multiplexed with an equivalent L2CAP data link onto a logical transport. For example, ACL-C and ACL-U are multiplexed onto an ACL logical transport, whereas ADVB-C and ADVB-U are multiplexed onto ADVB and PADVB logical transports. Subject to the rules defining the acknowledgment scheme, the control link traffic normally takes priority over the L2CAP link traffic.
- L2CAP links are used to transport L2CAP PDUs, which may carry the L2CAP signaling channel or framed user data submitted to user-instantiated L2CAP channels. L2CAP frames submitted to the baseband may be larger than the available Baseband packets. A link control protocol embedded within the LLID field preserves the frame-start and frame-continuation semantics when the frame is transmitted in a number of fragments to the receiver. Normally, L2CAP links give reliable delivery of data priority over timely delivery.
- Stream links are used to transport user data when timely delivery of the latest data has priority over reliability. Lost data may be replaced by padding at the receiver. On BR/EDR, these links (SCO and eSCO) have a fixed bandwidth and are always bidirectional between two devices; on LE, they have a variable bandwidth with a specified maximum and may be either bidirectional between two devices (CIS) or unidirectional broadcast (BIS).
- Periodic broadcast data is transmitted at regular intervals, possibly with jitter, by a device. It has no acknowledgment mechanism. The same data is transmitted until it is explicitly changed. This is sent on the PBD logical link on BR/EDR and on the ADVB-U logical link on LE.

### 3.5.4 Logical transports

#### 3.5.4.1 BR/EDR asynchronous connection-oriented (ACL)

The asynchronous connection-oriented (ACL) logical transport is used to carry LMP and L2CAP control signaling and best effort asynchronous user data. The ACL logical transport uses a 1-bit ARQN/SEQN scheme to provide simple channel reliability. Every active Peripheral within a piconet has one ACL logical transport to the piconet Central, known as the default ACL.

The default ACL is created between the Central and the Peripheral when a device joins a piconet (connects to the basic piconet physical channel). This default ACL is assigned a logical transport address (LT_ADDR) by the piconet Central. This LT_ADDR is also used to identify the active physical link when required (or as a piconet active member identifier, effectively for the same purpose).

The LT_ADDR for the default ACL is reused for synchronous connection-oriented logical transports between the same Central and Peripheral. Thus the LT_ADDR is not sufficient on its own to identify the default ACL. However the packet types used on the ACL are different from those used on the synchronous connection-oriented logical transport. Therefore, the ACL logical transport can be identified by the LT_ADDR field in the packet header in combination with the packet type field.

The default ACL may be used for isochronous data transport by configuring it to automatically flush packets after the packets have expired. Asynchronous traffic can be sent over an ACL logical transport configured for isochronous traffic by marking the asynchronous packets as non-automatically-flushable. This allows both isochronous and asynchronous traffic to be transferred at the same time to a single device.

If the default ACL is removed from the active physical link then all other logical transports that exist between the Central and the Peripheral are also removed. In the case of unexpected loss of synchronization to the piconet physical channel the physical link and all logical transports and logical links cease to exist at the time that this synchronization loss is detected.

#### 3.5.4.2 BR/EDR synchronous connection-oriented (SCO)

The synchronous connection-oriented (SCO) logical transport is a symmetric, point-to-point transport between the Central and a specific Peripheral. The SCO logical transport reserves slots on the physical channel and can therefore be considered as a circuit-switched connection between the Central and the Peripheral. SCO logical transports carry 64 kb/s of information synchronized with the piconet clock. Typically this information is an encoded voice stream. Three different SCO configurations exist, offering a balance between robustness, delay and bandwidth consumption.

Each SCO-S logical link is supported by a single SCO logical transport, which is assigned the same LT_ADDR as the default ACL logical transport between the devices. Therefore the LT_ADDR field is not sufficient to identify the destination of a received packet. Because the SCO links use reserved slots, a device uses a combination of the LT_ADDR, the slot numbers (a property of the physical channel) and the packet type to identify transmissions on the SCO link.

Although slots are reserved for the SCO, it is permissible to use a reserved slot for traffic from another channel that has a higher priority. This may be required as a result of QoS commitments, or to send LMP signaling on the default ACL when the physical channel bandwidth is fully occupied by SCOs. As SCOs carry different packet types to ACLs, the packet type is used to identify SCO traffic (in addition to the slot number and LT_ADDR).

There are no further architectural layers defined by the specification that are transported over a SCO link. A number of standard formats are defined for the 64 kb/s stream that is transported, or an unformatted stream is allowed where the application is responsible for interpreting the encoding of the stream.

#### 3.5.4.3 BR/EDR extended synchronous connection-oriented (eSCO)

The extended synchronous connection-oriented (eSCO) logical transport is a symmetric or asymmetric, point-to-point transport between the Central and a specific Peripheral. The eSCO reserves slots on the physical channel and can therefore be considered as a circuit-switched connection between the Central and the Peripheral. eSCO links offer a number of extensions over the standard SCO links, in that they support a more flexible combination of packet types and selectable data contents in the packets and selectable slot periods, allowing a range of synchronous bit rates to be supported.

eSCO links also can offer limited retransmission of packets (unlike SCO links where there is no retransmission). If these retransmissions are required they take place in the slots that follow the reserved slots, otherwise the slots may be used for other traffic.

Each eSCO-S logical link is supported by a single eSCO logical transport, identified by a LT_ADDR that is unique within the piconet for the duration of the eSCO. eSCO-S links are created using LM signaling and follow scheduling rules similar to SCO-S links.

There are no further architectural layers defined by the specification that are transported over an eSCO-S link. Instead applications may use the data stream for whatever purpose they require, subject to the transport characteristics of the stream being suitable for the data being transported.

#### 3.5.4.4 BR/EDR active Peripheral broadcast (APB)

The active Peripheral broadcast logical transport is used to transport LMP control signaling and connectionless L2CAP user traffic to all devices in the piconet that are

currently connected to the physical channel that is used by the APB. There is no acknowledgment protocol and the traffic is uni-directional from the piconet Central to the Peripherals. The APB channel may be used for L2CAP group traffic (a legacy of the 1.1 specification), and is never used for L2CAP connection-oriented channels or L2CAP control signaling.

The APB logical transport is inherently unreliable because of the lack of acknowledgment. To improve the reliability, each packet is transmitted a number of times. An identical sequence number is used to assist with filtering retransmissions at the Peripheral.

The APB logical transport is identified by a reserved LT_ADDR.

An APB is implicitly created whenever a piconet exists, and there is always one APB associated with each of the active physical links (whether operating over the basic or adapted piconet physical channel) that exist within the piconet. Because the basic and adapted piconet physical channels, and different channel maps on the adapted piconet physical channel, are mostly coincident, a Peripheral cannot always distinguish which of the APB channels is being used to transmit the packets. This adds to the general unreliability of the APB channel. (Although it is, perhaps, no more unreliable than general missed packets.)

A Central may decide to use only one of its two possible APBs (when it has both a basic and adapted piconet physical channel), or only one of the channel maps in use on the adapted piconet physical channel (when it has more than one map), as with sufficient retransmissions or careful selection of which slots to transmit on it is possible to address all Peripherals.

The APB channel is never used to carry L2CAP control signals.

#### 3.5.4.5 [This section is no longer used]

#### 3.5.4.6 LE asynchronous connection (LE ACL)

The LE asynchronous connection (LE ACL) logical transport is used to carry LL and L2CAP control signaling and best effort asynchronous user data. The LE ACL logical transport uses a 1-bit NESN/SN scheme to provide simple channel reliability. Every active Peripheral has one LE ACL logical transport to the piconet Central, known as the default LE ACL.

The default LE ACL is automatically created between the Central and the Peripheral when the piconet connecting them is created. This default LE ACL is assigned an Access Address by the piconet Central. This Access Address is also used to identify the active physical link and active piconet physical channel when required.

If the default LE ACL is removed from the LE active physical link then all other LE logical transports that exist between the Central and the Peripheral are also removed. In the case of unexpected loss of synchronization to the LE piconet physical channel the LE physical link and all LE logical transports and LE logical links cease to exist at the time that this synchronization loss is detected.

The default LE ACL can carry flushable data by configuring the default LE ACL with an ACL Flush Timeout. The ACL Flush Timeout only affects packets that the Host marks as flushable, in which case these packets are removed from the transmission queue when their flush timers expire if they have not yet begun their transmission over the air. LE ACL traffic not marked as flushable is not affected by the ACL Flush Timeout.

Figure 3.9 shows an example of a Host sending LE L2CAP PDUs which are marked as flushable. In this example, the Controller Connection Event Interval is lower than the Host interval in which the Host generates new L2CAP PDUs. In this example, the ACL Flush Timeout is set to a value that is lower than the Host interval. PDUs that remain longer than the length of the ACL Flush Timeout in the Controller buffers because of Link Layer delays are flushed before their first transmit opportunity and are not transmitted by the Link Layer, while newer L2CAP PDUs are sent instead at the Host interval rate. In Figure 3.9, ACL packet 3 was being retransmitted across multiple connection events, due to interference, while the Host submitted new L2CAP PDUs. L2CAP packets 4 and 5 exceeded their ACL Flush Timeout before their first transmit opportunity and therefore were flushed. When ACL packet 3 was finally acknowledged, newer L2CAP packets 6 and 7 were transmitted as their ACL Flush Timeout had not expired.

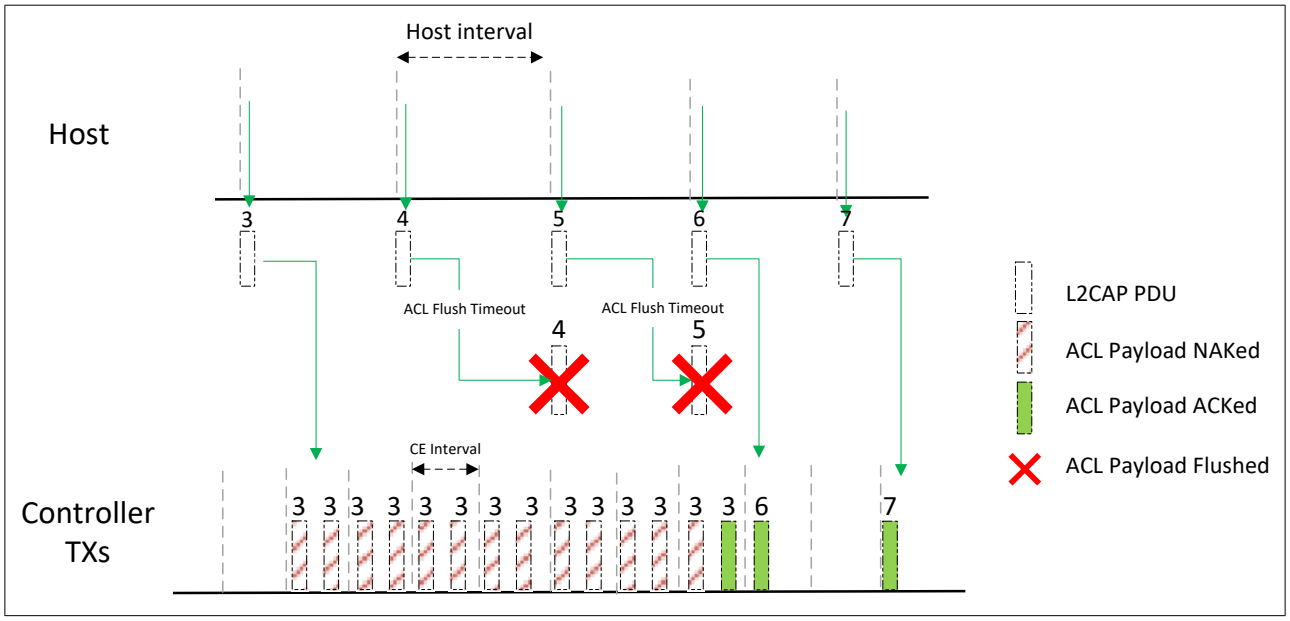



*Figure 3.9: Example of a Host sending flushable data to the Controller*

Figure 3.10 shows an example when the ACL Flush Timeout is set to infinite (i.e., no flushing) or when the Host is only sending L2CAP PDUs that are not marked as flushable. In this case, all PDUs are transmitted by the Link Layer and are queued by either the Host or the Controller, depending on the Controller's LE ACL buffer availability. In this example, L2CAP PDUs 4 and 5 were pending in the Controller queue while ACL packet 3 was being retransmitted by the Link Layer. When packet 3 was finally acknowledged, the queued L2CAP PDUs 4 and 5 were transmitted followed by L2CAP PDUs 6 and 7.

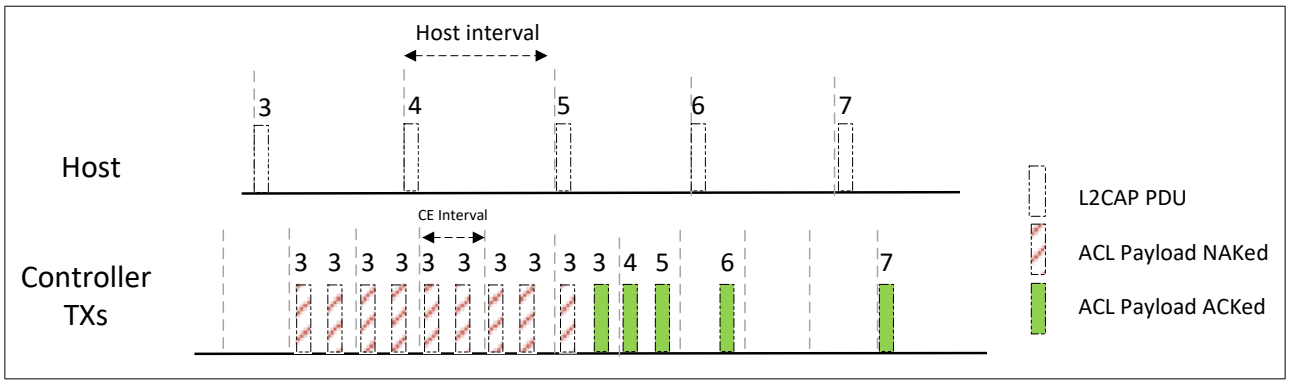



*Figure 3.10: Example of a Host sending data which is not flushable to the Controller*

### 3.5.4.7 LE advertising broadcast (ADVB)

The LE advertising broadcast logical transport is used to transport broadcast control and user data to all scanning devices in a given area. There is no acknowledgment protocol and the traffic is predominately unidirectional from the advertising device. A scanning device can send requests over the logical transport to get additional broadcast user data, or to form an LE ACL logical transport connection. The LE Advertising Broadcast logical transport data is carried only over the LE advertising broadcast link.

The ADVB logical transport is inherently unreliable because of the lack of acknowledgment. To improve the reliability, each packet is transmitted a number of times over the LE advertising broadcast link.

An ADVB is created whenever an advertising device begins advertising. The ADVB logical transport is identified by the advertiser's Bluetooth Device Address and advertising set.

### 3.5.4.8 Connectionless Peripheral Broadcast (CPB)

The CPB logical transport is used to transport profile broadcast data to all devices connected to the Connectionless Peripheral Broadcast logical transport. There is no acknowledgment scheme and the traffic is unidirectional from a Transmitter to zero or more Receivers. To improve reliability, profile broadcast data may be transmitted multiple times.

The CPB logical transport is created on the transmitter whenever the Connectionless Peripheral Broadcast is started. The CPB logical transport is created on the receiver whenever Connectionless Peripheral Broadcast reception is configured. The CPB logical transport is identified by a unique LT_ADDR within the piconet that is reserved specifically for that purpose by the Connectionless Peripheral Broadcast Transmitter.

#### 3.5.4.9 LE periodic advertising

##### 3.5.4.9.1 LE periodic advertising broadcast (PADVB)

The LE periodic advertising broadcast logical transport is used to transport periodic broadcast control and user data to all scanning devices in a given area. The data may be constant for several periods or may change frequently. There is no acknowledgment protocol and the traffic is unidirectional from the advertising device. The LE Periodic Advertising Broadcast logical transport data is carried only over the LE periodic physical link.

The PADVB logical transport is inherently unreliable because of the lack of acknowledgment. To improve the reliability, the period between transmissions can be shorter than the interval between changes to the data so that each packet can be transmitted a number of times over the LE periodic physical link.

A PADVB is created whenever an advertising device begins periodic advertising. The PADVB logical transport is identified by the advertiser's Bluetooth Device Address, timing, and advertising set.

##### 3.5.4.9.2 Periodic advertising with responses (PAwR)

The LE periodic advertising with responses logical transport is used to transport periodic broadcast control and user data to all scanning devices in a given area. These broadcasts are grouped into subevents, allowing a subset of the devices to be synchronized with each subevent. Each subevent can then contain information that is directed to the subset of devices synchronized to that subevent. This allows the broadcasting device to send data to many devices and the synchronized devices to only have to listen very infrequently for information directed to them. The data may be constant for several periods or may change frequently. The PAwR logical transport data is carried only over the LE periodic physical link.

The PAwR logical transport also includes response slots. The devices that have been directly addressed by the advertising device use response slots to send back responses. A higher layer specification determines the set of devices that can respond and when they respond. The PAwR logical transport is inherently unreliable, but the ability to use response slots allows higher layer acknowledgments to be used to provide reliability.

A PAwR logical transport is created whenever an advertising device begins periodic advertising configured to use subevents and responses. The PAwR logical transport is identified by the advertiser's Bluetooth Device Address, timing, and advertising set.

#### 3.5.4.10 Connected Isochronous Stream (CIS)

The CIS is a data-symmetric or data-asymmetric, point-to-point logical transport between the Central and a specific Peripheral. A CIS reserves transmission/reception (Tx/Rx) periods, known as subevents, on the isochronous physical channel and can be considered as a circuit switched connection between the Central and the Peripheral. The CIS supports a variable flushing period for payloads, variable size data contents in the packets, and a variable number of subevents, allowing a range of isochronous data rates, latencies, and re-transmissions to be supported. A CIS can be configured to retransmit packets by providing more subevents than required for transmitting the data. If retransmissions are required, they take place in the subevents (of the current or subsequent events) that follow.

Each LE-S or LE-F logical link is supported by a single CIS that is identified by a unique access address for the lifetime of the CIS. The LE-S or LE F links are created by using Link Layer procedures. The higher layer may use the data stream for whatever purpose it requires, as long as the transport characteristics of the stream are suitable for the data that is being transported. All isochronous connections are terminated when the associated LE piconet physical channel is terminated.

#### 3.5.4.11 Connected Isochronous Group (CIG)

Each CIS is part of a CIG. A CIG may have one or more CISes, all with the same Central but possibly different Peripherals. Multiple CISes in a CIG have a common timing reference based on the Central timing and are synchronized in time. The common timing reference of multiple CISes helps devices to synchronize their input or output data. For example, when the left and right channels of an audio stereo stream, which are received by separate devices, need to be rendered at the same time. Multiple CISes in a CIG can be scheduled sequentially or in an interleaved arrangement.

#### 3.5.4.12 Broadcast Isochronous Stream (BIS)

The BIS logical transport is used to transport one or more isochronous data streams to all devices for a BIS within range. The data may be fixed or variable size, framed or unframed. A BIS has one or more subevents for transmitting isochronous data packets. A BIS supports transmission of multiple new isochronous data packets in every BIS event. There is no acknowledgment protocol and the traffic is unidirectional from the broadcasting device. The BIS logical transport is inherently unreliable because of the lack of acknowledgment. To improve the reliability of delivery, the isochronous data packets can be unconditionally re-transmitted by increasing the number of subevents in every event. The reliability of delivery can also be improved by transmitting packets

in intervals earlier than the intervals they are associated with; this is called "pre-transmission". A BIS supports LE-S or LE-F logical links. A BIS is identified by a unique access address and the timing information. The access address and the timing information is transmitted in the packet that is sent using the associated Periodic Advertising Broadcast (PADVB) logical transport. A scanning device that supports the Synchronized Receiver role feature may receive isochronous data from a BIS after synchronizing to the BIS by using the timing information from the periodic advertising train.

#### 3.5.4.13 Broadcast Isochronous Group (BIG)

Each BIS is part of a BIG. A BIG may have one or more BISes. Multiple BISes in a BIG have a common timing reference based on the broadcaster and are synchronized in time. For example, when the left and right channels of an audio stereo stream, which are received by separate devices, need to be rendered at the same time. Multiple BISes in a BIG can be scheduled sequentially or in an interleaved arrangement. A BIG also supports a Low Energy Broadcast Control (LEB-C) logical link.

### 3.5.5 Logical links

Some logical transports are capable of supporting different logical links, either concurrently multiplexed, or one of the choice.

#### 3.5.5.1 BR/EDR logical links

Within BR/EDR logical transports, the logical link is identified by the logical link identifier (LLID) bits in the payload header of Baseband packets that carry a data payload. The logical links distinguish between a limited set of core protocols that are able to transmit and receive data on the logical transports. Not all of the logical transports are able to carry all of the logical links (the supported mapping is shown in Figure 3.2). In particular the BR/EDR SCO and eSCO logical transports are only able to carry constant data rate streams, and these are uniquely identified by the LT_ADDR. Such logical transports only use packets that do not contain a payload header, as their length is known in advance, and no LLID is necessary.

##### 3.5.5.1.1 ACL control logical links (ACL-C and APB-C)

The ACL Control Logical Links (ACL-C and APB-C) are used to carry BR/EDR LMP signaling between devices in the piconet. The ACL-C control link is normally only carried on the default ACL logical transport (though can also be carried in DV packets on the SCO logical transport) while the APB-C control link is only carried on the APB logical transport. Each control link is always given priority over the corresponding data link carried on the same logical transport.

#### 3.5.5.1.2 User asynchronous/isochronous logical links (ACL-U and APB-U)

The user asynchronous/isochronous logical links (ACL-U and APB-U) are used to carry all asynchronous and isochronous framed user data. The ACL-U link is normally only carried on the default ACL logical transport (though can also be carried in DV packets on the SCO logical transport) while the APB-U link is only carried on the APB logical transport. Packets on the ACL-U and APB-U link are identified by one of two reserved LLID values. One value is used to indicate that the Baseband packet contains the start of an L2CAP frame and the other indicates a continuation of a previous frame. This ensures correct synchronization of the L2CAP reassembler following flushed packets. The use of this technique removes the need for a more complex L2CAP header in every Baseband packet (the header is only required in the L2CAP start packets) because each L2CAP frame is completely transmitted before the next one starts. (An exception to this rule being the ability to flush a partially transmitted L2CAP frame in favor of another L2CAP frame.)

#### 3.5.5.1.3 User synchronous/extended synchronous logical links (SCO-S/eSCO-S)

Synchronous (SCO-S) and extended synchronous (eSCO-S) logical links are used to support isochronous data delivered in a stream without framing. These links are associated with a single logical transport, where data is delivered in constant sized units at a constant rate. There is no LLID within the packets on these transports, as only a single logical link can be supported, and the packet length and scheduling period are pre-defined and remain fixed during the lifetime of the link.

Variable rate isochronous data cannot be carried by the SCO-S or eSCO-S logical links. In this case the data must be carried on ACL-U or APB-U logical links, which use packets with a payload header.

#### 3.5.5.1.4 Profile Broadcast Data (PBD) logical link

The PBD logical link is used to broadcast isochronous unframed data to multiple receivers and resides on the CPB logical transport.

### 3.5.5.2 LE logical links

Within LE logical transports, the logical link is identified by the logical link identifier (LLID) bits in the payload header of Baseband packets that carry a data payload. The logical links distinguish between a limited set of core protocols that are able to transmit and receive data on the logical transports. Not all of the logical transports are able to carry all of the logical links (the supported mapping is shown in Figure 3.2).

#### 3.5.5.2.1 Control logical link (LE-C)

The LE ACL Control Logical Link (LE-C) is used to carry LE LL signaling between the two devices in the piconet. The control link is only carried on the default LE ACL logical transport.

#### 3.5.5.2.2 User asynchronous logical link (LE-U)

The user asynchronous logical link (LE-U) is used to carry all asynchronous and framed user data. The LE-U link is carried on the LE logical transport. Packets on the LE-U link are identified by one of two reserved LLID values. One value is used to indicate that the Baseband packet contains the start of an L2CAP frame and the other indicates a continuation of a previous frame or empty PDU. This ensures correct synchronization of the L2CAP re-assembler. The use of this technique removes the need for a more complex L2CAP header in every Baseband packet because each L2CAP frame is completely transmitted before the next one starts.

#### 3.5.5.2.3 Advertising Broadcast Control logical link (ADVB-C)

The LE Advertising Broadcast Control Logical Link (ADVB-C) is used to carry LE LL signaling between unconnected devices in a given area. This signaling is the control commands for gathering additional broadcast user data (scan requests) or connection requests. The control link is carried on the LE Advertising Broadcast and LE Periodic Advertising Broadcast logical transports.

#### 3.5.5.2.4 Advertising Broadcast User Data logical link (ADVB-U)

The LE Advertising Broadcast User Data Logical Link (ADVB-U) is used to carry LE Advertising Broadcast and LE Periodic Advertising Broadcast user data used between devices without the need for a connection or LE-U between the devices. The user data link is carried on the LE Advertising Broadcast logical transport for LE Advertising Broadcast user data and the LE Periodic Advertising Broadcast logical transport for LE Periodic Advertising Broadcast user data.

#### 3.5.5.2.5 Low Energy Stream (LE-S)

An LE-S is a logical link that is used to carry the unframed isochronous data packets of an isochronous data stream in a CIS or a BIS logical transport.

#### 3.5.5.2.6 Low Energy Framed (LE-F)

An LE-F is a logical link that is used to carry the framed isochronous data packets of an isochronous data stream in a CIS or a BIS logical transport.

#### 3.5.5.2.7 Low Energy Broadcast Control (LEB-C)

An LEB-C is a logical link that uses the BIS logical transport to carry the control information for all the BISes in a BIG.

#### 3.5.5.3 [This section is no longer used]

## 3.6 L2CAP channels

L2CAP provides a multiplexing role allowing many different applications to share an ACL-U, APB-U, or LE-U logical link. Applications and service protocols interface with L2CAP using a channel-oriented interface to create connections to equivalent entities on other devices.

L2CAP channel endpoints are identified to their clients by a Channel Identifier (CID). This is assigned by L2CAP, and each L2CAP channel endpoint on any device has a different CID.

L2CAP channels may be configured to provide an appropriate Quality of Service (QoS) to the application. L2CAP maps the channel onto the ACL-U, APB-U, or LE-U logical link.

L2CAP supports channels that are connection-oriented and others that are group-oriented. Group-oriented channels may be mapped onto the APB-U logical link, or implemented as iterated transmission to each member in turn over an ACL-U logical link.

Apart from the creation, configuration and dismantling of channels, the main role of L2CAP is to multiplex service data units (SDUs) from the channel clients onto the ACL-U, APB-U, or LE-U logical link, and to carry out a simple level of scheduling, selecting SDUs according to relative priority.

L2CAP can provide per channel flow control with the peer L2CAP layer (except on the APB-U logical link). This option is selected by the application when the channel is established. L2CAP can also provide enhanced error detection and retransmission to (a) reduce the probability of undetected errors being passed to the application and (b) recover from loss of portions of the user data when the Baseband performs a flush on the ACL-U logical link.

In the case where an HCI is present, the L2CAP is also required to segment L2CAP SDUs into fragments that will fit into the baseband buffers, and also to operate a token based flow control procedure over the HCI, submitting fragments to the baseband only when allowed to do so. This may affect the scheduling algorithm.

## 3.7 Isochronous Adaptation Layer (ISOAL)

The ISOAL provides a mechanism such that the timing used to generate or receive isochronous data in the upper layer can be independent of the timing used in the CIS or BIS logical transport used to carry the isochronous data. For example, audio codec data can be generated at a 10 ms interval while the value of ISO_Interval for the CIS can be 11.25 ms. The ISOAL converts upper layer isochronous data units to lower layer isochronous data packets (or the other way around). For more information, see [Vol 6] Part G.

## 3.8 Power control

The power control feature provides a mechanism to request a remote device to adjust its transmit power level based on local signal quality information. The feature is supported on BR/EDR active physical links (see Section 3.4.1.1) and on LE active physical links (see Section 3.4.3.1).

### 3.8.1 Power control in BR/EDR

In BR/EDR, two power control mechanisms (legacy and enhanced power control) are available. Both mechanisms support requesting an incremental change in transmit power level. The enhanced power control mechanism also allows requesting a change to maximum transmit power level. The requested change is applied on all supported modulations at once since the modulation can change dynamically between packets.

### 3.8.2 Power control in LE

In LE, a device can request a remote device to make a specified change in the remote device's power level on a given PHY. This allows a faster transition to the desired power level compared to incremental changes. A responding device can also return a value to indicate an acceptable reduction in the power level that allows the requesting device to further reduce its transmit power level to the minimum level possible and hence conserve energy. The local and remote devices can also share their current transmit power levels during this exchange to enable devices to calculate the link path loss between them. Devices are also allowed to do autonomous local transmit power level changes and indicate the change to the remote device.

In LE, an ACL connection can have associated connections like CIS(es). The power control for all the PHYs used on the ACL and associated connections is managed over the ACL connection. The Host can use the connection handle of the ACL connection to retrieve information for all PHYs used on the ACL and associated connections.

# 4 COMMUNICATION TOPOLOGY AND OPERATION

## 4.1 Piconet topology

### 4.1.1 BR/EDR topology

Any time a link is created using the BR/EDR Controller it is within the context of a piconet. Each link connects two devices, called the “Central” and “Peripheral”. A piconet consists of a single Central, known as the Central of the piconet, and all the Peripherals linked to it, known as the Peripherals in the piconet.

Connected BR/EDR devices communicate on the same physical channel by synchronizing with a common clock and hopping sequence. The common (piconet) clock is identical to the Bluetooth clock of the Central of the piconet and the hopping sequence is derived from the Central’s clock and the Central’s Bluetooth Device Address (different hopping sequences may be used for different Peripherals).

The terms Central and Peripheral are only used when describing these roles in a piconet.

A number of independent piconets may exist in close proximity. Each piconet has a different physical channel (that is a different Central and an independent timing and hopping sequence).

A Bluetooth device may participate concurrently in two or more piconets. It does this on a time-division multiplexing basis. A Bluetooth device can never be a Central of more than one piconet. (Since in BR/EDR the piconet is defined by synchronization to the Central’s Bluetooth clock it is impossible to be the Central of two or more piconets.) A Bluetooth device may be a Peripheral in many independent piconets.

A Bluetooth device that is a member of two or more piconets is said to be involved in a scatternet. Involvement in a scatternet does not necessarily imply any network routing capability or function in the Bluetooth device. The Bluetooth core protocols do not, and are not intended to offer such functionality, which is the responsibility of higher level protocols and is outside the scope of the specification.

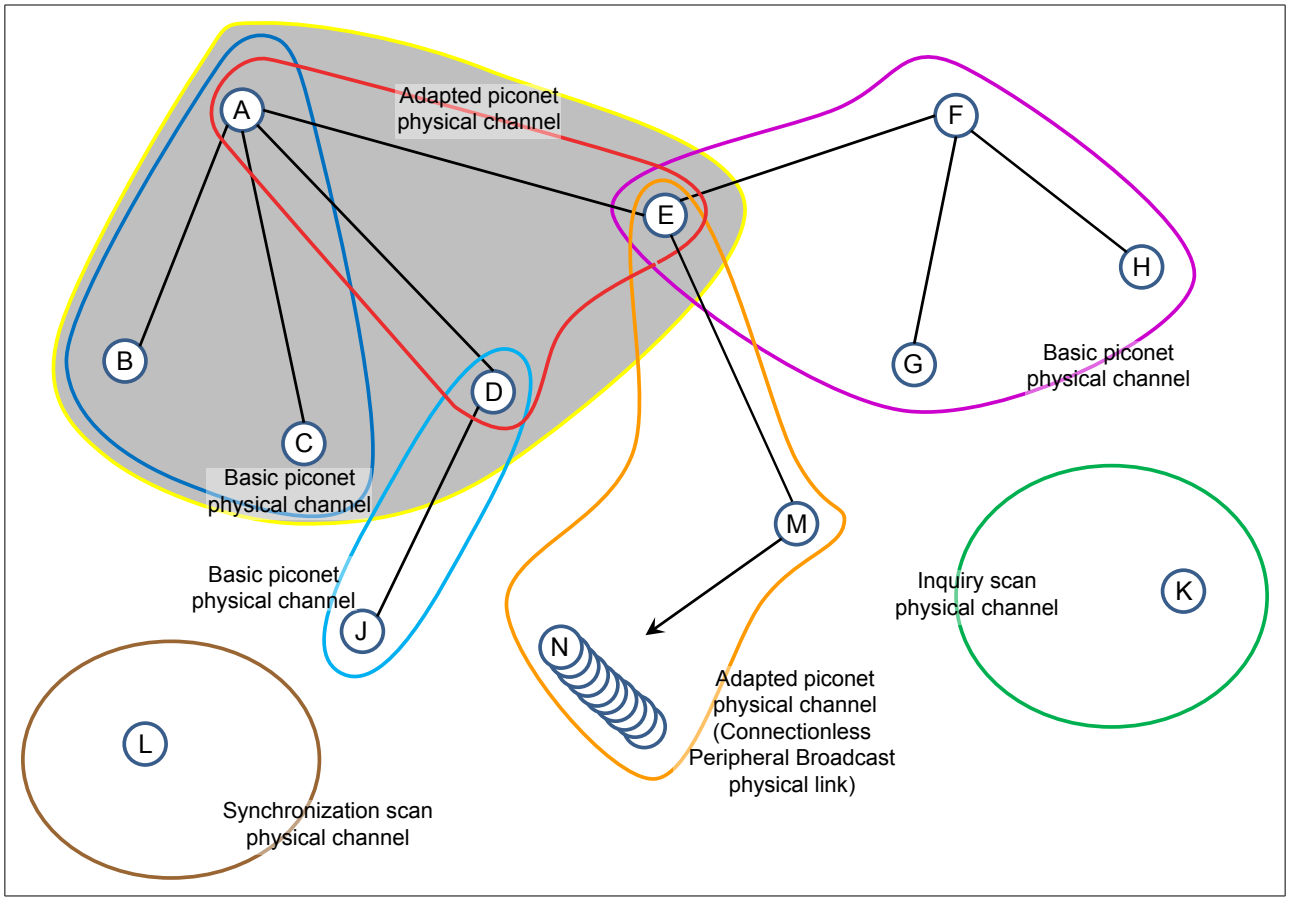



*Figure 4.1: Example Bluetooth BR/EDR topology*

In Figure 4.1 an example topology is shown that demonstrates a number of the architectural features described below. Device A is a Central in a piconet (represented by the shaded area, and known as piconet A) with devices B, C, D and E as Peripherals. Three other piconets are shown: a) one piconet with device F as Central (known as piconet F) and devices E, G and H as Peripherals, b) one piconet with device D as Central (known as piconet D) and device J as Peripheral, and c) one piconet with device M as Central (known as piconet M) and device E as a Peripheral and many devices N as Peripherals.

In piconet A there are two physical channels. Devices B and C are using the basic piconet physical channel (represented by the blue enclosure) as they do not support adaptive frequency hopping. Devices D and E are capable of supporting adaptive frequency hopping, and are using the adapted piconet physical channel (represented by the red enclosure). Device A is capable of adaptive frequency hopping, and operates in a TDM basis on both physical channels according to which Peripheral is being addressed.

Piconet D and piconet F are both using only a basic piconet physical channel (represented by the cyan and magenta enclosures respectively). In the case of piconet

D this is because device J does not support the adaptive hopping mode. Although device D supports adaptive hopping it cannot use it in this piconet. In piconet F device F does not support adaptive hopping, and therefore it cannot be used in this piconet.

Piconet M (represented by the orange enclosure) uses a Connectionless Peripheral Broadcast physical link over the adaptive piconet physical channel to send Profile Broadcast Data from the transmitter device M to many Receiver devices including E and N.

Device K is shown in the same locality as the other devices. It is not currently a member of a piconet, but has services that it offers to other Bluetooth devices. It is currently listening on its inquiry scan physical channel (represented by the green enclosure), awaiting an inquiry request from another device.

Device L is shown in the same locality as the other devices. It is not currently a member of a piconet, but is currently listening on its synchronization scan physical channel (represented by the brown enclosure), awaiting a synchronization train from another device.

### 4.1.2 LE topology

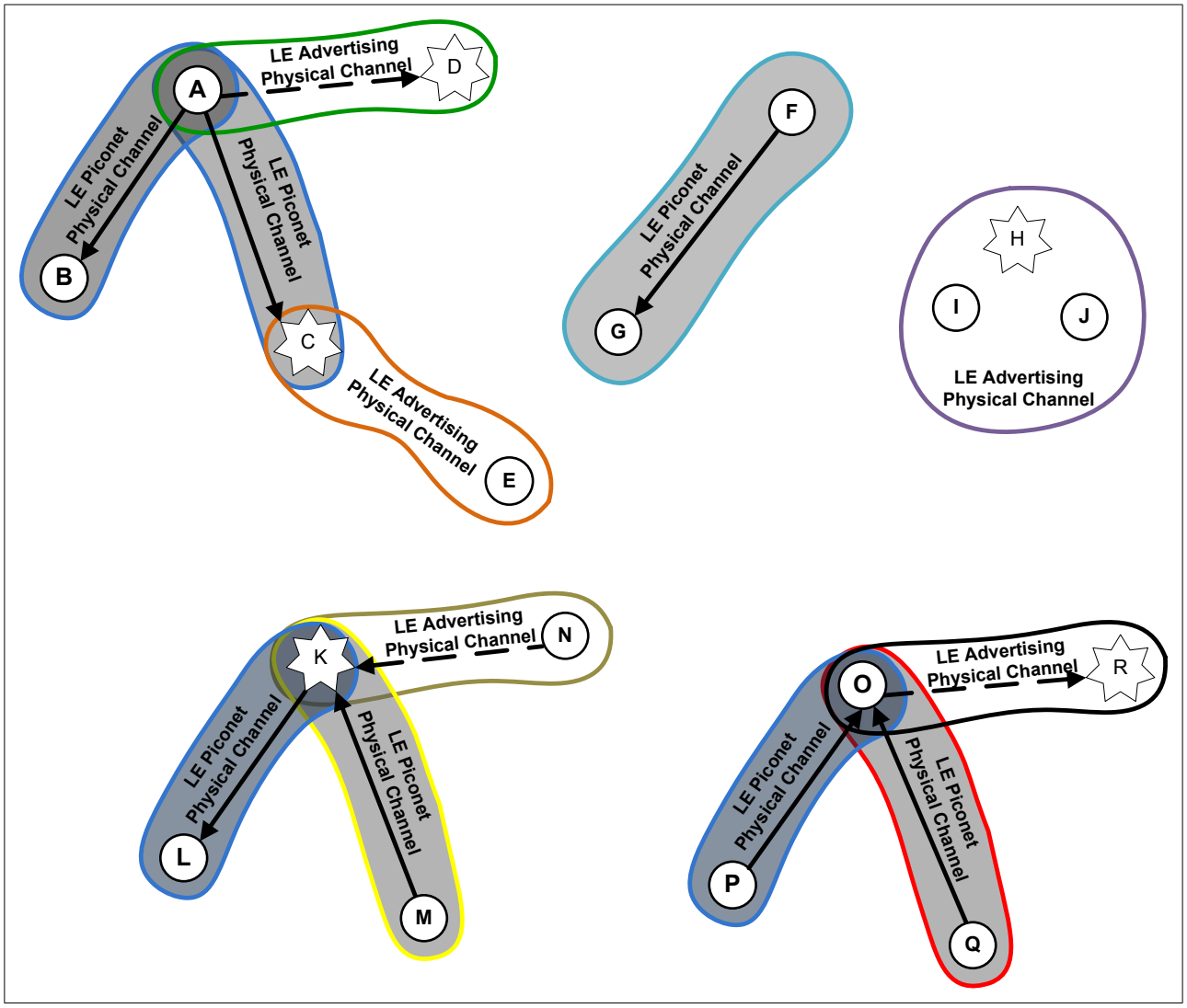



*Figure 4.2: Example of Bluetooth LE topology*

In Figure 4.2 an example topology is shown that demonstrates a number of the LE architectural features described below. Device A is a Central in two piconets (represented by the shaded areas) with devices B and C as the respective Peripherals. Unlike BR/EDR Peripherals, LE Peripherals do not share a single piconet or a common physical channel with the Central. Each Peripheral communicates on a separate physical channel with the Central. One other piconet is shown with device F as Central and device G as a Peripheral. Device K is in a scatternet. Device K is Central of one piconet with device L as the Peripheral and is Peripheral of a second piconet with device M as the Central. Device O is also in a scatternet. Device O is the Peripheral of two piconets, one with device P as the Central and the other with device Q as the Central. In the figure, solid arrows point from Central to Peripheral; dashed arrows,

indicating a connection initiation, point from initiator to advertiser using a connectable advertising event; devices that are advertising are indicated using stars.

There are five other groups of devices shown:

1. Device D is an advertiser and device A is also an initiator.
2. Device E is a scanner and device C is also an advertiser.
3. Device H is an advertiser and devices I and J are scanners.
4. Device K is also an advertiser and device N is an initiator.
5. Device R is an advertiser and device O is also an initiator.

Devices A and B are using one LE piconet physical channel (represented by the blue enclosure and a dark gray background). Devices A and C are using another LE piconet physical channel (represented by the blue enclosure and a lighter gray background). Device D is advertising using a connectable advertising event on the advertising physical channel (represented by the green enclosure) and device A is an initiator. Device A can form a connection with device D, creating a new piconet. Device C is also advertising on the advertising physical channel (represented by the orange enclosure) using any type of advertising events that are being captured by device E as a scanner. Devices C and D may be using different advertising PHY channels or different timings to avoid collisions.

Devices F and G are using a piconet and an LE piconet physical channel (represented by the aqua enclosure). Device F is the Central and device G is the Peripheral.

Devices H, I and J are using the LE advertising physical channel (represented by the purple enclosure). Device H is an advertiser and devices I and J are scanners.

In the scatternet involving device K, devices K and L are using one piconet and LE piconet physical channel. Devices K and M are using another piconet and LE piconet physical channel. Device K is also advertising using a connectable advertising event on the advertising physical channel and device N is an initiator. Device N can form a connection with device K resulting in device K being Peripheral of two devices and Central of one device at the same time.

In the scatternet involving device O, devices O and P are using one piconet and LE piconet physical channel. Devices O and Q are using another piconet and LE piconet physical channel. Device R is advertising using a connectable advertising event on the advertising physical channel and device O is an initiator. Device O can form a connection with device R resulting in device O being Peripheral of two devices and Central of one device at the same time.

## 4.2 Operational procedures and modes

The typical operational mode of a Bluetooth device is to be connected to other Bluetooth devices (in a piconet) and exchanging data with those Bluetooth devices. As Bluetooth is an ad-hoc wireless communications technology, there are a number of operational procedures that enable piconets to be formed so that the subsequent communications can take place. Procedures and modes are applied at different layers in the architecture and therefore a device may be engaged in a number of these procedures and modes concurrently.

### 4.2.1 BR/EDR procedures

#### 4.2.1.1 Inquiry (discovering) procedure

Bluetooth devices use the inquiry procedure to discover nearby devices, or to be discovered by devices in their locality.

The inquiry procedure is asymmetrical. A Bluetooth device that tries to find other nearby devices is known as an inquiring device and actively sends inquiry requests. Bluetooth devices that are available to be found are known as discoverable devices and listen for these inquiry requests and send responses. The inquiry procedure uses a special physical channel for the inquiry requests and responses.

Both inquiring and discoverable devices may already be connected to other Bluetooth devices in a piconet. Any time spent inquiring or occupying the inquiry scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

The inquiry procedure does not make use of any of the architectural layers above the physical channel, although a transient physical link may be considered to be present during the exchange of inquiry and inquiry response information.

##### 4.2.1.1.1 Extended Inquiry response

An Extended Inquiry Response can be used to provide miscellaneous information during the inquiry response procedure. Data types are defined for such things as local name and supported services, information that otherwise would have to be obtained by establishing a connection. A device that receives a local name and a list of supported services in an extended inquiry response does not have to connect to do a remote name request and an SDP service search, thereby shortening the time to useful information. It is recommended that a device includes all supported services and a significant portion of its local name, if that name is too long to be sent in its entirety, in the extended inquiry response.

Extended inquiry response data can be transmitted encrypted or unencrypted. Unencrypted data can be interpreted by any device. Encrypted data can be received by

any device but can only be decrypted and authenticated by devices that have previously shared the session key used to encrypt the data.

The extended inquiry response procedure is backwards compatible with the standard inquiry response procedure.

#### 4.2.1.2 Paging (connecting) procedure

The procedure for forming connections is asymmetrical and requires that one Bluetooth device carries out the page (connection) procedure while the other Bluetooth device is connectable (page scanning). The procedure is targeted, so that the page procedure is only responded to by one specified Bluetooth device.

The connectable device uses a special physical channel to listen for connection request packets from the paging (connecting) device. This physical channel has attributes that are specific to the connectable device, hence only a paging device with knowledge of the connectable device is able to communicate on this channel.

Both paging and connectable devices may already be connected to other Bluetooth devices. Any time spent paging or occupying the page scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

#### 4.2.1.3 Connected mode

After a successful connection procedure over the BR/EDR Controller, there is a piconet physical channel to which both devices are connected, there is a physical link between the devices, and there are default ACL-C, ACL-U, APB-C, and APB-U logical links. Two of these links (ACL-C and APB-C) transport the LMP control protocol and are invisible to the layers above the Link Manager. The ACL-U link transports the L2CAP signaling protocol and any multiplexed L2CAP best-effort channels. The APB-U link transports L2CAP channels that are broadcast to all Peripherals on the piconet. It is common to refer to a default ACL logical transport, which can be resolved by context, but typically refers to the default ACL-U logical link.

When in the connected mode it is possible to create and release additional logical links and to change the modes of the physical and logical links while remaining connected to the piconet physical channel. It is also possible for the device to carry out inquiry, paging or scanning procedures or to be connected to other piconets without needing to disconnect from the original piconet physical channel. These actions are done using the Link Manager, which exchanges Link Manager protocol messages with the remote Bluetooth device.

During the time that a Peripheral is actively connected to a piconet there is always a default ACL logical transport between the Peripheral and the Central. The only method of deleting the default ACL logical transport is to detach the device from the piconet

physical channel, at which time the entire hierarchy of L2CAP channels, logical links, and logical transports between the devices is deleted.

#### 4.2.1.4 Hold mode

Hold mode is not a general device mode, but applies to unreserved slots on the physical link. When in this mode, the physical link is only active during slots that are reserved for the operation of the synchronous link types SCO and eSCO. All asynchronous links are inactive. Hold modes operate once for each invocation and are then exited when complete, returning to the previous mode.

#### 4.2.1.5 Sniff mode

Sniff mode is not a general device mode, but applies to the default ACL logical transports. When in this mode the availability of these logical transports is modified by defining a duty cycle consisting of periods of presence and absence. Devices that have their default ACL logical transports in Sniff mode may use the absent periods to engage in activity on another physical channel, or to enter reduced power mode. Sniff mode only affects the default ACL logical transports (i.e. their shared ACL logical transport), and does not apply to any additional SCO or eSCO logical transports that may be active. The periods of presence and absence of the physical link on the piconet physical channel is derived as a union of all logical transports that are built on the physical link.

Sniff subrating provides a mechanism for further reducing the active duty cycle, thereby enhancing the power-saving capability of Sniff mode, by allowing a Host to specify maximum transmit and receive latencies. This allows the basebands to optimize the low power performance without having to exit and re-enter Sniff mode using Link Manager commands.

Broadcast logical transports have no defined expectations for presence or absence. A Central should aim to schedule broadcasts to coincide with periods of physical link presence within the piconet physical channel, but this is not always possible or practical. Repetition of broadcasts is defined to improve the possibilities for reaching multiple Peripherals without overlapping presence periods. However, broadcast logical transports cannot be considered to be reliable.

#### 4.2.1.6 [This section is no longer used]

#### 4.2.1.7 Role switch procedure

The role switch procedure is a method for swapping the roles of two devices connected in a piconet. The procedure involves moving from the physical channel that is defined by the original Central to the physical channel that is defined by the new Central. In the process of swapping from one physical channel to the next, the hierarchy of physical links and logical transports over the BR/EDR Controller are removed and rebuilt, with

the exception of the APB logical transport that is implied by the topology and is not preserved. After the role switch, the original piconet physical channel may cease to exist or may be continued if the original Central had other Peripherals that are still connected to the channel.

The procedure only moves the default ACL logical links and supporting layers to the new physical channel. Any additional logical transports are not copied by this procedure, and if required this must be carried out by higher layers. The LT_ADDRs of any affected transports will be reassigned on the new physical channel and, therefore, may change.

If there are any QoS commitments on the original logical transports, then these are not preserved after a role switch. These must be renegotiated after the role switch has completed.

#### 4.2.1.8 Enhanced Data Rate

Enhanced Data Rate is a method of extending the capacity and types of Bluetooth packets for the purposes of increasing the maximum throughput, providing better support for multiple connections, and lowering power consumption, while the remainder of the architecture is unchanged.

Enhanced Data Rate may be selected as a mode that operates independently on each logical transport. Once enabled, the packet type bits in the packet header are interpreted differently from their meaning in Basic Rate mode. This different interpretation is clarified in conjunction with the logical transport address field in the header. The result of this interpretation allows the packet payload header and payload to be received and demodulated according to the packet type. Enhanced Data Rate can be enabled only for the ACL and eSCO logical transports and cannot be enabled for the SCO and broadcast logical transports.

#### 4.2.1.9 Connectionless Peripheral Broadcast mode

Connectionless Peripheral Broadcast mode allows a piconet Central to transmit profile broadcast data to any number of connected Peripherals using the BR/EDR adapted piconet physical channel. To enter this mode, the Central reserves a specific logical transport address for the CPB logical transport and starts broadcasting data using the Connectionless Peripheral Broadcast physical link and the synchronization train procedure. A single Profile Broadcast Data logical link is defined, which carries profile broadcast data using the Connectionless Peripheral Broadcast logical transport. The profile broadcast data is unframed and bypasses L2CAP.

To receive the Connectionless Peripheral Broadcast packets, a device must connect with the Connectionless Peripheral Broadcast Transmitter which has already established a CPB logical transport. To connect, a device follows the Synchronization

Scan procedure to obtain the time schedule of the physical link and then starts receiving the Connectionless Peripheral Broadcast packets. Once connected, Connectionless Peripheral Broadcast receivers can receive profile broadcast data on the dedicated CPB logical transport and PBD logical link.

### 4.2.2 LE procedures

#### 4.2.2.1 Device filtering procedure

The device filtering procedure is a method for Controllers to reduce the number of devices requiring communication responses. Since it is not required to respond to requests from every device, it reduces the number of transmissions an LE Controller is required to make which reduces power consumption. It also reduces the communication the Controller would be required to make with the Host. This results in additional power savings since the Host does not have to be involved.

An advertising or scanning device may employ device filtering to restrict the devices from which it receives advertising packets, scan requests or connection requests. In LE, some advertising packets received by a scanning device require that the scanning device send a request to the advertising device. This advertisement can be ignored if device filtering is used and the advertising device is being filtered. A similar situation occurs with connection requests. Connection requests must be responded to by advertisers unless a device filter is used to limit the devices to which the advertiser is required to respond. Advertisers can also use device filters to limit the devices in which it will accept a scan request or connection request.

This device filtering is accomplished through the use of a “Filter Accept List” located in the LL block of the Controller. A Filter Accept List enumerates the remote devices that are allowed to communicate with the local device. When a Filter Accept List is in effect, transmissions from devices that are not in the Filter Accept List will be ignored by the LL. Since device filtering occurs in the LL it can have a significant impact on power consumption by filtering (or ignoring) advertising packets, scan requests or connection requests from being sent to the higher layers for handling.

The use of device filtering during certain procedures needs to be evaluated carefully to ensure devices are not unintentionally ignored, which may cause interoperability problems when attempting to establish connections or receive advertising broadcasts.

#### 4.2.2.2 Advertising procedure

An advertiser uses the advertising procedure to perform unidirectional broadcasts to devices in the area. The unidirectional broadcast occurs without a connection between the advertising device and the listening devices. The advertising procedure can be used to establish connections with nearby initiating devices or used to provide periodic broadcast of user data to scanning devices listening on the advertising physical

channel. The advertising procedure uses the primary advertising physical channel for all advertising broadcasts. However, the data may be offloaded on to the secondary advertising physical channel in one or more auxiliary packets to reduce both the occupancy of the primary advertising physical channel and the total on-air time and also to allow the data to be longer than the maximum that will fit into a single packet.

Advertising data can be transmitted encrypted or unencrypted. Unencrypted data can be interpreted by any scanning device. Encrypted data can be received by any scanning device but can only be decrypted and authenticated by devices that have previously obtained the session key used to encrypt the data.

An LE device connected in an LE piconet may also advertise using any type of advertising event. Time spent advertising while connected needs to be balanced with the connection requirements needed to maintain the already established connection(s).

Advertising devices may receive scan requests from listening devices in order to get additional user data from the advertising device. Scan responses are sent by the advertising device to the device making the scan request.

An advertising device may receive connection requests from initiator devices. If the advertising device was using a connectable advertising event and the initiating device is not being filtered by the device filtering procedure, the advertising device ceases advertising and enters the connected mode. The device can begin advertising again after it is in the connected mode.

When advertising on the LE Uncoded PHYs, scan requests and scan responses can take place on the same PHY channel as the original advertisement or can be offloaded to the secondary advertising physical channel. In some cases when advertising on the LE Uncoded PHYs, connection request and connection responses are offloaded to the secondary advertising physical channel. When advertising on the LE Coded PHY, scan requests, scan responses, connection requests, and connection responses are always offloaded. As with advertising data, offloading is carried out by having the initial advertisement on the primary advertising physical channel point to an auxiliary packet on the secondary advertising physical channel. Scan requests and scan responses, connection requests, and connection responses are made on the same PHY and same physical channel as the auxiliary packet.

#### 4.2.2.3 Scanning procedure

A scanning device uses the scanning procedure to listen for unidirectional broadcasts of user data from advertising devices using the advertising physical channel. A scanning device can request additional user data from an advertising device by making a scan request. The advertising device responds to these requests with additional user data sent to the scanning device over the advertising physical channel.

The scanning procedure can be used while connected to other LE devices. Time spent scanning while connected needs to be balanced with the connection requirements needed to maintain the already established connection with the other LE device in each piconet.

If the broadcasts are connectable advertising events and the scanning device is in the initiator mode, it can initiate a connection by sending a connection request on the primary advertising physical channel or secondary advertising physical channel to the advertising device. Once the connection request is sent on the primary advertising physical channel, the scanning device ceases the initiator mode scanning for additional broadcasts and enters the connected mode. When the connection request is sent on the secondary advertising physical channel, the scanning device leaves the initiator mode, ceasing scanning for additional broadcasts, and enters the connected mode when it receives the connection response. The device can use the scanning procedure after it enters the connected mode, allowing it to be the Central in more than one piconet at a time.

#### 4.2.2.4 Discovering procedure

Bluetooth devices use the advertising procedure and scanning procedure to discover nearby devices, or to be discovered by devices in a given area.

The discovery procedure is asymmetrical. A Bluetooth device that tries to find other nearby devices is known as a discovering device and listens for devices advertising scannable advertising events. Bluetooth devices that are available to be found are known as discoverable devices and actively broadcast scannable advertising events over the advertising broadcast physical channel.

Both discovering and discoverable devices may already be connected to other Bluetooth devices. Any time spent inquiring or occupying the advertising broadcast physical channel needs to be balanced with the connection requirements needed to maintain the already established connection with these other LE devices.

Using device filtering by the scanning device can prevent the scanning device from discovering all the devices in a given area.

#### 4.2.2.5 Connecting procedure

The procedure for forming connections is asymmetrical and requires that one Bluetooth device carries out the advertising procedure while the other Bluetooth device carries out the scanning procedure. The advertising procedure can be targeted, so that only one device will respond to the advertising. The scanning device can also target an advertising device by first discovering that the advertising device is present in a connectable manner, and in the given area, and then scanning only connectable advertising events from that device using the device filter. After receiving connectable

advertising events from the targeted advertising device, it can initiate a connection by sending the connection request to the targeted advertising device over the primary advertising physical channel or secondary advertising physical channel.

Time spent scanning while connected needs to be balanced with the connection requirements needed to maintain the already established connection with other LE devices.

#### 4.2.2.6 Connected mode

After a successful connection procedure, the devices are physically connected to each other within a piconet. This means that there is a piconet physical channel to which they are both connected, there is a physical link between the devices, and there are default LE-C and LE-U logical links. When in the connected mode it is possible to change the properties of the physical and logical links while remaining connected to the piconet physical channel, such as changing the adaptive frequency hopping sequence or changing the maximum length of data packets. It is also possible for the device to carry out advertising, scanning or discovery procedures without needing to disconnect from the original piconet physical channel.

Additional logical links are created using the Link Manager that exchanges LL Protocol messages with the remote Bluetooth device to negotiate the creation and settings for these links. One of these links (LE-C) transports the LL control protocol and is invisible to the layers above the Link Manager. The other link (LE-U) transports the L2CAP signaling protocol and any multiplexed L2CAP best-effort channels. It is common to refer to a default LE ACL logical transport, which can be resolved by context, but typically refers to the default LE-U logical link.

These two logical links share a logical transport.

During the time that a Peripheral is actively connected to a piconet there is always a default LE ACL logical transport between the Peripheral and the Central. The method of deleting the default LE ACL logical transport is to detach the device from the piconet physical channel, at which time the entire hierarchy of L2CAP channels, logical links, and logical transports between the devices is deleted.

##### 4.2.2.6.1 Connection Subrating

Connection subrating is a means of quickly modifying the effective connection interval of an existing LE ACL connection. It is accomplished by the Central and Peripheral skipping most of the underlying connection events; for example, if the subrating factor is set to 7 then only every $7^{th}$ connection event is used.

When connection subrating is applied to an existing connection, the active duty cycle may be quickly reduced, saving power, but the original connection interval can

be quickly restored for improved throughput when needed. Subrated connections rely on an underlying connection interval which is understood by both Central and Peripheral and, in so doing, allow reliable and immediate reductions and changes to the connection interval without waiting for connection updates which rely on an instant several connection events in the future. Devices that have subrated their default LE ACL logical transports may use the absent periods to engage in activity on another logical transport, or to enter reduced power mode. An LE subrated connection interval only affects the specified LE ACL logical transport and does not apply to any associated LE CIS logical transports that may be active.

#### 4.2.2.7 Periodic advertising procedure

An advertiser uses the periodic advertising procedure to perform unidirectional periodic broadcasts to devices in the area. The unidirectional broadcast occurs without a connection between the advertising device and the listening devices. The periodic advertising procedure can be used to synchronize with nearby devices to provide deterministic periodic broadcast of user data to scanning devices listening on the advertising physical channel. The advertising procedure uses the primary and secondary advertising physical channel to broadcast control information about the periodic advertising, referred to as the periodic advertising synchronization information. The advertiser can also pass the periodic advertising synchronization information to another connected device over the LE-C logical link.

An LE device synchronized with other LE devices on the periodic physical channel uses the periodic advertising event. Time spent periodic advertising while connected or synchronized with other LE devices needs to be balanced with the connection and synchronization requirements needed to maintain the already established connection(s) or synchronizations.

#### 4.2.2.8 Periodic advertising synchronization procedure

The procedure for synchronizing to periodic advertising consists of two parts: obtaining the periodic advertising synchronization information and then listening for the periodic advertisements. The first part may be done using either of two methods.

The first method requires that one Bluetooth device carries out the advertising procedure while the other Bluetooth device carries out the scanning procedure. The scanning device can target an advertising device by first discovering that the advertising device is present and broadcasting periodic advertisements in the given area. The scanning device then needs to receive the advertising events containing the periodic advertising synchronization information from the targeted advertising device.

The second method requires that a device that already has the periodic advertising synchronization information passes it to another connected device over the LE-C logical link.

Once the receiving device has obtained the periodic advertising synchronization information, the second part of the procedure is for it to listen directly for those periodic advertising events; the receiving device is synchronized when it has successfully received one such event.

Synchronizing devices may already be advertising, scanning, or be connected to other Bluetooth devices in a piconet or synchronized to other periodic advertisements. Any time spent synchronizing to periodic advertising needs to be balanced with the requirements needed to maintain already established connections or synchronizations.

#### 4.2.2.9 Periodic advertising synchronized mode

After a successful periodic advertising synchronization procedure, the devices are physically synchronized to each other. This means that there is a periodic physical channel to which they are both synchronized, there is a periodic physical link between the devices, and there is an ADVB-U and an ADVB-C logical link. It is also possible for the device to carry out advertising, scanning, or discovery procedures without needing to disconnect from the LE periodic physical channel.

A Link Layer that is listening to periodic advertising may report the data in the periodic advertisements to the Host. When it is not reporting the data to the Host, the Link Layer does not need to listen to as many events to maintain synchronization, thereby potentially providing more time for other procedures or reducing power consumption.

#### 4.2.2.10 Decision-based scanning

When advertising data is being offloaded to the secondary advertising physical channel, the advertiser can include information about the content of the auxiliary advertising packet in the packets on the primary advertising physical channel. This information (the “decision data”) can be provided by the Host to the Controller on the advertising device.

A scanning device can use the decision data provided by the advertiser to filter out auxiliary packets that its Host would not make use of. It does this by deciding, based on the decision data, whether or not to listen to those packets. This results in the scanning device maximizing the time that it spends scanning for packets on the primary advertising physical channel, which in turn leads to lower latency in discovery and connection procedures.

On the scanning device, the Host can provide instructions to its Controller on how to process the decision data to determine whether to accept or reject the received packet.

### 4.2.3 [This section is no longer used]

# 5 SECURITY OVERVIEW

## 5.1 Security architecture

The Bluetooth security model includes five distinct security features: pairing, bonding, device authentication, encryption and message integrity.

- Pairing: the process for creating one or more shared secret keys
- Bonding: the act of storing the keys created during pairing for use in subsequent connections in order to form a trusted device pair
- Device authentication: verification that the two devices have the same keys
- Encryption: message confidentiality
- Message integrity: protects against message forgeries

The Bluetooth Core security architecture has evolved over time and therefore there are several security mechanisms.

BR/EDR Legacy Pairing uses the E21 and E22 algorithms based on SAFER+. Device authentication is based on the E1 algorithm, which is also based on SAFER+. Encryption uses the E0 algorithm derived from the Massey-Rueppel algorithm. There is no provision for cryptographic message integrity. While the CRC provides some integrity protection, it is not considered to provide cryptographic integrity as it can be easily forged.

Secure Simple Pairing uses FIPS-approved algorithms (SHA-256, HMAC-SHA-256 and the P-192 elliptic curve) and four association models: Just Works, Numeric Comparison, Passkey Entry and Out-Of-Band (see Section 5.2). Device authentication and encryption are the same as BR/EDR Legacy Pairing.

LE Legacy Pairing uses AES-CCM encryption and four association models similar to, though not the same as, those in Secure Simple Pairing. It also provides a privacy feature.

Secure Connections on the BR/EDR physical transport upgrades Secure Simple Pairing to use the P-256 elliptic curve and device authentication to use FIPS-approved algorithms (HMAC-SHA-256 and AES-CTR). Secure Connections also adds message integrity (AES-CCM).

Secure Connections on the LE physical transport upgrades LE Legacy Pairing to use FIPS-approved algorithms (AES-CMAC and P-256 elliptic curve) and adapts the Numeric Comparison association model of Secure Simple Pairing to be used on the Bluetooth LE physical transport. It also includes provisions for a key generated using

Secure Connections on either physical transport to preclude the need for the user to pair a second time on the other physical transport.

The security key hierarchy for BR/EDR is shown in Figure 5.1. The key hierarchy is different depending on whether a physical link is using Secure Connections or legacy security procedures and algorithms.

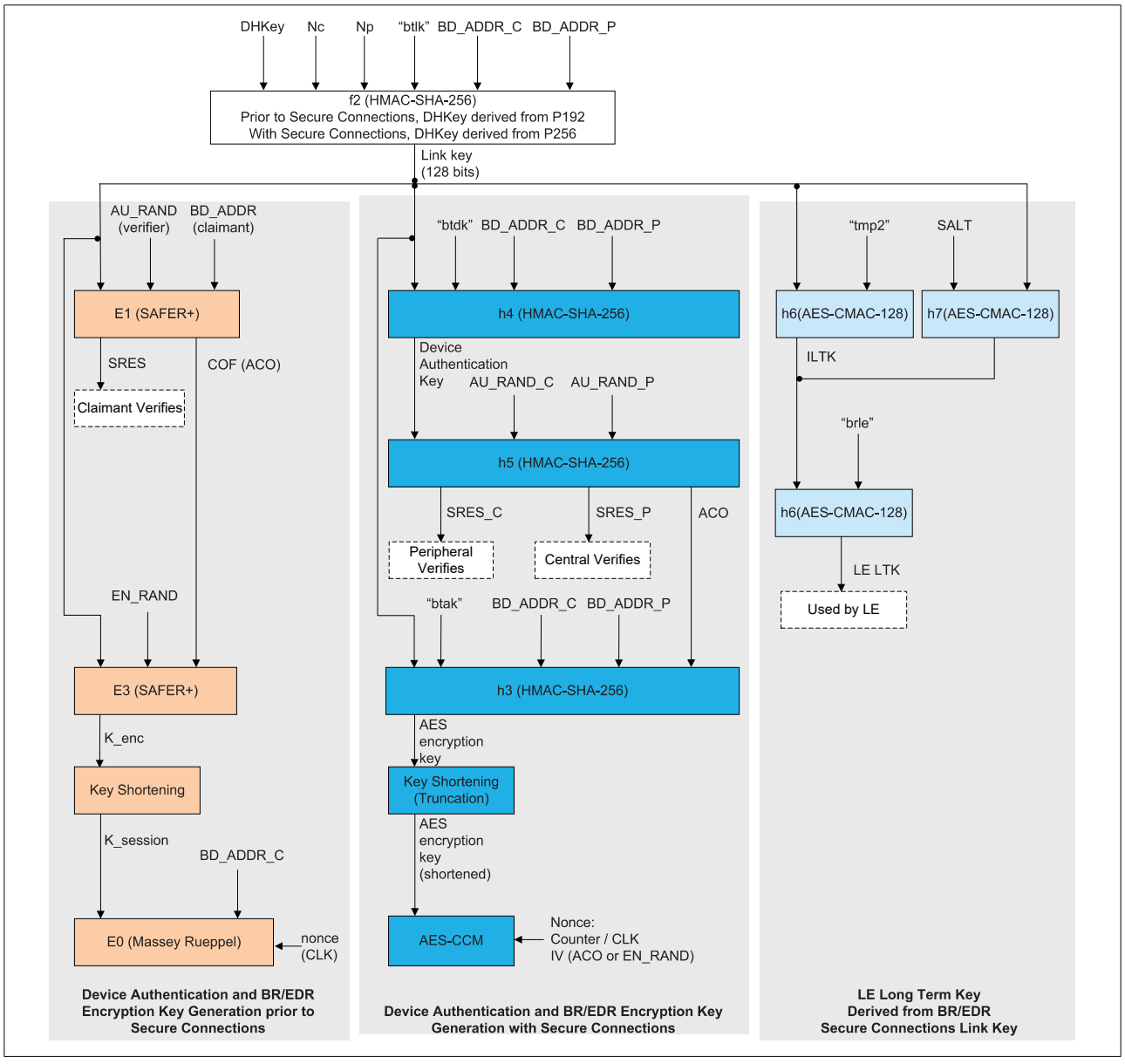



*Figure 5.1: BR/EDR key hierarchy*

The security key hierarchy for LE is shown in Figure 5.2. The key hierarchy is different depending on whether a physical link is using LE Secure Connections or LE legacy pairing procedures and algorithms.

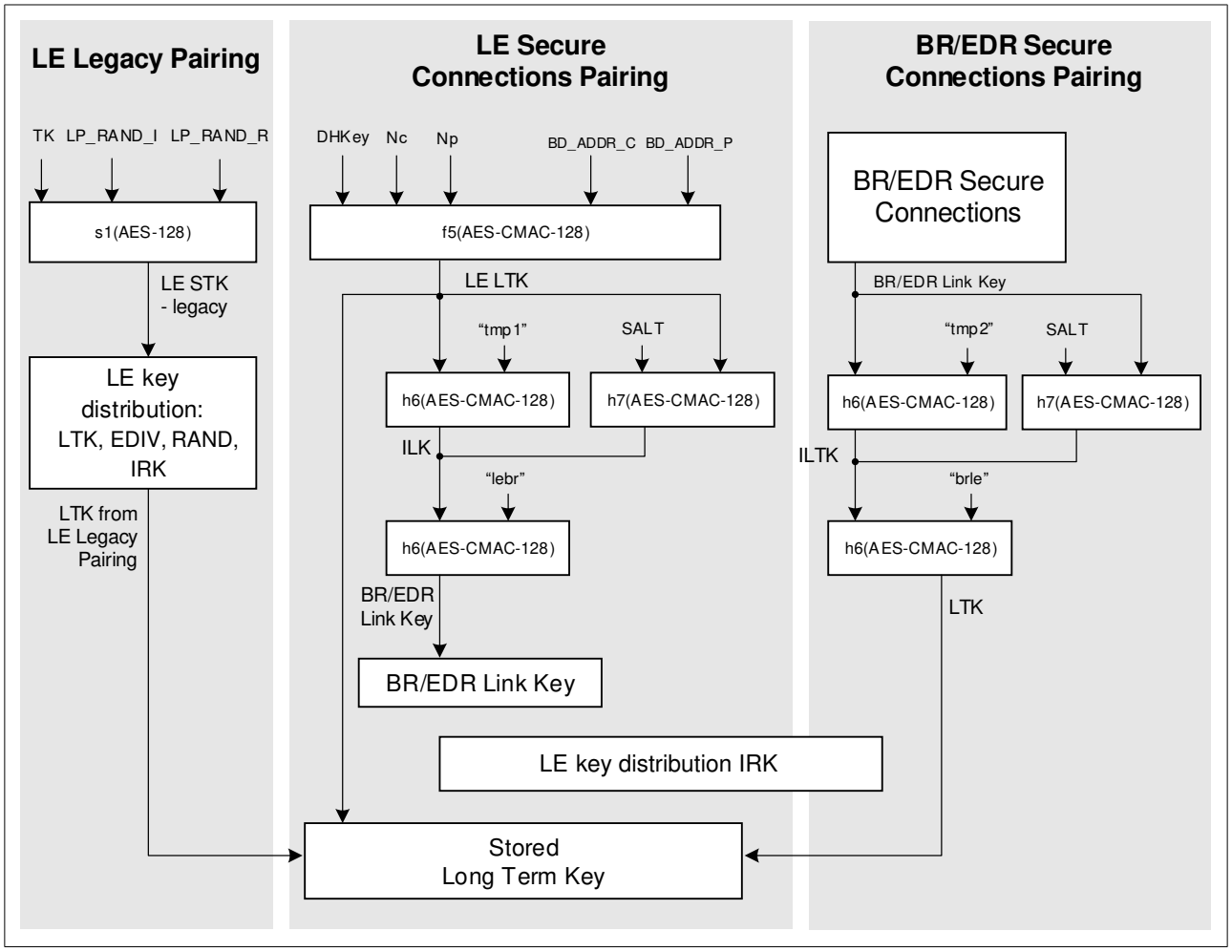



*Figure 5.2: LE key hierarchy*

The 5.2 version of the Core Specification adds a new LE security mode that enables transmission and reception of encrypted isochronous data over the Broadcast Isochronous Stream (BIS) logical transport.

Channel Sounding uses an AES-128 block based DRBG to exchange content that is used for reverification of Channel Sounding peers. The security features related to Channel Sounding are described in Section 9.4.

## 5.2 BR/EDR Secure Simple Pairing

The primary goal of Secure Simple Pairing is to simplify the pairing procedure for the user. Secondary goals are to maintain or improve the security in Bluetooth wireless technology. Since high levels of security and ease-of-use are often at opposite ends of the spectrum in many technologies and products, much care has been taken to maximize security while minimizing complexity from the end user's point of view.

### 5.2.1 Security goals

Secure Simple Pairing has two security goals: protection against passive eavesdropping and protection against man-in-the-middle (MITM) attacks (active eavesdropping). It is a goal of Secure Simple Pairing to exceed the maximum security level provided by the use of a 16 character, alphanumeric, case-sensitive PIN with BR/EDR Legacy Pairing, which often used a 4-digit PIN or a fixed PIN of commonly known values significantly limiting the security on the link.

### 5.2.2 Passive eavesdropping protection

A strong link key coupled with a strong encryption algorithm is necessary to give the user protection against passive eavesdropping. The strength of the link key is based on the amount of entropy (or randomness) in its generation process which would not be known by an attacker. Using legacy pairing, the only source of entropy is the PIN which, in many use cases, is typically four digits either selected by the user or fixed for a given product. Therefore, if the pairing procedure and one authentication exchange is recorded one can run an exhaustive search to find the PIN in a very short amount of time on commonly available computing hardware. With Secure Simple Pairing, the recording attack becomes much harder as the attacker must have solved a hard problem in public key cryptography in order to derive the link key from the recorded information. This protection is independent of the length of the passkey or other numeric values that the user must handle. Secure Simple Pairing gives the same resistance against the recording and passive eavesdropping attacks even when the user is not required to do anything.

Secure Simple Pairing uses Elliptic Curve Diffie Hellman (ECDH) public key cryptography as a means to thwart passive eavesdropping attacks. ECDH provides a very high degree of strength against passive eavesdropping attacks but it may be subject to MITM attacks, which however, are much harder to perform in practice than the passive eavesdropping attack (see Section 5.2.3).

Using BR/EDR Legacy Pairing with a 16 numeric digit PIN achieves about 53 bits of entropy whereas a 16 character, alphanumeric, case sensitive PIN yields about 95 bits of entropy when the entire 62 character set is used ([0, ... 9, 'A', ... 'Z', 'a', ... 'z']). For devices that do not support the Secure Connections feature, Secure Simple Pairing has approximately 96 bits of entropy using the FIPS approved P-192 elliptic curve which is at least as good as the entropy in BR/EDR Legacy Pairing using a 16 character, alphanumeric, case sensitive PIN. For devices that do support the Secure Connections feature, Secure Simple Pairing has approximately 128 bits of entropy using the FIPS-approved P-256 elliptic curve.

### 5.2.3 Man-in-the-middle protection

A man-in-the-middle (MITM) attack occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. The third device then relays information between the two devices giving the illusion that they are directly connected. The attacking device may even eavesdrop on communication between the two devices (known as active eavesdropping) and is able to insert and modify information on the connection. In this type of attack, all of the information exchanged between the two devices are compromised and the attacker may inject commands and information into each of the devices thus potentially damaging the function of the devices. Devices falling victim to the attack are capable of communicating only when the attacker is present. If the attacker is not active or out range, the two victim devices will not be able to communicate directly with each other and the user will notice it.

To prevent MITM attacks, Secure Simple Pairing offers two user assisted numeric methods: numeric comparison or passkey entry. If Secure Simple Pairing would use 16 decimal digit numbers, then the usability would be the same as using legacy pairing with 16 decimal digit PIN. The chance for a MITM to succeed inserting its own link keys in this case is a 1 in $10^{16} = 2^{53}$ pairing instances, which is an unnecessarily low probability.

Secure Simple Pairing protects the user from MITM attacks with a goal of offering a 1 in 1,000,000 chance that a MITM could mount a successful attack. The strength of the MITM protections was selected to minimize the user impact by using a six digit number for numeric comparison and Passkey entry. This level of MITM protection was selected since, in most cases, users can be alerted to the potential presence of a MITM attacker when the connection process fails as a result of a failed MITM attack. While most users feel that if they have not compromised their passkey, a 4-digit key is sufficient for authentication (i.e., bank card PIN codes), the use of six digits allows Secure Simple Pairing to be FIPS-compliant and this was deemed to have little perceivable usability impact.

### 5.2.4 Association models

Secure Simple Pairing uses four association models referred to as Numeric Comparison, Just Works, Out Of Band, and Passkey Entry. Each of these association models are described in more detail in the following sections.

The association model used is deterministic based on the IO capabilities of the two devices.

#### 5.2.4.1 Numeric Comparison

The Numeric Comparison association model is designed for scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter "yes" or "no". A good example of this model is the cell phone / PC scenario.

The user is shown a six digit number (from "000000" to "999999") on both displays and then asked whether the numbers are the same on both devices. If "yes" is entered on both devices, the pairing is successful.

The numeric comparison serves two purposes. First, since many devices do not have unique names, it provides confirmation to the user that the correct devices are connected with each other. Second, the numeric comparison provides protection against MITM attacks (see Section 5.2.3).

There is a significant difference from a cryptographic point of view between Numeric Comparison and the PIN entry model used by BR/EDR Legacy Pairing. In the Numeric Comparison association model, the six digit number is an artifact of the security algorithm and not an input to it, as is the case in the PIN entry model. Knowing the displayed number is of no benefit in decrypting the encoded data exchanged between the two devices.

#### 5.2.4.2 Just Works

The Just Works association model is primarily designed for scenarios where at least one of the devices does not have a display capable of displaying a six digit number nor does it have a keyboard capable of entering six decimal digits. A good example of this model is the cell phone/mono headset scenario where most headsets do not have a display.

The Just Works association model uses the Numeric Comparison protocol but the user is never shown a number and the application may simply ask the user to accept the connection (exact implementation is up to the manufacturer).

The Just Works association model provides the same protection as the Numeric Comparison association model against passive eavesdropping but offers no protection against the MITM attack.

When compared against today's experience of a headset with a fixed PIN, the security level of the Just Works association model is considerably higher since a high degree of protection against passive eavesdropping is realized.

#### 5.2.4.3 Out of Band

The Out of Band (OOB) association model is primarily designed for scenarios where an Out of Band mechanism is used to both discover the devices as well as to exchange or

transfer cryptographic numbers used in the pairing process. In order to be effective from a security point of view, the Out of Band channel should provide different properties in terms of security compared to the Bluetooth radio channel. The Out of Band channel should be resistant to MITM attacks. If it is not, then the Out of Band association model will not offer protection against MITM attacks and so is vulnerable to them, meaning security may be compromised during authentication.

The user's experience differs a bit depending on the Out of Band mechanism. As an example, with a Near Field Communication (NFC) solution, the user(s) will initially touch the two devices together, and is given the option to pair the first device with the other device. If "yes" is entered, the pairing is successful. This is a single touch experience where the exchanged information is used in both devices. The information exchanged includes discovery information (such as the Bluetooth Device Address) as well as cryptographic information. One of the devices will use a Bluetooth Device Address to establish a connection with the other device. The rest of the exchanged information is used during authentication.

The OOB mechanism may be implemented as either read only or read/write. If one side is read only, a one-way authentication is performed. If both sides are read/write, a two-way authentication is performed.

The OOB protocol is selected only when the pairing process has been activated by previous OOB exchange of information and one (or both) of the device(s) gives OOB as the IO capabilities. The protocol uses the information which has been exchanged and simply asks the user to confirm connection.

The OOB association model supports any OOB mechanism where cryptographic information and the Bluetooth Device Address can be exchanged. The OOB association model does not support a solution where the user has activated a Bluetooth connection and would like to use OOB for authentication only.

#### 5.2.4.4 Passkey Entry

The Passkey Entry association model is primarily designed for the scenario where one device has input capability but does not have the capability to display six digits and the other device has output capabilities. A good example of this model is the PC and keyboard scenario.

The user is shown a six digit number (from "000000" to "999999") on the device with a display, and is then asked to enter the number on the other device. If the value entered on the second device is correct, the pairing is successful.

There is a significant difference from a cryptographic point of view between Passkey Entry and the PIN entry model used by BR/EDR Legacy Pairing. In the Passkey Entry association model, the six digit number is independent of the security algorithm and not

an input to it, as is the case in the PIN entry model. Knowing the entered number is of no benefit in decrypting the encoded data exchanged between the two devices.

#### 5.2.4.5 Association model overview

The following diagram shows Secure Simple Pairing from the point of view of the technology used for discovery and then the different association possibilities.

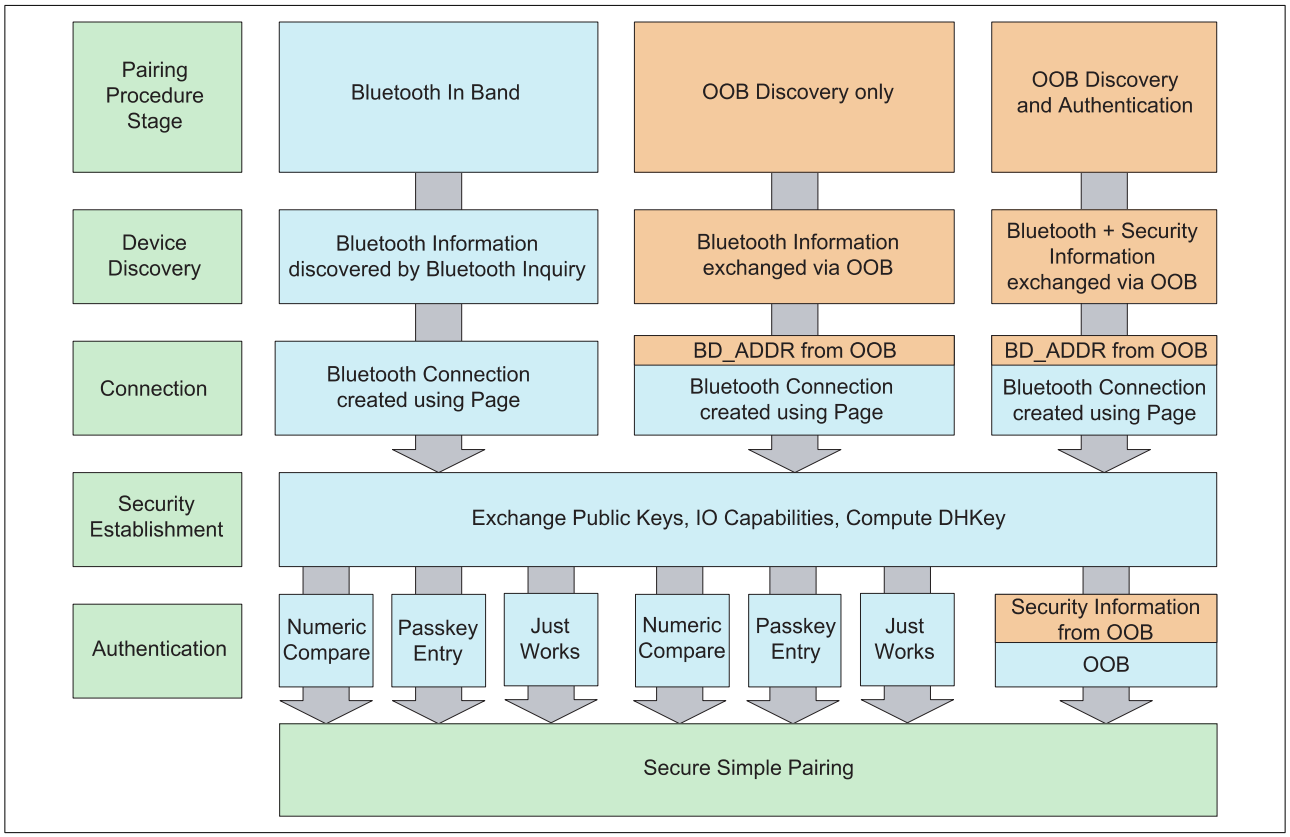



*Figure 5.3: Secure Simple Pairing association models*

## 5.3 Secure Connections Only mode

When a device requires that only FIPS-approved algorithms are used on the BR/EDR or LE physical transport, it should enter Secure Connections Only Mode. Secure Connections Only Mode is sometimes called a "FIPS Mode". This mode should be used when it is more important for a device to have high security than it is for it to maintain backwards compatibility with devices that do not support Secure Connections. The Host will enforce that the P-256 elliptic curve is used during pairing and that, on the BR/EDR physical transport, the secure authentication sequences are used and AES-CCM is used for encryption.

If a BR/EDR/LE device is configured in Secure Connections Only Mode, then both the BR/EDR and the LE transports will be in Secure Connections Only Mode.

## 5.4 LE security

LE Legacy Pairing has some differences in security aspects compared to BR/EDR security features such as Secure Simple Pairing. The association models are similar to BR/EDR Secure Simple Pairing from the user perspective and have the same names but have differences in the quality of the protection provided.

### 5.4.1 Association models

Bluetooth LE uses four association models referred to as Just Works, Numeric Comparison, Out of Band and Passkey Entry. LE legacy pairing does not have an equivalent of Numeric Comparison.

In LE legacy pairing, each of these association models is similar to BR/EDR Secure Simple Pairing with the following exceptions.

- Just Works and Passkey Entry do not provide any passive eavesdropping protection. This is because Secure Simple Pairing uses Elliptic Curve Diffie-Hellman and LE legacy pairing does not.

In LE Secure Connections pairing, the four association models are functionally equivalent to BR/EDR Secure Connections.

The use of each association model is based on the IO capabilities of the devices.

### 5.4.2 Key generation

Key generation in Bluetooth LE is performed by the Host on each LE device independent of any other LE device. By performing key generation in the Host, the key generation algorithms can be upgraded without the need to change the Controller.

Note: Key generation in BR/EDR is performed in the Controller.

Bluetooth LE uses multiple keys, each for a specific purpose, as follows:

- Confidentiality of data and device authentication
- Authentication of unencrypted data
- Device Identity

In LE, the key used to provide confidentiality and authentication is generated by combining contributions from each device during pairing.

### 5.4.3 Encryption

Encryption in Bluetooth LE uses AES-CCM cryptography. Like BR/EDR, in LE encryption is performed in the Controller.

### 5.4.4 [This section is no longer used]

### 5.4.5 Privacy feature

Bluetooth LE supports a feature that reduces the ability to track a LE device over a period of time by changing the Bluetooth Device Address on a frequent basis.

In order for a device using the privacy feature to reconnect to known devices, the device address, referred to as the private address, must be resolvable by the other device. The private address is generated using the device's identity resolving key (IRK) exchanged during the bonding procedure.

The term "resolution" means a process used by a device to calculate the device Identity Address from the received private address and the IRK, while the state "resolved" is the successful result of a resolution.

There are two variants of the privacy feature. In the first variant, private addresses are resolved and generated by the Host. In the second variant, private addresses are resolved and generated by the Controller without involving the Host after the Host provides the Controller device identity information. In addition, the second variant may involve the Host when the resolving list in the Controller cannot store all the device identity resolving keys necessary for bonded devices.

There are two modes of privacy: device privacy mode and network privacy mode. A device in device privacy mode is only concerned about the privacy of the device and will accept advertising packets from peer devices that contain their Identity Address as well as ones that contain a private address, even if the peer device has distributed its IRK in the past. In network privacy mode, a device will only accept advertising packets from peer devices that contain private addresses. By default, network privacy mode is used when private addresses are resolved and generated by the Controller.

The Host maintains a resolving list by adding and removing device identities. The Host may provide the Controller with a complete resolving list or a subset of the resolving list. A device identity consists of the peer's Identity Address and a local and peer's IRK pair.

When the Controller performs address resolution and the Host needs to refer to a peer device that is included in the resolving list, it uses the peer's device Identity Address. Likewise, all incoming events from the Controller to the Host will use the peer's device identity, provided that the peer's device address has been resolved. If the Controller cannot resolve the peer's device Identity Address in an advertisement, it may pass the event to the Host for resolution in the Host.

Device filtering becomes possible when address resolution is performed in the Controller because the peer's device Identity Address can be resolved prior to checking whether it is in the Filter Accept List.

Figure 5.4 shows a logical representation of the relationship between the Controller resolving list and the Controller Filter Accept List. Actual implementations of the resolving list and Filter Accept List are not required to follow this model. The resolving list may be independent of the Filter Accept List.

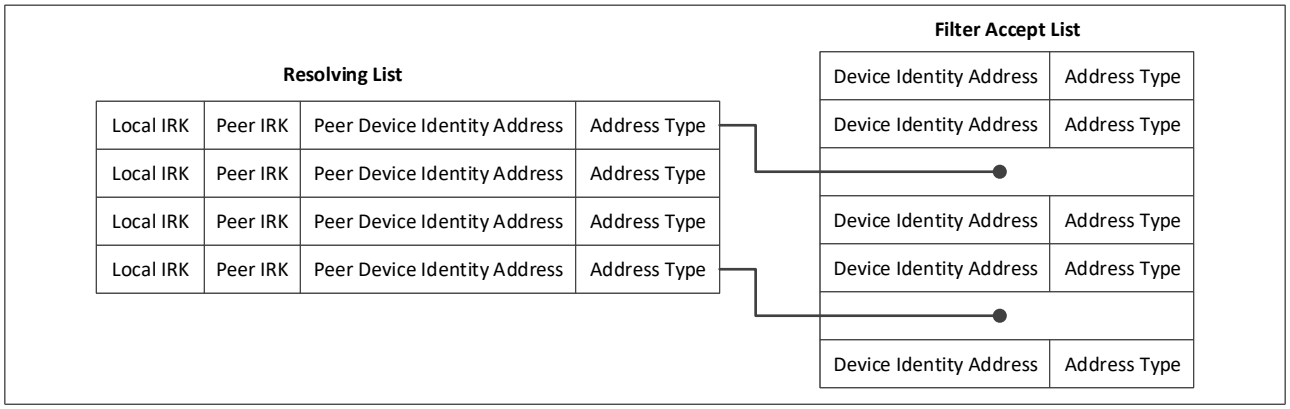



*Figure 5.4: Logical representation of the resolving list and Filter Accept List*

Note: Not all devices in the Filter Accept List are device Identity Addresses.

### 5.4.6 Encrypted Advertising Data

The privacy feature described in Section 5.4.5 allows the Bluetooth Device Address to be changed periodically while still allowing the fingerprinting of devices based on the contents of their advertising data. It is possible to encrypt advertising data by encapsulating the normal advertising data within an Encrypted Data data type using a pre-shared session key and a nonce. Because the encrypted advertising data nonce is changed whenever the private address is changed, the encrypted data before and after the change is also different. This approach prevents the tracking of devices based solely on the private address and advertising data.

The encrypted advertising data pre-shared session key is communicated only to peer devices that are authorized to receive such information. Only devices that have the key material can decrypt and authenticate messages and track the advertising device.

## 5.5 [This section is no longer used]

## 5.6 Key generation between BR/EDR and LE physical transports

When two BR/EDR/LE devices support Secure Connections over both transports, keys for both transports may be generated during a single pairing procedure. The ability to convert keys from one transport to the other prevents the need to pair twice, thus enabling a better user experience.

The link key for BR/EDR generated during Phase 4 of Secure Simple Pairing on the BR/EDR physical transport may be converted to a Long Term Key (LTK) for use on the

LE transport. Similarly, an LTK generated during Phase 2 of pairing on the LE physical transport may be converted to the BR/EDR Link Key for use on the BR/EDR physical transport.

# 6 BLUETOOTH APPLICATION ARCHITECTURE

## 6.1 Bluetooth profiles

Application interoperability in the Bluetooth system is accomplished by Bluetooth profiles. Bluetooth profiles define the required functions and features of each layer in the Bluetooth system from the PHY to L2CAP and any other protocols outside of this specification. The profile defines the vertical interactions between the layers as well as the peer-to-peer interactions of specific layers between devices.

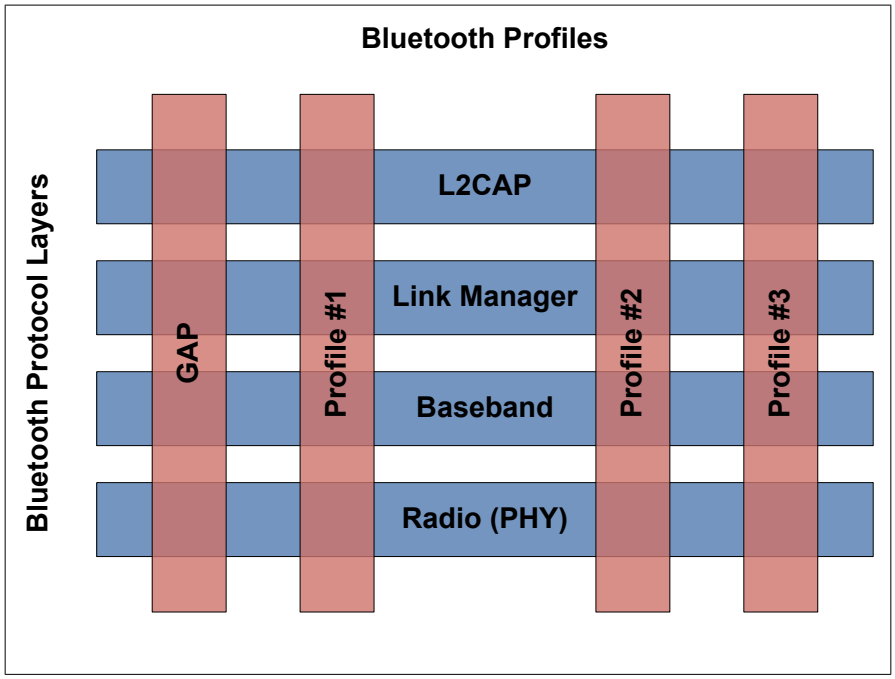



*Figure 6.1: Bluetooth profiles*

In addition, application behaviors and data formats are also defined by the profile. When two devices comply with all the requirements of a Bluetooth profile, interoperability of an application is enabled.

All profiles describe service discovery requirements necessary for devices to connect, find available application services and connection information necessary for making application level connections.

## 6.2 Generic Access Profile

The Bluetooth system defines a base profile which all Bluetooth devices implement. This profile is the Generic Access Profile (GAP), which defines the basic requirements of a Bluetooth device. For instance, for BR/EDR, it defines a Bluetooth device to include

the Radio Physical Layer, Baseband, Link Manager, L2CAP, and the Service Discovery protocol functionality; for LE, it defines the Radio Physical Layer, Link Layer, L2CAP, Security Manager, Attribute Protocol and Generic Attribute Profile. This ties all the various layers together to form the basic requirements for a Bluetooth device. It also describes the behaviors and methods for device discovery, connection establishment, security, authentication, association models and service discovery.

In BR/EDR, GAP defines a single role with functionality that may be present in each device. This functionality includes how devices discovery each other, establish connections and describes security association models used for authentication. In BR/EDR this functionality may be present in both devices. It may be necessary for a device to implement both the initiating and accepting functionality in order to discover or establish connections with all devices. A device may only include either the initiating or the accepting functionality but it requires the remote device to support the complimentary functionality to discovery or establish connections with the device. For BR/EDR, the Controller is required to support all the functionality, however the Host may limit this functionality based on the other profiles or use cases supported by the device.

In LE, GAP defines four specific roles: Broadcaster, Observer, Peripheral, and Central. A device may support multiple LE GAP roles provided that the underlying Controller supports those roles or role combinations. Each role specifies the requirements for the underlying Controller. This allows for Controllers to be optimized for specific use cases.

The Broadcaster role is optimized for transmitter only applications. Devices supporting the Broadcaster role use advertising to broadcast data. The Broadcaster role does not support connections. The Observer role is optimized for receiver only applications. Devices supporting the Observer role are the complementary device for a Broadcaster and receives broadcast data contained in advertisements. The Observer role does not support connections. The Peripheral role is optimized for devices that support a single connection and are less complex than Centrals; it uses the Link Layer Peripheral role within the connection. The Central role supports multiple connections and is the initiator for all connections with devices in the Peripheral role; it uses the Link Layer Central role within the connection. Devices supporting the Central role generally support more complex functions compared to the other LE GAP roles.

## 6.3 Profile hierarchy

Since all Bluetooth devices are required to implement GAP, any additional profiles implemented by a Bluetooth device become supersets of GAP. Depending on the complexity of an application or the ability to reuse common requirements of functionality of the Bluetooth system between many applications, additional generic profiles can be created that depend on GAP or other generic profiles, as well as enabling other profiles. A top level profile that describes application interoperability is called an Application Profile.

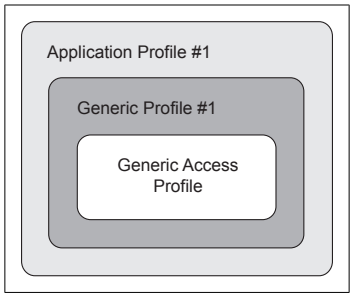



*Figure 6.2: Profile hierarchy*

Application profiles contain by reference GAP and any other generic profile that describes a set of common requirements of the Bluetooth system.

## 6.4 Generic Attribute Architecture

### 6.4.1 Attribute Protocol

To allow devices to read and write small data values held on a server, an Attribute Protocol (ATT) is defined. Each stored value, typically only a few octets, is known as an attribute. This protocol allows each attribute to be self-identifying using UUIDs to identify the type of data. These UUIDs can be well-known assigned numbers defined in the Assigned Numbers document and associated specifications, or a vendor assigned 128-bit UUID.

Attribute Protocol messages are sent over L2CAP channels, known as the ATT bearers.

Attribute Protocol defines two roles: Client and Server. A device can be both an ATT Client and an ATT Server at the same time. Attribute Protocol messages on a single ATT bearer allow a single transaction in each direction to be outstanding at a time. When a response to a message is received, the next transaction can be initiated. When multiple ATT bearers are created, each ATT bearer has a separate transaction model and therefore multiple ATT transactions can be outstanding at the same time, one per bearer. This can be used to allow multiple higher layer specifications to send messages concurrently.

The ATT Server stores the attributes and accepts Attribute Protocol requests, commands and confirmations from the ATT Client. The ATT Server sends responses to requests and, when configured by a higher layer, sends indications and notifications asynchronously to the ATT Client when specified events occur on the ATT Server.

### 6.4.2 Generic Attribute Profile

Generic Attribute Profile (GATT) is built on top of the Attribute Protocol (ATT) and establishes common operations and a framework for the data transported and stored by the Attribute Protocol. GATT defines two roles: Server and Client. A GATT Client

or Server is an ATT Client or Server respectively that conforms to the requirements in GATT. The GATT roles are not necessarily tied to specific GAP roles but may be specified by higher layer profiles. GATT and ATT are not transport specific and can be used in both BR/EDR and LE. However, GATT and ATT are mandatory to implement in LE since it is used for discovering services.

GATT also specifies the format of data contained on the GATT Server. Attributes, as transported by the Attribute Protocol, are formatted as Services and Characteristics. Services may contain a collection of characteristics. Characteristics contain a single value and any number of descriptors describing the characteristic value.

With the defined structure of services, characteristics and characteristic descriptors a GATT Client that is not specific to a profile can still traverse the GATT Server and display characteristic values to the user. The characteristic descriptors can be used to display descriptions of the characteristic values that may make the value understandable by the user.

## 6.5 GATT-Based Profile hierarchy

The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile.

The top level of the hierarchy is a profile. A profile is composed of one or more services necessary to fulfill a use case. A service is composed of characteristics or references to other services. Each characteristic contains a value and may contain information about the value. The service and characteristic and the components of the characteristic (i.e., value and descriptors) contain the profile data and are all stored in Attributes on the server.

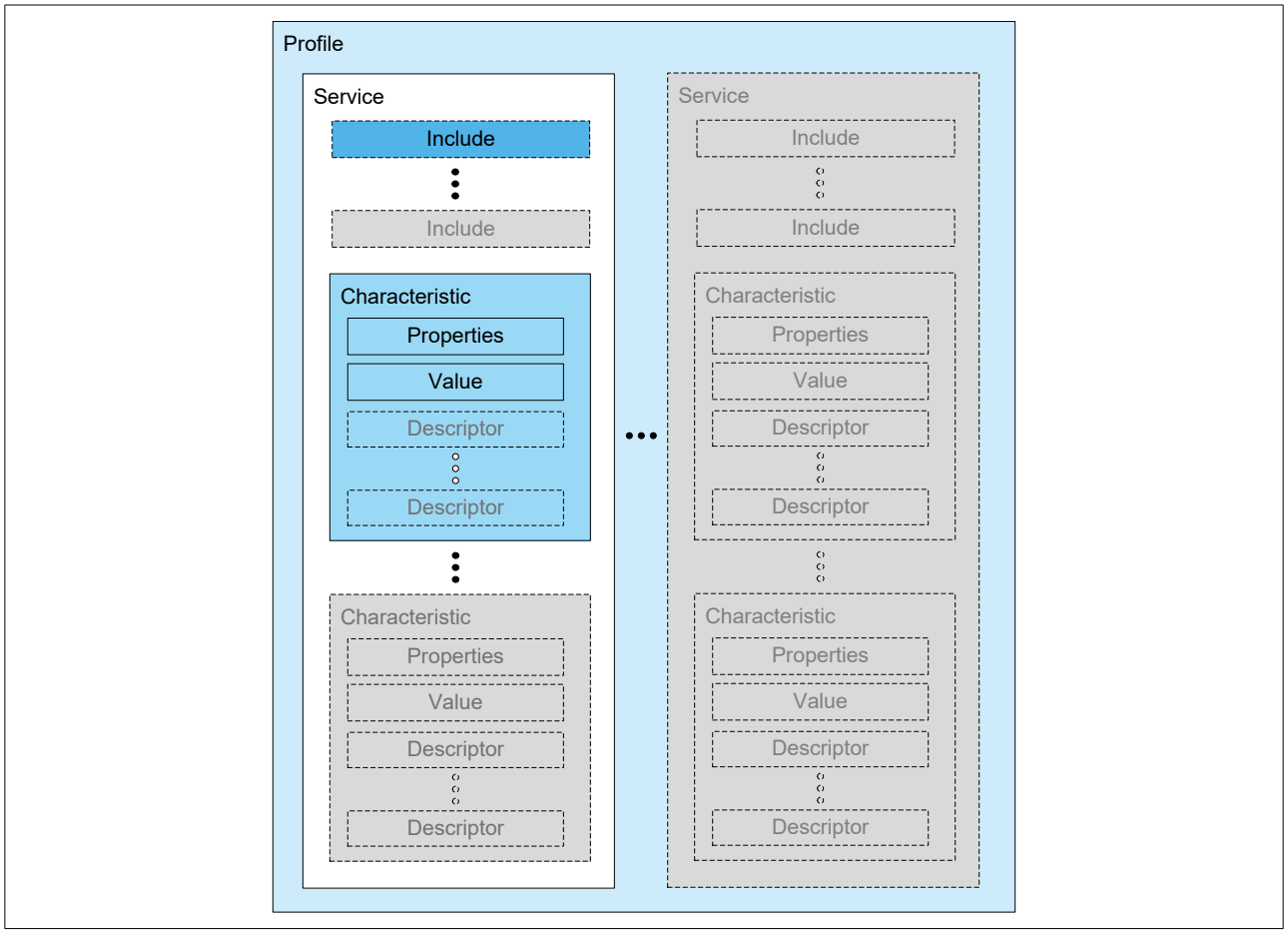



*Figure 6.3: GATT-Based Profile hierarchy*

### 6.5.1 Service

A service is a collection of data and associated behaviors to accomplish a particular function or feature of a device or portions of a device. A service may include other primary or secondary services and/or a set of characteristics that make up the service.

There are two types of services: primary and secondary. A primary service is a service that provides functionality of a device that can be used on its own. A secondary service is a service that provides additional functionality of a device in association with a primary service and is included from at least one primary service on the device.

To maintain backward compatibility with earlier clients, later revisions of a service definition can only add new included services or optional characteristics. Later revisions of a service definition are also forbidden from changing behaviors from previous revision of the service definition.

Services may be used in one or more profiles to fulfill a particular use case.

### 6.5.2 Included services

An included service is a method to incorporate another service definition on the server as part of the service including it. When a service includes another service, the entire included service becomes part of the new service including any nested included services and characteristics. The included service still exists as an independent service. There are no limits to the depth of nesting.

### 6.5.3 Characteristic

A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. A characteristic definition contains a characteristic declaration, characteristic properties, and a value. It may also contain descriptors that describe the value or permit configuration of the server with respect to the characteristic value.

## 6.6 [This section is no longer used]

# 7 COEXISTENCE AND COLLOCATION

Bluetooth devices operate in the unlicensed 2.4 GHz Industrial, Scientific and Medical (ISM) band. Many other technologies also make use of the ISM band, including wireless LAN, cordless phones, and microwave ovens. The ISM band is also close enough to other frequency bands that Bluetooth devices may be an interferer of or a victim of other technologies.

Radios may be collocated or non-collocated. The term "collocated" is a loose one - in this specification, collocated radios are assumed to be in the same product (a Multi-Radio Terminal or MRT) and may have mechanisms to coordinate their activity in order to mitigate interference.

Determining the amount of expected isolation between radios is important for choosing an appropriate coexistence mechanism. With sufficient isolation, frequency division duplexing (FDD) techniques are the most efficient. With insufficient isolation or a shared antenna, time division duplexing (TDD) techniques need to be used. In many cases, a combination of FDD and TDD techniques are required to achieve acceptable levels of performance.

This specification supports a variety of features that help mitigate interference to other devices and to minimize interference from other devices. Broadly, the types of solutions fall into the following categories:

| Type | Description |
|---|---|
| Frequency division | Simultaneous use of multiple radios enabled by filters and/or isolation |
| Time division | One radio may transmit or receive at a time through scheduling or prioritization |
| Time alignment | Activities of the collocated radios are aligned in the time domain to optimize performances by avoiding conflicting activities. E.g. Transmissions of multiple radios may occur simultaneously, multiple receptions may occur simultaneously, but it is not possible to transmit and receive simultaneously |
| Hybrid frequency and time division | Use of frequency division, time alignment, and time division techniques depending on the relative frequencies in use by the radios, filters and isolation |

*Table 7.1: Interference mitigation types*

## 7.1 Core features supporting coexistence and collocation

There are features in the specification to specifically target the reduction of interference from collocated or non-collocated devices.

| Feature | Version Introduced | Description |
|---|---|---|
| Adaptive Frequency Hopping | 1.2 | Allows devices to reduce the number of channels used in a piconet in order to avoid interferers |
| HCI Set Host Channel Classification | 1.2 | Allows a Host to inform the local Bluetooth Controller of the channels that are occupied by a collocated technology |
| Enhanced SCO (eSCO) | 1.2 | Added retransmissions to SCO for the purpose of combating interference |
| MWS Coexistence Signaling | CSA3 | Provides a standardized interface between collocated radios for communicating information necessary to enable some coexistence techniques |
| Piconet Clock Adjustment | 4.1 | Allows a Bluetooth device to align the piconet clock with an external technology, e.g. Long Term Evolution (LTE) |
| Train Nudging | 4.1 | Provides a mechanism to improve the success rate of page and inquiry when the slots to receive the respective responses are periodically not available |
| Generalized Interlaced Scanning | 4.1 | Provides a mechanism to improve the success rate of page scan and inquiry scan when some slots are periodically not available for scanning |
| Slot Availability Mask | 5.0 | Provides a mechanism for two Bluetooth devices to indicate to each other the availability of their time slots |

*Table 7.2: Interference mitigation features*

## 7.2 Adaptive Frequency Hopping

Adaptive Frequency Hopping (AFH) allows Bluetooth devices to improve their immunity to interference from and avoid causing interference to other devices in the 2.4 GHz ISM band. The basic principle is that Bluetooth channels are classified into two categories, used and unused, where used channels are part of the hopping sequence and unused channels are replaced in the hopping sequence by used channels in a pseudo-random way. This classification mechanism allows for the Bluetooth device to use either all or fewer than the channels available. The minimum number of channels allowed by the Bluetooth specification is 20 on BR/EDR and 2 on LE.

The specification defines the aspects of AFH necessary to ensure interoperability, including the hopping kernel, Baseband behavior, Link Manager Protocol (LMP) commands, Link Layer behavior and commands, and Host Controller interface (HCI) commands and events required to change and configure hopping sequences. The Bluetooth Specification also defines a mechanism that allows for a Peripheral to report channel classification information to the Central.

Adaptive Frequency Hopping makes use of metrics obtained through many sources. These metrics are analyzed and then the resulting Channel_Map is used by the adaptive frequency hopping kernel. The metrics may come from over-the-air measurements, data supplied by the Host (e.g., via HCI commands), or reports by the Peripheral or from other hardware coexistence interfaces.

While AFH is a critical element in coexistence, it is not enough in some circumstances.

## 7.3 Coexistence between Bluetooth Devices and Wireless LAN Devices

Coexistence between Bluetooth and Wireless LAN has traditionally been a combination of Adaptive Frequency Hopping (AFH) and proprietary techniques to prioritize traffic between the two protocols. The specification does not specify signaling between the Bluetooth Controller and a Wireless LAN device.

## 7.4 Mobile Wireless Standards (MWS) coexistence

Significant interference can be present between the Bluetooth radio and a collocated MWS radio operating in frequency bands adjacent to the 2.4 GHz ISM band. This interference can prevent one radio from receiving while the other radio is transmitting.

The “Filter recommendations for Coexistence with LTE and WiMAX” whitepaper[1] describes filter specifications that, in some cases, can reduce collocated interference to an acceptable level. The specification includes complementary solutions to traditional filtering including features for Bluetooth Controllers and Hosts as well as signaling and messaging mechanisms between collocated MWS and Bluetooth radios. Figure 7.1 illustrates the general architecture model for these mechanisms. This architecture assumes separate antennae with limited isolation.

[1] https://www.bluetooth.com/develop-with-bluetooth/build/white-papers

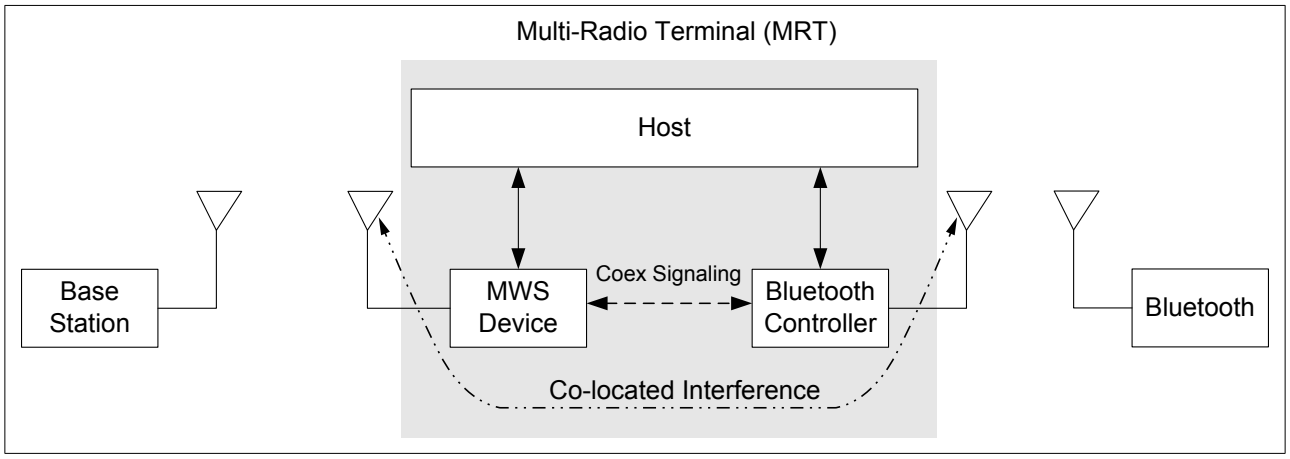



*Figure 7.1: MWS coexistence architecture*

Two types of solutions have been considered. In the first solution, both Bluetooth transmissions (TX) and receptions (RX) are constrained by collocated MWS activities. Solutions of this type are called Pure TDM (Time Division Multiplexed) Mode. In the second solution, only Bluetooth receptions are affected by collocated MWS transmissions and Bluetooth transmissions do not impact the operation of the collocated MWS. Solutions of this type are called Hybrid Mode and are achieved, for example, by using steep roll-off Band Select Filters (BSFs) for the ISM band in the Bluetooth transceiver. Hybrid Mode applies where the Bluetooth transmission's effect on MWS is sufficiently reduced via filtering that the Bluetooth device can transmit during the MWS downlink time. This requires a frequency guard band between the Bluetooth and MWS operational frequency ranges as well as constraints on both the Bluetooth BSF and the MWS BSF. The requirement for a time domain solution still remains, but only to protect Bluetooth reception.

These solutions are facilitated by an MWS coexistence signaling mechanism (see [Vol 7] Part A) and multiple transport layers (see [Vol 7] Part B and [Vol 7] Part C).

MWS technologies operate in licensed bands and use centralized scheduling to support Wide Area Network services. An MWS radio synchronizes both time and frequency with a network Base Station. The Base Station determines which MWS radio will transmit or receive and when. MWS radios have no control over when to transmit or receive. When Bluetooth transmissions interfere with MWS receptions in the MRT, the MWS radio can be rendered unusable if the Bluetooth radio transmits freely. Figure 7.2 shows how Bluetooth activity can interfere with every MWS reception opportunity and similarly how MWS transmissions can interfere with Bluetooth reception. In the example shown in Figure 7.2 the Bluetooth device in the MRT is operating as the Central of a piconet. Blocks marked with a "C" are single slot Central transmissions and those marked with a "P" are single slot Peripheral transmissions. The times at which reception by the MWS device may be corrupted by Bluetooth transmissions are marked with a red shade.

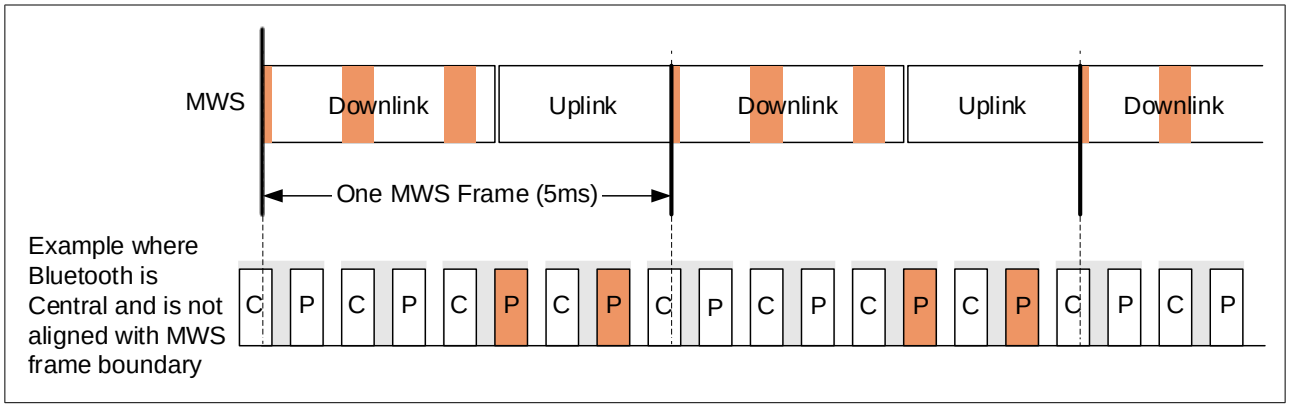



*Figure 7.2: MWS receptions interfered with by uncontrolled Bluetooth transmissions*

Even with the best relative timing relationship (when the Bluetooth slot boundary is aligned with the MWS frame boundary), the Bluetooth radio in the MRT suffers reduced transmission and reception opportunities due to time multiplexing with the collocated MWS radio. The Bluetooth radio only gets one transmission/reception opportunity every MWS frame, as shown in Figure 7.3.

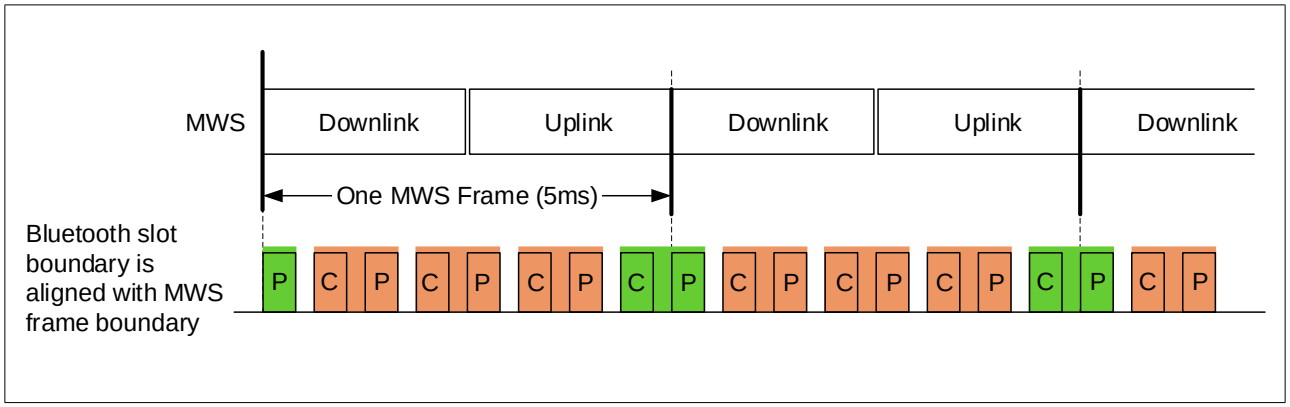



*Figure 7.3: Bluetooth radio has reduced transmission/receiving opportunities*

Consequently, three out of four (for 5 ms MWS frames) or seven out of eight (for 10 ms MWS frames) slot pairs of Bluetooth can be punctured by MWS activities. Furthermore, since Bluetooth inquiry and paging use a sequence of 16 channels at a time, when the Bluetooth radio in the MRT is performing inquiry or paging the channel sequence will repeat every 5 ms resulting in the same channels being repeatedly punctured by the collocated MWS activities, see Figure 7.4. As a result, there is a high probability that the remote scanning device will not be able to receive the page or inquiry IDs within the current timeout.

When the inquiry or paging channel sequence gets repeatedly punctured, Train Nudging can be used to add an additional offset to the clock bits in order to shift the channel sequence.

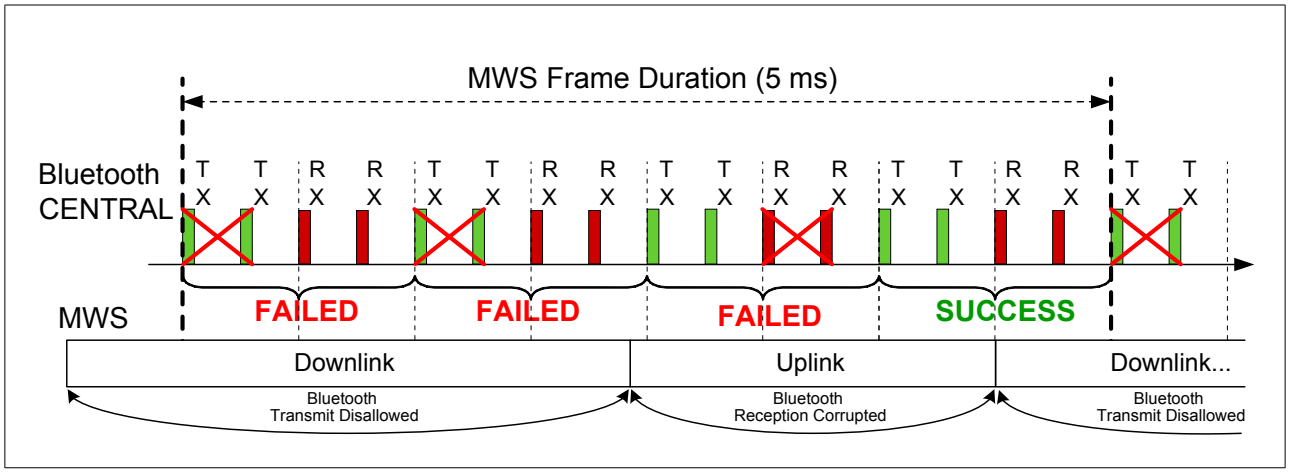



*Figure 7.4: Bluetooth radio can suffer inquiry/page failures*

As MWS transmissions interfere with Bluetooth receptions in the MRT, up to 50% of the transmitted IDs from the remote inquiry/page device will not be received by the Bluetooth radio in the MRT performing scanning, as shown in Figure 7.5.

Based on the pattern of slots that are not available for scanning, Generalized Interlaced Scanning can be used to tune the phase of the second scan during a back-to-back scan.

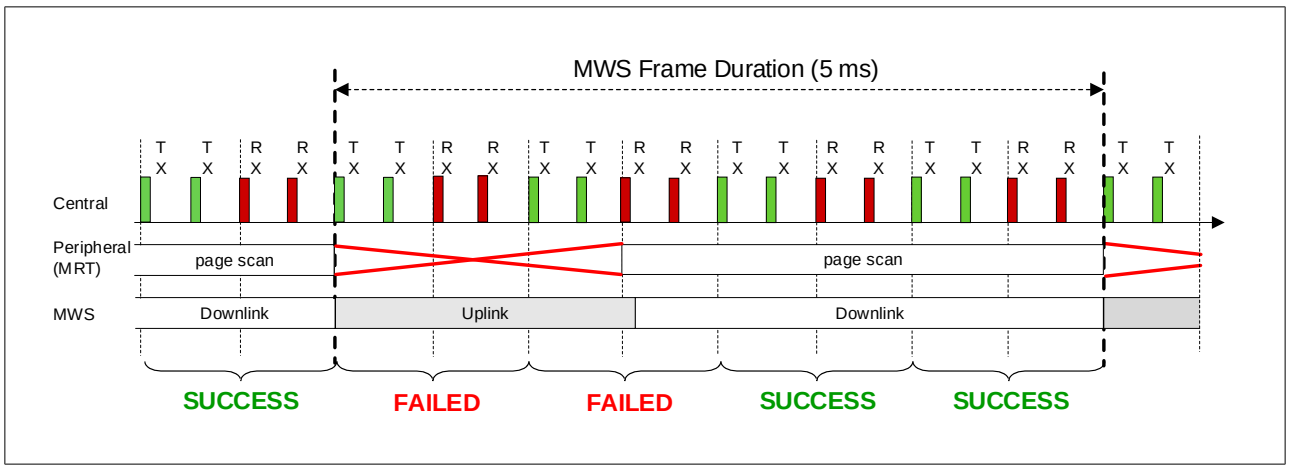



*Figure 7.5: Bluetooth radio can suffer inquiry scan/page scan failures*

## 7.5 Synchronizing Bluetooth with an external timing source

This section provides an example to illustrate the synchronization of the Bluetooth CLK with an MWS system so that the Bluetooth slots line up with the Downlink and Uplink times of the MWS system. The external frame in this example is LTE TDD Frame Configuration #1 and special subframe configuration #1; however the same mechanisms can be used for any external TDD/TDMA protocol. The example shows a time span of 10 ms.

[Vol 7] Part A defines the timing of the MWS frame as a fixed offset from the FRAME_SYNC signal in the coexistence signaling. This is shown in Figure 7.6 as FS. FS can be defined by the Host as any specific offset within the MWS frame. For a piconet Central, the most useful position for FS is the boundary between the uplink and the following downlink. This is because the Central needs to transmit in a Central slot during the uplink and then receive in the following Peripheral slot during the downlink. Putting FS at this boundary allows the Central to easily align its Bluetooth clock to put it between these slots. The situation is reversed in a Peripheral: FS is most useful on the boundary between the downlink and the following uplink.

The HCI command HCI_Set_External_Frame_Configuration ([Vol 4] Part E, Section 7.3.81) can be used to describe the MWS frame timing. This knowledge, together with FS, allows the Bluetooth Controller to align the Bluetooth clock with the MWS frame timing so as to minimize the effect of mutual interference. This is illustrated in Figure 7.6. The red ovals show slot pairs where both MWS and Bluetooth transmit and receive simultaneously and so do not interfere with each other. The MWS frame structure includes a downlink portion (“D”), an uplink portion (“U”), and a special portion (“S”) that includes a downlink and uplink portion separated by a guard period (“GP”).

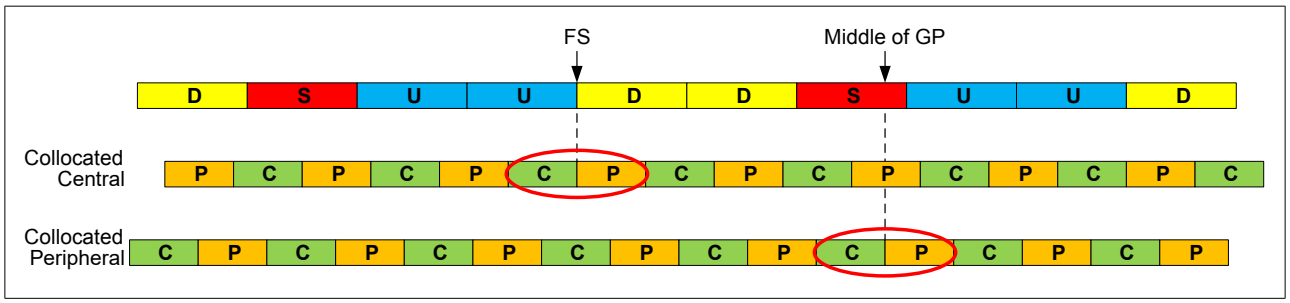



*Figure 7.6: Alignment of MWS frame timing and Bluetooth clock*

Note: In LTE, if the implementation does not have access to the exact LTE timing it can assume that the downlink-to-uplink boundary is the middle of the GP in a special LTE subframe.

## 7.6 Piconet clock adjustment

As discussed in the previous section, aligning the MWS and Bluetooth clocks correctly greatly improves throughput on both technologies. The Central has two mechanisms at its disposal to mitigate misalignments.

Coarse Clock Adjustments can be used to move the Bluetooth CLK using the LMP_CLK_ADJ PDU. The clk_adj_us parameter is used to align the slots with the MWS alignment point indicated by the FRAME_SYNC signal. The clk_adj_slots parameter can be used to move the CLK several slots forward in time. This can be useful to align e.g. eSCO. Coarse Clock Adjustment is expected to be used only rarely, for instance

at MWS connection or when the MWS frame timing changes due to roaming. Coarse Clock Adjustment can only be used when all Peripherals in the piconet support it.

The other option for the Central is to use Clock Dragging. This is a method of slowly adjusting the phase of the clock backward or forward by making the slots a few µs shorter or longer, respectively, until the desired CLK phase has been achieved. It should be noted that this is a very slow rate of adjustment, as it is designed to allow a legacy device to track the change, and therefore the requirements of [Vol 2] Part B, Section 1.1 mean that Clock Dragging must not be done at a faster rate than the maximum natural drift between devices. For this reason its main use is to facilitate small corrections over time if a misalignment with the MWS system is detected. If any device is connected that does not support Coarse Clock Adjustment, slowly moving the Peripheral using Clock Dragging is the only option.

It is recommended to let the collocated device be the Central, when possible, as it can react much faster to correct misalignments. If a Peripheral is the collocated device, doing a role switch to make it Central may be worth considering. Alternatively a Peripheral can send a Piconet Clock Adjustment Request LMP packet to the Central. The Central then has the option to perform a Coarse Clock Adjustment, Clock Dragging, or to reject the request.

## 7.7 Slot Availability Mask (SAM)

Slot Availability Mask (SAM) allows two Bluetooth devices to indicate to each other time slots that are available for transmission and reception. The SAM slot map specifies the availability or otherwise of Bluetooth slots. A slot could be unavailable because of external conditions (e.g., MWS coexistence) or internal conditions (e.g., scatternet commitments). SAM does not impose new mandatory rules for the scheduling of BR/EDR time slots. Instead, it merely provides information which allows Controllers to refine their scheduling of Bluetooth slots to improve performance.

SAM slot maps are calculated by the Controller itself based on its scheduling requirements. There are no HCI commands defined specifically for SAM, merely LMP sequences that enable devices to exchange maps and indicate the map in use. The HCI commands HCI_Set_External_Frame_Configuration (see [Vol 4] Part E, Section 7.3.81) and HCI_Set_MWS_PATTERN_Configuration (see [Vol 4] Part E, Section 7.3.85) and real-time signals (e.g., MWS_PATTERN_Index or FRAME_SYNC) provided by the Coexistence Logical Interface (see [Vol 7] Part A) contain information concerning the appropriate SAM_Index and SAM anchor point to use for MWS coexistence; these may therefore trigger these LMP sequences.

A Controller may choose to perform a Piconet Clock Adjustment before initiating an LMP_SAM_SWITCH sequence so as to increase the number of slot pairs available per MWS frame.

# 8 DIRECTION FINDING USING BLUETOOTH LOW ENERGY

An LE device can make its direction available for a peer device by transmitting direction finding enabled packets. Using direction information from several transmitters and profile-level information giving their locations, an LE radio can calculate its own position.

This feature is supported over the LE Uncoded PHYs, but not over the LE Coded PHY.

## 8.1 Angle of arrival (AoA) method

An LE device can make its direction available to a peer device by transmitting direction finding enabled packets using a single antenna.

The peer device, consisting of an RF switch and antenna array, switches antennae while receiving part of those packets and captures IQ samples. The IQ samples can be used to calculate the phase difference in the radio signal received using different elements of the antenna array, which in turn can be used to estimate the angle of arrival (AoA).

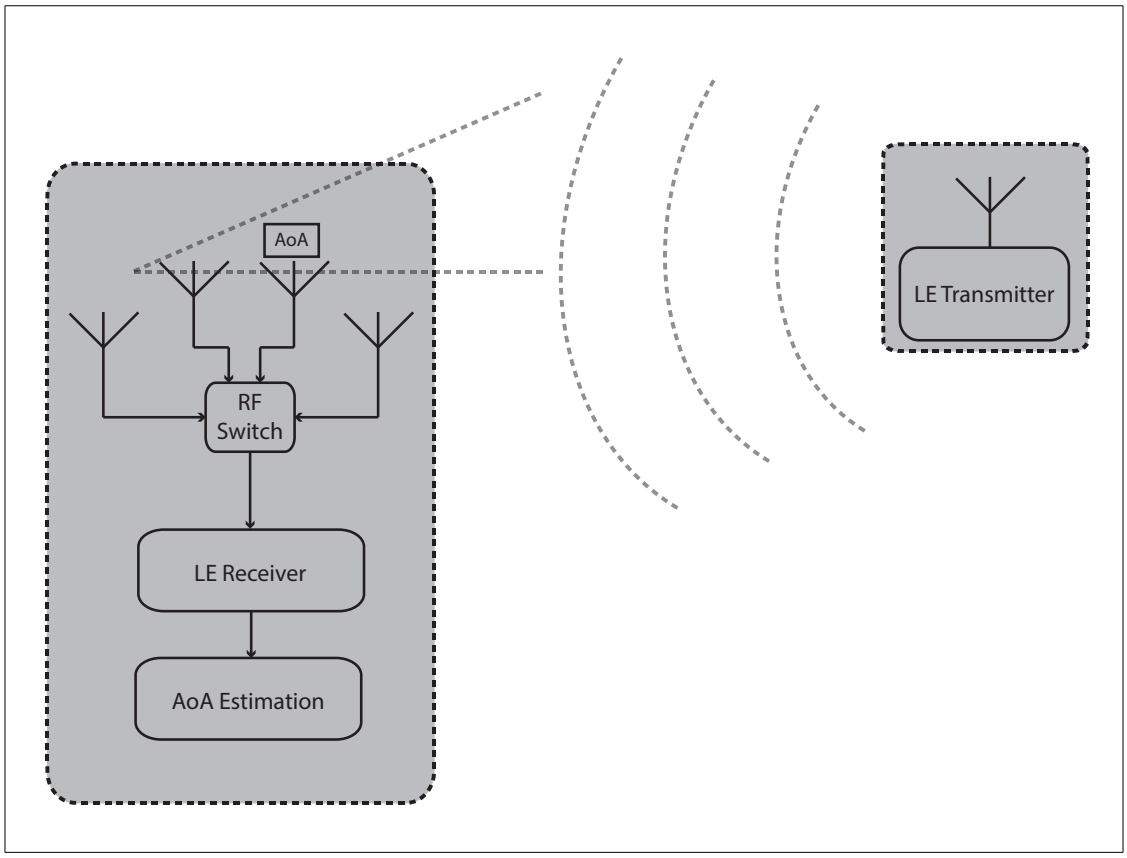



*Figure 8.1: Angle of Arrival method*

Consider a receiver device with an antenna array consisting of two antennae, separated by distance *d*. The transmitter device uses a single antenna to transmit a signal. As shown in Figure 8.2, a perpendicular line can be drawn from an incoming signal wave front extending to the furthest antenna (antenna 2) at the point of intersection to the closest antenna (antenna 1). The adjacent side of that right triangle represents the path difference relative to the angle of incidence of that wave front between both antennae. The phase difference, $\psi$, in the signal arriving at the two antennae is then

$$\psi = (2\pi d \cos(\theta)) \div \lambda$$

where $\lambda$ is the wavelength of the signal and $\theta$ is the angle of arrival (measured from a line connecting the two antennae in the receiver), and so

$$\theta = \arccos((\psi\lambda) \div (2\pi d))$$

Note: The distance d is profile-level information that is used by the receiving device to calculate the angle of arrival.

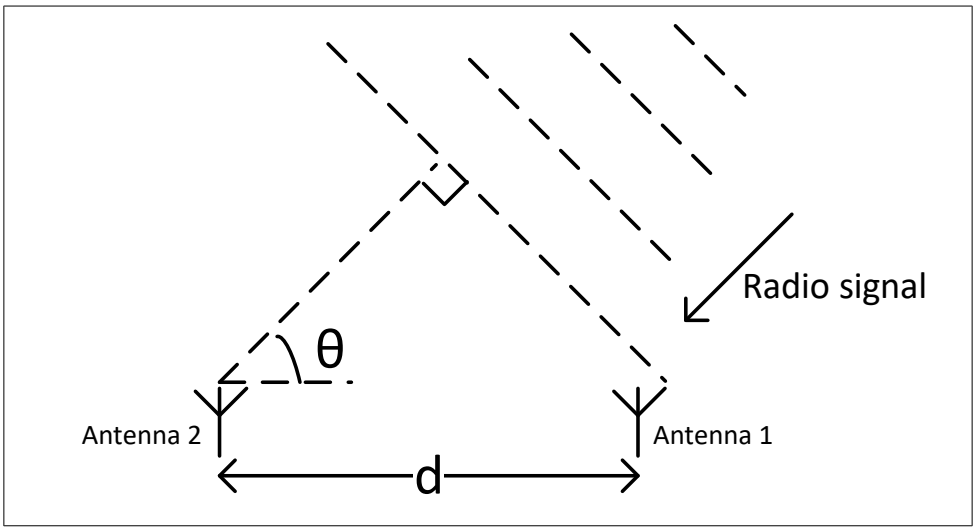



*Figure 8.2: Measuring the angle of arrival*

## 8.2 Angle of departure (AoD) method

A device consisting of an RF switch and antenna array can make its angle of departure (AoD) detectable by transmitting direction finding enabled packets, switching antennae during transmission.

The peer device receives those packets using a single antenna and captures IQ samples during part of those packets. Determination of the direction is based on the different propagation delays of the LE radio signal between the transmitting elements of the antenna array and a receiving single antenna. The propagation delays are detectable with IQ measurements. Any receiving LE radio with a single antenna that

supports the AoD feature can capture IQ samples and, with the aid of profile-level information specifying the antenna layout of the transmitter, calculate the angle of incidence of the incoming radio signal.

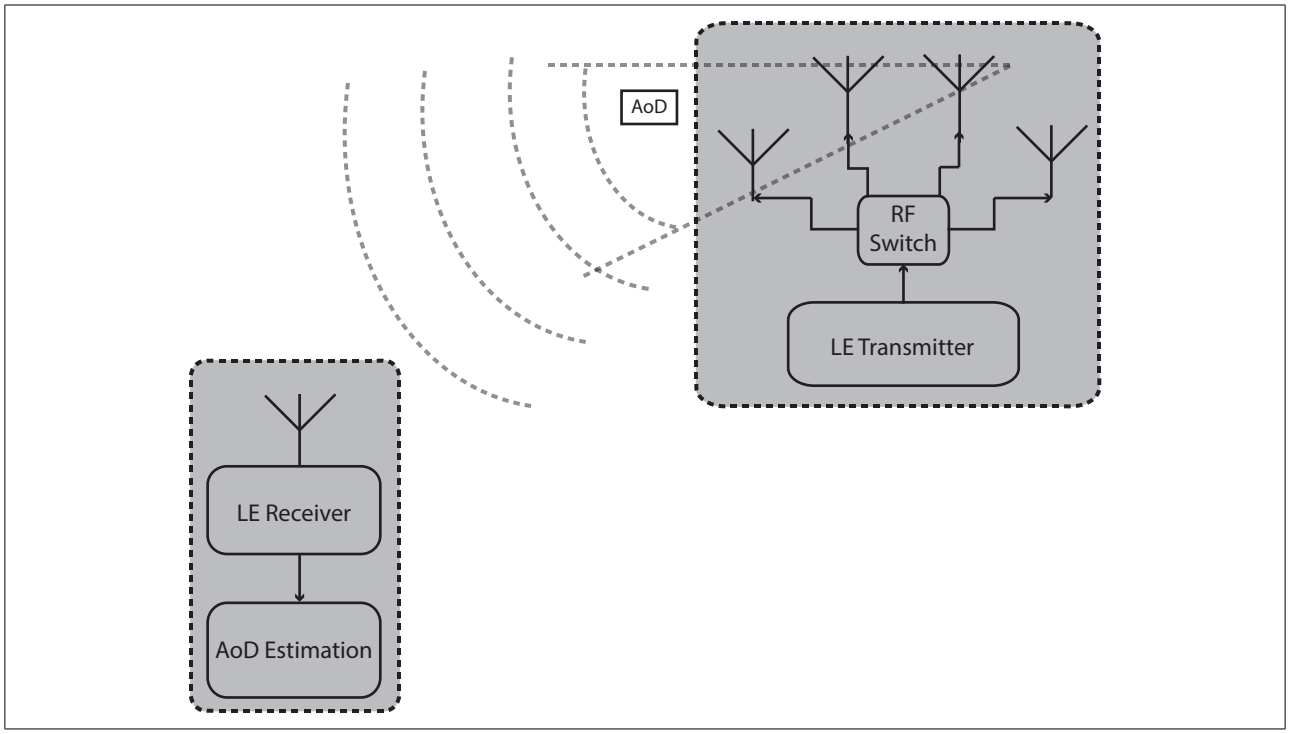



*Figure 8.3: Angle of departure method*

Consider a transmitter device with an antenna array consisting of two antennae, separated by distance *d*. The receiver device uses a single antenna to receive the signals. The phase difference, $\psi$, in the signal from antenna 1 and the signal from antenna 2 arriving at the receiver is then

$$\psi = (2\pi d \cos(\theta)) \div \lambda$$

where $\lambda$ is the wavelength of the signal and $\theta$ is the angle of departure (measured from a line connecting the two antennae in the transmitter), and so

$$\theta = \arccos((\psi\lambda) \div (2\pi d))$$

Note: The distance d is profile-level information that a transmitting device exchanges with the receiving device in order for the receiving device to calculate the angle of departure.

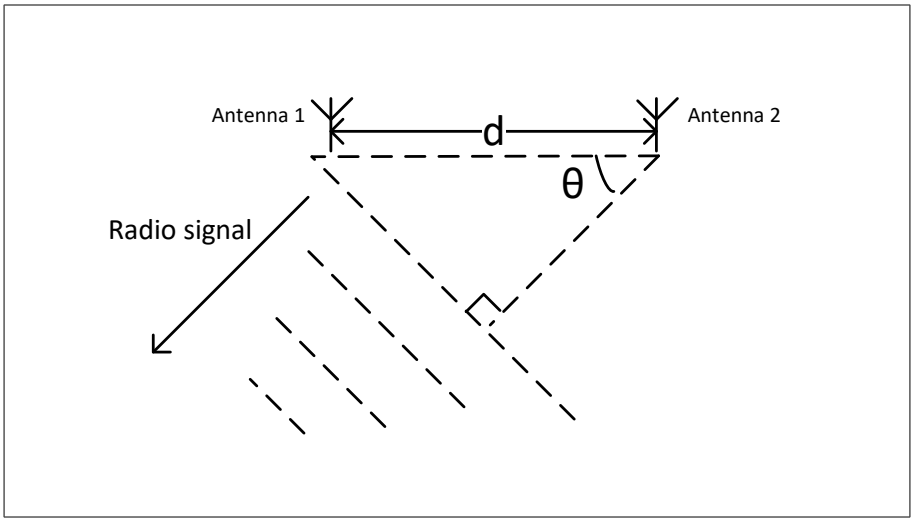



*Figure 8.4: Measuring the angle of departure*

# 9 CHANNEL SOUNDING USING BLUETOOTH LOW ENERGY

An LE device can use the Channel Sounding (CS) feature to characterize the propagation path between itself and a connected peer. The measurements obtained from a CS procedure enable a device to estimate its distance from the peer device. A device supporting the CS feature supports both round-trip time measurements and phase-based ranging measurements.

## 9.1 Channel Sounding procedure

The CS feature consists of a combination of Link Layer procedures and a dedicated air interface protocol that creates a tightly interlocked exchange of RF signals between two devices on multiple RF channels. This exchange is known as a CS procedure.

A CS procedure consists of CS events, subevents and steps. These define a set of time and frequency slots in which two devices exchange a combination of RF signals. The purpose of that exchange is to measure the physical characteristics of the transmission channel. These exchanges are bidirectional; both devices take turns sending and receiving RF signals.

A CS procedure is divided into one or more CS events. A CS event consists of one or more CS subevents. A CS subevent consists of two or more CS steps, each of which starts with a frequency change period called T_FCS. Time separation between CS subevents avoids continuous allocation of the RF resources of a given device and facilitates coexistence with other RF technologies. Within a CS subevent, the first CS steps are used to provide calibration information for the remaining CS steps within that CS subevent. Figure 9.1 shows the relationship between CS procedures, CS events, CS subevents, and CS steps.

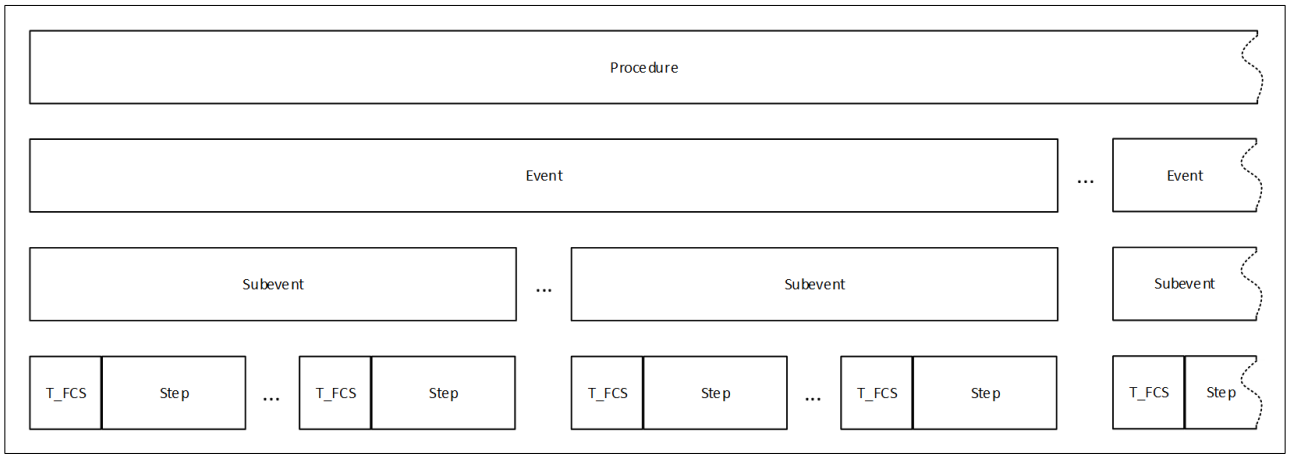



*Figure 9.1: CS procedure/event/subevent/step hierarchy*

Each CS subevent has a separate starting point. The Link Layer procedures that configure devices to run the CS procedure define the CS steps for these subevents, in terms of both time and frequency, relative to these starting points.

Four CS step types, known as mode-0 through mode-3, are defined. Each mode is used for a specific purpose. Mode-0 is used to calibrate one device to the other in terms of frequency and timing. Mode-1 is used to exchange a round-trip time (RTT) packet. Mode-2 is used to exchange phase-based ranging (PBR) CS tones to measure phase and amplitude of the communication channel. Mode-3 is used to exchange both RTT and phase/amplitude measurements.

After the CS procedure completes, each device retains information collected from each CS step that describes the communication channel. Mode-0 steps reflect local frequency adjustments. Mode-1 and mode-3 steps measure time of arrival and time of departure. Mode-2 and mode-3 steps measure phase and magnitude information in the form of in-phase and quadrature (IQ) measurements. The reflector communicates this information to higher layers to be sent back to the initiator.

## 9.2 Distance estimation based on phase and amplitude information

The Channel Sounding feature does not specify an algorithm for computing a distance estimate, but a mathematical representation is provided here.

Let $\theta_{CH}(f)$ represent the phase delay of the channel, where $f$ is the channel frequency, and $\Delta\theta_{LO}(f)$ represents the relative difference in phase of the RF carrier between the devices.

Then the relative phases of a carrier measured at the reflector and initiator's antenna is

$$\theta_{REFL}(f) = \theta_{CH}(f) + \Delta\theta_{LO}(f) \text{ and } \theta_{INIT}(f) = \theta_{CH}(f) - \Delta\theta_{LO}(f).$$

Further, let $A_{REFL}(f)$ and $A_{INIT}(f)$ represent the amplitude of that measured carrier at the reflector and initiator's antenna. Let a phase correction term (PCT) be defined by the angle that, if added to the internal angle of the local oscillator, would result in a phase identical to that of the incoming signal. Then IQ values represented by the PCT measured at the reflector and initiator are given by

$$PCT_{REFL}(f) = A_{REFL}(f)e^{i\theta_{REFL}(f)} \text{ and } PCT_{INIT}(f) = A_{INIT}(f)e^{i\theta_{INIT}(f)}$$

Assume that the communication channel is symmetrical between the initiator and reflector. Then the measured phases are dependent on both the communication channel and the relative difference in phase of the RF carrier between the devices.

The communication channel transfer function can then be estimated from

$$H^2(f) = A_{REFL}(f)e^{i\theta_{REFL}(f)} \times A_{INIT}(f)e^{i\theta_{INIT}(f)} = A_{CH}^2(f)e^{i2\theta_{CH}(f)}$$

Assume that from $H^2(f)$ it is possible to calculate the actual channel transfer function $H(f)$. Assuming that $H(f)$ is a linear-time invariant transfer function, then an under-resolved estimate of the impulse response $h(t)$ can be calculated from the inverse Fourier transform of $H(f)$. Assuming that there is only one propagation path, then the maximum peak in the estimate of the impulse response will occur at the delay between the two devices, and assuming communication is at the speed of light, it is possible to estimate the distance.

If the reflector supports Inline PCT Transfer (IPT) and this feature is enabled, then the reflector compensates the phase of its outgoing carrier using the measured PCT ($\theta_{REFL}(f)$). The relative phase of the carrier measured at the initiator's antenna is then

$$\theta_{INIT}(f) = \theta_{CH}(f) - \Delta\theta_{LO}(f) + \theta_{REFL}(f) = 2\theta_{CH}(f).$$

The IQ value represented by the PCT reported at the reflector with the IPT feature enabled is given by

$$PCT_{REFL}(f) = A_{REFL}(f)e^{i0}$$

Accordingly, the communication channel transfer function with the IPT feature enabled remains the same as before

$$H^2(f) = A_{REFL}(f)e^{i0} \times A_{INIT}(f)e^{i\theta_{INIT}(f)} = A_{CH}^2(f)e^{i2\theta_{CH}(f)}$$

In the case of a single propagation path, a simplification may be made. Here, the distance can also be estimated by the change in phase as a function of frequency. Assume that the distance between initiator and reflector is $x$. The total distance traveled by a reflected signal will then be $2x$.

The equation for wavelength is $\lambda = \frac{c}{f}$. Assume that the initial phase of the initiator is zero. If the channel is symmetrical, and if the initial phase of the reflector is zero; then the total phase change over a distance $x$ is given by

$$\varphi_{CH}(f) = -2 \times 2\pi\frac{x}{\lambda} = -2 \times 2\pi\frac{xf}{c}$$

From this equation, the derivative of the phase as a function of frequency is a constant and is proportional to distance.

$$\frac{d\varphi_{CH}(f)}{df} = -4\pi\frac{x}{c}$$

From the channel transfer function, the phase change can be calculated as a function of frequency.

$$\varphi_{CH}(f) = angle\left(H^2(f)\right) \quad \text{(EQ 1)}$$

An estimate of the distance, x, can then be calculated as follows.

$$x = -\frac{d\varphi_{CH}(f)}{df}\frac{c}{4\pi}$$

When channel measurements are taken at frequencies separated by 1 MHz, a distance ambiguity occurs in the equation above when

$$\varphi_{CH}(f + 1\ \ MHz) - \varphi_{CH}(f) \quad \text{(EQ 2)}$$

is an integer multiple of $2\pi$. This gives a distance ambiguity every $d_{wrap}$ meters, as shown below

$$d_{wrap} = \frac{2\pi}{1\ \text{MHz}}\frac{3\times10^8\ m/s}{4\pi} = 150\ \ m$$

To support distance estimation of larger distances, round-trip timing can be used to disambiguate distance results.

## 9.3 Distance estimation based on RTT packets

For RTT in the CS feature, the two devices exchange packets of the format known as CS_SYNC, which is described in [Vol 6] Part H, Section 2. Time of departure (ToD) and time of arrival (ToA) times are used to estimate the time of flight (ToF), as shown in Figure 9.2.

Assuming the same time base on both devices, sufficient precision of ToD and ToA for the required distance accuracy, and line of sight conditions with no reflections, then an estimated distance, x, can be calculated as shown in the equation below, where c is the speed of light.

$$x = \frac{T_{initator} - T_{reflector}}{2} \times c$$

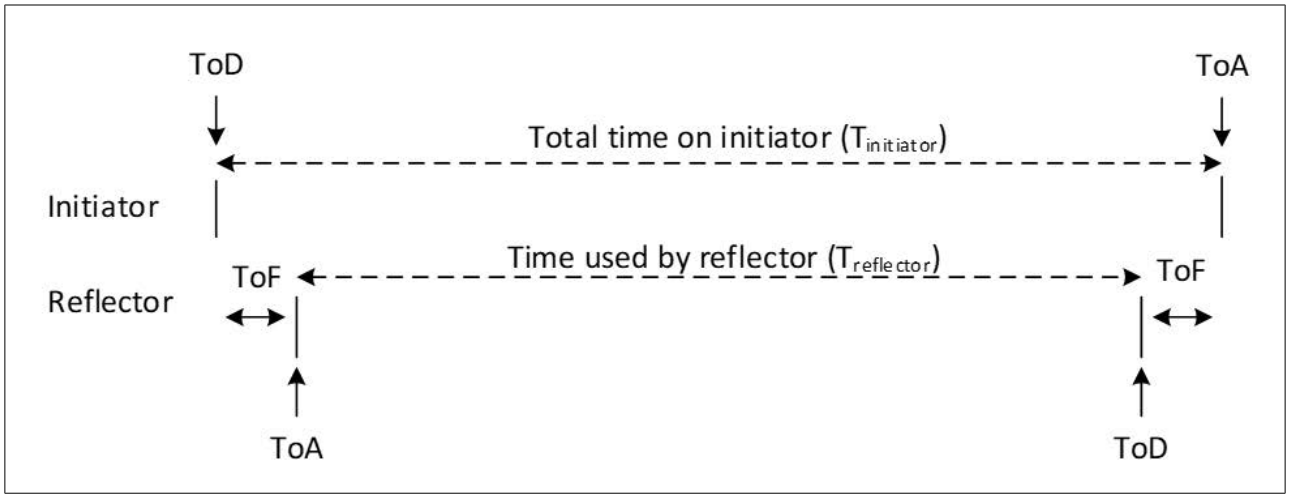



*Figure 9.2: Time of arrival (ToA), time of departure (ToD), and time of flight (ToF) for RTT estimation*

The accuracy of RTT ToF estimation depends on the capabilities of both devices. CS supports RTT based on timing extracted from a CS Access Address that consists of a 32-bit pseudo-noise bit sequence, and optionally based on timing extracted from a sounding sequence payload of up to 96 bits or a random sequence of up to 128 bits.

## 9.4 Security features

The CS feature includes security measures to reduce the probability of distance spoofing.

RTT and phase-based ranging can be employed in the same CS subevent or in the same CS step (CS step mode-3) to provide two partially independent estimates of distance.

CS step mode-1 and CS step mode-3 allow devices to potentially detect whether an attacker is present. With a random sequence, it is possible to measure how much a received GFSK modulated packet signal differs from the expected packet signal. With a sounding sequence, it is possible to detect the position of one or more marker signals.

A DRBG is employed by devices using the CS feature. The components used to generate the key material for this DRBG are exchanged between the device pair under an encrypted link and is never shared external to each device. This DRBG is used to cryptographically randomize the channel hop selection, the use of CS step modes, the modulation of tones, the transmission order in the case where multiple antenna paths are used, and the CS Access Address and payload content of CS_SYNC packets.

# ACRONYMS & ABBREVIATIONS

# CONTENTS

**1 List of acronyms and abbreviations ..... 354**

# 1 LIST OF ACRONYMS AND ABBREVIATIONS

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| 8DPSK | 8 phase Differential Phase Shift Keying | 3 Mb/s modulation type used by Enhanced Data rate |
| AAD | Additional Authenticated Data | |
| ACI | Antenna Configuration Index | |
| ACK | Acknowledge/Acknowledgment | |
| ACL | Asynchronous Connection-oriented [logical transport] | Reliable or time-bounded, bi-directional, point-to-point |
| ACL-C | ACL Control [logical link] (LMP) | |
| ACL-U | ACL User [logical link] (L2CAP) | |
| ACO | Authenticated Ciphering Offset | |
| AD | Advertising Data | |
| Adv_idx | Advertising channel index | |
| ADVB | LE Advertising Broadcast | |
| ADVB-C | LE Advertising Broadcast Control (Logical Link) | |
| ADVB-U | LE Advertising Broadcast User Data (Logical Link) | |
| ADI | AdvDataInfo | |
| AES | Advanced Encryption Standard | |
| AES-CCM | Advanced Encryption Standard - Counter with CBC-MAC | |
| AFH | Adaptive Frequency Hopping | |
| AHS | Adapted Hop Sequence | |
| AoA | Angle of Arrival | |
| AoD | Angle of Departure | |
| APB | Active Peripheral Broadcast [logical transport] | Unreliable, uni-directional broadcast to any devices synchronized with the physical channel |
| APB-C | APB Control [logical link] (LMP) | |
| APB-U | APB User [logical link] (L2CAP) | |
| ARQ | Automatic Repeat Request | |
| ASK | Amplitude Shift Keying | |
| ATT | Attribute Protocol | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| BB | Baseband | |
| BCH | Bose, Chaudhuri & Hocquenghem | Type of code<br>The persons who discovered these codes in 1959 (H) and 1960 (B&C) |
| BD_ADDR | Bluetooth Device Address | |
| BER | Bit Error Rate | |
| BIG | Broadcast Isochronous Group | A group of one or more time-related Broadcast Isochronous Streams |
| BIS | Broadcast Isochronous Stream | Connectionless isochronous logical transport |
| BT | Bandwidth Time | |
| C | Central | |
| C.# | Conditional | Any number may be used. See [Vol 1] Part E, Section 2.11 |
| CAC | Channel Access Code | |
| CBC-MAC | Cipher Block Chaining Message Authentication Code | |
| CCM | Counter with CBC-MAC | |
| CIG | Connected Isochronous Group | A group of one or more time-related Connected Isochronous Streams |
| CIS | Connected Isochronous Stream | Point-to-point isochronous logical transport |
| CLKN | Native Clock | |
| CLK | Central's Clock | |
| CLKE | Estimated Clock | |
| CODEC | COder DECoder | |
| COF | Ciphering Offset | |
| CP | CTEInfo Present | A field in the Data Channel PDU Header to indicate the presence of the CTEInfo field |
| CPB | Connectionless Peripheral Broadcast | The logical transport enabled by the Connectionless Peripheral Broadcast feature |
| CRC | Cyclic Redundancy Check | |
| CS | Channel Sounding | |
| CS Tone | Channel Sounding Tone | Unmodulated carrier associated with the phase-based ranging technique |
| CSA | Core Specification Addendum | (In plural Addenda) |
| CTE | Constant Tone Extension | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| CTEInfo | Constant Tone Extension Information | A field in the Data Channel PDU Header and the extended advertising header |
| CTS | Clear to send | |
| CVSD | Continuous Variable Slope Delta Modulation | |
| DAC | Device Access Code | |
| DCI | Default Check Initialization | |
| DEVM | Differential Error Vector Magnitude | Measure of modulation error used for Enhanced Data Rate transmitter testing |
| DFT | Discrete Fourier Transform | |
| DH | Data-High Rate | Data packet type for high rate data |
| DHK | Diversifier Hiding Key | |
| DIAC | Dedicated Inquiry Access Code | |
| DID | (Advertising) Data ID | |
| DIV | Diversifier | |
| DM | Data - Medium Rate | Data packet type for medium rate data |
| DPSK | Differential Phase Shift Keying | Generic description of Enhanced Data Rate modulation |
| DQPSK | Differential Quaternary Phase Shift Keying | Modulation type used by Enhanced Data Rate |
| DRBG | Deterministic Random Bit Generator | |
| DTM | Direct Test Mode | |
| DTX | Dirty Transmitter | RFPHY test packets with imperfections (in one or more of the following: center frequency offset, modulation index, symbol timing error, and frequency drift) introduced at the Lower Tester for device receiver testing. |
| DV | Data Voice | Data packet type for data and voice |
| E | Excluded | See [Vol 1] Part E, Section 2.11 |
| ECLD | Early Commit Late Detect | |
| EDIV | Encrypted Diversifier | |
| EDLC | Early Detect Late Commit | |
| EDR | Enhanced Data Rate | |
| EIR | Extended Inquiry Response | Host supplied information transmitted in the Inquiry Response substate |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| EIRP | Effective Isotropic Radiated Power | Equivalent power that an isotropic antenna must transmit to provide the same field power density |
| (e)SCO | Synchronous logical link or Extended Synchronous logical link | SCO or eSCO |
| eSCO | Extended Synchronous Connection-Oriented [logical transport] | Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission |
| eSCO-S | Stream eSCO (unframed) | Used to support isochronous data delivered in a stream without framing |
| ETSI | European Telecommunications Standards Institute | |
| FAE | Fractional Frequency Offset Actuation Error | |
| FCC | Federal Communications Commission | |
| FCS | Frame Check Sequence | |
| FDMA | Frequency Division Multiple Access | |
| FEC | Forward Error Correction code | |
| FFO | Fractional Frequency Offset | |
| FHS | Frequency Hop Synchronization | |
| FHSS | Frequency Hopping Spread Spectrum | |
| FIFO | First In First Out | |
| FIPS | Federal Information Processing Standards | |
| FM | Frequency Modulation | Modulation type |
| GAP | Generic Access profile | |
| GATT | Generic Attribute profile | |
| GFSK | Gaussian Frequency Shift Keying | |
| GIAC | General Inquiry Access Code | Used for GAP General Discoverable mode and General Inquiry procedure. See Assigned Numbers. |
| HCI | Host Controller interface | |
| HEC | Header-Error-Check | |
| HID | Human Interface Device | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| HV | High quality Voice | e.g. HV1 packet |
| HW | Hardware | |
| IAC | Inquiry Access Code | |
| IC | Industry Canada | |
| IEC | International Electrotechnical Commission | |
| IEEE | Institute of Electrical and Electronics Engineers | |
| IETF | Internet Engineering Task Force | |
| IFS | Inter Frame Space | |
| IP | Internet Protocol | |
| IPT | Inline PCT Transfer | |
| IPv4 | Internet Protocol version 4 | |
| IPv6 | Internet Protocol version 6 | |
| IQ | In-phase and Quadrature | |
| IrDA | Infra-red Data Association | |
| IRK | Identity Resolving Key | |
| ISM | Industrial, Scientific, Medical | |
| ISO | International Organization for Standardization | |
| ISO | Isochronous | |
| ISOAL | Isochronous Adaptation Layer | The layer that converts data units from the upper layer to data units in the Link Layer and vice versa |
| ITU | International Telecommunication Union | |
| IUT | Implementation Under Test | |
| IV | Initialization Vector | |
| IV_C | Initialization Vector (Central) | |
| IV_P | Initialization Vector (Peripheral) | |
| L2CAP | Logical Link Control and Adaptation protocol | |
| LAP | Lower Address Part | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| LC | Link Controller | Link Controller (or Baseband) part of the Bluetooth protocol stack. Low level Baseband protocol handler |
| LC | Link Control [logical link] | The control logical links LC and ACL-C are used at the link control level and link manager level, respectively. |
| LE | Low Energy | |
| LEB-C | Low Energy Broadcast Control | A logical link for control of Broadcast Isochronous Streams in a Broadcast Isochronous Group |
| LE-C | Low Energy Control (link) | |
| LE-F | Low Energy Framed | A logical link for transferring framed isochronous data packets using a Connected or Broadcast Isochronous Stream logical transport |
| LE-S | Low Energy Stream | A logical link for transferring unframed isochronous data packets using a Connected or Broadcast Isochronous Stream logical transport |
| LE-U | LE User [logical link] | |
| LFAE | Local Frequency Actuation Error | |
| LFSR | Linear Feedback Shift Register | |
| LIAC | Limited Inquiry Access Code | Used for GAP Limited Discoverable mode and Limited Inquiry procedure. See Assigned Numbers. |
| LL | Link Layer | |
| LLID | Logical Link Identifier | |
| LM | Link Manager | |
| LMP | Link Manager protocol | For LM peer to peer communication |
| LR | Loudness Rating | |
| LSB | Least Significant Bit | |
| LSO | Least Significant Octet | |
| LSTO | Link Supervision Timeout event | Controller can send LSTO event to Host |
| LT_ADDR | Logical Transport ADDRess | |
| LTK | Long Term Key | |
| M | Mandatory | See [Vol 1] Part E, Section 2.11 |
| MAC | Message Authentication Code | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| Mb/s | Megabits (millions of bits) per second | |
| MD | More Data | |
| MIC | Message Integrity Check | |
| MITM | Man-in-the-middle | |
| Msym/s | Megasymbols per second | |
| MSB | Most Significant Bit | |
| mSBC | modified Sub Band Codec | Hands-Free Profile v1.6 or later |
| MSC | Message sequence chart | |
| MSO | Most Significant Octet | |
| MTU | Maximum Transmission Unit | |
| MWS | Mobile Wireless Standards | For example LTE and WiMAX |
| N_AP | Number of Antenna Paths | |
| NADM | Normalized Attack Detector Metric | |
| NAK | Negative Acknowledge | |
| NAP | Non-significant Address Part | |
| NESN | Next Expected Sequence Number | |
| NIST | National Institute of Standards and Technology | |
| O | Optional | See [Vol 1] Part E, Section 2.11 |
| OBEX | OBject EXchange protocol | |
| OCF | Opcode Command Field | |
| OGF | Opcode Group Field | |
| OOB | Out of Band | |
| OTA | Over The Air | |
| π/4-DQPSK | π/4 Rotated Differential Quaternary Phase Shift Keying | 2 Mb/s modulation type used by Enhanced Data Rate |
| P | Peripheral | |
| PADVB | LE Periodic Advertising Broadcast (Logical Transport) | |
| PAwR | Periodic Advertising with Responses | |
| PBD | Profile Broadcast Data | The name of the logical link enabled by the Connectionless Peripheral Broadcast feature |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| PBF | Packet Boundary Flag | The device supports the capability to correctly handle HCI ACL Data packets |
| PBR | Phase-Based Ranging | |
| PCM | Pulse Coded Modulation | |
| PCT | Phase Correction Term | |
| PDU | Protocol Data Unit | A message |
| PHY | Physical Layer | |
| PIN | Personal Identification Number | |
| PN | Pseudo-random Noise | |
| ppm | Part Per Million | |
| PPP | Point-to-Point Protocol | |
| PRBS | Pseudo Random Bit Sequence | |
| PRNG | Pseudo Random Noise Generation | |
| PSK | Phase Shift Keying | Class of modulation types |
| ptt | Packet Type Table | The ptt parameter is used to select the logical transport types via LMP. |
| QoS | Quality of Service | |
| RAND | Random number | |
| RF | Radio Frequency | |
| RFC | Request For Comments | |
| RFCMode | Retransmission and Flow Control Mode | |
| RFCOMM | | Serial cable emulation protocol based on ETSI TS 07.10 |
| RFU | Reserved for future use | |
| RMS | Root Mean Square | |
| RPA | Resolvable Private Address | |
| RPL | Reference Power Level | |
| RSSI | Received Signal Strength Indication | |
| RTT | Round-Trip Time | |
| RX | Receive | |
| SAR | Segmentation and Reassembly | |
| SCA | Sleep Clock Accuracy | |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| SCO | Synchronous Connection-Oriented [logical transport] | Bi-directional, symmetric, point-to-point, AV channels |
| SCO-S | Stream SCO (unframed) | Used to support isochronous data delivered in a stream without framing |
| SDP | Service Discovery protocol | |
| SDU | Service Data Unit | |
| SEQN | Sequential Numbering scheme | |
| SID | (Advertising) Set ID | |
| SK | Session Key | |
| SKD_C | Session Key Diversifier (Central) | Central portion of the Session Key Diversifier |
| SKD_P | Session Key Diversifier (Peripheral) | Peripheral portion of the Session Key Diversifier |
| SM | Security Manager | |
| SMP | Security Manager Protocol | |
| SN | Sequence Number | |
| SNR | Signal-to-Noise Ratio | |
| SRES | Signed Response | |
| SSI | Signal Strength Indication | |
| SSP | Secure Simple Pairing | |
| STK | Short Term Key | |
| SW | Software | |
| T_FCS | Time for Frequency Change Spacing | |
| T_FM | Time for Frequency Measurement | |
| T_GD | Time for Guard period | |
| T_IFS | Time Inter Frame Space | Time interval between consecutive packets on same channel index in the situation indicated by the suffix. See [Vol 6] Part B, Section 4.1.1. |
| T_IP1 | Time for Interlude Period 1 (between CS packets) | |
| T_IP2 | Time for Interlude Period 2 (between CS tones) | |
| T_MCES | Time Minimum Connection Event Spacing | Minimum time interval between connection events. See [Vol 6] Part B, Section 4.1.5. |

| Acronym or abbreviation | Writing out in full | Comments |
|---|---|---|
| T_MSS | Time Minimum Subevent Spacing | Minimum time interval between subevents in the situation indicated by the suffix. See [Vol 6] Part B, Section 4.2.2. |
| T_PM | Time for Phase Measurement | |
| T_RD | Time for (transmission) Ramp-Down | |
| T_SY | Time for synchronization sequence (CS packet) | |
| TCP/IP | Transport Control Protocol/Internet Protocol | |
| TCS | Telephony Control protocol Specification | |
| TDD | Time-Division Duplex | |
| TDMA | Time Division Multiple Access | |
| TK | Temporary Key | |
| ToA | Time of Arrival | |
| ToD | Time of Departure | |
| ToF | Time of Flight | |
| TX | Transmit | |
| UAP | Upper Address Part | |
| UART | Universal Asynchronous receiver Transmitter | |
| UI | User Interface | |
| ULAP | Upper and Lower Address Parts | |
| USB | Universal Serial Bus | |
| UTF-8 | 8-bit UCS/Unicode Transformation Format | |
| UTP | Unified Test Protocol | Protocol for LE RFPHY test. See [Vol 6] Part F, Section 5. |
| UUID | Universal Unique Identifier | |

*Table 1.1: Acronyms and abbreviations*

Architecture, Change History,
And Conventions
Part C

# CORE SPECIFICATION CHANGE HISTORY

# CONTENTS

**1 Removed features ..... 368**

**2 Changes from v1.1 to v1.2 ..... 369**
2.1 New features ..... 369
2.2 Structure changes ..... 369
2.3 Deprecated features list ..... 370
2.4 Changes in wording ..... 370
2.5 Nomenclature changes ..... 370

**3 Changes from v1.2 to v2.0 + EDR ..... 371**
3.1 New features ..... 371
3.2 Deprecated features ..... 371

**4 Changes from v2.0 + EDR to v2.1 + EDR ..... 372**
4.1 New features ..... 372
4.2 Removed features ..... 372

**5 Changes from v2.1 + EDR to v3.0 + HS ..... 373**
5.1 New features ..... 373
5.2 Removed features ..... 373

**6 Changes from v3.0 + HS to v4.0 ..... 374**
6.1 New features ..... 374
6.2 Removed features ..... 374

**7 Changes from v4.0 to v4.1 ..... 375**
7.1 New features ..... 375
7.1.1 Features added in CSA 4 – integrated in v4.1 ..... 375
7.1.2 Features added in CSA 3 – integrated in v4.1 ..... 375
7.1.3 Features added in CSA 2 – integrated in v4.1 ..... 376
7.2 Removed features ..... 376

**8 Changes from v4.1 to v4.2 ..... 377**
8.1 New features ..... 377
8.2 Errata incorporated in v4.2 ..... 377

**9 Changes from v4.2 to v5.0 ..... 379**
9.1 New features ..... 379
9.1.1 Features added in CSA5 - integrated in v5.0 ..... 379
9.2 Removed features ..... 379
9.3 Privacy errata ..... 379

9.4 Errata incorporated in v5.0 ..... 380

**10 Changes from v5.0 to v5.1 ..... 384**
10.1 New features ..... 384
10.1.1 Features added in CSA6 – integrated in v5.1 ..... 384
10.2 Removed features ..... 384
10.3 Security erratum ..... 384
10.4 Errata incorporated in v5.1 ..... 384

**11 Changes from v5.1 to v5.2 ..... 387**
11.1 New features ..... 387
11.2 Security erratum ..... 387
11.3 Errata incorporated in v5.2 ..... 387

**12 Changes from v5.2 to v5.3 ..... 389**
12.1 New features ..... 389
12.2 Removed Features ..... 389
12.3 Errata incorporated in v5.3 ..... 389
12.4 Global terminology changes ..... 392

**13 Changes from v5.3 to v5.4 ..... 394**
13.1 New features ..... 394
13.2 Removed features ..... 394
13.3 Errata incorporated in v5.4 ..... 394

**14 Changes from v5.4 to v6.0 ..... 397**
14.1 New features ..... 397
14.2 Removed features ..... 397
14.3 Errata incorporated in v6.0 ..... 397

**15 Changes from v6.0 to v6.1 ..... 401**
15.1 New features ..... 401
15.2 Removed features ..... 401
15.3 Errata incorporated in v6.1 ..... 401

**16 Changes from v6.1 to v6.2 ..... 403**
16.1 New features ..... 403
16.2 Removed features ..... 403
16.3 Security errata ..... 403
16.4 Errata incorporated in v6.2 ..... 404

**17 Changes from v6.2 to v6.3 ..... 406**
17.1 New features ..... 406
17.2 Removed features ..... 406

17.3 Errata incorporated in v6.3 ..... 406

# 1 REMOVED FEATURES

Some features have been deemed no longer useful and have been removed. The term removed does not mean that they are no longer allowed, but that they are no longer recommended as the best way of performing a given function.

Removed features may be implemented using the latest qualifiable version (if any) of the specification that specifies the removed features.

# 2 CHANGES FROM V1.1 TO V1.2

## 2.1 New features

Several new features were introduced in the version 1.2. The major areas of improvement are:

- Architectural overview
- Faster connection
- Adaptive frequency hopping
- Extended SCO links
- Enhanced error detection and flow control
- Enhanced synchronization capability
- Enhanced flow specification

The feature descriptions are incorporated into the existing text in different core parts, described in Volumes 2 and 3.

## 2.2 Structure changes

Version 1.2 was significantly restructured for better consistency and readability. The most important structure changes have been performed in Baseband, LMP, HCI and L2CAP. The text in these sections has been rearranged to provide:

- Presentation of the information in a more logical progression
- Removal of redundant text and requirements
- Consolidation of Baseband-related requirements (for example, the Baseband Timers and Bluetooth Audio sections into the Baseband Specification)
- Alignment of the specification with the new architecture and terminology presented in the Architecture Overview (see [Vol 1] Part E).

In addition, new text and requirements were added for the new features as well as many changes throughout the specification to update the text to use IEEE language (see [Vol 1] Part E).

## 2.3 Deprecated features list

Features deprecated in version 1.2 are:

- The use of Unit Keys for security
- Optional Paging schemes
- 23 channel hopping sequence
- Page scan period mode and associated commands

## 2.4 Changes in wording

Two general classes of changes to the wording of the Bluetooth Specification have been done for version 1.2. They are a formalization of the language by using conventions established by the Institute of Electrical and Electronic Engineers (IEEE), and a regularization of Bluetooth wireless technology-specific terms. Many portions of the version 1.1 specification use imprecise or inaccurate terms to describe attributes of the protocol. A more accurate terminology described in Part E was introduced into the version 1.2 specification and have been applied in future versions.

## 2.5 Nomenclature changes

The nomenclature in Bluetooth 1.2 was also updated due to new concepts that are introduced together with the new features and the new architecture. See [Vol 1] Part A.

# 3 CHANGES FROM V1.2 TO V2.0 + EDR

## 3.1 New features

Version 2.0 + EDR introduces Enhanced Data Rate (EDR). EDR provides a set of additional packet types that use the new 2 Mb/s and 3 Mb/s modes.

In addition to EDR a set of errata provided in ESR02 has been incorporated into this version and revised to include changes caused by the addition of EDR.

These additions are incorporated into the existing text in different core parts described in Volumes 2 and 3.

## 3.2 Deprecated features

The only feature deprecated in version 2.0 + EDR is the Page Scan Period Mode and associated commands (based on Erratum 694 which is also included in ESR02).

# 4 CHANGES FROM V2.0 + EDR TO V2.1 + EDR

## 4.1 New features

Several new features are introduced in version 2.1 + EDR. The major areas of improvement are:

- Erroneous Data Reporting
- Encryption Pause and Resume
- Extended Inquiry Response
- Link Supervision Timeout Changed event
- Non-Automatically-Flushable Packet Boundary Flag
- Secure Simple Pairing
- Sniff Subrating
- Security Mode 4

## 4.2 Removed features

No features were removed in v2.1 + EDR.

# 5 CHANGES FROM V2.1 + EDR TO V3.0 + HS

## 5.1 New features

Several new features are introduced in version 3.0 + HS. The major areas of improvement are:

- AMP Manager protocol (A2MP)
- Enhancements to L2CAP including
  - Enhanced Retransmission Mode and Streaming Mode
  - Improvements to the L2CAP state machine for AMP channels
  - Fixed channel support
- Enhancements to HCI for AMP
- Enhancements to Security for AMP
- 802.11 Protocol Adaptation Layer
- Enhanced Power Control
- Unicast Connectionless Data
- HCI Read Encryption Key Size command
- Generic Test Methodology for AMP
- Enhanced USB and SDIO HCI Transports

## 5.2 Removed features

No features were removed in v3.0 + HS.

# 6 CHANGES FROM V3.0 + HS TO V4.0

## 6.1 New features

Several new features are introduced in version 4.0. The major areas of improvement are:

- Bluetooth Low Energy including
  - Low Energy Physical Layer
  - Low Energy Link Layer
  - Enhancements to HCI for Low Energy
  - Low Energy Direct Test Mode
  - AES Encryption
  - Enhancements to L2CAP for Low Energy
  - Enhancements to GAP for Low Energy
  - Attribute Protocol (ATT)
  - Generic Attribute profile (GATT)
  - Security Manager (SM)

## 6.2 Removed features

No features were removed in v4.0.

# 7 CHANGES FROM V4.0 TO V4.1

## 7.1 New features

Several new features are introduced in the version 4.1. The major areas of improvement are:

- BR/EDR Secure Connections
- Train nudging
- Generalized interlaced scan
- Low duty cycle directed advertising
- 32-bit UUID support in LE
- LE dual mode topology
- Piconet clock adjustment
- LE L2CAP connection-oriented channel support
- LE privacy v1.1
- LE Link Layer topology
- LE ping

### 7.1.1 Features added in CSA 4 – integrated in v4.1

- Connectionless Peripheral Broadcast
- Unencrypted unicast connectionless data support
- Fast advertising interval
- eSCO reserved slots clarification

### 7.1.2 Features added in CSA 3 – integrated in v4.1

- MWS coexistence HCI changes
- GAP connection parameters changes
- GAP authentication and lost bond
- Dual mode addressing
- Private address changes
- Common profile and service error code changes
- MWS coexistence signaling

- Wireless coexistence interface 1 (WCI-1) transport specification
- Wireless coexistence interface 2 (WCI-2) transport specification

### 7.1.3 Features added in CSA 2 – integrated in v4.1

- Limited discovery time changes
- EIR and AD data types in GAP
- Audio architecture HCI changes
- Audio architecture USB changes
- 802.11n enhancements to the 802.11 PAL

## 7.2 Removed features

No features were removed in v4.1.

# 8 CHANGES FROM V4.1 TO V4.2

## 8.1 New features

Several new features are introduced in version 4.2. The major areas of improvement are:

- LE Data Packet Length Extension
- LE Secure Connections
- Link Layer privacy
- Link Layer Extended Scanner Filter policies

## 8.2 Errata incorporated in v4.2

Table 8.1 lists the integrated ESR08 errata items and also some ESR09 items, without test impact, which were added to v4.2 prior to the ESR09 release.

| | ESR08 | ESR09 |
|---|---|---|
| **Global / Several Parts** | | 5782, 6001 |
| **V0B: Compliance Req.** | 5748 | |
| **V1A: Architecture Overview** | 4767, 4768, 4858, 5526, 5531, 5742, 5748 | 5965, 6086 |
| **V2A: Radio** | 5179 | |
| **V2B: Baseband** | 2645, 2668, 3909, 4997, 5185, 5447, 5636 | 5829, 5875, 5966 |
| **V2C: LMP** | 4934, 5118. 5756 | 5754, 5802, 5815, 5816, 5817, 5818, 5819 |
| **V2D: Error Codes** | 5645, 5660 | 5814 |
| **V2E: HCI** | 4998, 5097, 5186, 5286, 5340, 5358, 5397, 5416, 5589, 5630, 5637, 5660, 5736, 5737, 5738 | 5000, 5174, 5247, 5821, 5822, 5826, 5828, 5859, 5907, 5926, 5927, 5935, 5959, 5960 |
| **V2F: MSCs** | 5621 | 5762 |
| **V2G: Sample Data** | | |
| **V2H: Security Spec.** | 4815, 4830, 4903 | |
| **V3A: L2CAP** | 2765, 3253, 3767, 3768, 3901, 3902, 3903, 4603, 4734, 4735, 4776, 5419 | 5987 |
| **V3B: SDP** | 5258 | 5257 |

| | ESR08 | ESR09 |
|---|---|---|
| **V3C: GAP** | 2341, 4087, 4778, 4895, 4935, 4936, 5618, 5441, 5051 | 5803, 5938, 5939, 5964, 5989 |
| **V3F: ATT** | | 6006 |
| **V3G: GATT** | 3817, 4358, 5116, 5117 | |
| **V3H: Security Manager** | | 6050 |
| **V4B: USB Transport Layer** | 5273 | |
| **V6A: Physical Layer** | 5179 | |
| **V6B: Link Layer** | 5231, 5232, 5233, 5234, 5235, 5236, 5307, 5360, 5419, 5428, 5526, 5577, 5653, 5656, 5706, 5707, 5720, 5721, 5722, 5723 | 5824, 5922, 5947, 5950, 5954, 5994, 6116 |
| **V6C: Sample Data** | 4948 | |
| **V6D: MSCs** | 4797, 4798 | 6032 |
| **V6F: Direct Test Mode** | 4834, 5749, 5751 | |
| **V7A: MWS Coexistence Logical Signaling** | 5001 | |
| **V7B: WCI-1 Transport** | 5004, 5619 | |

*Table 8.1: Errata integrated in v4.2*

# 9 CHANGES FROM V4.2 TO V5.0

## 9.1 New features

Several new features are introduced in version 5.0. The major areas of improvement are:

- Slot Availability Mask (SAM)
- 2 Msym/s PHY for LE
- LE Long Range
- High Duty Cycle Non-Connectable Advertising
- LE Advertising Extensions
- LE Channel Selection Algorithm #2

### 9.1.1 Features added in CSA5 - integrated in v5.0

- Higher Output Power

## 9.2 Removed features

The following features were removed in this version of the specification:

- Park State

## 9.3 Privacy errata

The Privacy errata shown in Table 9.1 have been resolved and integrated in this version of the specification.

| Erratum | Title |
|---|---|
| 5433 | In Table 9.3, C1 condition should be removed |
| 6182 | Invalid behavior with resolving address list not clear in Core or Test Spec |
| 6214 | Statement mismatch between procedure description and figure |
| 6247 | Unclear when to use LE Directed Advertising Report event or LE Advertising Report event |
| 6356 | Defining scanner’s device address in “Scanner filter policy” |
| 6391 | Usage of peer address type not clear in privacy 1.2 |
| 6399 | LE Enhanced Connection Complete event |
| 6401 | Central Address Resolution Char in LE Peripheral |

| Erratum | Title |
|---|---|
| 6469 | Advertiser is not required to check AdvA |
| 6471 | AdvA not well defined in SCAN_RSP pdu, when privacy 1.2 enabled |
| 6508 | HCI command name mismatch |
| 6519 | When an advertiser device (or scanning performing active scanning) using privacy and Controller based resolvable private address generation is supposed to answer to a scan request (or advertising) received? |
| 6613 | Privacy Mode not clearly defined |
| 6629 | LE Directed Advertising Report event - Order of parameters |
| 6984 | How does Adv respond to RPA in Connect Req when there's no IRK and no WL |

*Table 9.1: Privacy errata*

## 9.4 Errata incorporated in v5.0

Table 9.2 lists the integrated ESR09 and ESR10 errata items.

| | ESR09 | ESR10 |
|---|---|---|
| **Global / Several Parts** | 5782, 6001, 6334 | 6570, 6751, 6855. 7076 |
| **V0B: Bluetooth Compliance Requirements** | | 6360, 6717 |
| **V1A: Architecture Overview** | 5900, 5965, 6086, 6120, 6193, 6240, 6242, 6277, 6284, 6288, 6290, 6303, 6361, 6363, 6364 | 6356, 6421, 6465, 6524, 6529, 6558, 6603, 6604, 6766, 6922, 7030, 7031, 7301 |
| **V1B: Acronyms & Abbreviations** | 6277, 6349 | 6717 |
| **V1C: Change History** | 6195, 6318 | |
| **V1E: IEEE Language** | 6208, 6381 | 6421, 6422, 7521 |
| **V2A: Radio** | | 6199, 6547 |
| **V2B: Baseband** | 5829, 5875, 5966, 6096, 6193, 6277, 6350 | 6421, 6443, 6446, 6480, 6505, 6523, 6531, 6537, 6618, 6646, 6647, 6668, 6714, 6721, 6749, 6953, 6964 |
| **V2C: LMP** | 5753, 5754, 5802, 5815, 5816, 5817, 5818, 5819, 6114, 6175, 6176, 6177, 6193, 6210, 6277, 6289, 6291, 6295, 6332, 6339, 6388, 6389, 6404 | 4783, 6447, 6531, 6606, 6735, 6773, 6776, 6852 |
| **V2D: Error Codes** | 5112, 5511, 5814, 6305, 6309 | 6421 |

| | ESR09 | ESR10 |
|---|---|---|
| **V2E: HCI** | 4354, 5000, 5174, 5247, 5266, 5821, 5822, 5826, 5828, 5859, 5905, 5907, 5926, 5927, 5935, 5959, 5960, 6072, 6110, 6115, 6211, 6265, 6277, 6278, 6286, 6293, 6301, 6302, 6326, 6338, 6339, 6374, 6409, 6419, 6420 | 4761, 4999, 5003, 6007, 6182, 6247, 6356, 6399, 6421, 6451, 6452, 6493, 6508, 6531, 6532, 6534, 6536, 6541, 6556, 6559, 6563, 6566, 6568, 6576, 6629, 6672, 6689, 6709, 6716, 6722, 6749, 6752, 6767, 6797, 6807, 6856, 6857, 6882, 6886, 6910, 6914, 6918, 6927, 6939, 6986, 6997, 6998, 7017, 7021, 7024, 7045, 7067, 7068, 7069, 7077, 7372 |
| **V2F: MSCs** | 5762, 5875, 6212, 6286, 6319 | 5089, 6218, 6428, 6531 |
| **V2H: Security Specification** | 6194, 6359 | 5791, 5792, 5882, 6421, 6902, 6906, 6907 |
| **V3A: L2CAP** | 5459, 5465, 5466, 5467, 5648, 5666, 5805, 5987, 6108, 6121, 6277, 6307, 6333 | 3548, 6417, 6421, 6567, 6702, 6898, 6939 |
| **V3B: SDP** | 5257, 6308 | 6421, 6694 |
| **V3C: GAP** | 4958, 5562, 5786, 5803, 5821, 5938, 5939, 5964, 5989, 6166, 6167, 6168, 6215, 6217, 6222, 6313, 6405, 6411 | 4348, 4777, 4872, 4919, 5110, 5399, 5407, 5418, 5433, 5435, 5708, 5778, 6214, 6356, 6401, 6421, 6439, 6502, 6510, 6574, 6575, 6599, 6732, 6745, 6764, 6897, 7032, 7033, 7354, 7355 |
| **V3D:Test Support** | | 6421, 6525, 7075 |
| **V3F: ATT** | 5953, 6006, 6132, 6357 | 4243, 5107, 5610, 5919, 6098, 6575 |
| **V3G: GATT** | 4253, 4836, 5059, 6251 | 4654, 5242, 5359, 5383, 5717, 5718, 5719, 6575, 6582, 7028, 7065, 7066, 7072 |
| **V3H: Security Manager** | 4793, 4794, 5427, 5625, 6050, 6207, 6405 | 4796, 6219, 6421, 6436, 6729, 6796, 6824, 6946, 7461 |
| **V4A: UART Transport** | | 6264 |
| **V4B: USB Transport Layer** | 5274 | 4165 |
| **V5A: 802.11 Protocol Adaptation** | | 6421, 7036 |
| **V6A: Physical Layer** | 6118, 6156 | 6228 |

| | ESR09 | ESR10 |
|---|---|---|
| **V6B: Link Layer** | 5174, 5534, 5745, 5824, 5922, 5947, 5950, 5954, 5986, 5994, 6116, 6119, 6123, 6131, 6153, 6164, 6167, 6168, 6188, 6192, 6197, 6268, 6294, 6306, 6310, 6311, 6321, 6331, 6339, 6345, 6371, 6384 | 6182, 6352, 6356, 6421, 6455, 6469, 6471, 6517, 6518, 6531, 6553, 6577, 6586, 6664, 6672, 6697, 6710, 6733, 6734, 6821, 6847, 6913, 6939, 6947, 6962, 6963, 6984, 7003, 7025, 7034, 7080, 7081, 7106, 7153 |
| **V6C: Sample Data** | | 5809 |
| **V6D: MSCs** | 6032, 6269 | 6563, 6672, 6847 |
| **V6E: LE Link Layer Security** | 6194 | |
| **V6F: Direct Test Mode** | 6390 | 6619, 6793, 6798, 7005, 7029 |
| **V7A: MWS Coexistence Logical Signaling** | 6234 | 5002 |
| **V7B: Wireless Coexistence Interface 1** | | 6915 |
| **V7C: Wireless Coexistence Interface 2** | | 6915 |

*Table 9.2: Errata integrated in v5.0*

Table 9.3 lists the integrated ESR11 errata items.

| | ESR11 | |
|---|---|---|
| **Global / Several Parts** | 7238, 7239, 7340 | |
| **V0C: Revision History and Acknowledgments** | 7895 | |
| **V1A: Architecture Overview** | 7421, 7422, 7423, 7484, 7772, 7777, 7980 | |
| **V1B: Acronyms & Abbreviations** | 7777, 7895 | |
| **V1C: Change History** | 7777 | |
| **V1D: Mixing of Spec Versions** | 7351, 7358 | |
| **V1E: IEEE Language** | | |
| **V2A: Radio** | 7278, 7526, 7662, 7777 | |
| **V2B: Baseband** | 7304, 7305, 7320, 7651, 7777 | |
| **V2C: LMP** | 7145, 7284, 7285, 7292, 7320, 7367, 7713 | |
| **V2D: Error Codes** | 7293, 8039 | |

| | ESR11 | |
|---|---|---|
| **V2E: HCI** | 7143, 7158, 7185, 7187, 7217, 7218, 7234, 7262, 7274, 7326, 7334, 7375, 7396, 7458, 7485, 7504, 7644, 7650, 7706, 7716, 7717, 7821, 7835, 7851, 7867, 7871, 7900, 7991, 8090 | |
| **V2F: MSCs** | 7509 | |
| **V3A: L2CAP** | 7115, 7116, 7527, 7869 | |
| **V3B: SDP** | 7819 | |
| **V3C: GAP** | 7873, 8024 | |
| **V3D:Test Support** | 7878 | |
| **V3F: ATT** | 8041 | |
| **V3G: GATT** | 7159, 7160, 7737, 7891, 7896, 7897, 8051, 8055 | |
| **V3H: Security Manager** | 7226, 7320, 7469 | |
| **V5A: 802.11 Protocol Adaptation** | 7875, 7876, 7878 | |
| **V6A: Physical Layer** | 7408, 7777 | |
| **V6B: Link Layer** | 7122, 7152, 7195, 7209, 7269, 7320, 7321, 7330, 7331, 7481, 7482, 7630, 7686, 7720, 7777, 7791, 7809, 7995, 8031 | |
| **V6D: MSCs** | 7509 | |
| **V6F: Direct Test Mode** | 7156, 7281, 7696 | |
| **V7A: MWS Coexistence Logical Signaling** | 7525 | |

*Table 9.3: Errata integrated in v5.0*

# 10 CHANGES FROM V5.0 TO V5.1

## 10.1 New features

Several new features are introduced in v5.1. The major areas of improvement are:

- Angle of Arrival (AoA) and Angle of Departure (AoD)
- Advertising Channel Index
- GATT Caching
- Minor Enhancements batch 1
  - HCI support for debug keys in LE Secure Connections
  - Sleep clock accuracy update mechanism
  - ADI field in scan response data
  - Interaction between QoS and Flow Specification
  - Host channel classification for secondary advertising
  - Allow the SID to appear in scan response reports
  - Specify the behavior when rules are violated
- Periodic Advertising Sync Transfer

### 10.1.1 Features added in CSA6 – integrated in v5.1

- Models
- Mesh-based model hierarchy

## 10.2 Removed features

The following features were removed in this version of the specification:

- Unit keys

## 10.3 Security erratum

Erratum 10734, which introduces new security requirements relating to pairing, has been resolved and integrated in this version of the specification.

## 10.4 Errata incorporated in v5.1

Table 10.1 lists the integrated ESR errata items.

| | ESR11 | ESR12 |
|---|---|---|
| **Global / Several Parts** | | 8407, 9085, 9088, 10278 |
| **V0C: Appendix** | 9065 | |
| **V1A: Architecture Overview** | 8132, 8211, 8218, 8347, 8632, 8804, 8808 | 9413, 9495, 9619, 9656, 9727, 9790, 10049, 10241, 10245, 10349 |
| **V1B: Acronyms & Abbreviations** | | 10193 |
| **V1C: Change History** | 9065 | |
| **V1D: Mixing of Spec Versions** | 9049 | |
| **V1E: General Terminology** | 8129 | 9222, 9223, 9499, 10295 |
| **V2A: Radio** | | 9477 |
| **V2B: Baseband** | 8153, 8184, 8185, 8187, 8188, 8211, 8218, 8531, 8552, 8555, 8562 | 9477, 9479, 9584, 9854, 9998, 10150, 10191, 10242, 10246, 10291 |
| **V2C: LMP** | 8180, 8185, 8221, 8224, 8355, 8364, 8366, 8367, 8368, 8406, 8533, 8554, 8702, 8911 | 8407, 8423, 9477, 9479, 9649, 9765, 9793, 10051, 10147, 10197, 10224, 10392, 10645 |
| **V2D: Error Codes** | 8803 | |
| **V2E: HCI** | 8106, 8117, 8134, 8149, 8170, 8172, 8176, 8181, 8202, 8236, 8262, 8267, 8329, 8352, 8387, 8522, 8531, 8556, 8557, 8559, 8560, 8561, 8562, 8596, 8597, 8621, 8622, 8647, 8649, 8650, 8651, 8653, 8654, 8658, 8660, 8661, 8689, 8752, 8848, 8858, 8859, 8868, 8909, 8914, 8920, 8932, 8934, 8953, 8970, 9044, 9106 | 8407, 9085, 9159, 9160, 9165, 9220, 9297, 9303, 9323, 9386, 9396, 9398, 9413, 9432, 9477, 9479, 9548, 9560, 9561, 9563, 9564, 9603, 9615, 9657, 9658, 9661, 9663, 9703, 9709, 9778, 9794, 9865, 9866, 9902, 9970, 10067, 10095, 10120, 10150, 10338, 10348, 10380, 10383, 10433, 10447, 10528, 10535, 10567, 10689, 10827 |
| **V2F: MSCs** | 8325 | |
| **V2H: Security** | 8215 | 9479, 9653, 10141, 10224, 10313, 10594, 10673 |
| **V3A: L2CAP** | 8419, 8562 | 9477, 9479, 10052,10154, 10170, 10216, 10847 |
| **V3B: SDP** | 8562 | 9317, 10171, 10773 |
| **V3C: GAP** | 8229, 8230, 8232, 8354, 9098, 9107 | 9370, 9371, 9391, 9592, 9692, 10183, 10399, 10401, 10412, 10418, 10691 |
| **V3D:Test Support** | 8135, 8562 | 9477, 10147, 10410 |
| **V3E: A2MP** | 8562 | |

| | ESR11 | ESR12 |
|---|---|---|
| **V3F: ATT** | 8208, 8412 | 9479 |
| **V3G: GATT** | 8562, 8563 | 9477, 9495, 9612, 9825, 9836, 9837, 10207 |
| **V3H: Security Manager** | 8562, 9074 | 9391, 9592, 9653, 9867, 10141 |
| **V4D: Three-wire UART** | 8562 | |
| **V5A: 802.11 Protocol Adaptation** | 8562 | 9479, 10446 |
| **V6A: Physical Layer** | | |
| **V6B: Link Layer** | 8001, 8097, 8180, 8203, 8204, 8228, 8246, 8249, 8250, 8278, 8300, 8315, 8324, 8365, 8387, 8424, 8425, 8431, 8432, 8433, 8447, 8564, 8595, 8598, 8699, 8745, 8773, 8801, 8803, 8812, 8854, 9056, 9060 | 9162, 9205, 9277, 9328, 9382, 9391, 9411, 9433, 9479, 9484, 9524, 9556, 9603, 9676, 9679, 9754, 9806, 9820, 9870, 9958, 9967, 9972, 9984, 10001, 10002, 10013, 10073, 10103, 10139, 10205, 10260, 10266, 10336, 10387, 10390, 10402, 10660, 10692, 10693, 10701, 10828, 11179 |
| **V6C: Sample Data** | 8179, 8235 | |
| **V6D: MSCs** | 8299, 8313, 8314, 8316, 8322, 8624, 8917, 9106 | 9432, 9603, 10631 |
| **V6F: Direct Test Mode** | 8619, 8543, 9011 | 9799 |
| **V7A: MWS Coexistence Logical Signaling** | | |

*Table 10.1: Errata integrated in v5.1*

# 11 CHANGES FROM V5.1 TO V5.2

## 11.1 New features

Several new features are introduced in v5.2. The major areas of improvement are:

- LE Isochronous Channels
- Enhanced Attribute Protocol
- LE Power Control

## 11.2 Security erratum

Erratum 11838, which introduces new security requirements and recommendations relating to encryption key size on BR/EDR, has been resolved and integrated in this version of the specification.

## 11.3 Errata incorporated in v5.2

Table 11.1 lists the integrated errata items.

| | **v5.2** |
|---|---|
| **Global / Several Parts** | |
| **V0A: Consolidated Table of Contents** | 11423 |
| **V0C: Revision History and Acknowledgments** | |
| **V1A: Architecture Overview** | 10961, 11422, 11529, 12232 |
| **V1B: Acronyms & Abbreviations** | |
| **V1C: Change History** | |
| **V1D: Mixing of Spec Versions** | 11870 |
| **V1E: General Terminology** | 8544, 10987 |
| **V1F: Controller Error Codes** | |
| **V2A: Radio** | |
| **V2B: Baseband** | 11234, 11422, 11499, 11911, 11973, 11974, 12035, 12057, 12445 |
| **V2C: LMP** | 8544, 10939, 10993, 11505, 11838, 11921, 12057 |
| **V2F: MSCs** | 10972, 11070 |
| **V2H: Security** | 11909, 11910, 11911 |

| | **v5.2** |
|---|---|
| **V3A: L2CAP** | 11505, 11506, 11673, 11679, 12434, 12275 |
| **V3B: SDP** | 11505 |
| **V3C: GAP** | 6375, 10940, 11118, 11419, 11838, 12275, 12377 |
| **V3D:Test Support** | |
| **V3E: A2MP** | |
| **V3F: ATT** | 11112, 12105 |
| **V3G: GATT** | 10925, 10926, 10937, 11187, 12210, 12275, 12070 |
| **V3H: Security Manager** | 11294 |
| **V4D: Three-wire UART** | |
| **V4E: HCI** | 8544, 10825, 10861, 10962, 10972, 10985, 10987, 11044, 11117, 11124, 11205, 11226, 11280, 11346, 11347, 11348, 11356, 11391, 11471, 11505, 11589, 11590, 11591, 11602, 11615, 11616, 11636, 11763, 11768, 11769, 11838, 11933, 12152, 12245, 12338, 12420, 12434, 12450, 12458, 12493, 12481, 12057, 12640 |
| **V5A: 802.11 Protocol Adaptation** | |
| **V6A: Physical Layer** | 10958 |
| **V6B: Link Layer** | 10909, 10951, 10957, 11021, 11049, 11066, 11084, 11308, 11355, 11357, 11386, 11443, 11482, 11485, 11498, 11499, 11505, 11602, 11640, 11701, 11828, 12232, 12236, 12263, 13012 |
| **V6C: Sample Data** | 11330, 11357 |
| **V6D: MSCs** | 12109 |
| **V6E: Low Energy Link Layer** | 11551, 11649, 11683 |
| **V6F: Direct Test Mode** | 10984, 11731 |
| **V7A: MWS Coexistence Logical Signaling** | 10985 |

*Table 11.1: Errata integrated in v5.2*

# 12 CHANGES FROM V5.2 TO V5.3

## 12.1 New features

Several new features are introduced in v5.3. The major areas of improvement are:

- AdvDataInfo in Periodic Advertising
- Host to Controller Encryption Key Control Enhancements
- LE Enhanced Connection Update
- LE Channel Classification

## 12.2 Removed Features

The following features were removed in this version of the specification:

- Alternative MAC/PHY
- AMP Manager protocol (A2MP)
- L2CAP Enhancements for AMP
- 802.11 PAL
- 802.11n Enhancements to the 802.11 PAL

## 12.3 Errata incorporated in v5.3

Table 12.1 lists the integrated errata items.

| | **v5.3** |
|---|---|
| **Global / Several Parts** | 11426, 11695, 12596, 14641, 15243, 15634 |
| **Front matter** | 15851, 16794 |
| **V0A: Consolidated Table of Contents** | 12495 |
| **V0B: Bluetooth Compliance Requirements** | 12495, 13250, 15754, 15874, 16979 |
| **V0C: Revision History and Acknowledgments** | 12495, 15631, 15632, 16032, 16312, 16605, 17067 |
| **V1A: Architecture Overview** | 10421, 11289, 11421, 11424, 11733, 11925, 11927, 12348, 12495, 14689, 14866, 15301, 15315, 15379, 15461, 15525, 15526, 15527, 15533, 15583, 16013, 16113, 16209, 16506, 16614, 16671, 16794, 16817, 16825, 16833, 16980, 17139 |

| | **v5.3** |
|---|---|
| **V1B: Acronyms & Abbreviations** | 10860, 11747, 12495, 15189 |
| **V1C: Change History** | 10860, 12495, 14930, 15368, 16032, 16163 |
| **V1D: Mixing of Spec Versions** | 12495, 14945, 15055, 15850, 16065 |
| **V1E: General Terminology** | 10306, 10419, 11702, 12495, 13190, 13500, 14670, 14705, 14773, 14972, 15368, 15461, 15583, 15851, 16163, 16635, 16671, 16672, 16794, 16843, 16906 |
| **V1F: Controller Error Codes** | 11747, 12495, 12664, 12720, 13346 |
| **V2A: Radio** | 11289, 11424, 12654, 13254, 15065, 15201, 15306, 16404, 16981 |
| **V2B: Baseband** | 10860, 11289, 11421, 11751, 12348, 13150, 13180, 13528, 14732, 14866, 14972, 15131, 15282, 15306, 15334, 15461, 15583, 15607, 15633, 15642, 15670, 15833, 15851, 16209, 16380, 16673, 16794, 16817 |
| **V2C: LMP** | 10419, 10860, 11421, 11424, 11747, 12093, 12589, 12908, 13147, 13167, 13180, 13195, 13417, 13473, 14646, 14689, 14790, 15039, 15115, 15117, 15162, 15176, 15220, 15273, 15334, 15388, 15467, 15541, 15583, 15639, 15834, 15851, 16500, 16566, 16794 |
| **V2F: MSCs** | 10860, 11289, 11895, 12187, 15197, 15464, 15583, 15630, 16191, 16572, 16646, 16817 |
| **V2G: Sample Data** | 10860, 15547, 15851, 16981, 17025 |
| **V2H: Security** | 10860, 11424, 12068, 12348, 12880, 13169, 13265, 13450, 15198, 15215, 15280, 15306, 15307, 15308, 15389, 15548, 15549, 15583, 15642, 15668, 15671, 15734, 15851, 16981, 16986 |
| **V3A: L2CAP** | 10860, 11289, 11421, 11895, 12713, 13147, 13151, 13185, 13235, 14605, 14749, 15254, 15256, 15306, 15313, 15323, 15461, 15554, 15578, 15583, 15630, 15833, 15850, 15851, 15944, 16065, 16116, 16187, 16188, 16191, 16266, 16493, 16518, 16549, 16550, 16794, 16817, 16982, 17046 |
| **V3B: SDP** | 10419, 11421, 11751, 12800, 13472, 15274, 16635, 16982, 16989 |
| **V3C: GAP** | 10419, 10860, 11289, 11421, 11787, 12322, 12348, 12379, 12725, 12973, 13147, 13152, 13162, 13182, 13184, 13186, 13187, 13188, 13335, 13336, 13406, 13407, 13425, 13603, 14797, 14896, 15006, 15039, 15255, 15302, 15314, 15334, 15384, 15385, 15400, 15403, 15406, 15408, 15497, 15546, 15583, 15630, 15831, 15850, 15851, 15871, 15944, 16198, 16209, 16310, 16381, 16471, 16506, 16517, 16518, 16536, 16557, 16609, 16982 |
| **V3D:Test Support** | 10860, 15409 |
| **V3E: A2MP** | 11421, 13236, 16065 |

| | **v5.3** |
|---|---|
| **V3F: ATT** | 10843, 13147, 13183, 13184, 13255, 13305, 13511, 13530, 14761, 14819, 14988, 15832, 16018, 16494, 16495, 16505, 16794, 16817, 16982 |
| **V3G: GATT** | 10419, 10843, 11925, 12348, 12526, 13067, 13147, 13153, 13184, 13189, 13247, 13263, 13304, 13307, 13332, 13418, 14988, 15142, 15334, 15550, 15586, 15764, 15831, 15850, 16178, 16188, 16266, 16479, 16505, 16671, 16759, 16920 |
| **V3H: Security Manager** | 10860, 11293, 11424, 12348, 12880, 13147, 13276, 13408, 14708, 14964, 15198, 15310, 15314, 15315, 15555, 15583, 15642, 15668, 15850, 15851, 16305 |
| **V4A: UART Transport Layer** | 14674, 16983 |
| **V4B: USB Transport Layer** | 11289, 11421 |
| **V4C: Secure Digital (SD) Transport Layer** | |
| **V4D: Three-wire UART** | 16635 |
| **V4E: HCI** | 10419, 10860, 11421, 11424, 11702, 11747, 11754, 11764, 11895, 12036, 12093, 12253, 12313, 12379, 12517, 12598, 12599, 12604, 12664, 12693, 12802, 12863, 12866, 12956, 13029, 13081, 13104, 13139, 13142, 13143, 13148, 13177, 13246, 13252, 13253, 13261, 13278, 13286, 13287, 13293, 13321, 13344, 13346, 13374, 13375, 13376, 13377, 13379, 13426, 13438, 13473, 13498, 13527, 13533, 13534, 13536, 13542, 13546, 13570, 13582, 13597, 14615, 14617, 14636, 14650, 14651, 14655, 14665, 14700, 14706, 14713, 14714, 14741, 14747, 14770, 14771, 14824, 14830, 14831, 14834, 14841, 14867, 14871, 14893, 14895, 14901, 14909, 14916, 14934, 14957, 14969, 14980, 14984, 14998, 15007, 15010, 15021, 15030, 15031, 15032, 15034, 15035, 15039, 15059, 15062, 15075, 15109, 15111, 15132, 15152, 15174, 15190, 15204, 15223, 15233, 15275, 15281, 15282, 15288, 15305, 15310, 15311, 15334, 15378, 15380, 15382, 15412, 15413, 15461, 15471, 15496, 15507, 15508, 15509, 15513, 15524, 15526, 15553, 15570, 15583, 15590, 15592, 15593, 15597, 15622, 15633, 15640, 15643, 15651, 15695, 15728, 15732, 15745, 15752, 15816, 15826, 15837, 15843, 15850, 15851, 15964, 16015, 16058, 16101, 16125, 16129, 16133, 16142, 16164, 16167, 16180, 16191, 16238, 16287, 16288, 16299, 16318, 16329, 16330, 16331, 16359, 16381, 16397, 16435, 16445, 16480, 16504, 16506, 16538, 16561, 16584, 16610, 16612, 16635, 16636, 16656, 16674, 16681, 16793, 16794, 16796, 16822, 16817, 16913, 16915, 16983 |
| **V5A: 802.11 Protocol Adaptation** | 11421, 12253, 13473, 15523 |
| **V6A: Physical Layer** | 12313, 13254, 15065, 15306, 15461, 15851, 16372, 16984 |

| | v5.3 |
|---|---|
| **V6B: Link Layer** | 11164, 11421, 11668, 11702, 11895, 12102, 12250, 12251, 12252, 12313, 12325, 12379, 12664, 12802, 12955, 13089, 13155, 13178, 13179, 13184, 13212, 13251, 13260, 13374, 13387, 13512, 13523, 13526, 13534, 13591, 13603, 14604, 14615, 14655, 14665, 14676, 14680, 14681, 14698, 14699, 14707, 14746, 14750, 14770, 14777, 14779, 14831, 14835, 14842, 14851, 14854, 14855, 14911, 14922, 14934, 14958, 14969, 14979, 15007, 15010, 15021, 15027, 15042, 15059, 15062, 15070, 15096, 15145, 15200, 15204, 15223, 15235, 15286, 15288, 15292, 15300, 15303, 15317, 15426, 15428, 15429, 15495, 15497, 15518, 15579, 15583, 15584, 15630, 15642, 15659, 15711, 15742, 15816, 15818, 15825, 15835, 15839, 15850, 15851, 15894, 15999, 16039, 16067, 16093, 16101, 16114, 16210, 16251, 16282, 16287, 16346, 16347, 16359, 16363, 16372, 16381, 16442, 16506, 16537, 16553, 16554, 16610, 16618, 16657, 16671, 16675, 16681, 16731, 16794, 16817, 16984 |
| **V6C: Sample Data** | 11895, 12197, 15200 |
| **V6D: MSCs** | 10860, 12345, 12922, 14721, 14747, 14934, 15007, 15021, 15147, 15223, 15305, 15401, 15579, 15583, 15630, 15691, 15999, 16817, 16984, 17046 |
| **V6E: Low Energy Link Layer** | 12068, 16039, 16093, 16681 |
| **V6F: Direct Test Mode** | |
| **V6G: Isochronous Adaptation Layer** | 12797, 13082, 13083, 13085, 13086, 13251, 13316, 13533, 14665, 14771, 14897, 14901, 15304, 15658, 15672, 15851, 15963, 16278, 16817 |
| **V7A: MWS Coexistence Logical Signaling** | |
| **V7B: Wireless Coexistence Interface 1 (WCI-1) Transport Specification** | 11987, 11988 |
| **V7C: Wireless Coexistence Interface 2 (WCI-2) Transport Specification** | |

*Table 12.1: Errata integrated in v5.3*

## 12.4 Global terminology changes

Certain terms that were identified as inappropriate have been replaced. As a consequence, some other terms have also been changed to retain consistency and some HCI command and event names had consequential changes. For a list of the original terms and names and their replacements, see the Appropriate Language Mapping Tables, https://www.bluetooth.com/language-mapping/Appropriate-Language-Mapping-Table.

These terminology changes were made under the following issues: 15328, 15334, 15336, 15338, 15342, 15343, 15344, 15345, 15346, 15347, 15348, 15349, 15350, 15351, 15352, 15353, 15354, 15355, 15356, 15357, 15358, 15359, 15360, 15361, 15362, 15363, 15389, 15529, 15531.

# 13 CHANGES FROM V5.3 TO V5.4

## 13.1 New features

Several new features are introduced in v5.4. The major areas of improvement are:

- Advertising Coding Selection
- Encrypted Advertising Data
- LE GATT Security Levels Characteristic
- Periodic Advertising with Responses

## 13.2 Removed features

No features were removed in v5.4.

## 13.3 Errata incorporated in v5.4

Table 13.1 lists the integrated errata items.

| | **v5.4** |
|---|---|
| **Global / Several Parts** | 17307, 20357, 20589 |
| **Front matter** | 17925 |
| **V0A: Consolidated Table of Contents** | |
| **V0B: Bluetooth Compliance Requirements** | 15534, 20402, 22281 |
| **V0C: Version History and Acknowledgments** | 17026, 17070, 19267, 19268, 19269, 20567, 22267, 22438 |
| **V1A: Architecture Overview** | 15535, 16097, 16927, 16940, 17231, 17232, 17332, 17366, 17464, 17746, 17755, 18603, 18654, 18765, 19112, 19149, 19206, 20369, 20604, 22220 22281 |
| **V1B: Acronyms & Abbreviations** | 17085, 17232, 17332, 18234, 19032, 19284, 20369 |
| **V1C: Change History** | 17232, 17332, 17993, 18302, 19025 |
| **V1D: Mixing of Spec Versions** | 15535, 17332 |
| **V1E: General Terminology** | 16542, 17028, 17090, 17888, 18234, 19202, 22345, 22439 |
| **V1F: Controller Error Codes** | 17332, 19090, 19198 |
| **V2A: Radio** | 17036, 17249, 20635 |

| | **v5.4** |
|---|---|
| **V2B: Baseband** | 15536, 15605, 17031, 17083, 17085, 17578, 17708, 17830, 17849, 18344, 19206 |
| **V2C: LMP** | 10969, 17031, 17083, 17532, 17565, 17591, 19206, 20605 |
| **V2F: MSCs** | 10969, 15536, 17039, 17391, 17566, 18068, 18946, 22220 |
| **V2G: Sample Data** | 10969, 17034 |
| **V2H: Security** | 15536, 16988, 17031, 17083, 17084, 17085, 17113, 17226, 17339, 17567, 22220 |
| **V3A: L2CAP** | 10969, 15248, 15537, 16591, 17042, 17043, 17044, 17050, 17051, 17085, 17232, 17319, 17332, 17464, 17468, 17560, 17568, 17580, 17709, 17849, 18303, 18341, 18343, 19054, 19098, 19206, 20635, 22220, 22441 |
| **V3B: SDP** | 10969, 15537, 17050, 17569, 17980 |
| **V3C: GAP** | 10969, 15537, 17044, 17050, 17214, 17332, 17398, 17470, 17571, 17702, 17832, 17946, 18083, 18234, 18303, 18668, 18766, 18819, 18820, 18896, 18905, 18962, 19042, 19093, 19149, 20369, 20374, 20385, 20420, 20452, 20605, 20606, 22185, 22193, 22220, 22222, 22441 |
| **V3D:Test Support** | 17083, 17572, 189051, 19217 |
| **V3E: A2MP** | 16263 |
| **V3F: ATT** | 10969, 16963, 17013, 17232, 17332, 17354, 17358, 17383, 17466, 17573, 17730, 19088, 22220 |
| **V3G: GATT** | 15248, 15537, 16947, 16962, 17232, 17332, 17574, 17698, 19169, 19194, 24889 |
| **V3H: Security Manager** | 10969, 15537, 16988, 17044, 17085, 17113, 17575, 17732, 18014, 18283, 19324, 22220 |
| **V4A: UART Transport Layer** | |
| **V4B: USB Transport Layer** | 15538, 17064, 17332, 17529 |
| **V4C: Secure Digital (SD) Transport Layer** | 17332 |
| **V4D: Three-wire UART** | 15538, 17085, 17107 |

| | **v5.4** |
|---|---|
| **V4E: HCI** | 15538, 16591, 16769, 16775, 16778, 16850, 16908, 16934, 16966, 16998, 17021, 17032, 17055, 17057, 17058, 17061, 17062, 17063, 17065, 17069, 17108, 17125, 17332, 17404, 17407, 17412, 17433, 17451, 17456, 17469, 17510, 17646, 17697, 17702, 17762, 17849, 17947, 18023, 18024, 18047, 18068, 18094, 18194, 18229, 18230, 18280, 18311, 18331, 18378, 18401, 18450, 18453, 18515, 18538, 18567, 18578, 18608, 18615, 18625, 18626, 18640, 18644, 18655, 18661, 18680, 18685, 18686, 18690, 18784, 18785, 18989, 18990, 19007, 19009, 19021, 19026, 19035, 19036, 19038, 19055, 19058, 19062, 19063, 19065, 19066, 19090, 19112, 19127, 19149, 19163, 19185, 19186, 19197, 19198, 19206, 19211, 19215, 19238, 20392, 20424, 20430, 20482, 20487, 22140, 22220, 22240, 22251, 22272, 22281, 22376, 22442 |
| **V5A: 802.11 Protocol Adaptation** | |
| **V6A: Physical Layer** | 17052, 18446, 18460, 18627, 20635, 20636, 22443 |
| **V6B: Link Layer** | 10969, 16097, 16591, 16878, 17052, 17083, 17085, 17089, 17234, 17332, 17352, 17356, 17361, 17390, 17433, 17489, 17509, 17531, 17583, 17702, 17849, 17971, 17991, 18070, 18100, 18226, 18233, 18234, 18267, 18311, 18456, 18476, 18555, 18566, 18567, 18632, 18648, 18653, 18701, 18742, 18766, 18805, 18833, 18834, 18970, 18989, 18990, 19007, 19026, 19060, 19080, 19087, 19089, 19112, 19132, 19149, 19151, 19200, 19206, 19238, 19266, 20385, 20389, 20401, 20444, 20447, 20472, 20499, 20605, 20624, 22169, 22170, 22219, 22220, 22371, 22443 |
| **V6C: Sample Data** | 17332 |
| **V6D: MSCs** | 10969, 15539, 17039, 17440, 17576, 17616, 18068, 18602, 18701, 20389, 20500, 22220 |
| **V6E: Low Energy Link Layer** | 16988, 17548, 20635, 22178 |
| **V6F: Direct Test Mode** | 17083, 17090, 17231, 17577, 20409 |
| **V6G: Isochronous Adaptation Layer** | 17090, 17462, 18001, 18002, 19149, 20476, 22281 |
| **V7A: MWS Coexistence Logical Signaling** | 22444 |
| **V7B: Wireless Coexistence Interface 1 (WCI-1) Transport Specification** | 10969, 19044, 22444 |
| **V7C: Wireless Coexistence Interface 2 (WCI-2) Transport Specification** | 17332, 19044, 22444 |

*Table 13.1: Errata integrated in v5.4*

# 14 CHANGES FROM V5.4 TO V6.0

## 14.1 New features

Several new features are introduced in v6.0. The major areas of improvement are:

- Channel Sounding, including Channel Sounding HCI Updates
- LL Extended Feature Set
- Decision-Based Advertising Filtering
- Enhancements for ISOAL
- Monitoring Advertisers
- Frame Space Update
- Core Configurations

## 14.2 Removed features

The following feature was removed in this version of the specification:

- Bluetooth Compliance Requirements

In addition, Mixing of Specification Versions was merged into the new Core Configurations feature.

## 14.3 Errata incorporated in v6.0

Table 14.1 lists the integrated errata items.

| | **v6.0** |
|---|---|
| **Global / Several Parts** | 23618, 25794 |
| **V0A: Consolidated Table of Contents** | |
| **V0B: Bluetooth Compliance Requirements** | 25429 |
| **V0C: Revision History and Acknowledgments** | 19321, 24281, 24284, 25264, 25429, 25553, 25794, 25795 |
| **V0D: Core Configurations** | 25429 |
| **V1A: Architecture Overview** | 17027, 18534, 22197, 22270, 22397, 22504, 22579, 22875, 22957, 22960, 23262, 23649, 23704, 24040, 24238, 24401, 24433, 24434, 24461, 24541, 24811, 25429, 25605 |

| | **v6.0** |
|---|---|
| **V1B: Acronyms & Abbreviations** | 22504, 23034, 24434, 25605, 25658 |
| **V1C: Change History** | 22489, 25795 |
| **V1D: Mixing of Spec Versions** | 23635, 25171, 25181, 25429 |
| **V1E: General Terminology** | 16543, 18534, 22270, 23135, 23548, 24044, 24798, 24800 |
| **V1F: Controller Error Codes** | 18534, 23446, 24855 |
| **V2A: Radio** | 22270, 23548, 23620, 24457, 25101, 25429, 25605 |
| **V2B: Baseband** | 17250, 17736, 22270, 23034, 23054, 23127, 23508, 23549, 23624, 23635, 23983, 25658 |
| **V2C: LMP** | 15466, 18534, 18960, 20613, 22262, 22270, 22504, 22839, 22926, 23064, 23219, 23578, 23579, 23739, 23983, 24042, 24171, 24284, 24402, 24582, 24675, 24786, 25101, 25429, 25658 |
| **V2F: MSCs** | 22488, 22489, 22490, 24617 |
| **V2G: Sample Data** | 24573, 24797, 25429 |
| **V2H: Security** | 17250, 17251, 17710, 20607, 22270, 22839, 23034, 23260, 23627, 23700, 24573, 24617, 25429, 25859 |
| **V3A: L2CAP** | 17049, 17250, 18534, 20607, 22270, 22534, 22535, 22594, 22641, 22693, 23048, 23991, 24217, 25429, 25605 |
| **V3B: SDP** | 16886, 20607, 22138, 22270, 22875, 23630, 23631, 24440 |
| **V3C: GAP** | 17049, 17874, 18082, 18534, 18703, 20657, 22255, 22289, 22492, 22589, 22726, 22783, 23034, 23190, 23303, 23635, 23704, 24057, 24058, 24202, 24312, 24394, 24440, 24461, 24493, 24617, 24620, 24762, 25021, 25429 |
| **V3D:Test Support** | 17250, 18534, 22270, 22504, 25429 |
| **V3F: ATT** | 22396, 22416, 22489, 22538, 22693, 23071, 23634, 24892, 25429, 25605 |
| **V3G: GATT** | 17355, 22270, 22487, 22693, 22894, 23537, 23635, 24126, 24247, 24440, 24473, 24580, 24889, 25021, 25429, 25605, 25658 |
| **V3H: Security Manager** | 17049, 22270, 22482, 23259, 23262, 23700, 24202, 24265, 24617, 25605 |
| **V4A: UART** | |
| **V4B: USB** | 22270 |
| **V4C: SD** | 16886, 22504, 24607 |
| **V4D: Three-wire UART** | 17250, 22270, 23983 |

| | **v6.0** |
|---|---|
| **V4E: HCI** | 11058, 16543, 16886, 18552, 18960, 18973, 20515, 20556, 20607, 22167, 22181, 22192, 22262, 22270, 22341, 22342, 22353, 22373, 22385, 22393, 22397, 22447, 22448, 22474, 22494, 22495, 22504, 22539, 22540, 22550, 22561, 22580, 22620, 22658, 22667, 22678, 22691, 22709, 22791, 22844, 22851, 22856, 22857, 22869, 22882, 22906, 22918, 23069, 23070, 23100, 23108, 23129, 23157, 23166, 23180, 23194, 23195, 23202, 23203, 23204, 23217, 23242, 23262, 23310, 23394, 23397, 23408, 23428, 23446, 23486, 23488, 23602, 23610, 23690, 23705, 23740, 23896, 23899, 23914, 23982, 23983, 24009, 24032, 24039, 24082, 24083, 24110, 24111, 24114, 24115, 24121, 24125, 24140, 24200, 24230, 24238, 24285, 24286, 24293, 24298, 24313, 24316, 24397, 24401, 24421, 24443, 24446, 24461, 24467, 24507, 24541, 24607, 24637, 24662, 24688, 24691, 24693, 24719, 24783, 24784, 24787, 24793, 24811, 24834, 24848, 24855, 24870, 24876, 24880, 24903, 24916, 24921, 24932, 24989, 25021, 25041, 25046, 25047, 25107, 25124, 25125, 25171, 25178, 25181, 25183, 25220, 25221, 25263, 25283, 25328, 25381, 25382, 25384, 25395, 25407, 25409, 25429, 25457, 25460, 25461, 25474, 25478, 25481, 25493, 25543, 25545, 25583, 25605, 25610, 25658, 25795 |
| **V6A: Physical Layer** | 22270, 22332, 22552, 23034, 23704, 24018, 24401, 24461, 24541, 24607, 24871, 24976, 25060, 25106, 25132, 25429, 25605, 25669, 25846, 25859 |
| **V6B: Link Layer** | 16543, 17250, 17438, 17874, 18534, 18703, 18973, 20451, 20485, 20515, 20658, 22167, 22181, 22197, 22270, 22311, 22332, 22479, 22496, 22530, 22540, 22553, 22554, 22555, 22556, 22565, 22591, 22648, 22649, 22677, 22700, 22835, 22895, 22973, 23034, 23050, 23069, 23192, 23194, 23295, 23398, 23402, 23434, 23446, 23510, 23530, 23616, 23635, 23704, 23741, 23983, 24032, 24112, 24134, 24195, 24202, 24238, 24286, 24370, 24371, 24372, 24373, 24374, 24401, 24420, 24431, 24436, 24437, 24444, 24445, 24461, 24475, 24476, 24477, 24541, 24573, 24576, 24590, 24646, 24652, 24688, 24719, 24811, 24833, 24875, 24883, 24888, 24917, 25052, 25084, 25127, 25151, 25171, 25181, 25220, 25234, 25263, 25293, 25297, 25368, 25383, 25398, 25512, 25513, 25605, 25658, 25795 |
| **V6C: Sample Data** | 18534, 22928, 23220, 23617, 23619, 23704, 24112, 24284, 24297, 24308, 24378, 24866, 25127, 25165, 25173 |
| **V6D: MSCs** | 24787, 25128, 25605 |
| **V6E: LE Link Layer Security** | 22270, 22584, 23398, 23704, 24377, 24383, 24385 |
| **V6F: Direct Test Mode** | 17250, 18534, 22270, 22504, 24401, 24442, 24607, 24647, 24763, 24787, 24823, 24833, 24935, 25429, 25434, 25557 |
| **V6G: Isochronous Adaptation Layer** | 20485, 20607, 22270, 22876, 23724, 23726, 24011, 24127, 24185, 24286, 24653, 24690, 24856 |

| | **v6.0** |
|---|---|
| **V6H: Channel Sounding** | 22270, 23154, 23155, 23414, 23423, 23464, 23518, 23704, 24018, 24064, 24152, 24296, 24320, 24401, 24434, 24541, 24575, 24607, 24638, 24647, 24795, 24796, 24811, 24916, 25058, 25060, 25072, 25126, 25172, 25293, 25296, 25425, 25542, 25605, 25658, 25674 |
| **V7A: MWS Coexistence Logical Signaling** | 18534, 22652 |
| **V7B: WCI-1** | 22270 |
| **V7C: WCI-2** | 22270 |

*Table 14.1: Errata integrated in v6.0*

# 15 CHANGES FROM V6.0 TO V6.1

## 15.1 New features

One new feature is introduced in v6.1:

- Randomized RPA Updates

## 15.2 Removed features

No features were removed in v6.1.

## 15.3 Errata incorporated in v6.1

Table 15.1 lists the integrated errata items.

| | v6.1 |
|---|---|
| **Global / Several Parts** | 26111 |
| **V0A: Consolidated Table of Contents** | |
| **V0B: Bluetooth Compliance Requirements** | |
| **V0C: Revision History and Acknowledgments** | 26264 |
| **V0D: Core Configurations** | 26814 |
| **V1A: Architecture Overview** | 19064, 25773, 26099, 26162 |
| **V1B: Acronyms & Abbreviations** | 26814 |
| **V1C: Change History** | 27211 |
| **V1D: Mixing of Spec Versions** | |
| **V1E: General Terminology** | 25625, 26381 |
| **V1F: Controller Error Codes** | 25131 |
| **V2A: Radio** | 26814 |
| **V2B: Baseband** | 24460, 26133, 26814 |
| **V2C: LMP** | 17738, 24460, 25502, 25832, 25833, 26814, 27211 |
| **V2F: MSCs** | |
| **V2G: Sample Data** | 26814 |
| **V2H: Security** | 25131, 26548, 26814 |
| **V3A: L2CAP** | 24555, 25439, 26664 |
| **V3B: SDP** | 26814 |

| | **v6.1** |
|---|---|
| **V3C: GAP** | 18947, 19226, 19323, 24891, 25255, 25458, 25773, 26158, 27211 |
| **V3D:Test Support** | |
| **V3F: ATT** | 20421, 23409, 25193, 25394, 25611, 27211 |
| **V3G: GATT** | 20375, 25193, 25227, 25611 |
| **V3H: Security Manager** | 25086, 26548, 26814 |
| **V4A: UART** | |
| **V4B: USB** | 26814 |
| **V4C: SD** | |
| **V4D: Three-wire UART** | 26814 |
| **V4E: HCI** | 11324, 22884, 23289, 23302, 24234, 24460, 24761, 25014, 25090, 25131, 25333, 25365, 25502, 25530, 25531, 25633, 25684, 25703, 25823, 25847, 25900, 25903, 25906, 25944, 26064, 26104, 26162, 26325, 26348, 26390, 26411, 26422, 26514, 26517, 26520, 26548, 26753, 26814, 27211 |
| **V6A: Physical Layer** | 25847, 26162, 26439 |
| **V6B: Link Layer** | 11667, 20623, 22557, 23376, 25056, 25212, 25333, 25392, 25393, 25480, 25633, 25718, 25720, 25765, 25817, 25820, 25823, 25847, 25899, 25926, 25982, 25995, 25996, 25997, 26060, 26104, 26112, 26164, 26325, 26391, 26398, 26490, 26533, 26539, 26548, 26567, 26593, 26645, 26658, 26788, 26814, 27211 |
| **V6C: Sample Data** | |
| **V6D: MSCs** | |
| **V6E: LE Link Layer Security** | |
| **V6F: Direct Test Mode** | |
| **V6G: Isochronous Adaptation Layer** | |
| **V6H: Channel Sounding** | 25633, 25847, 25876, 26062, 26063, 26078, 26082, 26112, 26116, 26162, 26814 |
| **V7A: MWS Coexistence Logical Signaling** | |
| **V7B: WCI-1** | |
| **V7C: WCI-2** | |

*Table 15.1: Errata integrated in v6.1*

# 16 CHANGES FROM V6.1 TO V6.2

## 16.1 New features

Several new features are introduced in v6.2. The major areas of improvement are:

- HCI USB LE Isochronous Support
- LE Test Mode Enhancements
- Shorter Connection Intervals
- Channel Sounding Amplitude-based Attack Resilience

## 16.2 Removed features

No features were removed in v6.2.

## 16.3 Security errata

The Security errata shown in Table 16.1 have been resolved and integrated in this version of the specification.

| Erratum | Title |
|---|---|
| 24489 | Implementations of Secure Simple Pairing and BR/EDR Secure Connections Pairing may generate insecure Passkeys. |
| 24490 | Implementations of LE Legacy Pairing and LE Secure Connections Pairing may generate insecure Passkeys. |
| 24491 | Implementations of LE Legacy Pairing may be vulnerable to reflection attacks on pairing Random and Confirm values. |
| 24557 | Random numbers used in security procedures may not be sourced from sufficiently secure random number generators. |
| 24558 | Controller implementations may offer a valid DHKey in response to a key generation request against an invalid remote peer public key. |
| 24560 | SDP Server implementations may accept an invalid continuation state. |
| 26039 | Conforming BR/EDR implementations shall no longer be able to negotiate or use a BR/EDR baseband link with an encryption key size of less than 7 octets |
| 26041 | Conforming implementations shall not permit a user to set the encryption key size |
| 26043 | Conforming implementations shall not retain a fixed public/private key pair if they support generation of a public/private key pair |
| 26047 | Conforming implementations shall be required to store the last verified SignCounter in the security database if data signing is used |

| Erratum | Title |
|---|---|
| 26048 | Conforming implementations shall not be permitted to populate a resolving list with zero IRK values when operating in device privacy mode |
| 26770 | Fix contradiction in legacy passkey entry security assessment |
| 27068 | Require the LL to ignore BIG Control PDUs when previous data has CSTF=0 |

*Table 16.1: Security errata*

## 16.4 Errata incorporated in v6.2

Table 16.2 lists the integrated errata items.

| | v6.2 |
|---|---|
| **Global / Several Parts** | |
| **V0A: Consolidated Table of Contents** | |
| **V0B: Bluetooth Compliance Requirements** | |
| **V0C: Revision History and Acknowledgments** | 26341, 28009, 28010 |
| **V0D: Core Configurations** | 26943, 27478, 27500 |
| **V1A: Architecture Overview** | 20603, 26341, 26387, 26389, 26417, 26535, 27187, 27672, 28009 |
| **V1B: Acronyms & Abbreviations** | 26559, 26560, 26627 |
| **V1C: Change History** | 28009 |
| **V1D: Mixing of Spec Versions** | |
| **V1E: General Terminology** | 27629 |
| **V1F: Controller Error Codes** | |
| **V2A: Radio** | 26341 |
| **V2B: Baseband** | 15163, 15530, 26387 |
| **V2C: LMP** | 26039, 26387, 27096, 27195 |
| **V2F: MSCs** | 15530, 26387, 26535, 27187 |
| **V2G: Sample Data** | 23622, 26039, 26389 |
| **V2H: Security** | 24489, 26039, 26041, 26043, 26389, 26535 |
| **V3A: L2CAP** | 15163, 24985, 25037, 26387, 26389, 26535, 26667, 27718, 28009 |
| **V3B: SDP** | 23629, 24560, 26387, 28009 |
| **V3C: GAP** | 25701, 26048, 26341, 26387, 26389, 26449, 26535, 27129, 27246, 27436, 27668, 27848 |
| **V3D:Test Support** | 26387 |

| | **v6.2** |
|---|---|
| **V3F: ATT** | 19328, 26387, 26535 |
| **V3G: GATT** | 15163, 19172, 26387, 26535, 26873, 27431, 27628, 27718 |
| **V3H: Security Manager** | 15163, 24490, 24491, 24557, 26047, 26309, 26387, 26535, 27504, 26770 |
| **V4A: UART** | |
| **V4B: USB** | |
| **V4C: SD** | |
| **V4D: Three-wire UART** | |
| **V4E: HCI** | 12498, 15506, 15530, 22269, 24558, 25661, 25672, 25732, 25755, 25808, 25974, 25975, 26039, 26083, 26387, 26389, 26435, 26462, 26471, 26535, 26558, 26561, 26568, 26662, 26810, 26916, 26939, 26940, 26994, 27069, 27126, 27214, 27235, 27228, 27442, 27472, 27493, 27495, 27498, 27646, 27719, 27750, 27774, 27797, 27824, 27871, 28022, 28023 |
| **V6A: Physical Layer** | 26341, 26387, 26389, 26578, 28023 |
| **V6B: Link Layer** | 15163, 20570, 25808, 26130, 26387, 26389, 26412, 26450, 26463, 26467, 26471, 26488, 26535, 26545, 26558, 26562, 26941, 26945, 27068, 27234, 27235, 27246, 27495, 27499, 27543, 27672, 27677, 27720, 27775, 27789, 27797, 27887, 28009, 28022 |
| **V6C: Sample Data** | 26999, 28009 |
| **V6D: MSCs** | 15163, 25808, 26535, 26810, 26916, 27473, 27672, 28009 |
| **V6E: LE Link Layer Security** | 28009 |
| **V6F: Direct Test Mode** | 26201, 26204, 26412, 26441, 26464, 26465, 26466, 26471, 26472, 26504, 26559, 26563, 26564, 26565, 26627, 26774, 27268, 27402 |
| **V6G: Isochronous Adaptation Layer** | 24326, 26389, 26521, 27187, 27827 |
| **V6H: Channel Sounding** | 15163, 26387, 26389, 27246, 27650, 27846, 28022 |
| **V7A: MWS Coexistence Logical Signaling** | |
| **V7B: WCI-1** | 26535 |
| **V7C: WCI-2** | 26535 |

*Table 16.2: Errata integrated in v6.2*

# 17 CHANGES FROM V6.2 TO V6.3

## 17.1 New features

Several new features are introduced in v6.3. The major areas of improvement are:

- Channel Sounding Inline Phase Correction Term Transfer
- Channel Sounding PHY-specific RTT Accuracy
- Running Out of Bits

## 17.2 Removed features

The following feature was removed in this version of the specification:

- Data signing

## 17.3 Errata incorporated in v6.3

Table 17.1 lists the integrated errata items.

| | **v6.3** |
|---|---|
| **Global / Several Parts** | 9480 |
| **V0A: Consolidated Table of Contents** | |
| **V0B: Bluetooth Compliance Requirements** | |
| **V0C: Revision History and Acknowledgments** | 18901, 28188, 28600 |
| **V0D: Core Configurations** | 28600 |
| **V1A: Architecture Overview** | 17734, 18901, 26497, 28031, 28451, 28600 |
| **V1B: Acronyms & Abbreviations** | 18901, 28411 |
| **V1C: Change History** | 18901, 26122, 28600 |
| **V1D: Mixing of Spec Versions** | |
| **V1E: General Terminology** | 18031, 28782 |
| **V1F: Controller Error Codes** | 27398 |
| **V2A: Radio** | 25649, 28031, 28108 |
| **V2B: Baseband** | 27501, 27826, 28031, 28188 |
| **V2C: LMP** | 26497, 27398, 27464, 27501, 27826, 28188, 28782 |
| **V2F: MSCs** | 23102, 27826, 28188 |

| | **v6.3** |
|---|---|
| **V2G: Sample Data** | 18031 |
| **V2H: Security** | 17791, 23102, 27398, 27826, 28031, 28158, 28782 |
| **V3A: L2CAP** | 26580, 27398, 27686, 27826, 28031, 28188 |
| **V3B: SDP** | 28031, 28140, 28188 |
| **V3C: GAP** | 18901, 22714, 26054, 26856, 27398, 27674, 27758, 27826, 28031, 28188, 28262, 28782 |
| **V3D: Test Support** | 26497 |
| **V3F: ATT** | 18901, 28031, 28188 |
| **V3G: GATT** | 18901, 27398 |
| **V3H: Security Manager** | 17791, 18031, 18901, 22154, 26580, 27398, 27734, 28031, 28158 |
| **V4A: UART** | |
| **V4B: USB** | |
| **V4C: SD** | 26497 |
| **V4D: Three-wire UART** | 27826 |
| **V4E: HCI** | 12417, 22714, 23214, 25017, 25494, 25647, 26328, 26330, 26339, 26349, 26497, 26540, 26690, 26771, 26793, 26795, 26829, 26831, 26854, 27156, 27210, 27248, 27276, 27398, 27464, 27501, 27505, 27657, 27735, 27736, 27922, 28073, 28180, 28188, 28208, 28262, 28394, 28488, 28493, 28584, 28589, 28600, 28616, 28618, 28782 |
| **V6A: Physical Layer** | 26187, 28008, 28600 |
| **V6B: Link Layer** | 22714, 22830, 26236, 26399, 26403, 26625, 26643, 26824, 27252, 27276, 27398, 27413, 27804, 27826, 27925, 28018, 28031, 28137, 28188, 28230, 28264, 28451, 28497, 28584, 28600, 28782 |
| **V6C: Sample Data** | 18031 |
| **V6D: MSCs** | 26231, 27398, 27826, 28042, 28411, 28782 |
| **V6E: LE Link Layer Security** | |
| **V6F: Direct Test Mode** | 26185, 26187, 26497, 26771, 27398, 27623, 27844, 27959, 28188 |
| **V6G: Isochronous Adaptation Layer** | 27398 |
| **V6H: Channel Sounding** | 26497, 26540, 26760, 26802, 26824, 27210, 27398, 27796, 28007, 28052, 28059, 28067, 28326 |

| | **v6.3** |
|---|---|
| **V7A: MWS Coexistence Logical Signaling** | |
| **V7B: WCI-1** | |
| **V7C: WCI-2** | |

*Table 17.1: Errata integrated in v6.3*

# [THIS PART IS NO LONGER USED]

*Mixing requirements are located in [Vol 0] Part D.*

Architecture, Change History,
And Conventions
Part E

# GENERAL TERMINOLOGY AND INTERPRETATION

# CONTENTS

**1 Language conventions ..... 412**
1.1 [This section is no longer used] ..... 413
1.2 [This section is no longer used] ..... 413
1.3 [This section is no longer used] ..... 413
1.4 [This section is no longer used] ..... 413
1.5 [This section is no longer used] ..... 413
1.6 [This section is no longer used] ..... 413
1.7 Implementation Alternatives ..... 413
1.8 Discrepancies ..... 413
1.9 Appropriate Language ..... 413

**2 General interpretation rules ..... 414**
2.1 Binary and hexadecimal values ..... 414
2.2 Bit numbers and bit fields ..... 414
2.3 Specification of bit values ..... 414
2.4 Values with restricted purposes ..... 415
2.4.1 Reserved for future use ..... 415
2.4.2 Previously used ..... 415
2.4.3 Reserved for specification development purposes ..... 416
2.5 Use of invalid values in checksums and other calculations ..... 416
2.6 Assigned number requirements ..... 416
2.7 Responding to invalid behavior ..... 416
2.8 Ranges of values ..... 418
2.9 Type Names ..... 418
2.9.1 Basic types ..... 419
2.9.2 Array types ..... 419
2.9.3 Variable length types ..... 420
2.10 Mathematical conventions ..... 420
2.11 Requirement status symbols ..... 422
2.12 Table structure ..... 422
2.13 References to HCI commands and events ..... 423

**3 Naming conventions ..... 424**
3.1 BR/EDR ..... 424
3.2 Bluetooth Low Energy ..... 424
3.2.1 Link Layer PDUs ..... 424

**4 Sample Data ..... 427**

# 1 LANGUAGE CONVENTIONS

In the development of a specification, the Bluetooth SIG has established the following conventions for use of the terms *“shall”*, *"mandatory"*, *“shall not”*, *“should”*, *“should not”*, *“may”*, *"optional"*, *“must”*, and *“can”*. In this Bluetooth specification, the terms in Table 1.1 have the specific meanings given in that table, irrespective of other meanings that exist.

| | |
|---|---|
| shall<br>*or*<br>mandatory | —used to express what is required by the specification and is to be implemented exactly as written without deviation |
| shall not | —used to express what is forbidden by the specification |
| should<br>*or*<br>may<br>*or*<br>optional | —not mandatory. Used to express either:<br>1. what is recommended by the specification without forbidding anything ("should")<br>2. what is permissible within the limits of the specification ("may" or "optional") |
| should not | —used to indicate that something is discouraged but not forbidden by the specification |
| must | —used to indicate either:<br>1. an indisputable statement of fact that is always true regardless of the circumstances<br>2. an implication or natural consequence if a separately-stated requirement is followed |
| can | —used to express a statement of possibility or capability |

*Table 1.1: Language conventions terms and definitions*

Where more than one item is permitted but not required, the choices to include or support those items are independent from one another unless the specification explicitly states otherwise. Each item that is implemented shall be implemented exactly as written without deviation.

## 1.1 [This section is no longer used]

## 1.2 [This section is no longer used]

## 1.3 [This section is no longer used]

## 1.4 [This section is no longer used]

## 1.5 [This section is no longer used]

## 1.6 [This section is no longer used]

## 1.7 Implementation Alternatives

When specification content indicates that there are multiple alternatives to satisfy specification requirements, if one alternative is explained or illustrated in an example it is not intended to limit other alternatives that the specification requirements permit.

## 1.8 Discrepancies

It is the goal of Bluetooth SIG that specifications are clear, unambiguous, and do not contain discrepancies. However, members can report any perceived discrepancy by filing an erratum and can request a test case waiver as appropriate.

## 1.9 Appropriate Language

Certain terms that were identified as inappropriate have been replaced. As a consequence, some other terms have also been changed to retain consistency and some HCI command and event names had consequential changes. For a list of the original terms and names and their replacements, see the Appropriate Language Mapping Tables, https://www.bluetooth.com/language-mapping/Appropriate-Language-Mapping-Table.

# 2 GENERAL INTERPRETATION RULES

The following rules apply throughout the specification except where explicitly overridden.

## 2.1 Binary and hexadecimal values

Binary numbers are normally written with a “0b” prefix, so 0b1101 is the same as the decimal number 13.

In some places a sequence of bits is written in quotation marks thus: '1010'. Such sequences are not normally intended to be interpreted as numbers. The order that the bits are to be processed will always be specified.

Hexadecimal numbers are written with a "0x" prefix, so 0x42 is the same as the decimal number 66. The letters "a" to "f" are used to represent the digits 10 to 15, so 0x1A is the same as the decimal number 26. The case of letters in a hexadecimal number is not significant.

Underscore characters may be placed between the digits of binary or hexadecimal numbers to make them easier to interpret; these underscores shall not affect the value. For example, 0b0010_1011 and 0b00101011 both equal the decimal number 43.

All numbers not written in one of the above ways are decimal.

## 2.2 Bit numbers and bit fields

In some cases the specification needs to refer to some of the bits of an integer value. Bits are always numbered from 0 as the least significant bit, so bit 0 of 0b1011 equals 1 while bit 2 equals 0. A single bit will be notated with a subscript, as in $CLK_5$.

Sometimes it is necessary to refer to a consecutive set of bits; for example, given a value CLK it may be necessary to refer to bits 2 to 4 of CLK (that is, the value equal to (CLK ÷ 4) *mod* 8. This will be notated either by a subscript with a dash or by brackets and a colon; the bit numbers will always be inclusive and the most significant bit number is given first. For example, bits 2 to 4 of CLK are written $CLK_{4-2}$ or CLK[4:2].

## 2.3 Specification of bit values

Some values in the specification are divided into individual bits, each of which has a description. If explicit bit values are not given then this description represents the

meaning when the bit equals 1 and the opposite applies when the value is 0. For example, a description of:

Bit 3: use 3-slot packets

means the same as:

Bit 3 = 1: use 3-slot packets;

Bit 3 = 0: do not use 3-slot packets.

## 2.4 Values with restricted purposes

Where a field in a packet, PDU, or other data structure, a parameter, or another variable object is described as being split into components (e.g., when each bit of a field has a separate meaning), the rules in this section apply to each component separately.

### 2.4.1 Reserved for future use

Where a field in a packet, PDU, or other data structure is described as "Reserved for future use" (irrespective of whether in upper or lower case), the device creating an instance of the structure to send to another device (including messages sent between a Host and a Controller over HCI) shall set it to zero. Any device receiving or interpreting the structure shall ignore that field; in particular, it shall not reject the structure because of the value of the field.

Where a field, parameter, or other variable object can take a range of values and some values are described as "Reserved for future use", devices shall not set the object to any of those values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous; however, this does not apply in a context where the object is described as being ignored or if it is specified to ignore unrecognized values.

Where a field, parameter, or other variable object only has meanings specified for some values, all other values are reserved for future use.

The abbreviation "RFU" is equivalent to "Reserved for future use".

### 2.4.2 Previously used

The term "Previously used" (irrespective of whether in upper or lower case) indicates that a field or value was used for a removed feature (see [Vol 1] Part C, Section 1). Devices that do not implement that feature shall treat the field or value as reserved for future use.

Note: These fields and values will not be used for new features.

### 2.4.3 Reserved for specification development purposes

The term "Reserved for specification development purposes" (irrespective of whether in upper or lower case) indicates that a value will not appear in published specifications but may be used during specification development.

Other than during specification development, these values shall be treated as reserved for future use. Implementations shall not use these values for vendor-specific features or purposes.

## 2.5 Use of invalid values in checksums and other calculations

Where a field or value is used as part of a checksum, CRC, cryptographic key, or other similar calculation, and the value sent to or received from the peer is not valid (for example, it is an RFU field that has not been set to 0 by the sender), the actual value sent or received shall be used in the calculation and it shall not be replaced by a different valid value.

## 2.6 Assigned number requirements

The Bluetooth SIG maintains a published set of assigned numbers on the Bluetooth SIG Assigned Numbers web page. These assigned numbers are grouped in various number spaces. Numbers assigned may overlap with other assigned numbers in different number spaces, but no number within a number space is ever reused. The various number spaces are defined in the specification that defines the usage of the assigned numbers.

All assigned numbers within a given number space shall only be designated by the Bluetooth SIG and shall only be used for their intended purposes when used within a field, parameter, or other variable object defined to take on a value within that number space. All values not explicitly assigned within a given number space are Reserved for future use and subject to the requirements in Section 2.4.

All 16-bit and 32-bit UUIDs as defined in [Vol 3] Part B, Section 2.5.1, are considered assigned numbers. All other UUID values may be used in any context where a UUID is permitted provided they are generated according to the recommendations in ITU-T Rec. X.667(10/2012), alternatively known as ISO/IEC 9834.8:2014.

## 2.7 Responding to invalid behavior

If a device receives a packet or PDU that it supports but is not permitted in its current state, or has contents not permitted by the specification, the sending device

has exhibited invalid behavior. Unless the specification states a particular action to be taken, the receiving device should respond in one of the following ways:

- Ignore the packet or PDU.
- Attempt to recover the situation while both not violating the specification and being prepared for the sending device to be in a state where one or both devices cannot recover.
- If the packet or PDU was received from the peer device in a connection, either immediately terminate that connection or consider the connection to be lost. This may be done at any appropriate layer; e.g., following invalid behavior on an L2CAP channel, a device could choose to terminate either the channel or the underlying ACL connection.

Methods for recovering from invalid behavior include, but are not limited to:

- Sending a rejection response
- If a packet or PDU is too long, ignoring the extra contents
- If a value is out of range, using the nearest permitted value
- If a field is missing, using any default value specified

Examples of packets or PDUs not permitted in the current state include:

- A POLL packet sent by a Peripheral (see [Vol 2] Part B, Section 6.5.1.3)
- An LMP_DETACH PDU received while the device is in Hold or Sniff mode (see [Vol 2] Part C, Section 4.1.2)
- Any packet sent on the primary advertising physical channel using the LE 2M PHY or an ADV_IND PDU sent on the LE Coded PHY (see [Vol 6] Part B, Section 2.3)
- Two consecutive LL_FEATURE_REQ PDUs sent by a Central with no LL_FEATURE_RSP sent by the Peripheral between them (see [Vol 6] Part B, Section 5.1.4.1)
- A second LL_VERSION_IND PDU sent by the same device during the same connection (see [Vol 6] Part B, Section 5.1.5)
- A second ATT request or indication before the device has sent a response or confirmation in reply to the first one

Examples of packets and PDUs with contents not permitted by the specification include:

- A packet or PDU which is required to have a specific length but does not have that length
- A PDU where the value of a specific parameter or field is out of range (also see Section 2.4)

- An L2CAP C-frame sent over fixed channel CID 0x0005 that contains more than one command (see [Vol 3] Part A, Section 4) or has a length longer than that of the enclosed command packet

Note: The appropriate response to invalid behavior will depend on the specific circumstances and the choice made can affect the user experience. For example, a POLL packet sent by a Peripheral could actually be a NULL packet with the TYPE field corrupted, so it would be reasonable to ignore the POLL packet rather than terminate the connection. On the other hand, invalid behavior during a security procedure can indicate an attack by a third party and immediate disconnection would be appropriate.

Note: If the sending device supports different features or a different version of the specification, invalid behavior can occur because there are different requirements and one device has not checked for this possibility, such as by checking feature masks.

## 2.8 Ranges of values

Where a range is given (e.g. "the value is between 1 and 10", "in the range 7 to Nmax", or "Size: 1-31") then the range always includes both endpoints unless explicitly stated otherwise.

## 2.9 Type Names

The names defined in this section are used in the specification to indicate the type and representation of a value.

Values shall be stored in little-endian order. Except in arrays (see Section 2.9.2), a value of a given type shall be stored in the smallest number of octets that can contain the value; the value shall occupy the whole of each octet except the most significant bits of the last octet. All other bits in the last octet shall be reserved for future use. For example, if a value has a size of 22 bits, then it occupies 3 octets:

- The 8 least significant bits ($value_{7-0}$) are stored in the first octet.
- The next 8 bits ($value_{15-8}$) are stored in the middle octet.
- The 6 most significant bits ($value_{21-16}$) are stored in the 6 least significant bits of the last octet.
- The 2 most significant bits of the last octet are reserved for future use.

Where a value appears in an SDP service record (see [Vol 3] Part B, Section 2.2), the octets representing the value shall be reversed in order to match the big-endian order used by SDP. The order of elements of an array shall be unchanged.

### 2.9.1 Basic types

The names *uint#,* where # is a decimal number other than zero, indicate an unsigned integer with the specified number of bits; therefore *uint16* is an unsigned integer with 16 bits. The value shall be represented in binary, so *uint16* has the range of values 0 to 65535.

The names *sint#*, where # is a decimal number other than 0 or 1, indicate a signed integer with the specified number of bits including sign; therefore *sint12* is a signed integer with 12 bits. The value shall be represented in two's-complement binary with the sign bit as the most significant bit, so *sint12* has the range of values -2048 to +2047.

The name *boolean* indicates a single bit representing a truth value: a 0 bit represents false and a 1 bit represents true.

The names *float#,* where # is 16, 32, 64, 128, or 256, indicate a floating point number using the IEEE 754 interchange format with that number of bits.

The names *medfloat16* and *medfloat32* indicate the 16 and 32 bit floating point types defined in ISO/IEEE 11073-20601.

The names *UUID16*, *UUID32*, and *UUID128* indicate the three lengths of UUID (see [Vol 3] Part B, Section 2.5.1). The name *UUID* indicates a UUID whose length is specified separately to the value.

### 2.9.2 Array types

Any basic type may be converted to an array type of a specific length. The name of the array type is formed from the name of the basic type by adding the length, in brackets, after the name. For example, *sint12[3]* indicates an array of 3 values, each of type *sint12*. Empty brackets indicate that the length of the array is specified elsewhere. The original type is called the base type of the array.

If the size of the base type is 1 bit, each group of 8 consecutive elements is packed into an octet starting at the least significant bit for the first value.

If the size of the base type is 2 bits, each group of 4 consecutive elements is packed into an octet using bits 1-0 for the first value, bits 3-2 for the second value, bits 5-4 for the third value, and bits 7-6 for the fourth value.

If the size of the base type is 3 bits, each consecutive pair of elements is packed into an octet using bits 2-0 for the first value and bits 5-3 for the second value; bits 6 and 7 of each octet are reserved for future use.

If the size of the base type is 4 bits, each consecutive pair of elements is packed into an octet using bits 3-0 for the first value and bits 7-4 for the second value.

If the number of elements is not a multiple of 8, 4, or 2 as appropriate, the bits of the last octet that are not occupied by values are reserved for future use.

If the size of the base type is 5 bits or more, each element of the array occupies separate octets.

For example, the type *uint2[5]* occupies two octets, with the first value occupying bits 1-0 of the first octet, the fourth value occupying bits 7-6 of the first octet, the last value occupying bits 1-0 of the second octet, and bits 7-2 of the second octet reserved for future use. The type *sint12[3]* occupies six octets in total, two for each element.

### 2.9.3 Variable length types

The name *utf8s* indicates a variable length array of octets holding a string encoded using UTF-8. A specific number of octets may be indicated by appending the number in braces to the type name, so *utf8s{6}* indicates an array of 6 octets. If the actual string is shorter than the size of the array, the first unused octet shall be zero. If the number in braces is followed by the letter “z”, the remaining octets shall also be zero. For example, the string “Café” is represented in the type *utf8s{8z}* by the octet sequence 0x43, 0x61, 0x66, 0xC3, 0xA9, 0x00, 0x00, 0x00. If a specific length is not indicated, the length is specified separately to the value.

The name *utf16s* indicates a variable length array of *uint16* values holding a string encoded using UTF-16LE; the length is specified separately to the value.

The name *struct* indicates a variable length array of octets whose length and internal format are specified separately to the value.

## 2.10 Mathematical conventions

These conventions apply to how mathematical symbols are used in this specification, irrespective of how they are interpreted in any other context.

The operators +, −, ×, and ÷ have their usual meanings. Division is done using real numbers unless the result is being assigned to or used as an integer, in which case the quotient is rounded towards zero. The × operator is sometimes omitted if the meaning is clear. In figures these operators may appear inside square boxes.

The operator *mod* indicates the remainder from division; x *mod* y is the remainder when x is divided by y. It is only used with both operands being integers and y greater than zero. The division is rounded down so that the remainder is always non-negative; in other words, x *mod* y always equals $x - \lfloor x \div y \rfloor \times y$.

The notation “(*mod* m)” at the end of an equation indicates that the equality is tested after applying the *mod* operator on each side; in other words, “a = b (*mod* m)” is

equivalent to “a *mod* m = b *mod* m”. The three-way inequality “a ≤ b < c (*mod* m)”—with both relational operators in the same direction but each either including or excluding equality—is equivalent to “0 ≤ (b - a) *mod* m < (c - a) *mod* m”.

The operators + and – have equal precedence. The operators ×, ÷, and *mod* have equal precedence, which is higher than that of + and −. Exponentiation has higher precedence than all of these operators. Expressions involving operators of equal precedence are evaluated from left to right. Precedence can be overridden using parentheses; unless clear from the context, brackets and braces are equivalent to parentheses and are used for clarity. Exponents and subscripts are evaluated separately as if they were in parentheses; for example, $x^{2-y}$ means x raised to the power 2 − y, not the square of x with y then subtracted.

Signed integers shall be represented as two’s complement.

Where a field, variable, parameter, or similar is specified as an integer with a specific number of bits B, then any requirement to store a value not representable in B bits shall be carried out by adding or subtracting $2^B$ until the value is in range. In particular, adding 1 to an unsigned variable with value $2^B - 1$ results in setting it to zero.

The relational operators <, >, ≤, ≥, =, and ≠ have lower precedence than all arithmetic operators. Multiple relational operators are logically combined; e.g., x < y ≥ z means that x is required to be less than y and that y is required to be at least as great as z.

The bitwise operators NOT, AND, OR, and XOR are applied to each bit of their operands separately; for the last three, the operands will have the same number of bits.

The symbols in Table 2.1 have the meanings given in that table.

| Symbol | Meaning |
|---|---|
| $\lfloor x \rfloor$ | the largest integer less than or equal to x |
| $\lceil x \rceil$ | the smallest integer greater than or equal to x |
| $\lvert x \rvert$ | the absolute value of x |
| $\sqrt{x}$ | the square root of x |
| min (x, y) | the minimum value of x and y; there can be more than two arguments |
| max (x, y) | the maximum value of x and y; there can be more than two arguments |
| Re (z) | the real part of the complex number z |
| Im (z) | the imaginary part of the complex number z |
| exp (x) | e (the number 2.71828...) raised to the power of x |
| $x \pm y$ | any value v such that x − y ≤ v ≤ x + y; the precedence is the same as the + operator |

| Symbol | Meaning |
|---|---|
| $\log_B x$ | logarithm base B of x |
| ln x *or* ln(x) | $\log_e x$ |
| x \|\| y | concatenation of bit sequences |
| x! | the factorial of x |
| $\oplus$ | bitwise XOR |
| $\subseteq$ | subset of |

*Table 2.1: Assorted mathematical symbols*

## 2.11 Requirement status symbols

In this document (such as in requirements tables), the following symbols are used:

- “M” for mandatory (see Section 1).
- “O” for optional (see Section 1).
- “E” for excluded. "Excluded" means not permitted in this context; cannot be supported or included for this purpose. The item can still be supported or included if allowed for some other purpose (e.g., a feature can be mandatory for one role and excluded for another; a device that supports both roles must support this feature).
- “C.#” for conditional. “Conditional” means that an item is required, optional, or prohibited based on whether one or more other items are supported or included (# represents any number). Within the definition of the condition, if those other items mean that “not permitted” applies, it has the same meaning as “E”.
- “X” for reserved for future use; excluded in this context but might change status in a future version of this document.

## 2.12 Table structure

A blank cell in a table indicates that there is no useful information that can be placed in this cell. Examples of this are:

- When there is no comment to make in a "comments" column
- In a column specifying properties when the relevant item is reserved for future use (and therefore does not have any properties)
- In a "units" column when the relevant item is unitless

Where an explicit absence is indicated, the cell will contain *"none"*. Examples of this are:

- In the "condition" column of the description of a finite state machine where a rule is unconditional
- In the "action" column of the description of a finite state machine where a rule has no action
- In a "restrictions" column where there are no applicable restrictions
- In an interface description where there are no parameters of a specific type

## 2.13 References to HCI commands and events

Outside Volume 4, references to HCI commands and events indicate a possible way of communicating between the Controller and Host and do not, of themselves, require HCI to be supported. Instead, vendor-specific communication mechanisms may be used when HCI is not used.

# 3 NAMING CONVENTIONS

## 3.1 BR/EDR

This section is not currently used.

## 3.2 Bluetooth Low Energy

### 3.2.1 Link Layer PDUs

A consistent naming scheme is used for Link Layer PDUs to make their purpose and usage clearer.

The PDU name consists of up to five components. Each component is entirely uppercase. Those components that are present are separated by a single underscore (e.g., if only three of the five components are present, there are two underscores, not four). In order, these components are:

1. Where the PDU is used (optional)
2. When the PDU is used (mandatory)
3. What the PDU does (optional but usually present)
4. Version (optional)
5. How the PDU is used (mandatory)

The first component ("where") indicates which physical channel the PDU is used on. The values currently used are shown in Table 3.1.

| Value | Meaning |
|---|---|
| *none* | PDU is used on the primary advertising physical channel or any non-advertising physical channel |
| AUX | PDU is used on the secondary advertising physical channel |

*Table 3.1: First component (“where”) values*

The second component ("when") indicates which kind of Link Layer procedure makes use of the PDU. The values currently used are shown in Table 3.2.

| Value | Meaning |
|---|---|
| ADV | Normal Advertising |
| SYNC | Periodic Advertising |
| SCAN | Scanning |

| Value | Meaning |
|---|---|
| CONNECT | Connecting |
| CHAIN | Fragmented Data |
| LL | Control PDU on the Data logical transport |
| BIG | Control PDU in a Broadcast Isochronous Group on the isochronous logical transport |
| DATA | Reliable Data [Note 1] |
| CIS | Isochronous data PDU in a Connected Isochronous Stream |
| BIS | Isochronous data PDU in a Broadcast Isochronous Stream |

*Table 3.2: Second component (“when”) values*

[Note 1] This name is not currently used in the specification.

The third component ("what") distinguishes different PDUs that are found in the same context. While it is normally present, it is sometimes omitted where there is a "default" or "usual" case. This component may contain more than one word separated by underlines.

For example, the different cases for legacy advertising are shown in Table 3.3.

| Value | Meaning |
|---|---|
| *none* | Various |
| DIRECT | Connectable directed |
| NONCONN | Non-connectable and non-scannable undirected |
| SCAN | Scannable undirected |

*Table 3.3: Third component (“what”) values for legacy advertising*

As another example, each Link Layer PDU has a value for this component based on the specific procedure it is used for.

The fourth component ("version") distinguishes between PDUs with the same purpose but different contents (usually because the original PDU was found to be insufficient to handle new features). The values currently used are shown in Table 3.4.

| Value | Meaning |
|---|---|
| *none* | Original version of the PDU |
| EXT | Extended version of the PDU |

*Table 3.4: Fourth component (“version”) values*

The fifth and final component ("how") indicates how the PDU fits into a procedure. The values currently used are shown in Table 3.5.

| Value | Meaning |
|---|---|
| IND | An indication that doesn't expect a reply |
| REQ | A request that expects a response |
| RSP | A response to a request |

*Table 3.5: Fifth component ("how") values*

Some examples of this convention in use, showing how the PDU name breaks up into the five components, are given in Table 3.6. A blank cell indicates that the component is omitted.

| PDU name | Components | | | | |
|---|---|---|---|---|---|
| | where | when | what | version | how |
| ADV_IND | | ADV | | | IND |
| ADV_DIRECT_IND | | ADV | DIRECT | | IND |
| ADV_EXT_IND | | ADV | | EXT | IND |
| AUX_ADV_IND | AUX | ADV | | | IND |
| AUX_CHAIN_IND | AUX | CHAIN | | | IND |
| SCAN_REQ | | SCAN | | | REQ |
| AUX_SYNC_IND | AUX | SYNC | | | IND |
| LL_PHY_UPDATE_IND | | LL | PHY_UPDATE | | IND |
| LL_LENGTH_REQ | | LL | LENGTH | | REQ |
| LL_LENGTH_RSP | | LL | LENGTH | | RSP |
| LL_REJECT_IND | | LL | REJECT | | IND |
| LL_REJECT_EXT_IND | | LL | REJECT | EXT | IND |
| BIG_CHANNEL_MAP_IND | | BIG | CHANNEL_MAP | | IND |

*Table 3.6: Examples*

Thus AUX_SYNC_IND is a PDU used for synchronous advertising on the secondary advertising physical channel that does not expect a response.

# 4 SAMPLE DATA

All sample data are provided for reference purpose only; they are intended as a complement to the behavior described elsewhere in the specification. They can be used to check the behavior of an implementation and avoid misunderstandings.

These sample data have been checked against the specification and various implementations. Any discrepancies should be reported by filing an erratum; until such an erratum is resolved, implementers should assume that the data, rather than the specification, are erroneous.

This version of the specification contains sample data in:

- [Vol 2] Part G
- [Vol 3] Part H, Appendix D
- [Vol 6] Part C

# CONTROLLER ERROR CODES

# CONTENTS

**1 Overview of error codes ..... 431**
1.1 Usage descriptions ..... 431
1.2 [This section is no longer used] ..... 431
1.3 List of error codes ..... 431

**2 Error code descriptions ..... 434**
2.1 Unknown HCI Command (0x01) ..... 434
2.2 Unknown Connection Identifier (0x02) ..... 434
2.3 Hardware Failure (0x03) ..... 434
2.4 Page Timeout (0x04) ..... 434
2.5 Authentication Failure (0x05) ..... 434
2.6 PIN or Key Missing (0x06) ..... 434
2.7 Memory Capacity Exceeded (0x07) ..... 434
2.8 Connection Timeout (0x08) ..... 435
2.9 Connection Limit Exceeded (0x09) ..... 435
2.10 Synchronous Connection Limit to a Device Exceeded (0x0A) ..... 435
2.11 Connection Already Exists (0x0B) ..... 435
2.12 Command Disallowed (0x0C) ..... 435
2.13 Rejected due to Limited Resources (0x0D) ..... 435
2.14 Rejected due to Security Reasons (0x0E) ..... 435
2.15 Rejected due to Unacceptable BD_ADDR (0x0F) ..... 436
2.16 Connection Accept Timeout Exceeded (0x10) ..... 436
2.17 Unsupported Feature or Parameter Value (0x11) ..... 436
2.18 Invalid HCI Command Parameters (0x12) ..... 436
2.19 Remote User Terminated Connection (0x13) ..... 436
2.20 Remote Device Terminated Connection due to Low Resources (0x14) ..... 437
2.21 Remote Device Terminated Connection due to Power Off (0x15) ..... 437
2.22 Connection Terminated by Local Host (0x16) ..... 437
2.23 Repeated Attempts (0x17) ..... 437
2.24 Pairing Not Allowed (0x18) ..... 437
2.25 Unknown LMP PDU (0x19) ..... 437
2.26 Unsupported Remote Feature (0x1A) ..... 437
2.27 SCO Offset Rejected (0x1B) ..... 437
2.28 SCO Interval Rejected (0x1C) ..... 438
2.29 SCO Air Mode Rejected (0x1D) ..... 438
2.30 Invalid LMP Parameters / Invalid LL Parameters (0x1E) ..... 438
2.31 Unspecified Error (0x1F) ..... 438

2.32 Unsupported LMP Parameter Value / Unsupported LL Parameter Value (0x20) ..... 438
2.33 Role Change Not Allowed (0x21) ..... 438
2.34 LMP Response Timeout / LL Response Timeout (0x22) ..... 439
2.35 LMP Error Transaction Collision / LL Procedure Collision (0x23) 439
2.36 LMP PDU Not Allowed (0x24) ..... 439
2.37 Encryption Mode Not Acceptable (0x25) ..... 439
2.38 Link Key cannot be Changed (0x26) ..... 439
2.39 Requested QoS Not Supported (0x27) ..... 439
2.40 Instant Passed (0x28) ..... 439
2.41 Pairing with Unit Key Not Supported (0x29) ..... 439
2.42 Different Transaction Collision (0x2A) ..... 440
2.43 QoS Unacceptable Parameter (0x2C) ..... 440
2.44 QoS Rejected (0x2D) ..... 440
2.45 Channel Assessment Not Supported (0x2E) ..... 440
2.46 Insufficient Security (0x2F) ..... 440
2.47 Parameter Out of Mandatory Range (0x30) ..... 440
2.48 Role Switch Pending (0x32) ..... 440
2.49 Reserved Slot Violation (0x34) ..... 440
2.50 Role Switch Failed (0x35) ..... 441
2.51 Extended Inquiry Response Too Large (0x36) ..... 441
2.52 Secure Simple Pairing Not Supported by Host (0x37) ..... 441
2.53 Host Busy - Pairing (0x38) ..... 441
2.54 Rejected due to No Suitable Channel Found (0x39) ..... 441
2.55 Controller Busy (0x3A) ..... 441
2.56 Unacceptable Connection Parameters (0x3B) ..... 441
2.57 Advertising Timeout (0x3C) ..... 442
2.58 Connection Terminated due to MIC Failure (0x3D) ..... 442
2.59 Connection Failed to be Established / Synchronization Timeout (0x3E) ..... 442
2.60 [This section is no longer used] ..... 442
2.61 Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging (0x40) ..... 442
2.62 Type0 Submap Not Defined (0x41) ..... 442
2.63 Unknown Advertising Identifier (0x42) ..... 442
2.64 Limit Reached (0x43) ..... 443
2.65 Operation Cancelled by Host (0x44) ..... 443
2.66 Packet Too Long (0x45) ..... 443
2.67 Too Late (0x46) ..... 443
2.68 Too Early (0x47) ..... 443
2.69 Insufficient Channels (0x48) ..... 443

# 1 OVERVIEW OF ERROR CODES

This document lists the various possible error codes. When a command fails, or an LMP or LL message needs to indicate a failure, error codes are used to indicate the reason for the error. Error codes have a size of one octet.

## 1.1 Usage descriptions

The purpose of this section is to give descriptions of how the error codes should be used. It is beyond the scope of this document to give detailed descriptions of all situations where error codes can be used, especially as this is implementation dependent.

## 1.2 [This section is no longer used]

## 1.3 List of error codes

The error code of 0x00 means *Success*. The possible range of failure error codes is 0x01 to 0xFF. Section 2 provides an error code usage description for each failure error code.

Values marked as “Reserved for future use” or not listed in Table 1.1 can be used in future versions of the specification. A Host shall consider any error code that it does not explicitly understand equivalent to the error code *Unspecified Error* (0x1F).

| Error Code | Name |
|---|---|
| 0x00 | Success |
| 0x01 | Unknown HCI Command |
| 0x02 | Unknown Connection Identifier |
| 0x03 | Hardware Failure |
| 0x04 | Page Timeout |
| 0x05 | Authentication Failure |
| 0x06 | PIN or Key Missing |
| 0x07 | Memory Capacity Exceeded |
| 0x08 | Connection Timeout |
| 0x09 | Connection Limit Exceeded |
| 0x0A | Synchronous Connection Limit to a Device Exceeded |
| 0x0B | Connection Already Exists |
| 0x0C | Command Disallowed |

| Error Code | Name |
|---|---|
| 0x0D | Rejected due to Limited Resources |
| 0x0E | Rejected due to Security Reasons |
| 0x0F | Rejected due to Unacceptable BD_ADDR |
| 0x10 | Connection Accept Timeout Exceeded |
| 0x11 | Unsupported Feature or Parameter Value |
| 0x12 | Invalid HCI Command Parameters |
| 0x13 | Remote User Terminated Connection |
| 0x14 | Remote Device Terminated Connection due to Low Resources |
| 0x15 | Remote Device Terminated Connection due to Power Off |
| 0x16 | Connection Terminated by Local Host |
| 0x17 | Repeated Attempts |
| 0x18 | Pairing Not Allowed |
| 0x19 | Unknown LMP PDU |
| 0x1A | Unsupported Remote Feature |
| 0x1B | SCO Offset Rejected |
| 0x1C | SCO Interval Rejected |
| 0x1D | SCO Air Mode Rejected |
| 0x1E | Invalid LMP Parameters / Invalid LL Parameters |
| 0x1F | Unspecified Error |
| 0x20 | Unsupported LMP Parameter Value / Unsupported LL Parameter Value |
| 0x21 | Role Change Not Allowed |
| 0x22 | LMP Response Timeout / LL Response Timeout |
| 0x23 | LMP Error Transaction Collision / LL Procedure Collision |
| 0x24 | LMP PDU Not Allowed |
| 0x25 | Encryption Mode Not Acceptable |
| 0x26 | Link Key cannot be Changed |
| 0x27 | Requested QoS Not Supported |
| 0x28 | Instant Passed |
| 0x29 | Pairing With Unit Key Not Supported |
| 0x2A | Different Transaction Collision |
| 0x2B | Reserved for future use |
| 0x2C | QoS Unacceptable Parameter |
| 0x2D | QoS Rejected |

| Error Code | Name |
|---|---|
| 0x2E | Channel Assessment Not Supported |
| 0x2F | Insufficient Security |
| 0x30 | Parameter Out of Mandatory Range |
| 0x31 | Reserved for future use |
| 0x32 | Role Switch Pending |
| 0x33 | Reserved for future use |
| 0x34 | Reserved Slot Violation |
| 0x35 | Role Switch Failed |
| 0x36 | Extended Inquiry Response Too Large |
| 0x37 | Secure Simple Pairing Not Supported by Host |
| 0x38 | Host Busy - Pairing |
| 0x39 | Rejected due to No Suitable Channel Found |
| 0x3A | Controller Busy |
| 0x3B | Unacceptable Connection Parameters |
| 0x3C | Advertising Timeout |
| 0x3D | Connection Terminated due to MIC Failure |
| 0x3E | Connection Failed to be Established / Synchronization Timeout |
| 0x3F | *Previously used* |
| 0x40 | Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging |
| 0x41 | Type0 Submap Not Defined |
| 0x42 | Unknown Advertising Identifier |
| 0x43 | Limit Reached |
| 0x44 | Operation Cancelled by Host |
| 0x45 | Packet Too Long |
| 0x46 | Too Late |
| 0x47 | Too Early |
| 0x48 | Insufficient Channels |

*Table 1.1: List of possible error codes*

# 2 ERROR CODE DESCRIPTIONS

## 2.1 Unknown HCI Command (0x01)

The *Unknown HCI Command* error code indicates that the Controller does not understand the HCI Command packet opcode that the Host sent. The opcode given might not correspond to any of the opcodes specified in this document, or any vendor-specific opcodes, or the command may have not been implemented.

## 2.2 Unknown Connection Identifier (0x02)

The *Unknown Connection Identifier* error code indicates that a command was sent from the Host that should identify a connection, but that connection does not exist or does not identify the correct type of connection.

## 2.3 Hardware Failure (0x03)

The *Hardware Failure* error code indicates to the Host that something in the Controller has failed in a manner that cannot be described with any other error code. The meaning implied with this error code is implementation dependent.

## 2.4 Page Timeout (0x04)

The *Page Timeout* error code indicates that a page timed out because of the Page Timeout configuration parameter. This error code shall only be used with the HCI_Remote_Name_Request and HCI_Create_Connection commands or with equivalent mechanisms when HCI is not supported.

## 2.5 Authentication Failure (0x05)

The *Authentication Failure* error code indicates that pairing or authentication failed due to incorrect results in the pairing or authentication procedure. This could be due to an incorrect PIN or Link Key.

## 2.6 PIN or Key Missing (0x06)

The *PIN or Key Missing* error code is used when pairing failed because of a missing PIN, or authentication failed because of a missing Key.

## 2.7 Memory Capacity Exceeded (0x07)

The *Memory Capacity Exceeded* error code indicates to the Host that the Controller has run out of memory to store new parameters.

## 2.8 Connection Timeout (0x08)

The *Connection Timeout* error code indicates that either the link supervision timeout has expired for a given connection or that the synchronization timeout has expired for a given broadcast.

## 2.9 Connection Limit Exceeded (0x09)

The *Connection Limit Exceeded* error code indicates that an attempt to create another connection failed because the Controller is already at its limit of the number of connections it can support. The number of connections a device can support is implementation dependent.

## 2.10 Synchronous Connection Limit to a Device Exceeded (0x0A)

The *Synchronous Connection Limit to a Device Exceeded* error code indicates that the Controller has reached the limit to the number of synchronous connections that can be achieved to a device. The number of synchronous connections a device can support is implementation dependent.

## 2.11 Connection Already Exists (0x0B)

The *Connection Already Exists* error code indicates that an attempt was made to create a new Connection to a device when there is already a connection to this device and multiple connections to the same device are not permitted.

## 2.12 Command Disallowed (0x0C)

The *Command Disallowed* error code indicates that the command requested cannot be executed because the Controller is in a state where it cannot process this command at this time. This error shall not be used for command opcodes where the error code *Unknown HCI Command* (0x01) is valid.

## 2.13 Rejected due to Limited Resources (0x0D)

The *Rejected due to Limited Resources* error code indicates that a connection or other request was rejected due to limited resources.

## 2.14 Rejected due to Security Reasons (0x0E)

The *Rejected due to Security Reasons* error code indicates that a connection or other request was rejected due to security requirements not being fulfilled, like authentication or pairing.

## 2.15 Rejected due to Unacceptable BD_ADDR (0x0F)

The *Rejected due to Unacceptable BD_ADDR* error code indicates that a connection or other request was rejected because this device does not accept the BD_ADDR. This may be because the device will only accept connections from specific BD_ADDRs.

## 2.16 Connection Accept Timeout Exceeded (0x10)

The *Connection Accept Timeout Exceeded* error code indicates that the Connection Accept Timeout has been exceeded for this connection attempt.

## 2.17 Unsupported Feature or Parameter Value (0x11)

The *Unsupported Feature or Parameter Value* error code indicates that a feature or parameter value in the HCI command is not supported. This error code shall not be used in an LMP PDU.

## 2.18 Invalid HCI Command Parameters (0x12)

The *Invalid HCI Command Parameters* error code indicates that at least one of the HCI command parameters is invalid.

This shall be used when:

- the parameter total length is invalid.
- a command parameter is an invalid type.
- a connection identifier does not match the corresponding event.
- a parameter is odd when it is required to be even.
- a parameter is outside of the specified range.
- two or more parameter values have inconsistent values.

Note: An invalid type can be, for example, when a SCO Connection_Handle is used where an ACL Connection_Handle is required.

## 2.19 Remote User Terminated Connection (0x13)

The *Remote User Terminated Connection* error code indicates that the user on the remote device either terminated the connection or stopped broadcasting packets.

## 2.20 Remote Device Terminated Connection due to Low Resources (0x14)

The *Remote Device Terminated Connection due to Low Resources* error code indicates that the remote device terminated the connection because of low resources.

## 2.21 Remote Device Terminated Connection due to Power Off (0x15)

The *Remote Device Terminated Connection due to Power Off* error code indicates that the remote device terminated the connection because the device is about to power off.

## 2.22 Connection Terminated by Local Host (0x16)

The *Connection Terminated by Local Host* error code indicates that either the local device terminated the connection, terminated synchronization with a broadcaster, or terminated broadcasting packets.

## 2.23 Repeated Attempts (0x17)

The *Repeated Attempts* error code indicates that the Controller is disallowing an authentication or pairing procedure because too little time has elapsed since the last authentication or pairing attempt failed.

## 2.24 Pairing Not Allowed (0x18)

The *Pairing Not Allowed* error code indicates that the device does not allow pairing. For example, when a device only allows pairing during a certain time window after some user input allows pairing.

## 2.25 Unknown LMP PDU (0x19)

The *Unknown LMP PDU* error code indicates that the Controller has received an unknown LMP opcode.

## 2.26 Unsupported Remote Feature (0x1A)

The *Unsupported Remote Feature* error code indicates that the remote device does not support the feature associated with the issued command, LMP PDU, or Link Layer Control PDU.

## 2.27 SCO Offset Rejected (0x1B)

The *SCO Offset Rejected* error code indicates that the offset requested in the LMP_SCO_LINK_REQ PDU has been rejected.

## 2.28 SCO Interval Rejected (0x1C)

The *SCO Interval Rejected* error code indicates that the interval requested in the LMP_SCO_LINK_REQ PDU has been rejected.

## 2.29 SCO Air Mode Rejected (0x1D)

The *SCO Air Mode Rejected* error code indicates that the air mode requested in the LMP_SCO_LINK_REQ PDU has been rejected.

## 2.30 Invalid LMP Parameters / Invalid LL Parameters (0x1E)

The *Invalid LMP Parameters / Invalid LL Parameters* error code indicates that some LMP PDU / LL Control PDU parameters were invalid. This shall be used when:

- the PDU length is invalid.
- a parameter is odd when it is required to be even.
- a parameter is outside of the specified range.
- two or more parameters have inconsistent values.

## 2.31 Unspecified Error (0x1F)

The *Unspecified Error* error code indicates that no other error code specified is appropriate to use.

## 2.32 Unsupported LMP Parameter Value / Unsupported LL Parameter Value (0x20)

The *Unsupported LMP Parameter Value / Unsupported LL Parameter Value* error code indicates that an LMP PDU or an LL Control PDU contains at least one parameter value that is not supported by the Controller at this time. This is normally used after a long negotiation procedure, for example during an LMP_HOLD_REQ, LMP_SNIFF_REQ and LMP_ENCRYPTION_KEY_SIZE_REQ PDU exchanges. This may be used by the Link Layer, for example during the Connection Parameters Request Link Layer Control procedure.

## 2.33 Role Change Not Allowed (0x21)

The *Role Change Not Allowed* error code indicates that a Controller will not allow a role change at this time.

## 2.34 LMP Response Timeout / LL Response Timeout (0x22)

The *LMP Response Timeout / LL Response Timeout* error code indicates that an LMP transaction failed to respond within the LMP response timeout or an LL transaction failed to respond within the LL response timeout.

## 2.35 LMP Error Transaction Collision / LL Procedure Collision (0x23)

The *LMP Error Transaction Collision / LL Procedure Collision* error code indicates that an LMP transaction or LL procedure has collided with the same transaction or procedure that is already in progress.

## 2.36 LMP PDU Not Allowed (0x24)

The *LMP PDU Not Allowed* error code indicates that a Controller sent an LMP PDU with an opcode that was not allowed.

## 2.37 Encryption Mode Not Acceptable (0x25)

The *Encryption Mode Not Acceptable* error code indicates that the requested encryption mode is not acceptable at this time.

## 2.38 Link Key cannot be Changed (0x26)

The *Link Key cannot be Changed* error code indicates that a link key cannot be changed because a fixed unit key is being used.

## 2.39 Requested QoS Not Supported (0x27)

The *Requested QoS Not Supported* error code indicates that the requested Quality of Service is not supported.

## 2.40 Instant Passed (0x28)

The *Instant Passed* error code indicates that an LMP PDU or LL PDU that includes an instant cannot be performed because the instant when this would have occurred has passed.

## 2.41 Pairing with Unit Key Not Supported (0x29)

The *Pairing With Unit Key Not Supported* error code indicates that it was not possible to pair as a unit key was requested and it is not supported.

## 2.42 Different Transaction Collision (0x2A)

The *Different Transaction Collision* error code indicates that an LMP transaction or LL Procedure was started that collides with an ongoing transaction.

## 2.43 QoS Unacceptable Parameter (0x2C)

The *QoS Unacceptable Parameter* error code indicates that the specified quality of service parameters could not be accepted at this time, but other parameters may be acceptable.

## 2.44 QoS Rejected (0x2D)

The *QoS Rejected* error code indicates that the specified quality of service parameters cannot be accepted and QoS negotiation should be terminated.

## 2.45 Channel Assessment Not Supported (0x2E)

The *Channel Assessment Not Supported* error code indicates that the Controller cannot perform channel assessment because it is not supported.

## 2.46 Insufficient Security (0x2F)

The *Insufficient Security* error code indicates that the HCI command or LMP PDU sent is only possible on an encrypted link.

## 2.47 Parameter Out of Mandatory Range (0x30)

The *Parameter Out of Mandatory Range* error code indicates that a parameter value requested is outside the mandatory range of parameters for the given HCI command or LMP PDU and the recipient does not accept that value.

## 2.48 Role Switch Pending (0x32)

The *Role Switch Pending* error code indicates that a Role Switch is pending. This can be used when an HCI command or LMP PDU cannot be accepted because of a pending role switch. This can also be used to notify a peer device about a pending role switch.

## 2.49 Reserved Slot Violation (0x34)

The *Reserved Slot Violation* error code indicates that the current Synchronous negotiation was terminated with the negotiation state set to Reserved Slot Violation.

## 2.50 Role Switch Failed (0x35)

The *Role Switch Failed* error code indicates that a role switch was attempted but it failed and the original piconet structure is restored. The switch may have failed because the TDD switch or piconet switch failed.

## 2.51 Extended Inquiry Response Too Large (0x36)

The *Extended Inquiry Response Too Large* error code indicates that the extended inquiry response, with the requested requirements for FEC, is too large to fit in any of the packet types supported by the Controller.

## 2.52 Secure Simple Pairing Not Supported by Host (0x37)

The *Secure Simple Pairing Not Supported by Host* error code indicates that the IO capabilities request or response was rejected because the sending Host does not support Secure Simple Pairing even though the receiving Link Manager does.

## 2.53 Host Busy - Pairing (0x38)

The *Host Busy - Pairing* error code indicates that the Host is busy with another pairing operation and unable to support the requested pairing. The receiving device should retry pairing again later.

## 2.54 Rejected due to No Suitable Channel Found (0x39)

The *Rejected due to No Suitable Channel Found* error code indicates that the Controller could not calculate an appropriate value for the Channel selection operation.

## 2.55 Controller Busy (0x3A)

The *Controller Busy* error code indicates that the operation was rejected because the Controller was busy and unable to process the request.

## 2.56 Unacceptable Connection Parameters (0x3B)

The *Unacceptable Connection Parameters* error code indicates that the remote device either terminated the connection or rejected a request because of one or more unacceptable connection parameters.

## 2.57 Advertising Timeout (0x3C)[1]

The *Advertising Timeout* error code indicates that advertising for a fixed duration completed or, for directed advertising, that advertising completed without a connection being created.

## 2.58 Connection Terminated due to MIC Failure (0x3D)

The *Connection Terminated due to MIC Failure* error code indicates that either the connection or the synchronization was terminated because the Message Integrity Check (MIC) failed on a received packet.

## 2.59 Connection Failed to be Established / Synchronization Timeout (0x3E)

The *Connection Failed to be Established / Synchronization Timeout* error code indicates that the LL initiated a connection or initiated synchronization but the connection has failed to be established or the Link Layer failed to synchronize within the specified time.

## 2.60 [This section is no longer used]

## 2.61 Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging (0x40)

The *Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging* error code indicates that the Central, at this time, is unable to make a coarse adjustment to the piconet clock, using the supplied parameters. Instead the Central will attempt to move the clock using clock dragging.

## 2.62 Type0 Submap Not Defined (0x41)

The *Type0 Submap Not Defined* error code indicates that the LMP PDU is rejected because the Type 0 submap is not currently defined.

## 2.63 Unknown Advertising Identifier (0x42)

The *Unknown Advertising Identifier* error code indicates that a command was sent from the Host that should identify an Advertising or Sync handle, but the Advertising or Sync handle does not exist.

[1]Formerly called Directed Advertising Timeout

## 2.64 Limit Reached (0x43)

The *Limit Reached* error code indicates that number of operations requested has been reached and has indicated the completion of the activity (e.g., advertising or scanning).

## 2.65 Operation Cancelled by Host (0x44)

The *Operation Cancelled by Host* error code indicates a request to the Controller issued by the Host and still pending was successfully canceled.

## 2.66 Packet Too Long (0x45)

The *Packet Too Long* error code indicates that an attempt was made to send or receive a packet that exceeds the maximum allowed packet length.

## 2.67 Too Late (0x46)

The *Too Late* error code indicates that information was provided too late to the Controller.

## 2.68 Too Early (0x47)

The *Too Early* error code indicates that information was provided too early to the Controller.

## 2.69 Insufficient Channels (0x48)

The *Insufficient Channels* error code indicates that the result of the requested operation would yield too few physical channels.

# BR/EDR Controller

**Specification of the *Bluetooth*® System**

**Volume 2**

**Covered Core Package Version: v6.3**

**Version Date: 2026-05-05**

BR/EDR Controller
Part A

# RADIO PHYSICAL LAYER SPECIFICATION

*This Part describes the BR/EDR radio physical layer.*

# CONTENTS

**1 Scope ..... 448**
1.1 Requirements ..... 448
1.1.1 π/4-DQPSK modulation ..... 449
1.1.2 8DPSK modulation ..... 449
1.1.3 3-slot packets ..... 449
1.1.4 5-slot packets ..... 450
1.1.5 Power control ..... 450
1.1.6 Enhanced power control ..... 450

**2 Frequency bands and channel arrangement ..... 451**

**3 Transmitter characteristics ..... 452**
3.1 Basic Rate ..... 452
3.1.1 Modulation characteristics ..... 452
3.1.2 Spurious emissions ..... 453
3.1.2.1 In-band spurious emission ..... 453
3.1.3 Radio frequency tolerance ..... 453
3.2 Enhanced Data Rate ..... 454
3.2.1 Modulation characteristics ..... 454
3.2.1.1 Modulation method overview ..... 454
3.2.1.2 Differential phase encoding ..... 455
3.2.1.3 Pulse shaping ..... 455
3.2.1.4 Modulation accuracy ..... 456
3.2.2 Spurious emissions ..... 456
3.2.2.1 In-band spurious emission ..... 457
3.2.3 Radio frequency tolerance ..... 457
3.2.4 Relative transmit power ..... 458

**4 Receiver characteristics ..... 459**
4.1 Basic Rate ..... 459
4.1.1 Actual sensitivity level ..... 459
4.1.2 Interference performance ..... 459
4.1.3 Out-of-band blocking ..... 460
4.1.4 Intermodulation characteristics ..... 460
4.1.5 Maximum usable level ..... 461
4.1.6 Received Signal Strength Indication ..... 461
4.1.7 Reference signal definition ..... 461
4.2 Enhanced Data Rate ..... 461
4.2.1 Actual sensitivity level ..... 461
4.2.2 BER floor performance ..... 461

4.2.3 Interference performance ..... 461
4.2.4 Maximum usable level ..... 462
4.2.5 Out-of-band and intermodulation characteristics ..... 462
4.2.6 Reference signal definition ..... 463

**5 Power management ..... 464**
5.1 Power classes ..... 464
5.2 Power control ..... 464
5.3 Enhanced power control ..... 465

**Appendix A Test conditions ..... 466**
A.1 Nominal test conditions ..... 466
A.1.1 Nominal temperature ..... 466
A.1.2 Nominal power source ..... 466
A.2 [This section is no longer used] ..... 466

**Appendix B [This Appendix is no longer used] ..... 467**

**Appendix C Modulation accuracy definition ..... 468**
C.1 Enhanced Data Rate modulation accuracy ..... 468
C.1.1 RMS DEVM ..... 470
C.1.2 Peak DEVM ..... 470

# 1 SCOPE

Bluetooth devices operate in the unlicensed 2.4 GHz ISM (Industrial Scientific Medical) band. A frequency hop transceiver is applied to combat interference and fading.

Two modulation modes are defined. A mandatory mode, called Basic Rate, uses a shaped, binary FM modulation to minimize transceiver complexity. An optional mode, called Enhanced Data Rate, uses PSK modulation and has two variants: π/4-DQPSK and 8DPSK. The symbol rate for all modulation modes is 1 Msym/s. The gross air data rate is 1 Mb/s for Basic Rate, 2 Mb/s for Enhanced Data Rate using π/4-DQPSK and 3 Mb/s for Enhanced Data Rate using 8DPSK.

A Time Division Duplex (TDD) scheme is used in both modes. The specification defines the requirements for a Bluetooth radio for the Basic Rate and Enhanced Data Rate modes.

Requirements are defined for two reasons:

- Provide compatibility between radios used in the system
- Define the quality of the system

The Bluetooth radio shall fulfil the stated requirements under the operating conditions specified in Appendix A. The radio parameters shall be measured according to the methods described in the RF Test Specification.

The Bluetooth SIG maintains regulatory content associated with Bluetooth technology in the 2.4 GHz ISM band on its web site, at https://www.bluetooth.com/regulatory-requirements/.

## 1.1 Requirements

A device supporting RF shall meet the requirements of Section 3 for any packet it is able to transmit and the requirements of Section 4 for any packet it is able to receive.

The device shall be able to transmit and receive GFSK packets, according to the requirements in Section 3.1 and Section 4.1, of any length between 68 µs and 426 µs.

The device shall be able to transmit and receive any packet type it supports on all of the 79 channels specified in Section 2.

RF has the following optional features:

- π/4-DQPSK modulation
- 8DPSK modulation
- 3-slot packets
- 5-slot packets
- Power control
- Enhanced power control
- Power class 1 (see Section 5.1)

For each row of Table 1.1, if a device supports the feature named in the first column then that device shall also support the feature named in the second column.

| Feature | Required feature |
|---|---|
| 8DPSK modulation | π/4-DQPSK modulation |
| 5-slot packets | 3-slot packets |
| Enhanced power control | Power control |
| Power class 1 | Power control |

*Table 1.1: Features that require other features*

### 1.1.1 π/4-DQPSK modulation

A device that supports π/4-DQPSK modulation shall be able to transmit and receive packets using π/4-DQPSK modulation (Enhanced Data Rate 2 Mb/s packets) according to the requirements in Section 3.2 and Section 4.2 with a payload field of up to 248 symbols.

### 1.1.2 8DPSK modulation

A device that supports 8DPSK modulation shall be able to transmit and receive packets using 8DPSK modulation (Enhanced Data Rate 3 Mb/s packets) according to the requirements in Section 3.2 and Section 4.2 with a payload field of up to 246 symbols.

### 1.1.3 3-slot packets

A device that supports 3-slot packets shall be able to transmit and receive Basic Rate packets of any length between 68 µs and 1686 µs and, if supported, Enhanced Data Rate packets with a payload field of up to 1500 symbols.

### 1.1.4 5-slot packets

A device that supports 5-slot packets shall be able to transmit and receive Basic Rate packets of any length between 68 µs and 2916 µs and, if supported, Enhanced Data Rate packets with a payload field of up to 2748 symbols.

### 1.1.5 Power control

A device that supports power control shall be capable of adjusting its transmission power, shall be capable of responding to legacy power control requests (see [Vol 2] Part C, Section 4.1.3), and shall meet the requirements in Section 5.2.

### 1.1.6 Enhanced power control

A device that supports enhanced power control shall be capable of responding to enhanced power control requests (see [Vol 2] Part C, Section 4.1.3.1) and shall meet the requirements in Section 5.3.

# 2 FREQUENCY BANDS AND CHANNEL ARRANGEMENT

The Bluetooth system operates in the 2.4 GHz ISM band. This frequency band is 2400 MHz to 2483.5 MHz.

| Frequency Range | RF Channels |
|---|---|
| 2.400 GHz to 2.4835 GHz | f=2402+k MHz, k=0,...,78 |

*Table 2.1: Operating frequency bands*

RF channels are spaced 1 MHz and are ordered in channel number k as shown in Table 2.1.

# 3 TRANSMITTER CHARACTERISTICS

The requirements stated in this section are given as power levels at the antenna connector of the Bluetooth device; this is also referred to as the radiative transmit power level of the device. If the device does not have a connector, a reference antenna with 0 dBi gain is assumed. Power level values used in HCI commands, HCI events, and Link Manager Protocol (LMP) PDUs shall be assumed to be the radiative transmit power level of the device unless specified otherwise.

Due to difficulty in measurement accuracy in radiated measurements, systems with an integral antenna should provide a temporary antenna connector during RF qualification testing.

## 3.1 Basic Rate

### 3.1.1 Modulation characteristics

The Modulation is GFSK (Gaussian Frequency Shift Keying) with a bandwidth-bit period product BT=0.5. The Modulation index shall be between 0.28 and 0.35. A binary one shall be represented by a positive frequency deviation, and a binary zero shall be represented by a negative frequency deviation. The symbol timing shall be less than ±20 ppm.

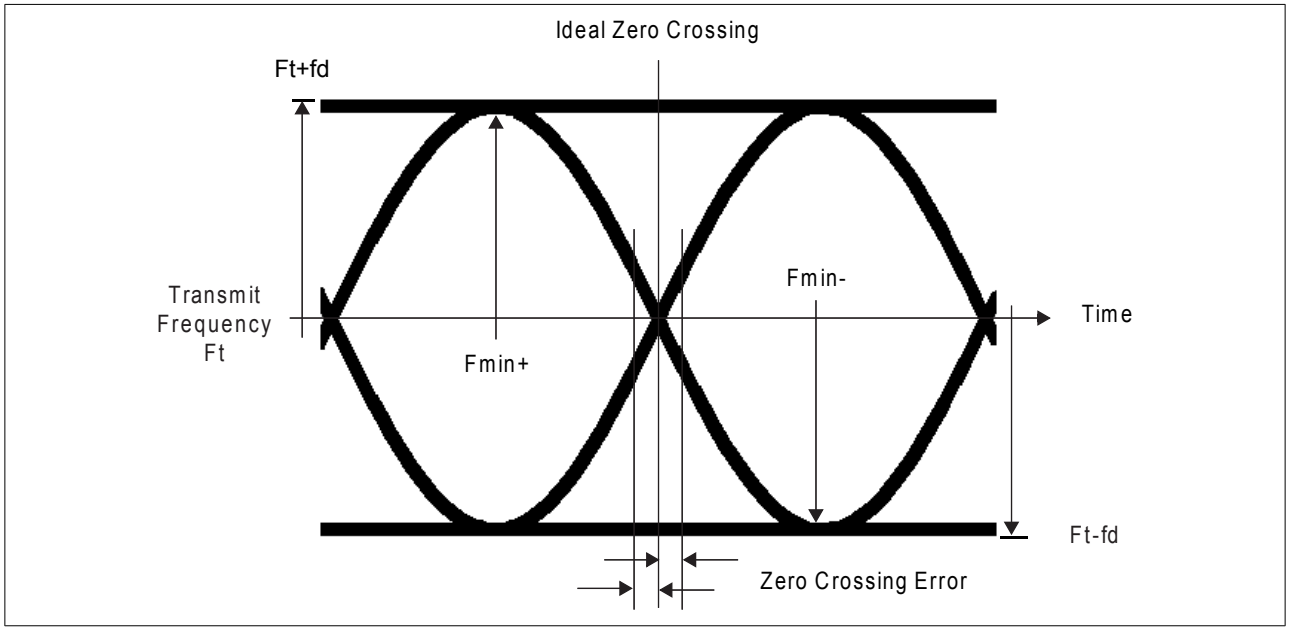



*Figure 3.1: GFSK parameters definition*

For each transmission, the minimum frequency deviation, Fmin = min{|Fmin+|, Fmin-}, which corresponds to 1010 sequence shall be no smaller than ±80% of the frequency

deviation (fd) with respect to the transmit frequency Ft, which corresponds to a 00001111 sequence.

In addition, the minimum frequency deviation shall never be smaller than 115 kHz. The data transmitted has a symbol rate of 1 Msym/s.

The zero crossing error is the time difference between the ideal symbol period and the measured crossing time. This shall be less than ± 1/8 of a symbol period.

See Figure 3.1 for the definitions of some symbols and terms in these requirements.

### 3.1.2 Spurious emissions

In-band spurious emissions shall be measured with a frequency hopping radio transmitting on one RF channel and receiving on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same transmit RF channel.

#### 3.1.2.1 In-band spurious emission

Within the ISM band the transmitter shall pass a spectrum mask, given in Table 3.1. An adjacent channel power is specified for channels at least 1 MHz from the carrier. This adjacent channel power is defined as the sum of the measured power in a 1 MHz bandwidth. The spectrum measurement shall be performed with a 100 kHz resolution bandwidth and an average detector. The device shall transmit on the RF channel with the center frequency M and the adjacent channel power shall be measured on a 1 MHz channel centered on RF frequency N. The transmitter shall transmit a pseudo-random data pattern in the payload throughout the test.

| Frequency offset | Transmit Power |
|---|---|
| 2 MHz (\|M-N\| = 2) | -20 dBm |
| 3 MHz or greater (\|M-N\| ≥ 3) | -30 dBm |

*Table 3.1: Transmit Spectrum mask*

Exceptions are allowed in up to three bands of 1 MHz width centered on a frequency which is an integer multiple of 1 MHz. They shall have a value of –20 dBm or less.

### 3.1.3 Radio frequency tolerance

The transmitted initial center frequency shall be within ±75 kHz from $F_c$. The initial frequency accuracy is defined as being the frequency accuracy before any packet information is transmitted.

The frequency drift requirement is not included in the ±75 kHz.

The limits on the transmitter center frequency drift within a packet are specified in Table 3.2. The different packets are defined in the Baseband Specification.

| Duration of Packet | Frequency Drift |
|---|---|
| Max length one slot packet | ±25 kHz |
| Max length three slot packet | ±40 kHz |
| Max length five slot packet | ±40 kHz |
| Maximum drift rate[1] | 400 Hz/µs |

*Table 3.2: Maximum allowable frequency drifts in a packet*

[1]The maximum drift rate is allowed anywhere in a packet.

## 3.2 Enhanced Data Rate

A key characteristic of the Enhanced Data Rate mode is that the modulation mode is changed within the packet. The access code and packet header, as defined in [Vol 2] Part B, Table 6.1, are transmitted with the Basic Rate 1 Mb/s GFSK modulation mode, whereas the subsequent synchronization sequence, payload, and trailer sequence are transmitted using the Enhanced Data Rate PSK modulation mode.

### 3.2.1 Modulation characteristics

During access code and packet header transmission the Basic Rate GFSK modulation mode shall be used. During the transmission of the synchronization sequence, payload, and trailer sequence a PSK type of modulation with a data rate of 2 Mb/s or optionally 3 Mb/s shall be used. The following subsections specify the PSK modulation for this transmission.

#### 3.2.1.1 Modulation method overview

The PSK modulation format defined for the 2 Mb/s transmission shall be π/4 rotated differential encoded quaternary phase shift keying (π/4-DQPSK).

The PSK modulation format defined for the 3 Mb/s transmission shall be differential encoded 8-ary phase shift keying (8DPSK).

The modulation shall employ square-root raised cosine pulse shaping to generate the equivalent lowpass information-bearing signal v(t). The output of the transmitter shall be a bandpass signal that can be represented as

$$S(t) = Re\left[v(t)e^{j2\pi F_c t}\right] \quad \text{(EQ 1)}$$

with $F_c$ denoting the carrier frequency.

#### 3.2.1.2 Differential phase encoding

For the M-ary modulation, the binary data stream {$b_n$}, n=1,2,3, ...N, shall be mapped onto a corresponding sequence {Sk}, k=1,2, ...N÷$\log_2$(M) of complex valued signal points. M=4 applies for 2 Mb/s and M=8 applies for 3 Mb/s. Gray coding shall be applied as shown in Table 3.3 and Table 3.4. In the event that the length of the binary data stream N is not an integer multiple of $\log_2$(M), the last symbol of the sequence {$S_k$} shall be formed by appending data zeros to the appropriate length. The signal points $S_k$ shall be defined by:

$$S_k = S_{k-1} e^{j\varphi_k} \qquad k = 1, 2, \ ..N \div log_2(M) \tag{EQ 2}$$

$$S_0 = e^{j\phi} \qquad \text{with } \phi \in [0, 2\pi) \tag{EQ 3}$$

The relationship between the binary input $b_k$ and the phase $\phi_k$ shall be as defined in Table 3.3 for the 2 Mb/s transmission and in Table 3.4 for the 3 Mb/s transmission.

| $b_{2k-1}$ | $b_{2k}$ | $\varphi_k$ |
|---|---|---|
| 0 | 0 | π/4 |
| 0 | 1 | 3π/4 |
| 1 | 1 | -3π/4 |
| 1 | 0 | -π/4 |

*Table 3.3: π/4-DQPSK mapping*

| $b_{3k-2}$ | $b_{3k-1}$ | 3k | $\varphi_k$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | π/4 |
| 0 | 1 | 1 | π/2 |
| 0 | 1 | 0 | 3π/4 |
| 1 | 1 | 0 | π |
| 1 | 1 | 1 | -3π/4 |
| 1 | 0 | 1 | -π/2 |
| 1 | 0 | 0 | -π/4 |

*Table 3.4: 8DPSK mapping*

#### 3.2.1.3 Pulse shaping

The lowpass equivalent information-bearing signal v(t) shall be generated according to

$$v(t) = \sum_k S_k p\left(t - kT\right) \tag{EQ 4}$$

in which the symbol period T shall be 1 µs.

The frequency spectrum P(f), which corresponds to the square-root raised cosine pulse p(t) of the pulse shaping filter is:

$$|P(f)| = \begin{cases} 1 & \text{when} \quad 0 \le |f| \le \frac{1-\beta}{2T} \\ \sqrt{\frac{1}{2}\left(1 - \sin\left(\frac{\pi(2|f|T-1)}{2\beta}\right)\right)} & \text{when} \quad \frac{1-\beta}{2T} \le |f| \le \frac{1+\beta}{2T} \\ 0 & \text{elsewhere} \end{cases} \quad \text{(EQ 5)}$$

The roll off factor β shall be 0.4.

#### 3.2.1.4 Modulation accuracy

The measurement of modulation accuracy uses differential error vector magnitude (DEVM) with tracking of the carrier frequency drift. The definition of DEVM and the derivation of the RMS and peak measures of DEVM are given in Appendix C.

The DEVM shall be measured over the synchronization sequence and payload portions of the packet, but not the trailer symbols. For each modulation method and each measurement carrier frequency, the DEVM measurement is made over a total of 200 non-overlapping blocks, where each block contains 50 symbols.

The transmitted packets shall be the longest supported packet type for each modulation method, as defined in [Vol 2] Part B, Table 6.8 and [Vol 2] Part B, Table 6.9.

The DEVM is measured using a square-root raised cosine filter, with a roll-off of 0.4 and a 3 dB bandwidth of ±500 kHz.

##### 3.2.1.4.1 RMS DEVM

The RMS DEVM for any of the measured blocks shall not exceed 0.20 for π/4-DQPSK and 0.13 for 8DPSK.

##### 3.2.1.4.2 99% DEVM

The 99% DEVM (defined as the DEVM value for which 99% of measured symbols have a lower DEVM) shall not exceed 0.30 for π/4-DQPSK and 0.20 for 8DPSK.

##### 3.2.1.4.3 Peak DEVM

The Peak DEVM shall not exceed 0.35 for π/4-DQPSK and 0.25 for 8DPSK.

### 3.2.2 Spurious emissions

In-band spurious emissions shall be measured with a frequency hopping radio transmitting on one RF channel and receiving on a second RF channel; this means

that the synthesizer may change RF channels between reception and transmission, but always returns to the same transmit RF channel.

#### 3.2.2.1 In-band spurious emission

Within the ISM band the power spectral density of the transmitter shall comply with the following requirements when sending pseudo random data. All power measurements shall use a 100 kHz resolution bandwidth and an average detector. The adjacent channel power for channels at least 2 MHz from the carrier is defined as the sum of the power measurements over a 1 MHz channel and shall not exceed -20 dBm for the second adjacent channels and -30 dBm for the third and subsequent adjacent channels.

Exceptions are allowed in up to 3 bands of 1 MHz width centered on a frequency which is an integer multiple of 1 MHz. They shall have a value of –20 dBm or less.

### 3.2.3 Radio frequency tolerance

The same carrier frequencies $F_c$ as used for Basic Rate transmissions shall be used for the Enhanced Data Rate transmissions. The transmitted initial center frequency accuracy shall be within ±75 kHz from $F_c$. The maximum excursion from $F_c$ (frequency offset plus drift) shall not exceed ±75 kHz.

The initial frequency accuracy is defined as being the frequency accuracy before any information is transmitted.

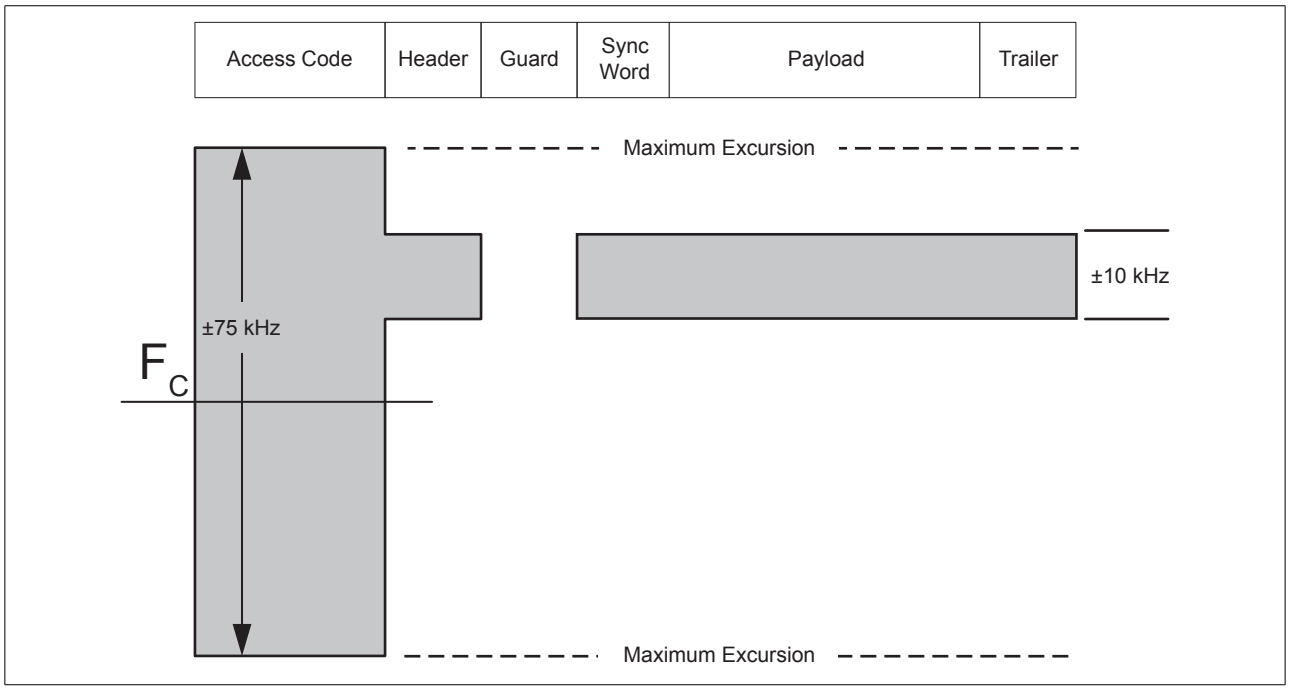



*Figure 3.2: Carrier frequency mask*

The requirements on accuracy and stability are illustrated by Figure 3.2 for the Enhanced Data Rate packet format defined in Definition: Baseband Specification . The higher frequency accuracy requirement shall start at the first symbol of the header. The maximum drift over the header, synchronization sequence and payload shall be ±10 kHz.

### 3.2.4 Relative transmit power

The requirement on the relative power of the GFSK and DPSK portions of the Enhanced Data Rate packet is defined as follows. The average power level during the transmission of access code and header is denoted as $P_{GFSK}$ and the average power level during the transmission of the synchronization sequence and the payload is denoted as $P_{DPSK}$. The following inequalities shall be satisfied independently for each Enhanced Data Rate packet transmitted:

$(P_{GFSK} - 4\ dB) < P_{DPSK} < (P_{GFSK} + 1\ dB)$

# 4 RECEIVER CHARACTERISTICS

The receiver characteristics shall be measured using loopback as defined in [Vol 3] Part D, Section 1.

The reference sensitivity level referred to in this section is -70 dBm.

## 4.1 Basic Rate

### 4.1.1 Actual sensitivity level

The actual sensitivity level is defined as the input level for which a raw bit error rate (BER) of 0.1% is met. The receiver sensitivity shall be below or equal to –70 dBm when receiving signals specified in Section 3.1.

### 4.1.2 Interference performance

The interference performance shall be measured with a wanted signal of -67 dBm. If the frequency of an interfering signal is outside of the band 2400 MHz to 2483.5 MHz, the out-of-band blocking specification (see Section 4.1.3) shall apply. The interfering signal shall be Bluetooth-modulated (see Section 4.1.7). The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 4.1:

| Frequency of Interference | Ratio |
|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 11 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$ | 0 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | -17 dB |
| Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$ | -27 dB |
| Image frequency Interference[1,2,3], $C/I_{Image}$ | -9 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image \pm 1MHz}$ | -15 dB |

*Table 4.1: Interference performance*

[1]If two adjacent channel specifications from Table 4.1 are applicable to the same channel, the more relaxed specification applies.

[2]In-band image frequency

[3]If the image frequency ≠ n MHz, then the image reference frequency is defined as the closest n MHz frequency for integer n.

These specifications are only to be tested at nominal temperature conditions with a device receiving on one RF channel and transmitting on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same receive RF channel.

RF channels where the requirements are not met are called spurious response RF channels. Five spurious response RF channels are allowed at RF channels with a distance of ≥2 MHz from the wanted signal. On these spurious response RF channels a relaxed interference requirement C/I = -17 dB shall be met.

### 4.1.3 Out-of-band blocking

The out-of-band suppression (or rejection) shall be measured with the wanted signal 3 dB over the reference sensitivity level. The interfering signal shall be a continuous wave signal. The BER shall be ≤0.1%. The out-of-band blocking shall fulfil the following requirements:

| Interfering Signal Frequency | Interfering Signal Power Level |
|---|---|
| 30 MHz to 2000 MHz | -10 dBm |
| 2000 MHz to 2399 MHz | -27 dBm |
| 2484 MHz to 3000 MHz | -27 dBm |
| 3000 MHz to 12.75 GHz | -10 dBm |

*Table 4.2: Out-of-band suppression (or rejection) requirements*

24 exceptions are permitted which are dependent upon the given RF channel and are centered at a frequency which is an integer multiple of 1 MHz. For at least 19 of these spurious response frequencies, a reduced interference level of at least -50 dBm is allowed in order to achieve the required BER = 0.1% performance, whereas for a maximum of 5 of the spurious frequencies the interference level may be assumed arbitrarily lower.

### 4.1.4 Intermodulation characteristics

The reference sensitivity performance, BER = 0.1%, shall be met under the following conditions:

- The wanted signal shall be at frequency $f_0$ with a power level 6 dB over the reference sensitivity level.
- A static sine wave signal shall be at a frequency $f_1$ with a power level of –39 dBm.
- A Bluetooth modulated signal (see Section 4.1.7) shall be at $f_2$ with a power level of -39 dBm.

Frequencies $f_0$, $f_1$ and $f_2$ shall be chosen such that $f_0=2f_1-f_2$ and $|f_2-f_1|=n$ MHz, where n can be 3, 4, or 5. The system shall fulfill at least one of the three alternatives (n=3, 4, or 5).

### 4.1.5 Maximum usable level

The maximum usable input level that the receiver operates at shall be greater than -20 dBm. The BER shall be less than or equal to 0.1% at –20 dBm input power.

### 4.1.6 Received Signal Strength Indication

If a device supports Received Signal Strength Indication (RSSI) the accuracy shall be ±6 dB. If the device is aware that the RSSI varies across frequencies, then it should update the RSSI value of a packet depending on the frequency that the packet was received on before using the value, e.g., before reporting it to the Host.

### 4.1.7 Reference signal definition

A Bluetooth modulated interfering signal shall be defined as:

Modulation = GFSK
Modulation index = 0.32±1%
BT= 0.5±1%
Bit Rate = 1 Mb/s ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
Frequency accuracy better than ±1 ppm.

## 4.2 Enhanced Data Rate

### 4.2.1 Actual sensitivity level

The actual sensitivity level shall be defined as the input level for which a raw bit error rate (BER) of 0.01% is met. The requirement for a Bluetooth π/4-DQPSK and 8DPSK Enhanced Data Rate receiver shall be an actual sensitivity level of –70 dBm or better. The receiver shall achieve the –70 dBm sensitivity level when receiving signals specified in Section 3.2.

### 4.2.2 BER floor performance

The receiver shall achieve a BER less than 0.001% at 10 dB above the reference sensitivity level.

### 4.2.3 Interference performance

The interference performance shall be measured with a wanted signal of -67 dBm. The requirements in this section shall only apply if the frequency of the interferer is inside of the band 2400 MHz to 2483.5 MHz.

The interfering signal for co-channel interference shall be similarly modulated as the desired signal. The interfering signal for other channels shall be equivalent to a nominal Bluetooth Basic Rate GFSK transmitter. The interfering signal shall carry random data.

A BER of 0.1% or better shall be achieved for the signal to interference ratios defined in Table 4.3.

| Frequency of Interference | π/4-DQPSK Ratio | 8DPSK Ratio |
|---|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 13 dB | 21 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$ | 0 dB | 5 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | -17 dB | -12 dB |
| Adjacent (≥3 MHz) interference[1] | -27 dB | -20 dB |
| Image frequency interference[1,2,3], $C/I_{Image}$ | -7 dB | 0 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1,2,3], $C/I_{Image\ \pm 1\ MHz}$ | -15 dB | -8 dB |

*Table 4.3: Interference performance*

[1]If two adjacent channel specifications from Table 4.3 are applicable to the same channel, the more relaxed specification applies.

[2]In-band image frequency.

[3]If the image frequency is not equal to n MHz, then the image reference frequency is defined as the closest n MHz frequency for integer n.

These specifications are only to be tested at nominal temperature conditions with a receiver hopping on one frequency; this means that the synthesizer may change frequency between receive slot and transmit slot, but always returns to the same receive frequency.

Frequencies where the requirements are not met are called spurious response frequencies. Five spurious response frequencies are allowed at frequencies with a distance of ≥2 MHz from the wanted signal. On these spurious response frequencies a relaxed interference requirement C/I = -15 dB for π/4-DQPSK and C/I = -10 dB for 8DPSK shall be met.

### 4.2.4 Maximum usable level

The maximum usable input level that the receiver operates at shall be greater than -20 dBm. The BER shall be less than or equal to 0.1% at -20 dBm input power.

### 4.2.5 Out-of-band and intermodulation characteristics

The Basic Rate out-of-band blocking and intermodulation requirements also apply to Enhanced Data Rate since they result in adequate performance. No additional requirements apply to Enhanced Data Rate.

### 4.2.6 Reference signal definition

A 2 Mb/s Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = π/4-DQPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within ±1 dB

A 3 Mb/s Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = 8DPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within ±1 dB

# 5 POWER MANAGEMENT

## 5.1 Power classes

Bluetooth devices are classified into three power classes based on their output power capabilities level at the maximum power setting ($P_{max}$) the device supports, as defined in Table 5.1.

| Power Class | Requirements |
|---|---|
| 1 | 100 mW (20 dBm) ≥ $P_{max}$ > 2.5 mW (4 dBm) |
| 2 | 2.5 mW (4 dBm) ≥ $P_{max}$ > 1 mW (0 dBm) |
| 3 | 1 mW (0 dBm) ≥ $P_{max}$ |

*Table 5.1: Power classes*

If a device supports more than one modulation mode, then the power class is determined by the mode with the highest maximum power setting and any power control requirements shall apply to all modulation modes. For example, if a device has a maximum power of 6 dBm on BR but 2 dBm on EDR, then it is a class 1 device and must support power control on EDR as well as BR.

A class 1 device shall be able to adjust its transmit power down to 4 dBm or less.

If a class 1 device is paging or inquiring very close to another device, the input power can be larger than the requirement in Section 4.1.5. This can cause the receiving device to fail to respond. It may therefore be useful to page at class 2 or 3 power in addition to paging at power class 1.

If the receiving device in a connection does not support the necessary messaging for sending power control messages (see [Vol 2] Part C, Section 4.1.3), then the output power of the transmitting device shall not exceed the maximum output power of power class 2.

## 5.2 Power control

A device that supports power control shall have a set of transmission power levels; the difference in power level between adjacent levels forms a “power step”. The power steps shall form a monotonic sequence, with a maximum step size of 8 dB and a minimum step size of 2 dB. The power level at the lowest power step should be less than -30 dBm.

The transmit power level difference between the packet headers of all supported packet types at any given power step shall not exceed 10 dB.

A device that supports power control shall control the output power in a physical link in response to LMP commands received from a peer device that is capable of sending such requests (see [Vol 2] Part C, Section 4.1.3).

Using high transmit power in use cases where short ranges could be encountered can cause the receiver on the remote device to be saturated and result in link failure. Power control requests can be used to adjust a connected remote device's transmit power level based on the receiver's signal level.

A power class 1 device can use power control to limit the transmitted power of a remote device to no more than +4 dBm. A power class 2 or class 3 device can use power control to optimize the power consumption and reduce the overall interference level for all devices that use the same spectrum that Bluetooth devices use.

## 5.3 Enhanced power control

When communicating with a device that does not support enhanced power control, a device that supports enhanced power control shall have an equal number of power steps for each supported modulation scheme so that all supported modulation modes shall reach their respective maximum (or minimum) levels at the same time. The power levels may vary per modulation mode.

# Appendix A Test conditions

## A.1 Nominal test conditions

### A.1.1 Nominal temperature

The nominal temperature conditions for tests shall be +15 to +35 °C. When it is impractical to carry out the test under this condition a note to this effect, stating the ambient temperature, shall be recorded. The actual value during the test shall be recorded in the test report.

### A.1.2 Nominal power source

The normal test voltage for the equipment shall be the nominal voltage for which the equipment was designed.

## A.2 [This section is no longer used]

# Appendix B [This Appendix is no longer used]

# Appendix C Modulation accuracy definition

## C.1 Enhanced Data Rate modulation accuracy

The Enhanced Data Rate modulation accuracy is defined by the differential error vector, being the difference between the vectors representing consecutive symbols of the transmitted signal, after passing the signal through a specified measurement filter, sampling it at the symbol rate with an optimum sampling phase and compensating it for carrier frequency error and for the ideal carrier phase changes. The magnitude of the normalized differential error vector is called the Differential Error Vector Magnitude (DEVM). The objective of the DEVM is to estimate the modulation errors that would be perceived by a differential receiver.

In an ideal transmitter, the input bit sequence $\{b_j\}$ is mapped onto a complex valued symbol sequence $\{S_k\}$. Subsequently, this symbol sequence is transformed into a baseband signal S(t) by means of a pulse-shaping filter.

In an actual transmitter implementation, the bit sequence $\{b_j\}$ generates a baseband equivalent transmitted signal Y(t). This signal Y(t) contains, besides the desired component S(t), multiple distortion components. This is illustrated in Figure C.1 (in Figure C.1 and Figure C.2 a circle with an X indicates a mixer).

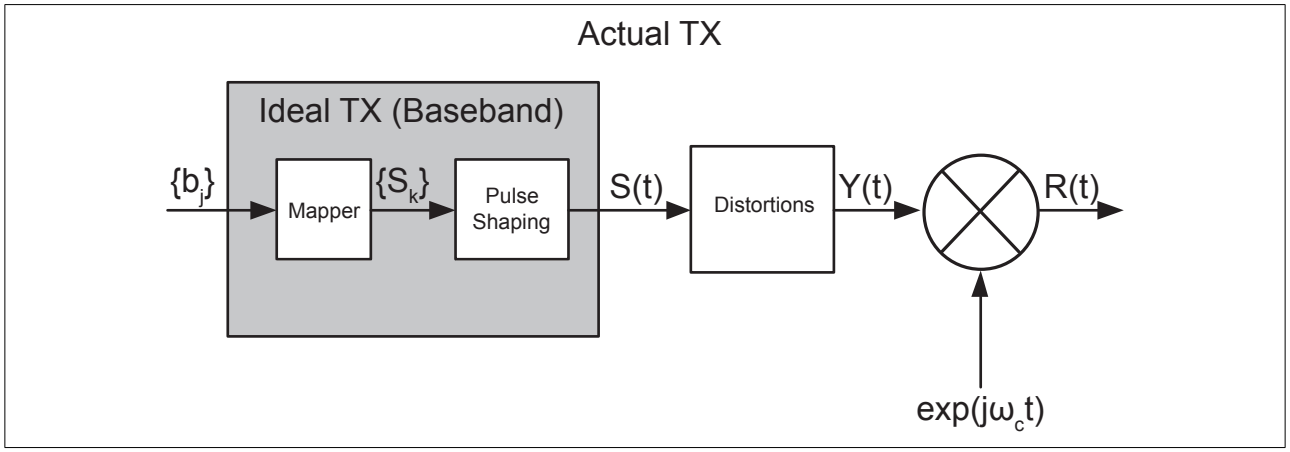



*Figure C.1: TX model*

Let Z(t) be the output of the measurement filter after first compensating the received signal for the initial center frequency error, $\omega_i$, of the received packet, i.e. the output after down conversion and filtering the transmit signal R(t) (see Figure C.2).The measurement filter is defined by a square-root raised cosine shaping filter with a roll-off factor equal to 0.4 and 3 dB bandwidth of ±500 kHz.

Let $\{Z_k(\varepsilon)\}$ be the sequence of samples obtained by sampling the signal Z(t) with a sampling period equal to the symbol period T and a sampling phase equal to ε such that $Z_k(\varepsilon)=Z((k+\varepsilon)T)$.

This sequence $\{Z_k(\varepsilon)\}$ would coincide with the symbol sequence $\{S_k\}$ if no distortion is present and the correct timing phase ε is chosen.

To reflect the behavior of a typical differential receiver, the sample sequence $\{Z_k(\varepsilon)\}$ should be compensated for carrier frequency drift. Therefore, the sequence $\{Z_k(\varepsilon)\}$ is multiplied by a factor $W^{-k}$ in which $W = e^{j\omega T}$ accounts for the frequency offset ω. A constant value of ω is used for each DEVM block of N = 50 symbols, but ω may vary between DEVM blocks (the values of ω can be used to measure carrier frequency drift).

In addition, $\{Z_k(\varepsilon)\}$ is compensated for the ideal phase changes between symbols by multiplying it with the complex conjugate of the symbol sequence $\{S_k\}$. However, it is not necessary to compensate $\{Z_k(\varepsilon)\}$ for initial carrier phase or output power of the transmitter.

Let $\{Q_k(\varepsilon,\omega)\}$ denote the compensated sequence $\{Z_k(\varepsilon)\}$, where the ideal phase changes have been removed and ε and ω are chosen optimally to minimize the DEVM, (i.e. remove time and frequency uncertainty). For a transmitter with no distortions other than a constant frequency error, $\{Q_k(\varepsilon,\omega)\}$ is a complex constant that depends on the initial carrier phase and the output power of the transmitter.

The differential error sequence $\{E_k(\varepsilon,\omega)\}$ is defined as the difference between $\{Q_k(\varepsilon,\omega)\}$ and $\{Q_{k-1}(\varepsilon,\omega)\}$. This reflects the modulation errors that would be perceived by a differential receiver. For a transmitter with no distortions other than a constant frequency error, $\{E_k(\varepsilon,\omega)\}$ is zero.

The definitions of the DEVM measures are based upon this differential error sequence $\{E_k(\varepsilon,\omega)\}$. The generation of the error sequence is depicted in Figure C.2 (the circle with the + indicates a direct summation function; the input with the "−" is negated before being summed).

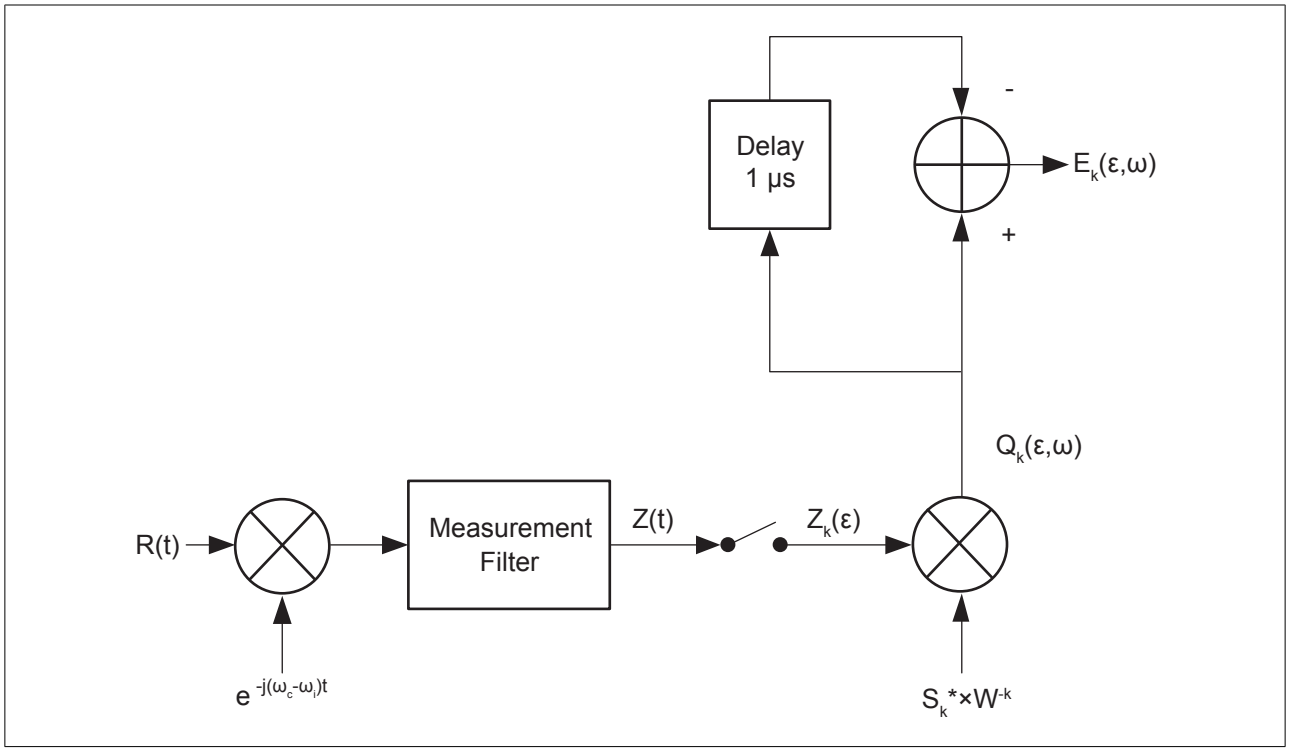



*Figure C.2: Error sequence generation*

### C.1.1 RMS DEVM

The root mean squared DEVM (RMS DEVM) computed over N = 50 symbols is defined as:

$$RMS\ \ DEVM = \min_{\varepsilon,\omega}\left\{\sqrt{\frac{\sum_{k=1}^{N}|E_k(\varepsilon,\omega)|^2}{\sum_{k=1}^{N}|Q_k(\varepsilon,\omega)|^2}}\right\} \quad \text{(EQ 6)}$$

As can be seen from the expression above, the RMS DEVM is the square-root of the normalized power of the error.

### C.1.2 Peak DEVM

The DEVM at symbol k is defined as:

$$DEVM\,(k) = \sqrt{\frac{|E_k(\varepsilon_0,\omega_0)|^2}{\sum_{j=1}^{N}|\,Q_j(\varepsilon_0,\omega_0)|^2/N}} \quad \text{(EQ 7)}$$

where $\varepsilon_0$ and $\omega_0$ are the values for $\varepsilon$ and $\omega$ used to calculate the RMS DEVM.

The peak DEVM is defined as:

$$Peak\ DEVM = \max_{k}\{\mathrm{DEVM}\ (k)\} \quad \text{(EQ 8)}$$

BR/EDR Controller
Part B

# BASEBAND SPECIFICATION

*This Part describes the specification of the Bluetooth Link Controller which carries out the Baseband protocols and other low-level link routines.*

Bluetooth SIG Proprietary

# CONTENTS

**1 General description ..... 480**
1.1 Bluetooth clock ..... 481
1.2 Bluetooth Device addressing ..... 482
1.2.1 Reserved addresses ..... 483
1.3 Access codes ..... 484

**2 Physical channels ..... 485**
2.1 Physical channel definition ..... 486
2.2 Basic piconet physical channel ..... 486
2.2.1 Central and Peripheral roles ..... 486
2.2.2 Hopping characteristics ..... 487
2.2.3 Time slots ..... 487
2.2.4 Piconet clocks ..... 488
2.2.5 Transmit/receive timing ..... 488
2.2.5.1 Piconet physical channel timing ..... 489
2.2.5.2 Piconet physical channel re-synchronization ..... 491
2.3 Adapted piconet physical channel ..... 492
2.3.1 Hopping characteristics ..... 492
2.4 Page scan physical channel ..... 492
2.4.1 Clock estimate for paging ..... 493
2.4.2 Hopping characteristics ..... 493
2.4.3 Paging procedure timing ..... 493
2.4.4 Page response timing ..... 494
2.5 Inquiry scan physical channel ..... 496
2.5.1 Clock for inquiry ..... 496
2.5.2 Hopping characteristics ..... 497
2.5.3 Inquiry procedure timing ..... 497
2.5.4 Inquiry response timing ..... 497
2.6 Hop selection ..... 498
2.6.1 General selection scheme ..... 499
2.6.2 Selection kernel ..... 502
2.6.2.1 First addition operation ..... 503
2.6.2.2 XOR operation ..... 503
2.6.2.3 Permutation operation ..... 504
2.6.2.4 Second addition operation ..... 505
2.6.2.5 Register bank ..... 505
2.6.3 Adapted hop selection kernel ..... 506
2.6.3.1 Channel re-mapping function ..... 506

2.6.4 Control word ..... 507
2.6.4.1 Page scan and inquiry scan hopping sequences ..... 509
2.6.4.2 Page hopping sequence ..... 509
2.6.4.3 Peripheral page response hopping sequence ..... 509
2.6.4.4 Central page response hopping sequence ..... 510
2.6.4.5 Inquiry hopping sequence ..... 510
2.6.4.6 Inquiry response hopping sequence ..... 511
2.6.4.7 Basic and adapted channel hopping sequence ..... 511
2.6.4.8 Synchronization train RF channels ..... 511
2.7 Synchronization scan physical channel ..... 511
2.7.1 Hopping characteristics ..... 511
2.7.2 Synchronization Train procedure timing ..... 512
2.7.3 Synchronization Scan procedure timing ..... 513

**3 Physical links ..... 514**
3.1 Link supervision for active physical links ..... 514
3.2 Link supervision for Connectionless Peripheral Broadcast physical links ..... 515
3.3 Authenticated payload timeout for active links ..... 515

**4 Logical transports ..... 516**
4.1 General ..... 516
4.2 Logical transport address (LT_ADDR) ..... 516
4.3 Synchronous logical transports ..... 517
4.4 Asynchronous logical transport ..... 517
4.5 Transmit/receive routines ..... 518
4.5.1 TX routine ..... 518
4.5.1.1 ACL traffic ..... 519
4.5.1.2 SCO traffic ..... 520
4.5.1.3 Mixed data/voice traffic ..... 520
4.5.1.4 eSCO traffic ..... 521
4.5.1.5 Default packet types ..... 521
4.5.2 RX routine ..... 521
4.5.3 Flow control ..... 522
4.5.3.1 Destination control ..... 523
4.5.3.2 Source control ..... 523
4.6 Active Peripheral broadcast transport ..... 523
4.7 [This section is no longer used] ..... 524
4.8 Connectionless Peripheral Broadcast logical transport ..... 524

**5 Logical links ..... 525**
5.1 Link Control logical link (LC) ..... 525
5.2 ACL Control logical links (ACL-C and APB-C) ..... 525
5.3 User asynchronous/isochronous logical links (ACL-U and APB-U) ..... 526
5.3.1 Pausing the ACL-U logical link ..... 526
5.4 User synchronous data logical link (SCO-S) ..... 526
5.5 User extended synchronous data logical link (eSCO-S) ..... 526
5.6 Logical link priorities ..... 526
5.7 Profile broadcast data logical link ..... 527

**6 Packets ..... 528**
6.1 General format ..... 528
6.1.1 Basic Rate ..... 528
6.1.2 Enhanced Data Rate ..... 528
6.2 Bit ordering ..... 529
6.3 Access code ..... 529
6.3.1 Access code types ..... 529
6.3.2 Preamble ..... 530
6.3.3 Sync word ..... 530
6.3.3.1 Synchronization word definition ..... 530
6.3.3.2 Pseudo-random noise sequence generation 533
6.3.4 Trailer ..... 534
6.4 Packet header ..... 534
6.4.1 LT_ADDR ..... 535
6.4.2 TYPE ..... 535
6.4.3 FLOW ..... 535
6.4.4 ARQN ..... 535
6.4.5 SEQN ..... 536
6.4.6 HEC ..... 536
6.5 Packet types ..... 536
6.5.1 Common packet types ..... 539
6.5.1.1 ID packet ..... 539
6.5.1.2 NULL packet ..... 539
6.5.1.3 POLL packet ..... 539
6.5.1.4 FHS packet ..... 539
6.5.1.5 DM1 packet ..... 541
6.5.2 SCO packets ..... 541
6.5.2.1 HV1 packet ..... 542
6.5.2.2 HV2 packet ..... 542
6.5.2.3 HV3 packet ..... 542
6.5.2.4 DV packet ..... 542
6.5.3 eSCO packets ..... 542

6.5.3.1 EV3 packet ..... 543
6.5.3.2 EV4 packet ..... 543
6.5.3.3 EV5 packet ..... 543
6.5.3.4 2-EV3 packet ..... 543
6.5.3.5 2-EV5 packet ..... 543
6.5.3.6 3-EV3 packet ..... 544
6.5.3.7 3-EV5 packet ..... 544
6.5.4 ACL packets ..... 544
6.5.4.1 DM1 packet ..... 544
6.5.4.2 DH1 packet ..... 544
6.5.4.3 DM3 packet ..... 545
6.5.4.4 DH3 packet ..... 545
6.5.4.5 DM5 packet ..... 545
6.5.4.6 DH5 packet ..... 545
6.5.4.7 AUX1 packet ..... 545
6.5.4.8 2-DH1 packet ..... 545
6.5.4.9 2-DH3 packet ..... 546
6.5.4.10 2-DH5 packet ..... 546
6.5.4.11 3-DH1 packet ..... 546
6.5.4.12 3-DH3 packet ..... 546
6.5.4.13 3-DH5 packet ..... 546
6.6 Payload format ..... 546
6.6.1 Synchronous data field ..... 547
6.6.2 Asynchronous data field ..... 548
6.7 Packet summary ..... 552

**7 Bit stream processing ..... 554**
7.1 Error checking ..... 555
7.1.1 HEC generation ..... 556
7.1.2 CRC generation ..... 557
7.2 Data whitening ..... 558
7.3 Error correction ..... 559
7.4 FEC code: rate 1/3 ..... 560
7.5 FEC code: rate 2/3 ..... 560
7.6 ARQ scheme ..... 561
7.6.1 Unnumbered ARQ ..... 561
7.6.2 Retransmit filtering ..... 565
7.6.2.1 Initialization of SEQN at start of new connection ..... 566
7.6.2.2 ACL and SCO retransmit filtering ..... 566
7.6.2.3 eSCO retransmit filtering ..... 567
7.6.2.4 FHS retransmit filtering ..... 567

7.6.2.5 Extended inquiry response retransmit filtering ..... 567
7.6.2.6 Packets without CRC retransmit filtering ..... 568
7.6.3 Flushing payloads ..... 568
7.6.4 Multi-Peripheral considerations ..... 569
7.6.5 Active Peripheral Broadcast packets ..... 569
7.7 Erroneous synchronous data reporting ..... 570
7.8 Message Integrity Check ..... 570

**8 Link Controller operation ..... 571**
8.1 Overview of states ..... 571
8.2 Standby state ..... 572
8.3 Connection establishment substates ..... 572
8.3.1 Page Scan substate ..... 572
8.3.2 Page substate ..... 574
8.3.3 Page response substates ..... 576
8.3.3.1 Peripheral Page Response substate ..... 579
8.3.3.2 Central Page Response substate ..... 580
8.4 Device discovery substates ..... 581
8.4.1 Inquiry Scan substate ..... 582
8.4.2 Inquiry substate ..... 583
8.4.3 Inquiry Response substate ..... 584
8.5 Connection state ..... 586
8.6 Active mode ..... 587
8.6.1 Polling in the Active mode ..... 588
8.6.2 SCO ..... 589
8.6.3 eSCO ..... 590
8.6.4 Broadcast scheme ..... 593
8.6.5 Role switch ..... 593
8.6.6 Scatternet ..... 595
8.6.6.1 Inter-piconet communications ..... 596
8.6.7 Hop sequence switching ..... 597
8.6.8 Channel classification and channel map selection ..... 600
8.6.9 Power management ..... 601
8.6.9.1 Packet handling ..... 601
8.6.9.2 Slot occupancy ..... 601
8.6.9.3 Recommendations for low-power operation 601
8.6.9.4 Enhanced Data Rate ..... 602
8.6.10 Piconet clock adjustment ..... 602
8.6.10.1 Coarse clock adjustment ..... 602
8.6.10.2 Coarse Clock Adjustment Recovery mode .. 604
8.6.10.3 Clock Dragging ..... 605
8.6.11 Slot Availability Mask (SAM) ..... 605

8.6.11.1 SAM anchor point ..... 609
8.6.11.2 SAM scheduling ..... 610
8.7 Sniff mode ..... 612
8.7.1 Sniff Transition mode ..... 614
8.7.2 Sniff subrating ..... 614
8.7.2.1 Sniff mode timeout ..... 615
8.7.2.2 Sniff Subrating mode ..... 615
8.8 Hold mode ..... 616
8.9 [This section is no longer used] ..... 616
8.10 Connectionless Peripheral Broadcast mode ..... 616
8.10.1 Connectionless Peripheral Broadcast transmit operation ..... 617
8.10.2 Connectionless Peripheral Broadcast receive operation ..... 618
8.10.3 AFH in Connectionless Peripheral Broadcast ..... 619
8.11 Synchronization establishment substates ..... 619
8.11.1 Synchronization Scan substate ..... 619
8.11.2 Synchronization Train substate ..... 619

**9 Audio ..... 623**
9.1 LOG PCM codec ..... 623
9.2 CVSD codec ..... 623
9.3 Error handling ..... 626
9.4 General audio requirements ..... 626
9.4.1 Signal levels ..... 626
9.4.2 CVSD audio quality ..... 626

**Appendix A General audio recommendations ..... 627**
A.1 Maximum sound pressure ..... 627
A.2 [This section is no longer used] ..... 627
A.3 Audio levels for Bluetooth ..... 627
A.4 Microphone path ..... 628
A.5 Loudspeaker path ..... 628
A.6 Bluetooth voice interface ..... 628
A.7 Frequency mask ..... 629

**Appendix B Timers ..... 631**
B.1 List of timers ..... 631
B.1.1 inquiryTO ..... 631
B.1.2 pageTO ..... 631
B.1.3 extended_pageTO ..... 631
B.1.4 pagerespTO ..... 631
B.1.5 newconnectionTO ..... 631

B.1.6 supervisionTO ..... 632
B.1.7 CPB_supervisionTO ..... 632
B.1.8 synchronization_trainTO ..... 632
B.1.9 synchronization_scanTO ..... 632
B.1.10 authenticatedPayloadTO ..... 633
B.1.11 CLK_adj_dragTO ..... 633

**Appendix C Recommendations for AFH operation in Hold, Sniff, and Connectionless Peripheral Broadcast modes ..... 634**

C.1 Operation at the Central ..... 634
C.2 [This section is no longer used] ..... 635
C.3 AFH operation in Sniff mode ..... 635
C.4 AFH operation in Hold mode ..... 635
C.5 AFH operation in Connectionless Peripheral Broadcast ..... 635

# 1 GENERAL DESCRIPTION

This part specifies the normal operation of a Bluetooth Baseband.

The Bluetooth system provides a point-to-point connection or a point-to-multipoint connection, see (a) and (b) in Figure 1.1. In a point-to-point connection the physical channel is shared between two Bluetooth devices. In a point-to-multipoint connection, the physical channel is shared among several Bluetooth devices. Two or more devices sharing the same physical channel form a *piconet*. One Bluetooth device acts as the Central of the piconet, whereas the other device(s) act as Peripheral(s). Up to seven Peripherals can be active in the piconet. The channel access is controlled by the Central. An unlimited number of Peripherals can receive data on the physical channel carrying the Connectionless Peripheral Broadcast physical link.

Piconets that have common devices are called a *scatternet,* see (c) in Figure 1.1. Each piconet only has a single Central, however, Peripherals can participate in different piconets on a time-division multiplex basis. In addition, a Central in one piconet can be a Peripheral in other piconets. Piconets shall not be frequency synchronized and each piconet has its own hopping sequence.

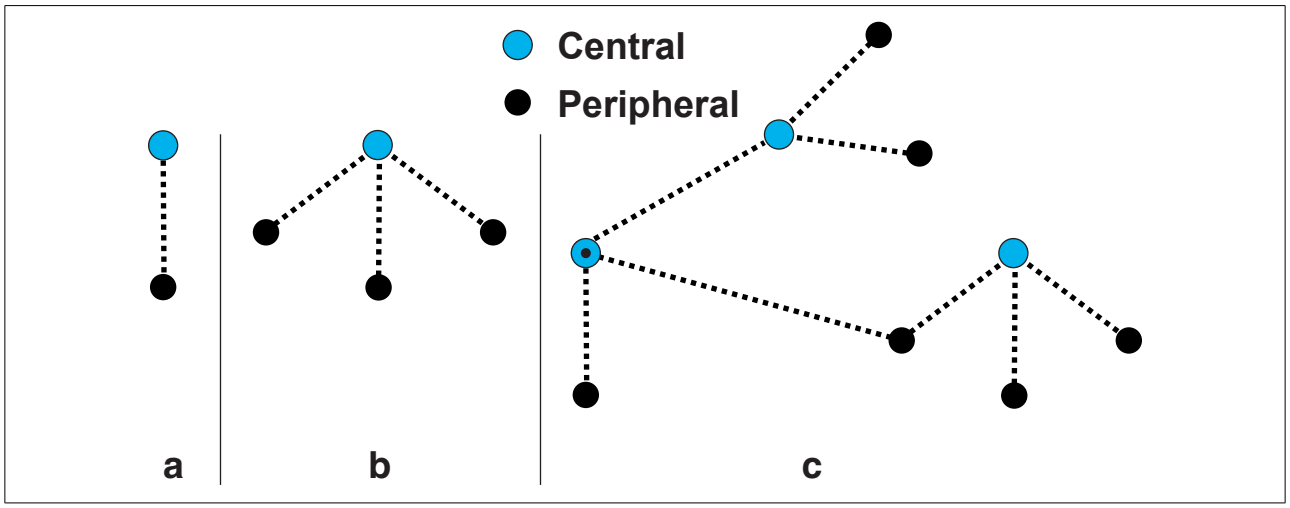



*Figure 1.1: Piconets with a single Peripheral operation (a), a multi-Peripheral operation (b) and a scatternet operation (c)*

Data is transmitted over the air in packets. Two modes are defined: a mandatory mode called Basic Rate and an optional mode called Enhanced Data Rate. The symbol rate for all modulation modes is 1 Msym/s. The gross air data rate is 1 Mb/s for Basic Rate. Enhanced Data Rate has a primary modulation mode that provides a gross air data rate of 2 Mb/s, and a secondary modulation mode that provides a gross air data rate of 3 Mb/s.

The general Basic Rate packet format is shown in Figure 1.2. Each packet consists of 3 entities: the access code, the header, and the payload.

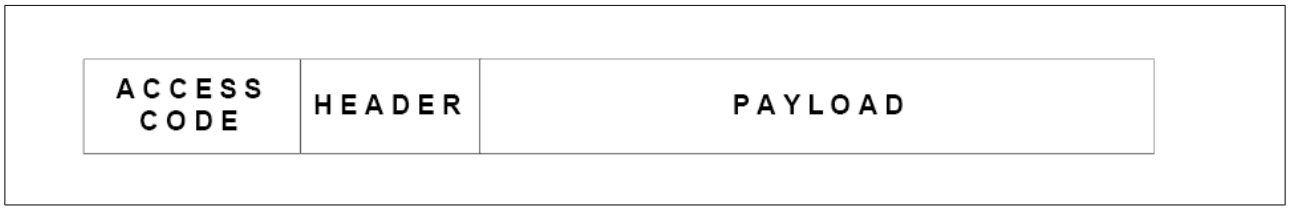



*Figure 1.2: Standard Basic Rate packet format*

The general Enhanced Data Rate packet format is shown in Figure 1.3. Each packet consists of 6 entities: the access code, the header, the guard period, the synchronization sequence, the Enhanced Data Rate payload and the trailer. The access code and header use the same modulation mode as for Basic Rate packets while the synchronization sequence, the Enhanced Data Rate payload and the trailer use the Enhanced Data Rate modulation mode. The guard time allows for the transition between the modulation modes.

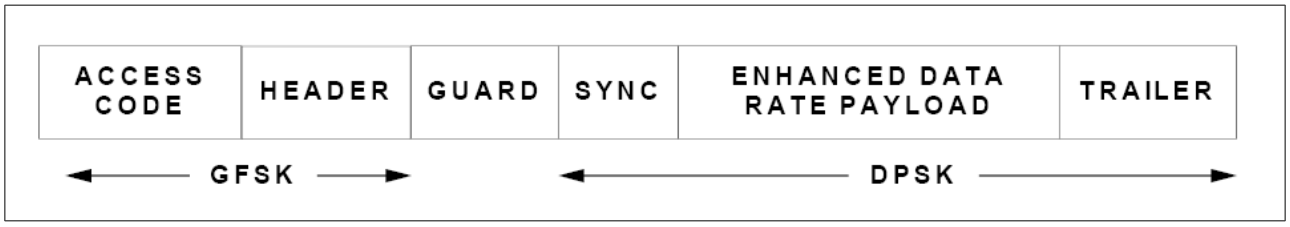



*Figure 1.3: Standard Enhanced Data Rate packet format*

## 1.1 Bluetooth clock

Every Bluetooth device shall have a native clock that shall be derived from a free running reference clock. Offsets may be added to the reference clock to synchronize the native clock with other non-Bluetooth systems. For synchronization with other Bluetooth devices, offsets are used that, when added to the native clock, provide temporary Bluetooth clocks that are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it may therefore be initialized to any value. The clock has a cycle of about a day. If the clock is implemented with a counter, a 28-bit counter is required that shall wrap around at $2^{28}$ -1. The least significant bit (LSB) shall tick in units of 312.5 µs (i.e. half a time slot), giving a clock rate of 3.2 kHz.

The clock determines critical periods and triggers the events in the device. Four periods are important in the Bluetooth system: 312.5 µs, 625 µs, 1.25 ms, and 1.28 s; these periods correspond to the timer bits $CLK_0$, $CLK_1$, $CLK_2$, and $CLK_{12}$, respectively, see Figure 1.4.

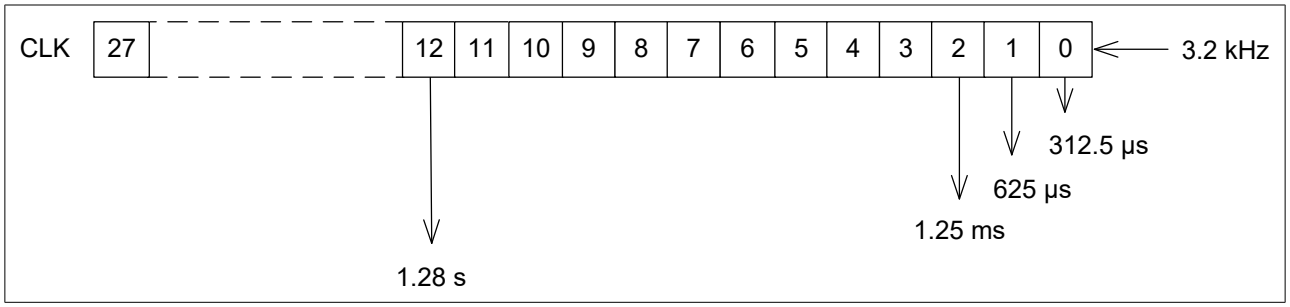



*Figure 1.4: Bluetooth clock*

In the different modes and states a device can reside in, the clock has different appearances:

- CLKR reference clock
- CLKN native clock
- CLKE estimated clock
- CLK Central's clock

CLKR is the reference clock driven by the free running system clock. CLKN may be offset from the reference clock by a timing offset. In Standby state and in Hold, Sniff, and Connectionless Peripheral Broadcast modes the reference clock shall have a worst case accuracy of ±250 ppm. In all other circumstances, it shall have a worst case accuracy of ±20 ppm; this accuracy shall also be used by the piconet Central while performing Piconet Clock Adjustment (see Section 8.6.10).

See Section 2.2.4 for the definition of CLK and Section 2.4.1 for the definition of CLKE.

The Central may adjust its native clock during the existence of the piconet within certain limits (see Section 8.6.10.3). The Central may also perform a coarse adjustment of the native clock by using the LMP_CLK_ADJ sequence.

## 1.2 Bluetooth Device addressing

Each Bluetooth device shall be allocated a unique 48-bit Bluetooth Device Address (BD_ADDR). The address shall be a 48-bit extended unique identifier (EUI-48) created in accordance with section 8.2 ("Universal addresses") of the IEEE 802-2014 standard (http://standards.ieee.org/findstds/standard/802-2014.html).

Creation of a valid EUI-48 requires one of the following MAC Address Block types to be obtained from the IEEE Registration Authority:

- MAC Address Block Large (MA-L)
- MAC Address Block Medium (MA-M)
- MAC Address Block Small (MA-S)

See http://standards.ieee.org/develop/regauth/index.html for information on obtaining one of these MAC Address Blocks. See also the IEEE's guidelines for use of these addresses (https://standards.ieee.org/develop/regauth/tut/eui.pdf).

Figure 1.5 illustrates how the LAP, UAP, and NAP map to the EUI-48. The bit pattern in Figure 1.5 is an example BD_ADDR.

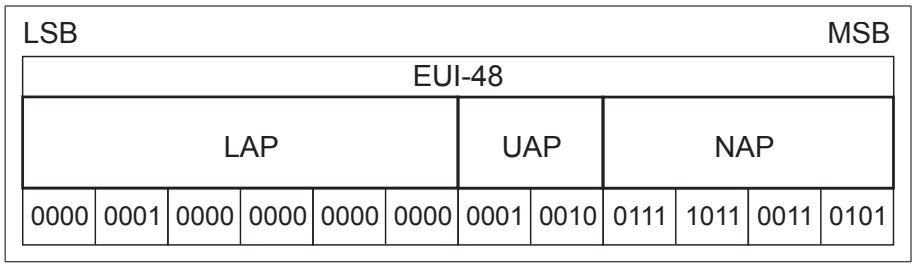



*Figure 1.5: Format of BD_ADDR*

The BD_ADDR may take any values except those that would have any of the 64 reserved LAP values for general and dedicated inquiries (see Section 1.2.1).

### 1.2.1 Reserved addresses

A block of 64 contiguous LAPs is reserved for inquiry operations; one LAP common to all devices is reserved for general inquiry, the remaining 63 LAPs are reserved for dedicated inquiry of specific classes of devices (see Assigned Numbers). The same LAP values are used regardless of the contents of UAP and NAP. Consequently, none of these LAPs can be part of a user BD_ADDR.

The reserved LAP addresses are 0x9E8B00 to 0x9E8B3F. The general inquiry LAP is 0x9E8B33. All addresses have the LSB at the rightmost position, hexadecimal notation. The default check initialization (DCI) is used as the UAP whenever one of the reserved LAP addresses is used. The DCI is defined to be 0x00 (hexadecimal).

## 1.3 Access codes

In the Bluetooth system all transmissions over the physical channel begin with an access code. Three different access codes are defined, see also Section 6.3.1:

- device access code (DAC)
- channel access code (CAC)
- inquiry access code (IAC)

All access codes are derived from the LAP of a device address or an inquiry address. The device access code is used during Page, Page Scan, Central Page Response, and Peripheral Page Response substates and shall be derived from the paged device's BD_ADDR. The channel access code is used in the Connection state, Synchronization Train substate, and Synchronization Scan substate, and forms the beginning of all packets exchanged on the piconet physical channel. The channel access code shall be derived from the LAP of the Central's BD_ADDR. Finally, the inquiry access code shall be used in the Inquiry substate. There is one general IAC (GIAC) for general inquiry operations and there are 63 dedicated IACs (DIACs) for dedicated inquiry operations.

The access code also indicates to the receiver the arrival of a packet. It is used for timing synchronization and offset compensation. The receiver correlates against the entire synchronization word in the access code, providing very robust signaling.

# 2 PHYSICAL CHANNELS

The lowest architectural layer in the Bluetooth system is the physical channel. A number of types of physical channels are defined. All Bluetooth physical channels are characterized by the combination of a basic pseudo-random frequency hopping sequence (which, for physical links on the adapted piconet physical channel, is then modified by the *AFH_channel_map* (defined in [Vol 2] Part C, Section 5.2) for that link), the specific slot timing of the transmissions, the access code and packet header encoding. These aspects, together with the range of the transmitters, define the signature of the physical channel. For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference.

Two devices that communicate use a shared physical channel for this communication. To achieve this, their transceivers must be tuned to the same RF frequency at the same time, and they must be within a nominal range of each other.

Given that the number of RF carriers is limited and that many Bluetooth devices could be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.

Several Bluetooth physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (the basic piconet channel and adapted piconet channel) are used for communication between connected devices and are associated with a specific piconet. Other physical channels are used for discovering (the inquiry scan channel) and connecting (the page scan channel) Bluetooth devices. The synchronization scan physical channel is used by devices to obtain timing and frequency information about the Connectionless Peripheral Broadcast physical link or to recover the current piconet clock.

A Bluetooth device can only use one of these physical channels at any given time. In order to support multiple concurrent operations the device uses time-division multiplexing between the channels. In this way a Bluetooth device can appear to operate simultaneously in several piconets, as well as being discoverable and connectable.

Whenever a Bluetooth device is synchronized to the timing, frequency and access code of a physical channel it is said to be 'connected' to this channel (whether or not it is

actively involved in communications over the channel.) At a minimum, a device need only be capable of connection to one physical channel at a time, however, advanced devices may be capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

## 2.1 Physical channel definition

Physical channels, apart from the synchronization scan physical channel (which uses a set of fixed RF channels), are defined by a basic pseudo-random RF channel hopping sequence, the packet (slot) timing, and an access code. The hopping sequences are determined by the UAP and LAP of a Bluetooth Device Address, the selected basic hopping sequence, and – for the adapted piconet physical channel – the *AFH_channel_map* being used on a physical link. The phase in the hopping sequence is determined by the Bluetooth clock. All physical channels are subdivided into time slots. Within the physical channel, each reception or transmission event is associated with a time slot or time slots. For each reception or transmission event an RF channel is selected by the hop selection kernel (see Section 2.6).The maximum hop rate is 1600 hops per second in the Connection state, the Synchronization Train substate, and the Synchronization Scan substate and the maximum is 3200 hops per second in the Inquiry and Page substates.

The following physical channels are defined:

- basic piconet physical channel
- adapted piconet physical channel
- page scan physical channel
- inquiry scan physical channel
- synchronization scan physical channel

## 2.2 Basic piconet physical channel

During the Connection state the basic piconet physical channel is used by default. The adapted piconet physical channel may also be used. The adapted piconet physical channel is identical to the basic piconet physical channel except for the differences listed in Section 2.3.

### 2.2.1 Central and Peripheral roles

The basic piconet physical channel is defined by the Central of the piconet. The Central controls the traffic on the piconet physical channel by a polling scheme (see Section 8.5).

By definition, the device that initiates a connection by paging is the Central. Once a piconet has been established, the Central and Peripheral may exchange roles. This is described in Section 8.6.5.

### 2.2.2 Hopping characteristics

The basic piconet physical channel is characterized by a pseudo-random hopping through all 79 RF channels. The frequency hopping in the piconet physical channel is determined by the Bluetooth clock and BD_ADDR of the Central. When the piconet is established, the Central's clock is communicated to the Peripherals. Each Peripheral shall add an offset to its native clock to synchronize with the Central's clock. Since the clocks are independent, the offsets must be updated regularly. All devices participating in the piconet are time-synchronized and hop-synchronized to the channel.

The basic piconet physical channel uses the basic channel hopping sequence and is described in Section 2.6.

### 2.2.3 Time slots

The basic piconet physical channel is divided into time slots, each 625 µs in length. The time slots are numbered according to the most significant 27 bits of the Bluetooth clock $CLK_{27-1}$ of the piconet Central. The slot numbering ranges from 0 to $2^{27}$-1 and is cyclic with a cycle length of $2^{27}$. The time slot number is denoted as k.

A TDD scheme is used where Central and Peripheral alternately transmit, see Figure 2.1. The packet start shall be aligned with the slot start. Packets may extend over up to five time slots.

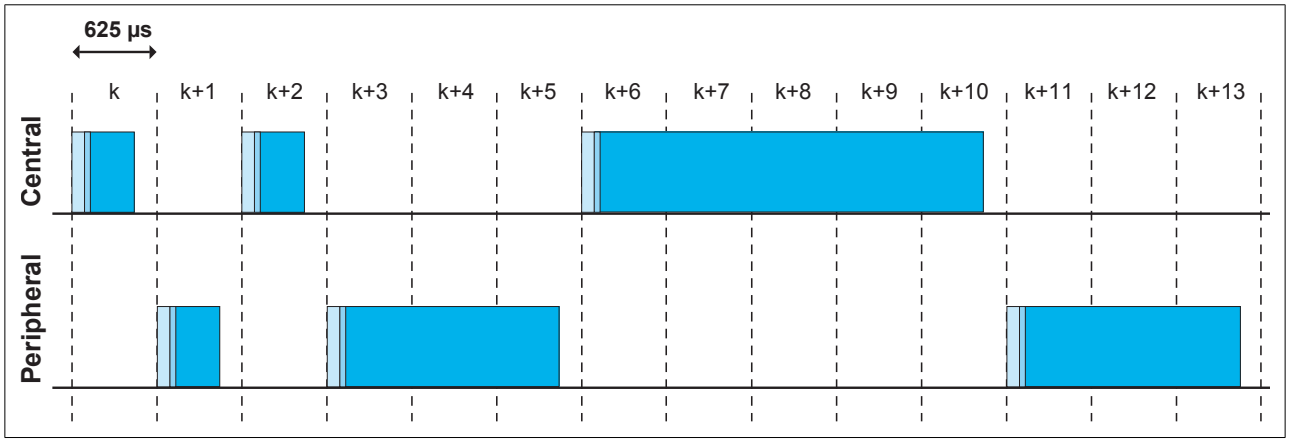



*Figure 2.1: Multi-slot packets*

The term *slot pairs* is used to indicate two adjacent time slots starting with a Central-to-Peripheral transmission slot.

### 2.2.4 Piconet clocks

CLK is the clock of the Central of the piconet. It shall be used for all timing and scheduling activities in the piconet. All devices shall use the CLK to schedule their transmission and reception. The CLK shall be derived from the reference clock CLKR (see Section 1.1) by adding a time_base_offset and a peripheral_clock_offset, see Figure 2.2. The time_base_offset is a value that devices may use to store a locally generated offset to CLKN caused by alignment to an external time base. The peripheral_clock_offset shall be zero for the Central since CLK is identical to its own native clock CLKN. Each Peripheral shall add an appropriate peripheral_clock_offset to its CLKN such that the CLK corresponds to the CLKN of the Central. Although all CLKNs in the devices run at the same nominal rate, mutual drift causes inaccuracies in CLK. Therefore, the peripheral_clock_offset in the Peripherals must be regularly updated such that CLK is approximately CLKN of the Central.

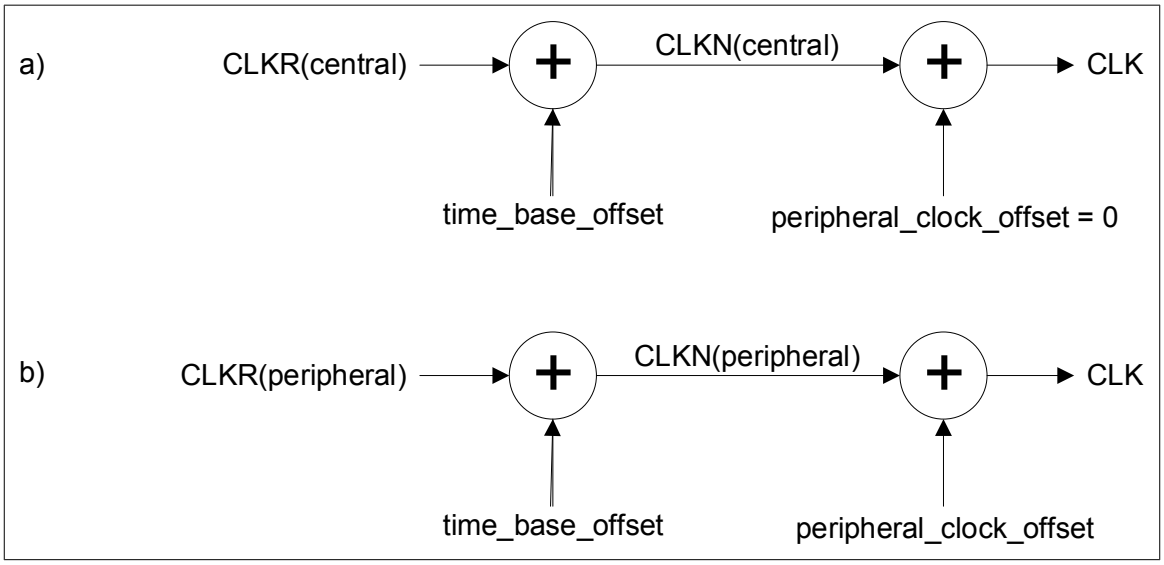



*Figure 2.2: Derivation of CLK in Central (a) and in Peripheral (b)*

Changes in time_base_offset are only made by the Central of a piconet; a device acting only as a Peripheral has no need to distinguish CLKR and CLKN in normal operation. In a scatternet situation, a Controller may be changing both time_base_offset to align its CLKN with an external clock, either collocated or at the request of a Peripheral, and peripheral_clock_offset to maintain synchronization with a different piconet clock. In some cases, it is not possible to determine how much of an observed offset is caused by external frame timing alignment (time_base_offset) and how much is caused by the offset between Central and Peripheral (peripheral_clock_offset).

### 2.2.5 Transmit/receive timing

The Central transmission shall always start at even numbered time slots ($CLK_1$=0) and the Peripheral transmission shall always start at odd numbered time slots ($CLK_1$=1).

Due to packet types that cover more than a single slot, Central transmission may continue in odd numbered slots and Peripheral transmission may continue in even numbered slots, see Figure 2.1.

All timing diagrams shown in Section 2 are based on the signals as present at the antenna. The term “exact” when used to describe timing refers to an ideal transmission or reception and neglects timing jitter and clock frequency imperfections.

The average timing of packet transmission shall not drift faster than 20 ppm relative to the ideal slot timing of 625 µs. The instantaneous timing shall not deviate more than 1 µs from the average timing. Thus, the absolute packet transmission timing $t_k$ of slot boundary $k$ shall fulfill the equation:

$$t_k = \left( \sum_{i=1}^{k} (1 + d_i) T_N \right) + j_k + \text{ offset}, \qquad \text{(EQ 1)}$$

where $T_N$ is the nominal slot length (625 µs), $j_k$ denotes jitter ($|j_k| \leq 1$ µs) at the start of slot $k$, and, $d_k$, denotes the drift ($|d_k| \leq 20$ ppm) within slot $k$. The jitter and drift can vary arbitrarily within the given limits for every slot, while *offset* is an arbitrary but fixed constant. For Hold and Sniff modes the drift and jitter parameters specified in Link Manager Protocol [Vol 2] Part C, Section 4.3.1 apply.

#### 2.2.5.1 Piconet physical channel timing

In the figures, only single-slot packets are shown as an example.

The Central TX/RX timing is shown in Figure 2.3. In Figure 2.3 and Figure 2.4 the channel hopping frequencies are indicated by *f(k)* where k is the time slot number. After transmission, a return packet is expected N × 625 µs after the start of the TX packet where N is an odd, integer larger than 0. N depends on the type of the transmitted packet.

To allow for some time slipping, an uncertainty window is defined around the exact receive timing. During normal operation, the window length shall be 20 µs, which allows the RX packet to arrive up to 10 µs too early or 10 µs too late. It is recommended that Peripherals implement variable sized windows or time tracking to accommodate a Central's absence of more than 250 ms.

During the beginning of the RX cycle, the access correlator shall search for the correct channel access code over the uncertainty window. If an event trigger does not occur the receiver may go to sleep until the next RX event. If in the course of the search, it becomes apparent that the correlation output will never exceed the final threshold, the receiver may go to sleep earlier. If a trigger event occurs, the receiver shall remain open to receive the rest of the packet unless the packet is for another device, a non-recoverable header error is detected, or a non-recoverable payload error is detected.

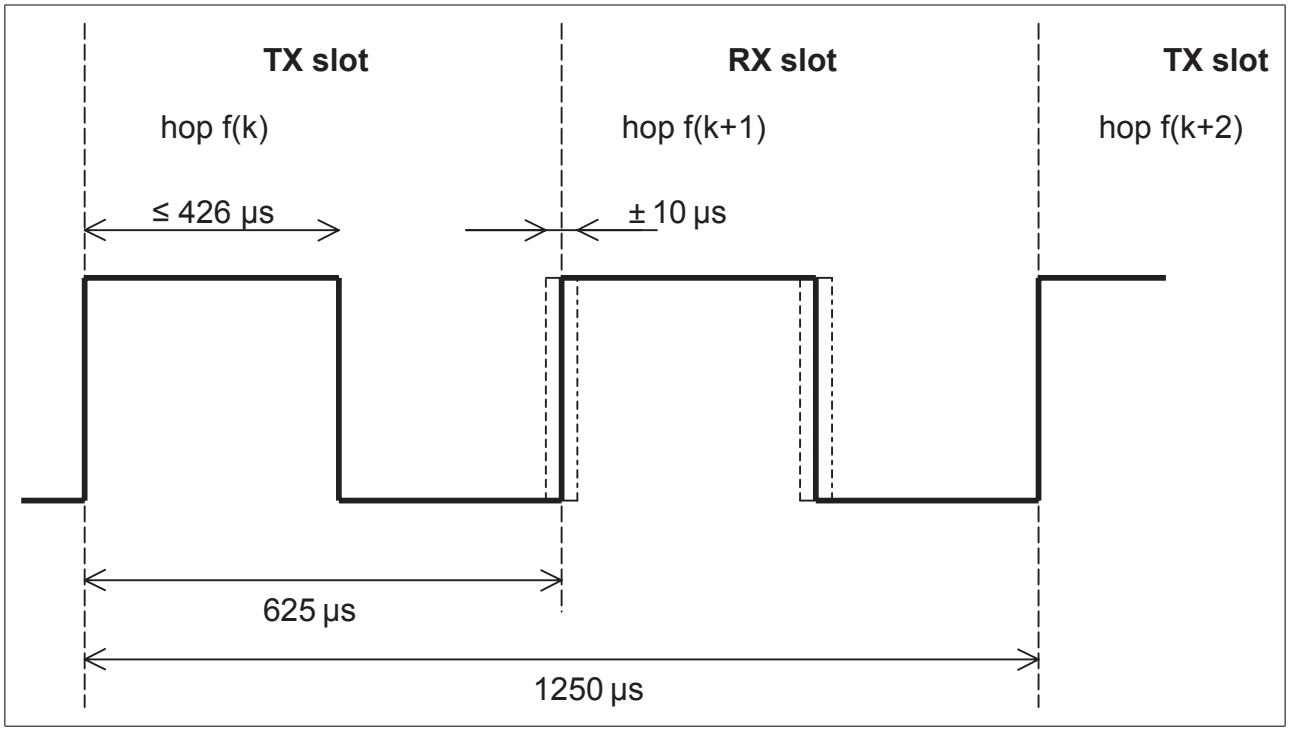



*Figure 2.3: RX/TX cycle of Central transceiver in normal mode for single-slot packets*

Each Central transmission shall be derived from bit 2 of the Central's native Bluetooth clock, thus the current transmission will be scheduled M × 1250 µs after the start of the previous Central TX burst where M depends on the transmitted and received packet type and is an even, integer larger than 0. The Central TX timing shall be derived from the Central's native Bluetooth clock, and thus it will not be affected by time drifts in the Peripheral(s).

Peripherals maintain an estimate of the Central's native clock by adding a timing offset to the Peripheral's native clock (see Section 2.2.4). This offset shall be updated each time a packet is received from the Central. By comparing the exact RX timing of the received packet with the estimated RX timing, Peripherals shall correct the offset for any timing misalignments. Since only the channel access code is required to synchronize the Peripheral, Peripheral RX timing can be corrected with any packet sent in the Central-to-Peripheral transmission slot.

The Peripheral's TX/RX timing is shown in Figure 2.4. The Peripheral's transmission shall be scheduled N × 625 µs after the start of the Peripheral's RX packet where N is an odd, positive integer larger than 0. If the Peripheral's RX timing drifts, so will its TX timing. During periods when a Peripheral is in the Active mode (see Section 8.6) and is prevented from receiving valid channel access codes from the Central due to local RF interference, Peripheral activity in a different piconet, or any other reason, the Peripheral may increase its receive uncertainty window and/or use predicted timing drift to increase the probability of receiving the Central's bursts when reception resumes.

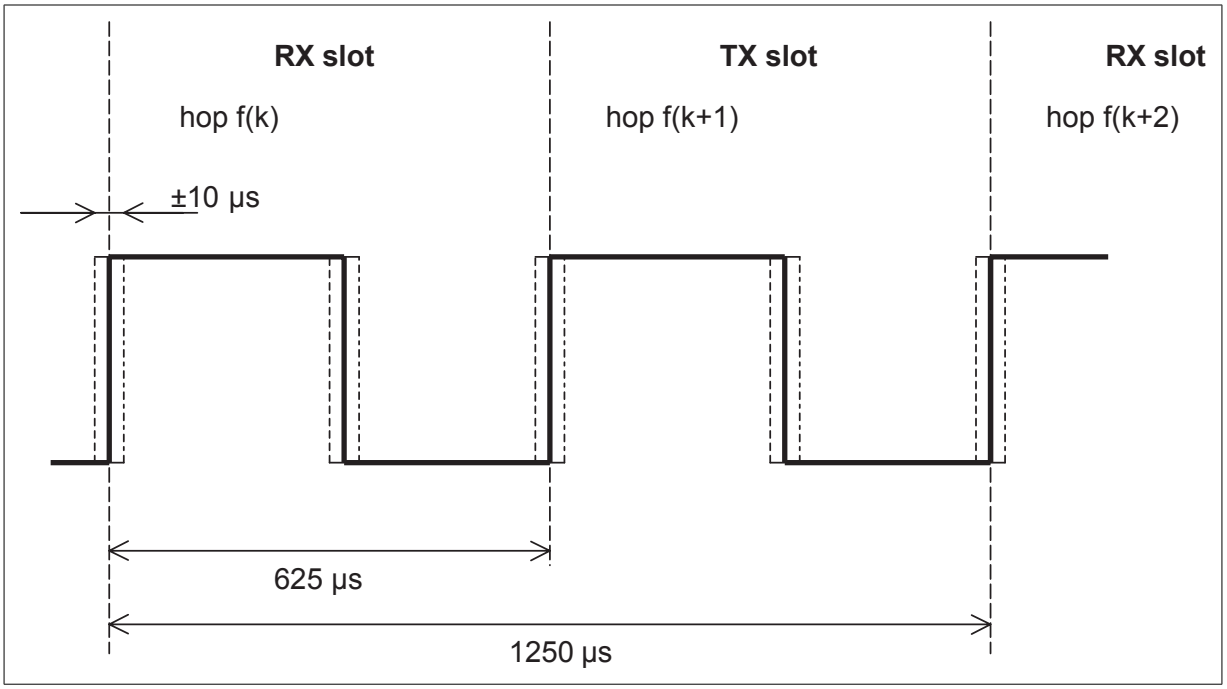



*Figure 2.4: RX/TX cycle of Peripheral transceiver in normal mode for single-slot packets*

### 2.2.5.2 Piconet physical channel re-synchronization

In the piconet physical channel, a Peripheral can lose synchronization if it does not receive a packet from the Central at least every 200 ms (or less if the low power clock is used). The Central may fail to transmit to the Peripheral due to the Central being busy with other tasks such as maintaining connections to other devices in Sniff or Hold modes, due to SCO, eSCO, or Connectionless Peripheral Broadcast activity, due to the Central being involved in a scatternet, or due to interference. When re-synchronizing to the piconet physical channel a Peripheral shall listen for the Central before it may send information (except for a Connectionless Peripheral Broadcast Receiver device which shall listen for the Transmitter but does not send information). In this case, the length of the synchronization search window in the Peripheral may be increased from 20 µs to a larger value X µs as illustrated in Figure 2.5. Only RX hop frequencies are used: the hop frequency used in the Central-to-Peripheral (RX) slot shall also be used in the uncertainty window, even when it is extended into the preceding time interval normally used for the Peripheral-to-Central (TX) slot.

If the length of search window, X, exceeds 1250 µs, consecutive windows shall avoid overlapping search windows. Consecutive windows should instead be centered at *f(k), f(k+4),... f(k+4i)* (where 'i' is an integer), which gives a maximum value X=2500 µs, or even at *f(k), f(k+6),...f(k+6i)* which gives a maximum value X=3750 µs. The RX hop frequencies used shall correspond to the Central-to-Peripheral transmission slots.

It is recommended that single slot packets are transmitted by the Central during Peripheral re-synchronization.

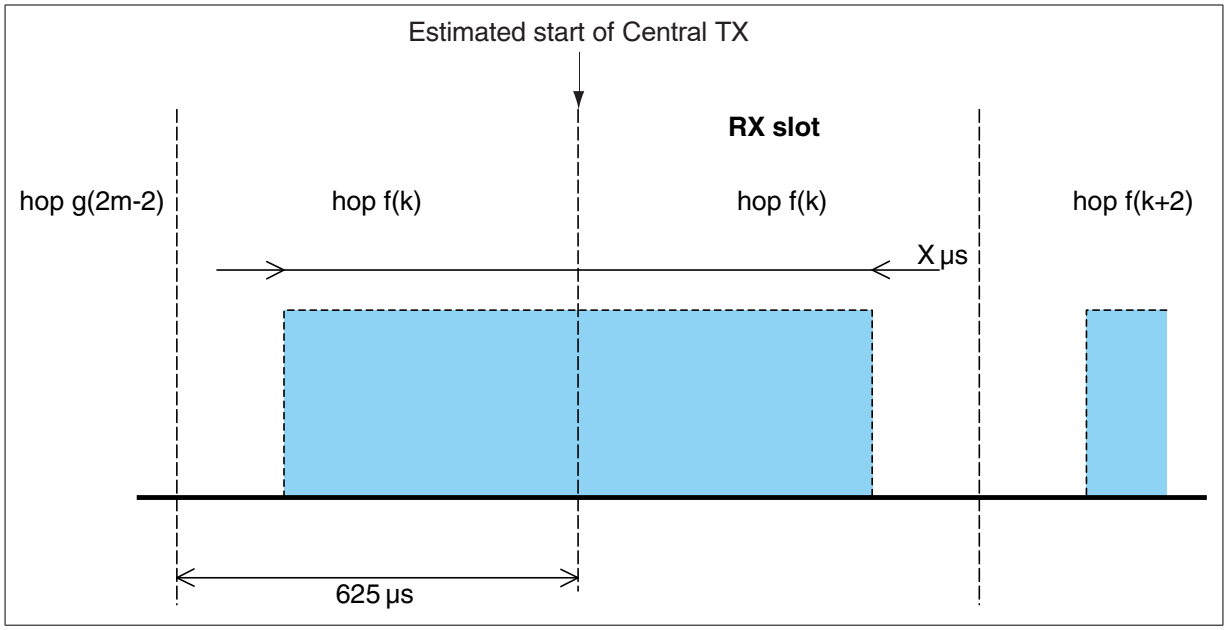



*Figure 2.5: RX timing of Peripheral returning from Hold mode*

## 2.3 Adapted piconet physical channel

For devices that enter Connectionless Peripheral Broadcast mode, the device that transmits Connectionless Peripheral Broadcast packets is the Central of the piconet and any device that receives Connectionless Peripheral Broadcast packets is a Peripheral of the piconet.

### 2.3.1 Hopping characteristics

Each physical link on the adapted piconet physical channel shall use at least $N_{min}$ RF channels (where $N_{min}$ is 20).

The adapted piconet physical channel uses the adapted channel hopping sequence described in Section 2.6.

Adapted piconet physical channels can be used for connected devices that have adaptive frequency hopping (AFH) enabled. There are two distinctions between basic and adapted piconet physical channels. The first is the same channel mechanism that makes the Peripheral frequency the same as the preceding Central transmission. The second aspect is that the adapted piconet physical channel may be based on less than the full 79 frequencies of the basic piconet physical channel. Each physical link on an adapted piconet physical channel may use a different set of frequencies.

## 2.4 Page scan physical channel

Although Central and Peripheral roles are not defined prior to a connection, the term *Central* is used for the paging device (that becomes a Central in the Connection state)

and *Peripheral* is used for the page scanning device (that becomes a Peripheral in the Connection state).

### 2.4.1 Clock estimate for paging

A paging device uses an estimate of the native clock of the page scanning device, CLKE; i.e. an offset shall be added to the CLKN of the pager to approximate the CLKN of the recipient, see Figure 2.6. CLKE shall be derived from the native CLKN by adding an offset. By using the CLKN of the recipient, the pager might be able to speed up the connection establishment.

Note: CLKR is never used for deriving CLKE or for any other control of the hopping kernel.

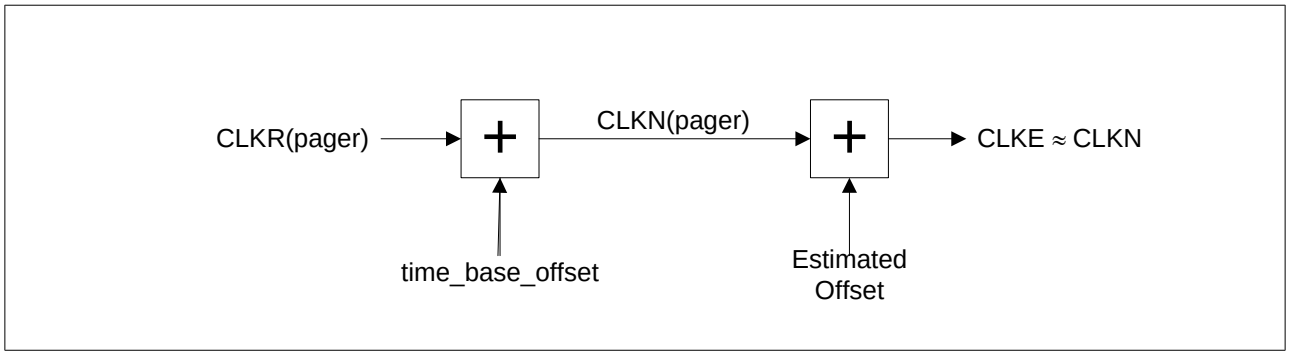



*Figure 2.6: Derivation of CLKE*

### 2.4.2 Hopping characteristics

The page scan physical channel follows a slower hopping pattern than the basic piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the page scan channel shall be determined by the native Bluetooth clock of the scanning device. The frequency hopping sequence is determined by the Bluetooth address of the scanning device.

The page scan physical channel uses the page, Central page response, Peripheral page response, and page scan hopping sequences specified in Section 2.6.

### 2.4.3 Paging procedure timing

During the paging procedure, the Central shall transmit paging messages (see Table 8.3) corresponding to the Peripheral to be connected. Since the paging message is a very short packet, the hop rate is 3200 hops per second. In a single TX slot interval, the paging device shall transmit on two different hop frequencies. In Figure 2.7 to Figure 2.11, *f(k)* is used for the frequencies of the page hopping sequence and *f'(k)* denotes the corresponding page response sequence frequencies. The first transmission starts where $CLK_0 = 0$ and the second transmission starts where $CLK_0 = 1$.

In a single RX slot interval, the paging device shall listen for the Peripheral page response message on two different hop frequencies. Similar to transmission, the nominal reception starts where $CLK_0$ = 0 and the second reception nominally starts where $CLK_0$ = 1; see Figure 2.7. During the TX slot, the paging device shall send the paging message at the TX hop frequencies *f(k)* and *f(k+1)*. In the RX slot, it shall listen for a response on the corresponding RX hop frequencies *f'(k)* and *f'(k+1)*. The listening periods shall be exactly timed 625 µs after the corresponding paging packets, and shall include a ±10 µs uncertainty window.

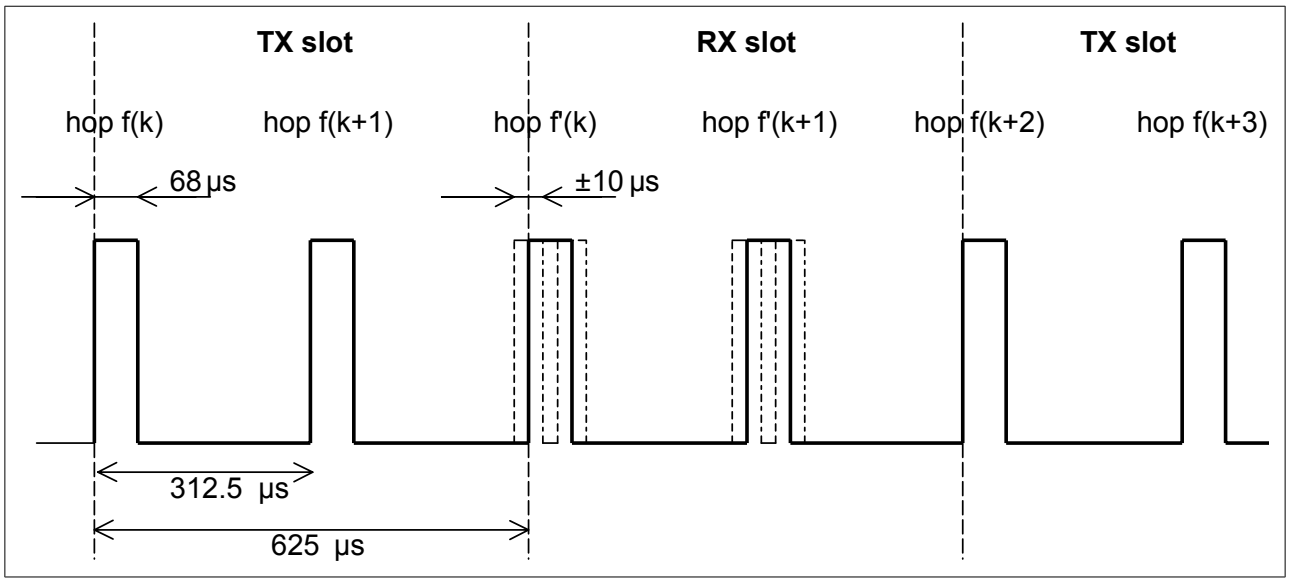



*Figure 2.7: RX/TX cycle of transceiver in Page mode*

### 2.4.4 Page response timing

At connection setup a Central page response packet is transmitted from the Central to the Peripheral (see Table 8.3). This packet establishes the timing and frequency synchronization. After the Peripheral has received the page message, it shall return a response message that consists of the Peripheral page response packet and shall follow 625 µs after the receipt of the page message. The Central shall send the Central page response packet in the TX slot following the RX slot in which it received the Peripheral's response, according to the RX/TX timing of the Central. The time difference between the Peripheral page response and Central page response message will depend on the timing of the page message the Peripheral received. In Figure 2.8, the Peripheral receives the paging message sent **first** in the Central-to-Peripheral slot. It then responds with a first Peripheral page response packet in the first half of the Peripheral-to-Central slot. The timing of the Central page response packet is based on the timing of the page message sent first in the preceding Central-to-Peripheral slot: there is an exact 1250 µs delay between the first page message and the Central page response packet. The packet is sent at the hop frequency *f(k+1)* which is the hop frequency following the hop frequency *f(k)* the page message was received in.

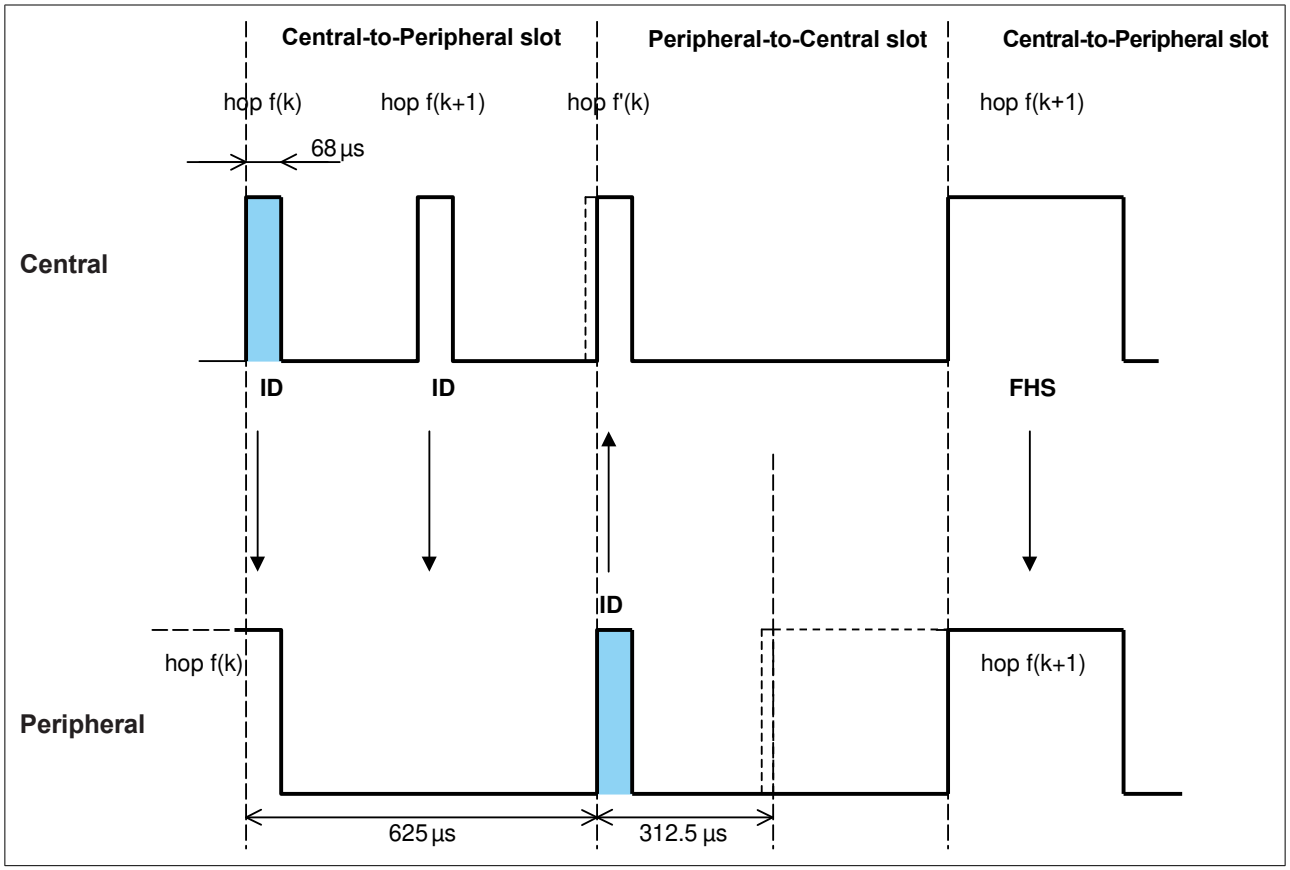



*Figure 2.8: Timing of page response packets on successful page in first half slot*

In Figure 2.9, the Peripheral receives the paging message sent **second** in the Central-to-Peripheral slot. It then responds with a Peripheral page response packet in the second half of the Peripheral-to-Central slot exactly 625 µs after the receipt of the page message. The timing of the Central page response packet is still based on the timing of the page message sent **first** in the preceding Central-to-Peripheral slot: there is an exact 1250 µs delay between the **first** page message and the Central page response packet. The packet is sent at the hop frequency $f(k+2)$ which is the hop frequency following the hop frequency $f(k+1)$ the page message was received in.

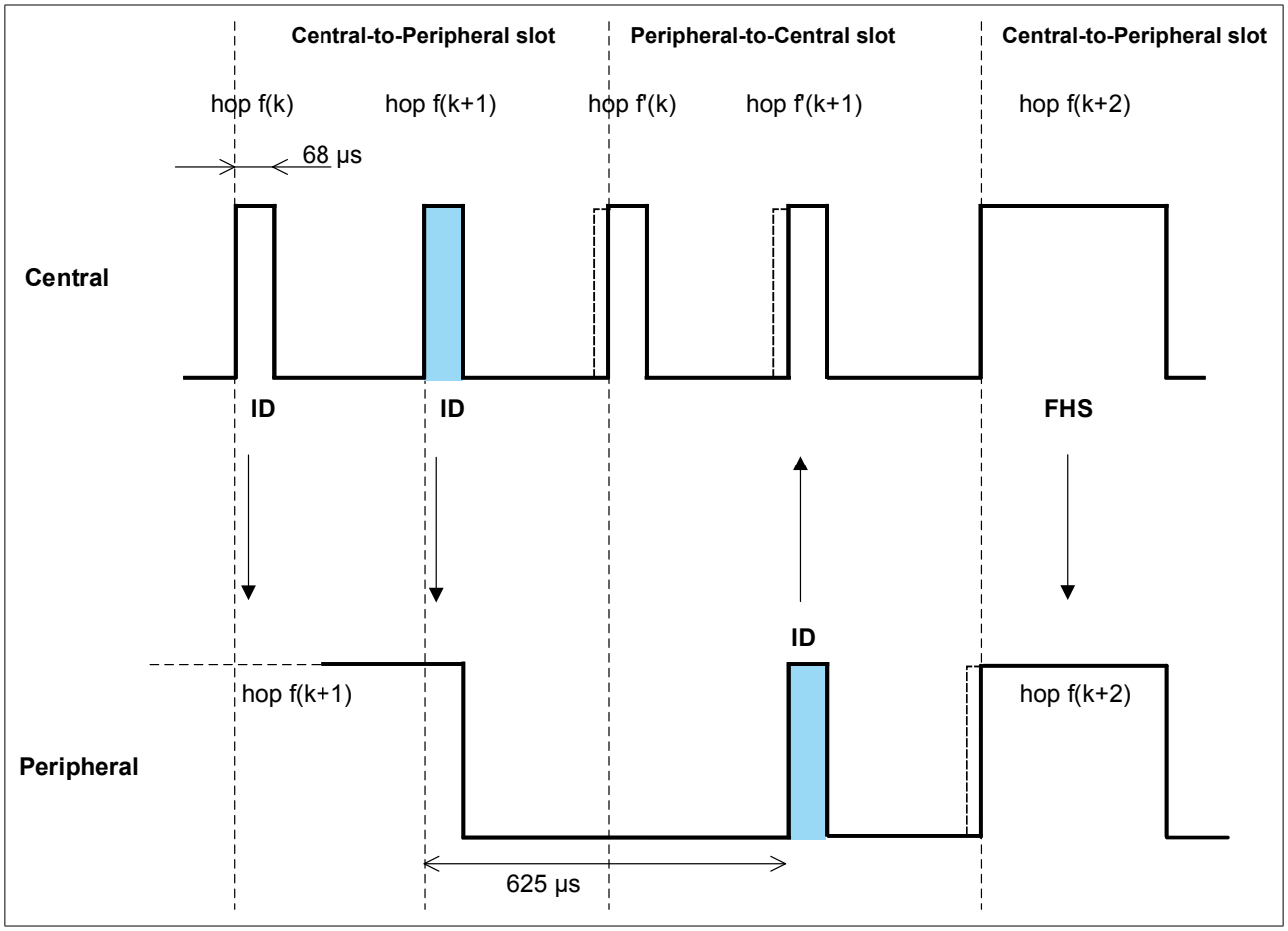



*Figure 2.9: Timing of page response packets on successful page in second half slot*

The Peripheral shall adjust its RX/TX timing according to the reception of the Central page response packet (and not according to the reception of the page message). That is, the second Peripheral page response message that acknowledges the reception of the Central page response packet shall be transmitted 625 µs after the start of the Central page response packet.

## 2.5 Inquiry scan physical channel

An inquiring device transmits inquiry messages. A device scanning for inquiries is the discoverable device.

The inquiring device transmits in slots that are designated Central-to-Peripheral slots in other contexts while the discoverable device transmits in slots that are designated Peripheral-to-Central slots in other contexts. These terms are used in describing the inquiry process to avoid the need for two names for the same concept in other places.

### 2.5.1 Clock for inquiry

The clock used for inquiry and inquiry scan shall be the device's native clock.

### 2.5.2 Hopping characteristics

The inquiry scan channel follows a slower hopping pattern than the piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the inquiry scan channel is determined by the native Bluetooth clock of the scanning device while the frequency hopping sequence is determined by the general inquiry access code.

The inquiry scan physical channel uses the inquiry, inquiry response, and inquiry scan hopping sequences described in Section 2.6.

### 2.5.3 Inquiry procedure timing

During the inquiry procedure, the inquiring device shall transmit inquiry messages with the general or dedicated inquiry access code. The timing for inquiry is the same as for paging (see Section 2.4.3).

### 2.5.4 Inquiry response timing

An inquiry response packet is transmitted from the discoverable device to the inquiring device after the discoverable device has received an inquiry message (see Table 8.5). This packet contains information necessary for the inquiring device to page the discoverable device (see definition of the FHS packet in Section 6.5.1.4) and follows 625 µs after the receipt of the inquiry message. If the discoverable device transmits an extended inquiry response packet, then it shall be transmitted 1250 µs after the start of the inquiry response packet.

In Figure 2.10 and Figure 2.11, *f(k)* is used for the frequencies of the inquiry hopping sequence and f'(k) denotes the corresponding inquiry response sequence frequency. The inquiry response packet and the extended inquiry response packet are received by the inquiring device at the hop frequency *f'(k)* when the inquiry message received by the discoverable device was first in the Central-to-Peripheral slot.

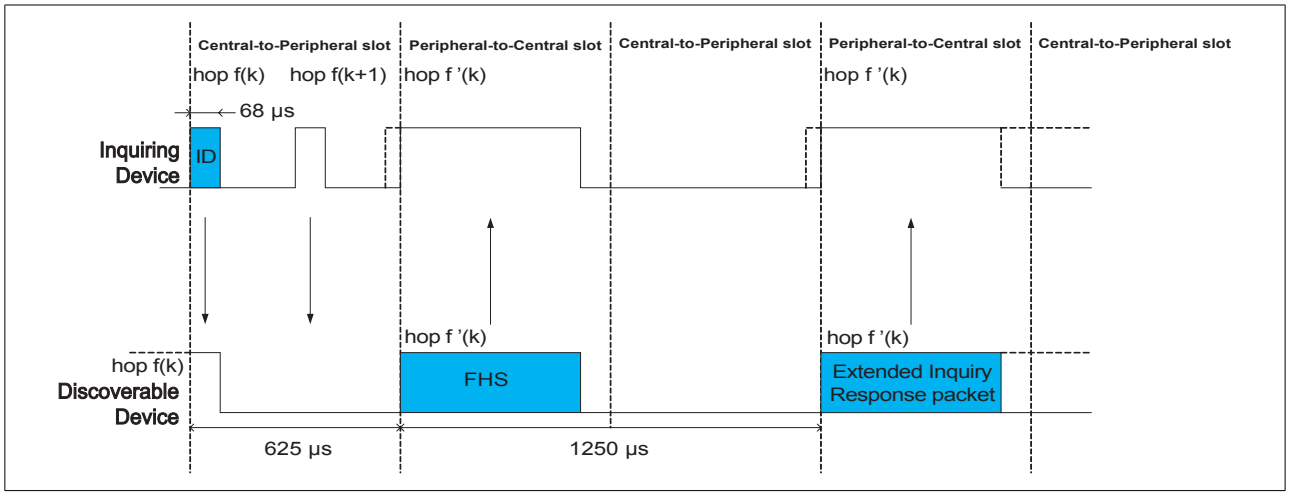



*Figure 2.10: Timing of inquiry response packets on successful inquiry in first half slot*

When the inquiry message received by the discoverable device was the second in the Central-to-Peripheral slot the inquiry response packet and the extended inquiry response packet are received by the inquiring device at the hop frequency *f'(k+1)*.

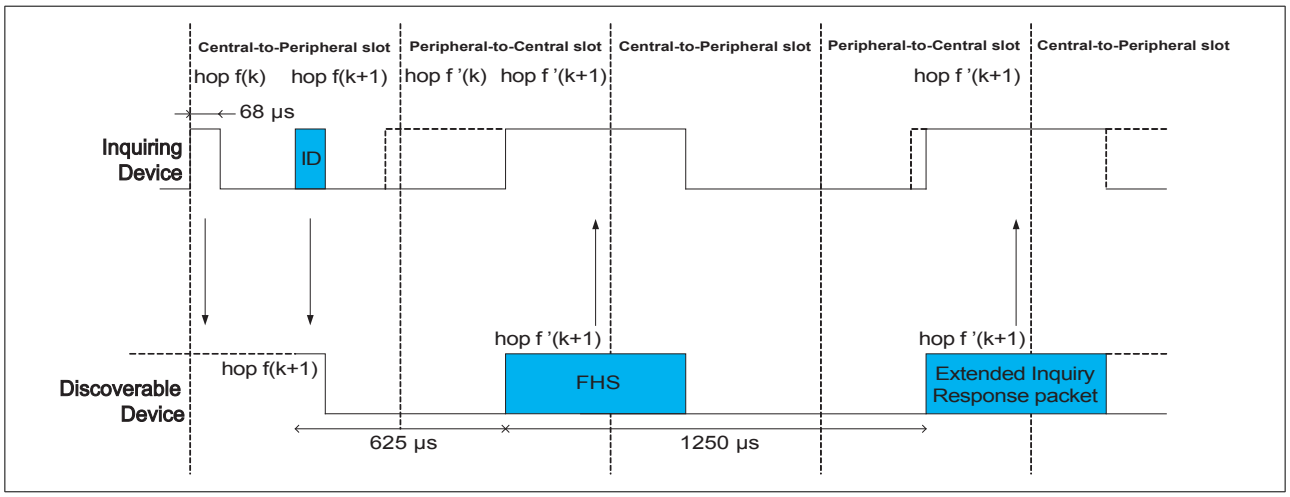



*Figure 2.11: Timing of inquiry response packets on successful inquiry in second half slot*

## 2.6 Hop selection

Bluetooth devices shall use the hopping kernel as defined in the following sections.

In total, six types of hopping sequence are defined – five for the basic hop system and one for an adapted set of hop locations used by adaptive frequency hopping (AFH). These sequences are:

- A page hopping sequence with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32.

- A page response hopping sequence covering 32 response frequencies that are in a one-to-one correspondence to the current page hopping sequence. The Central and Peripheral use different rules to obtain the same sequence.
- An inquiry hopping sequence with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32.
- An inquiry response hopping sequence covering 32 response frequencies that are in a one-to-one correspondence to the current inquiry hopping sequence.
- A basic channel hopping sequence which has a very long period length, which does not show repetitive patterns over a short time interval, and which distributes the hop frequencies equally over the 79 MHz during a short time interval.
- An adapted channel hopping sequence derived from the basic channel hopping sequence which uses the same channel mechanism and may use fewer than 79 frequencies. The adapted channel hopping sequence is used in place of the basic channel hopping sequence and for Connectionless Peripheral Broadcast. All other hopping sequences are not affected by hop sequence adaptation.

In addition, a set of synchronization train RF channels with 3 fixed frequencies is defined.

### 2.6.1 General selection scheme

The selection scheme consists of two parts:

- selecting a sequence;
- mapping this sequence onto the hop frequencies.

The general block diagram of the hop selection scheme is shown in Figure 2.12. The mapping from the input to a particular RF channel index is performed in the selection box.

The inputs to the selection box are the selected clock, frozen clock, N, $k_{offset}$, interlace_offset, address, sequence selection and AFH_channel_map. The source of the clock input depends on the hopping sequence selected. Additionally, each hopping sequence uses different bits of the clock (see Table 2.2). N, interlace_offset, and $k_{offset}$ are defined in Section 2.6.4.

The *sequence selection* input can be set to the following values:

- page scan
- inquiry scan
- page

- inquiry
- Central page response
- Peripheral page response
- inquiry response
- basic channel
- adapted channel

The address input consists of 28 bits as specified in Table 2.3 (see Section 2.6.4). The hopping sequence is selected by the sequence selection input to the selection box.

When the adapted channel hopping sequence is selected, the *AFH_channel_map* is an additional input to the selection box. The *AFH_channel_map* indicates which channels shall be *used* and which shall be *unused*. These terms are defined in Section 2.6.3.

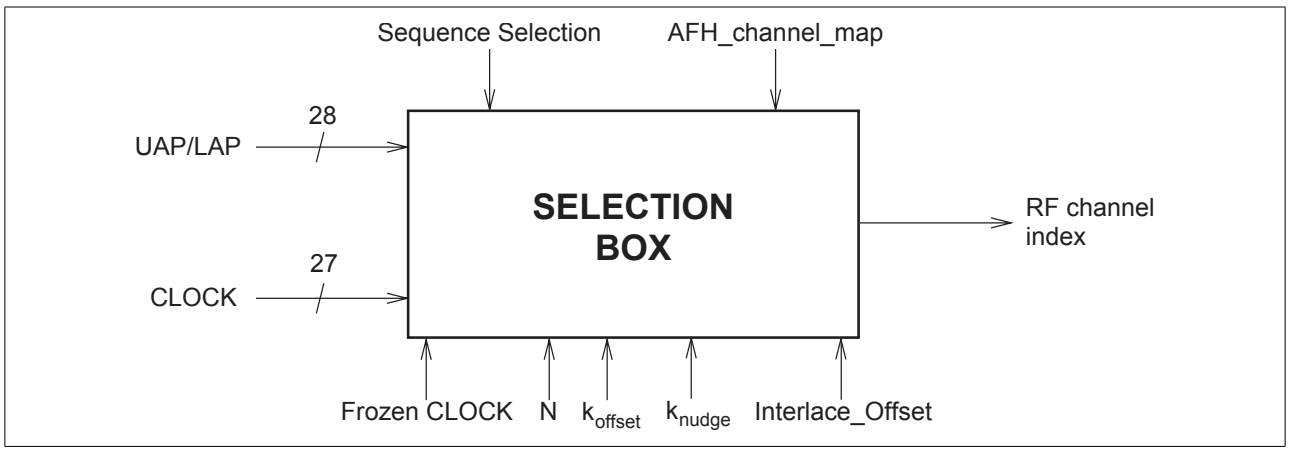



*Figure 2.12: General block diagram of hop selection scheme*

The output, *RF channel index*, constitutes a pseudo-random sequence. The RF channel index is mapped to RF channel frequencies using the equation in [Vol 2] Part A, Table 2.1.

The selection scheme chooses a segment of 32 hop frequencies spanning about 64 MHz and visits these hops in a pseudo-random order. Next, a different 32-hop segment is chosen, etc. In the page, Central page response, Peripheral page response, page scan, inquiry, inquiry response and inquiry scan hopping sequences, the same 32-hop segment is used all the time (the segment is selected by the address; different devices will have different paging segments). When the basic channel hopping sequence is selected, the output constitutes a pseudo-random sequence that slides through the 79 hops. The principle is depicted in Figure 2.13.

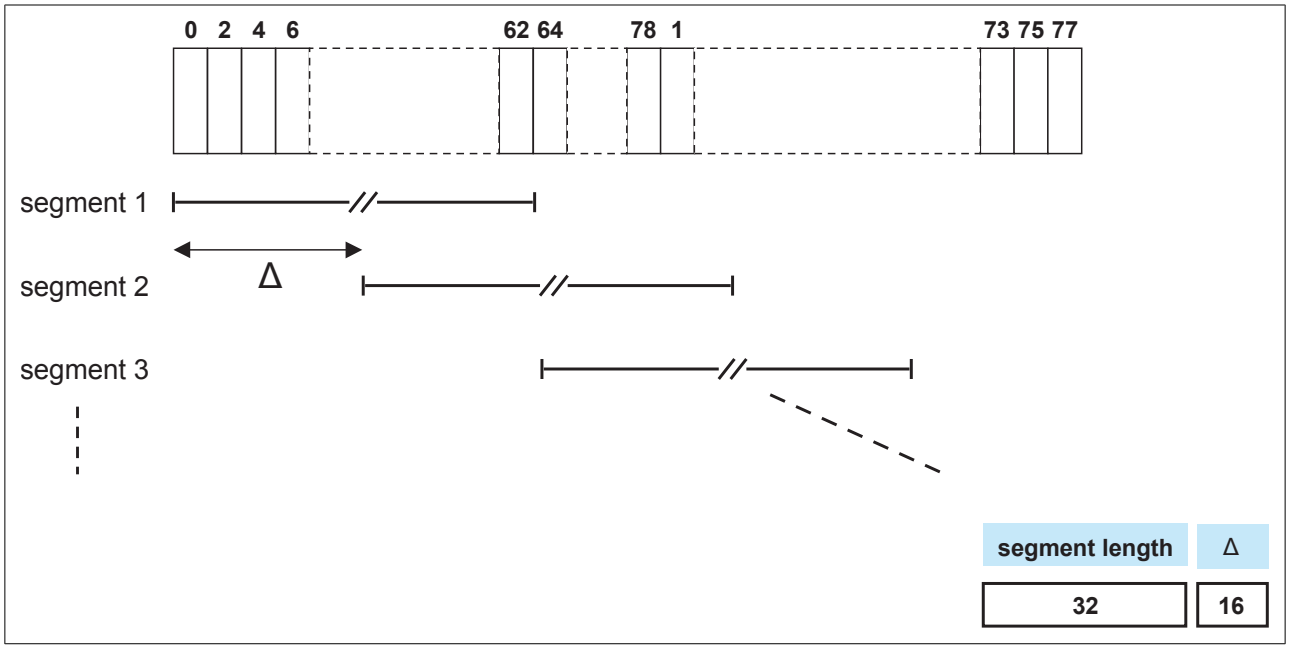



*Figure 2.13: Hop selection scheme in Connection state*

The RF frequency shall remain fixed for the duration of the packet. The RF frequency for the packet shall be derived from the Bluetooth clock value in the first slot of the packet. The RF frequency in the first slot after a multi-slot packet shall use the frequency as determined by the Bluetooth clock value for that slot. Figure 2.14 illustrates the hop definition on single- and multi-slot packets.

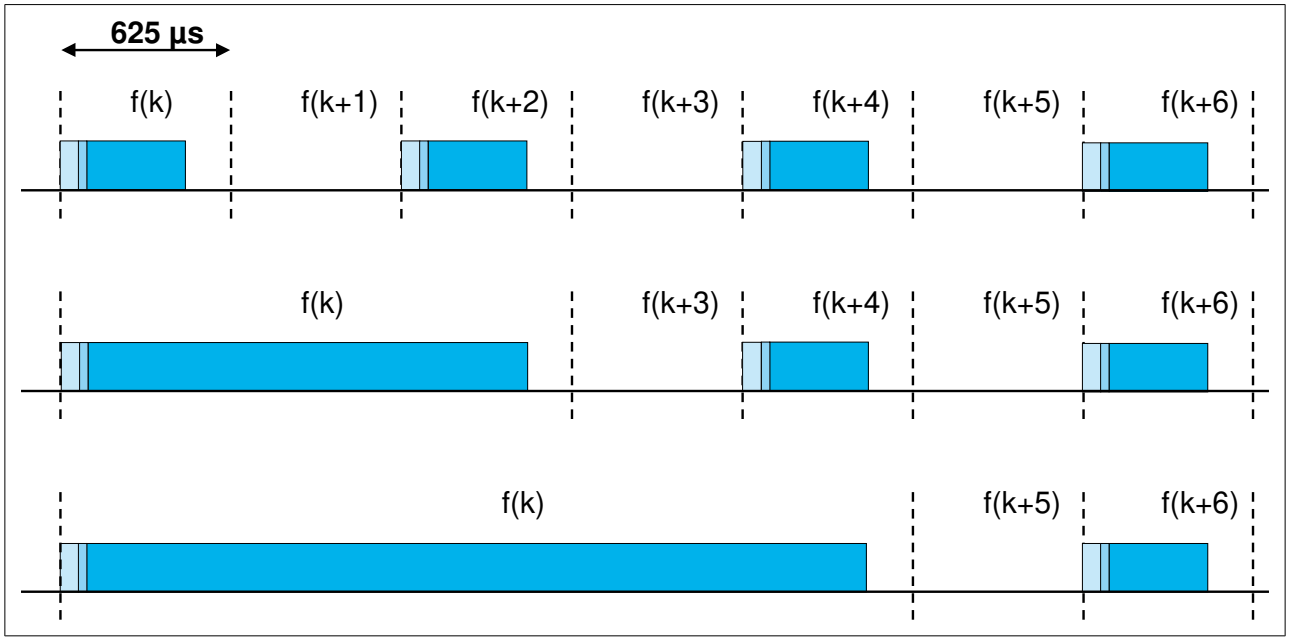



*Figure 2.14: Single- and multi-slot packets*

When the adapted channel hopping sequence is used, the pseudo-random sequence contains only frequencies that are in the RF channel set defined by the *AFH_channel_map* input. The adapted sequence has similar statistical properties to the non-adapted hop sequence. In addition, the Peripheral responds with its packet on the same RF channel that was used by the Central to address that Peripheral (or would have been in the case of a synchronous reserved slot without a validly received Central-to-Peripheral transmission). This is called the *same channel mechanism* of AFH. Thus, the RF channel used for the Central to Peripheral packet is also used for the immediately following Peripheral to Central packet. An example of the same channel mechanism is illustrated in Figure 2.15. The same channel mechanism shall be used whenever the adapted channel hopping sequence is selected.

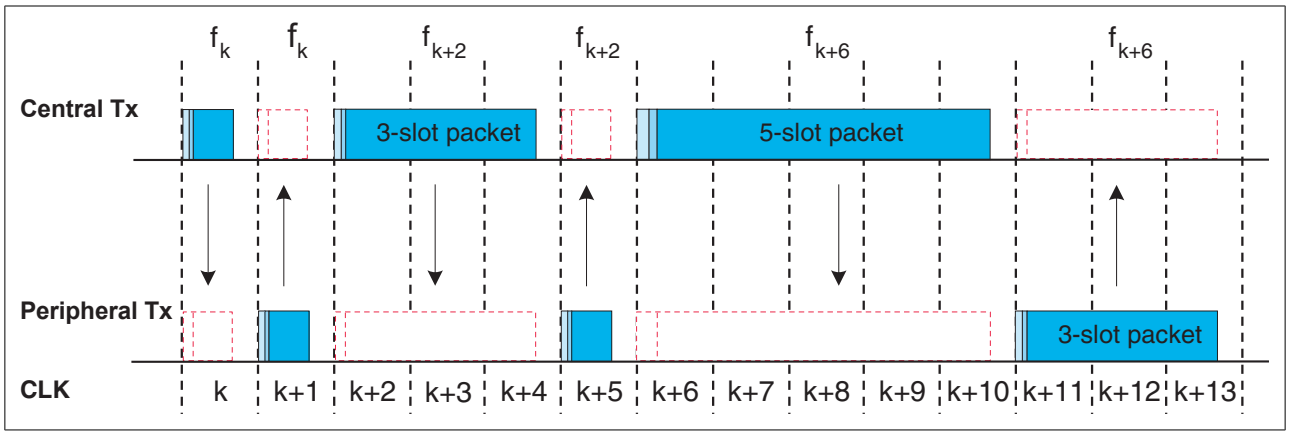



*Figure 2.15: Example of the same channel mechanism*

### 2.6.2 Selection kernel

The basic hop selection kernel shall be as shown in Figure 2.16 and is used for the page, page response, inquiry, inquiry response and basic channel hopping selection kernels. In these substates the AFH_channel_map input is unused. The adapted channel hopping selection kernel is described in Section 2.6.3. The X input determines the phase in the 32-hop segment, whereas Y1 and Y2 selects between Central-to-Peripheral and Peripheral-to-Central. The inputs A to D determine the ordering within the segment, the inputs E and F determine the mapping onto the hop frequencies. The kernel addresses a register containing the RF channel indices. This list is ordered so that first all even RF channel indices are listed and then all odd hop frequencies. In this way, a 32-hop segment spans about 64 MHz.

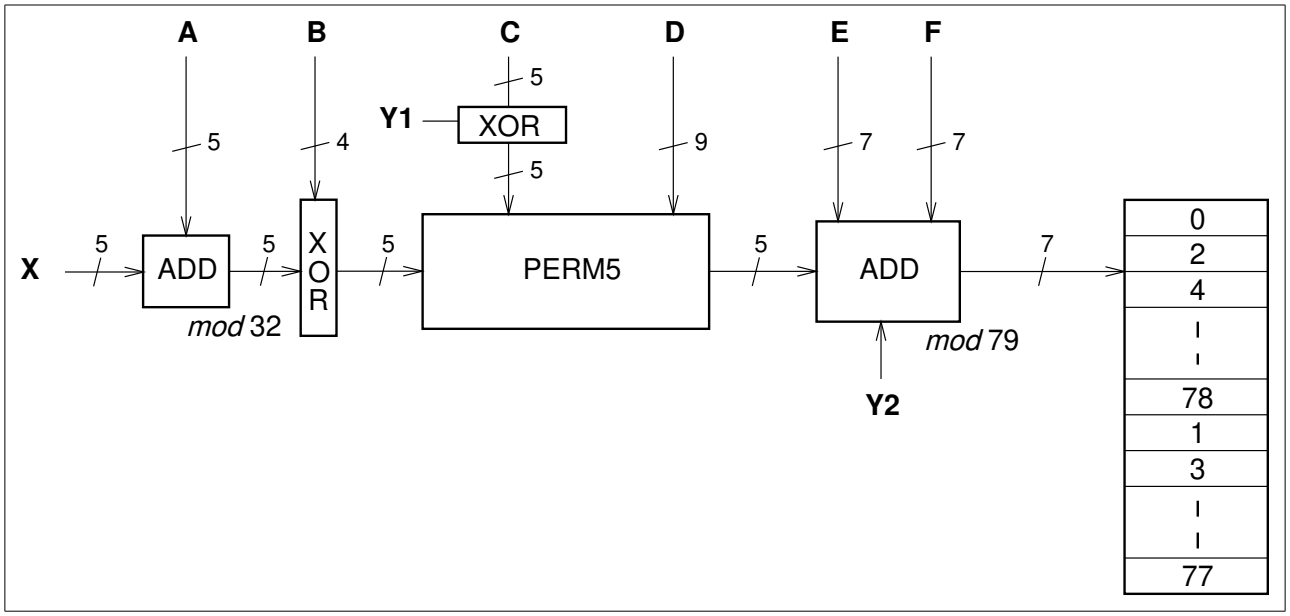



*Figure 2.16: Block diagram of the basic hop selection kernel for the hop system*

The selection procedure consists of an addition, an XOR operation, a permutation operation, an addition, and finally a register selection. In Section 2.6.2 and Table 2.2, the notation $A_i$ is used for bit i of the BD_ADDR.

### 2.6.2.1 First addition operation

The first addition operation only adds a constant to the phase and applies a *mod* 32 operation. For the page hopping sequence, the first addition is redundant since it only changes the phase within the segment. However, when different segments are concatenated (as in the basic channel hopping sequence), the first addition operation will have an impact on the resulting sequence.

### 2.6.2.2 XOR operation

In the XOR operation, the four LSBs of the output of the first addition (designated Z′ here) are XORed with the address bits $A_{22\text{-}19}$, while the $Z'_4$ bit is left unaltered. The operation is illustrated in Figure 2.17.

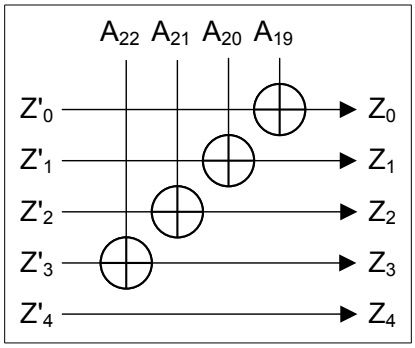



*Figure 2.17: XOR operation for the hop system*

### 2.6.2.3 Permutation operation

The permutation operation involves the switching from 5 inputs to 5 outputs for the hop system, controlled by the control word. The permutation or switching box shall be as shown in Figure 2.18. It consists of 7 stages of butterfly operations. The control of the butterflies by the control signals P is shown in Table 2.1. $P_{0\text{-}8}$ corresponds to $D_{0\text{-}8}$, and, $P_{i+9}$ corresponds to $C_i \oplus Y1$ for $i = 0 \ldots 4$ in Figure 2.16.

| Control signal | Butterfly |
|---|---|
| $P_0$ | $\{Z_0,Z_1\}$ |
| $P_1$ | $\{Z_2,Z_3\}$ |
| $P_2$ | $\{Z_1,Z_2\}$ |
| $P_3$ | $\{Z_3,Z_4\}$ |
| $P_4$ | $\{Z_0,Z_4\}$ |
| $P_5$ | $\{Z_1,Z_3\}$ |
| $P_6$ | $\{Z_0,Z_2\}$ |

| Control signal | Butterfly |
|---|---|
| $P_7$ | $\{Z_3,Z_4\}$ |
| $P_8$ | $\{Z_1,Z_4\}$ |
| $P_9$ | $\{Z_0,Z_3\}$ |
| $P_{10}$ | $\{Z_2,Z_4\}$ |
| $P_{11}$ | $\{Z_1,Z_3\}$ |
| $P_{12}$ | $\{Z_0,Z_3\}$ |
| $P_{13}$ | $\{Z_1,Z_2\}$ |

*Table 2.1: Control of the butterflies for the hop system*

The Z input is the output of the XOR operation as described in the previous section. The butterfly operation can be implemented with multiplexers as depicted in Figure 2.19.

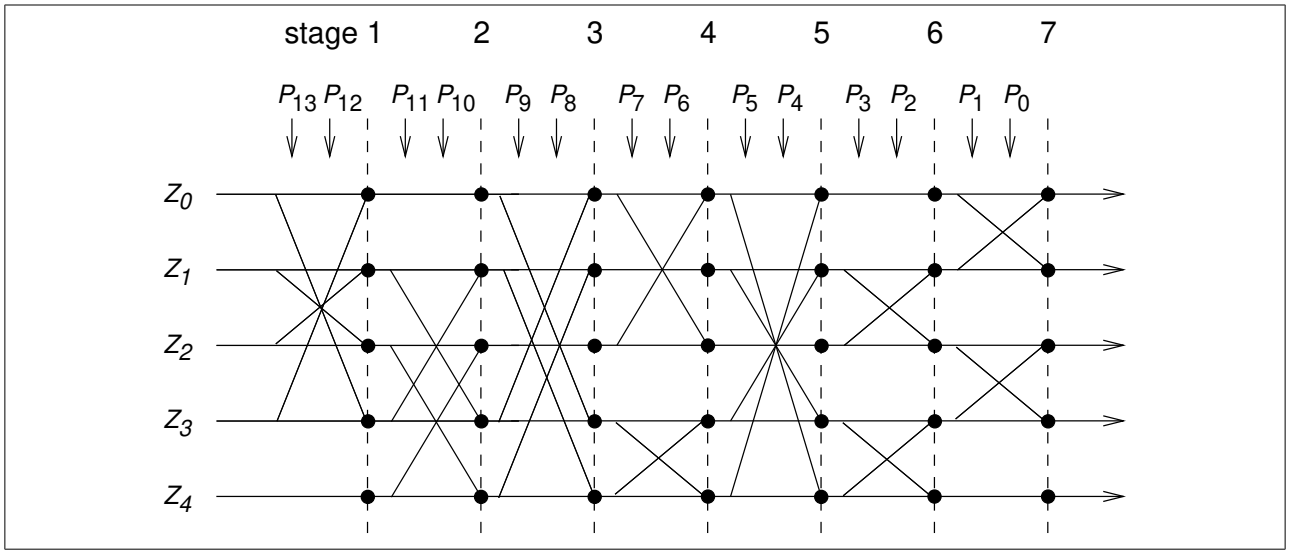



*Figure 2.18: Permutation operation for the hop system*

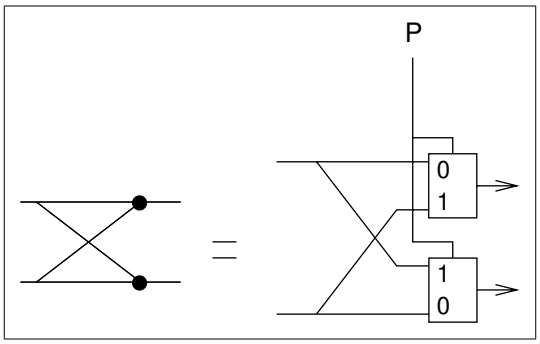



*Figure 2.19: Butterfly implementation*

#### 2.6.2.4 Second addition operation

The addition operation selects the hop frequencies to be used by the current segment and also switches between Central-to-Peripheral and Peripheral-to-Central. It adds the output of the current permutation (which selects a channel within the segment), a constant, the clock bit selecting Central or Peripheral slots, and a value used to switch each segment to a new set of channels in Connection state. The addition is applied *mod* 79.

#### 2.6.2.5 Register bank

The output of the adder addresses a bank of 79 registers. The registers are loaded with the synthesizer code words corresponding to the hop frequencies 0 to 78.

The upper half of the bank contains the even hop frequencies, whereas the lower half of the bank contains the odd hop frequencies.

### 2.6.3 Adapted hop selection kernel

The adapted hop selection kernel is based on the basic hop selection kernel defined in the preceding sections.

The inputs to the adapted hop selection kernel are the same as for the basic hop system kernel except that the input *AFH_channel_map* (defined in Link Manager Protocol [Vol 2] Part C, Section 5.2) is used. The *AFH_channel_map* indicates which RF channels shall be *used* and which shall be *unused*. When hop sequence adaptation is enabled, the number of *used* RF channels may be reduced from 79 to some smaller value N. All devices shall be capable of operating on an adapted hop sequence (AHS) with $N_{min} \leq N \leq 79$, with any combination of *used* RF channels within the *AFH_channel_map* that meets this constraint. $N_{min}$ is defined in Section 2.3.1.

Adaptation of the hopping sequence is achieved through two additions to the basic channel hopping sequence according to Figure 2.16:

- *Unused* RF channels are re-mapped uniformly onto *used* RF channels. That is, if the hop selection kernel of the basic system generates an *unused* RF channel, an alternative RF channel out of the set of *used* RF channels is selected pseudo-randomly.
- The *used* RF channel generated for the Central-to-Peripheral packet is also used for the immediately following Peripheral-to-Central packet (see Section 2.6.1).

#### 2.6.3.1 Channel re-mapping function

When the adapted hop selection kernel is selected, the basic hop selection kernel according to Figure 2.16 is initially used to determine an RF channel. If this RF channel is *unused* according to the *AFH_channel_map*, the *unused* RF channel is re-mapped by the re-mapping function to one of the *used* RF channels. If the RF channel determined by the basic hop selection kernel is already in the set of *used* RF channels, no adjustment is made. The hop sequence of the (non-adapted) basic hop equals the sequence of the adapted selection kernel on all locations where *used* RF channels are generated by the basic hop. This property facilitates all Peripherals remaining synchronized even if they are not all using the same hopping sequence.

A block diagram of the re-mapping mechanism is shown in Figure 2.20. The re-mapping function is a post-processing step to the selection kernel from Figure 2.16, denoted as ‘Hop selection of the basic hop’. The output $f_k$ of the basic hop selection kernel is an RF channel number that ranges between 0 and 78. This RF channel will either be in the set of *used* RF channels or in the set of *unused* RF channels.

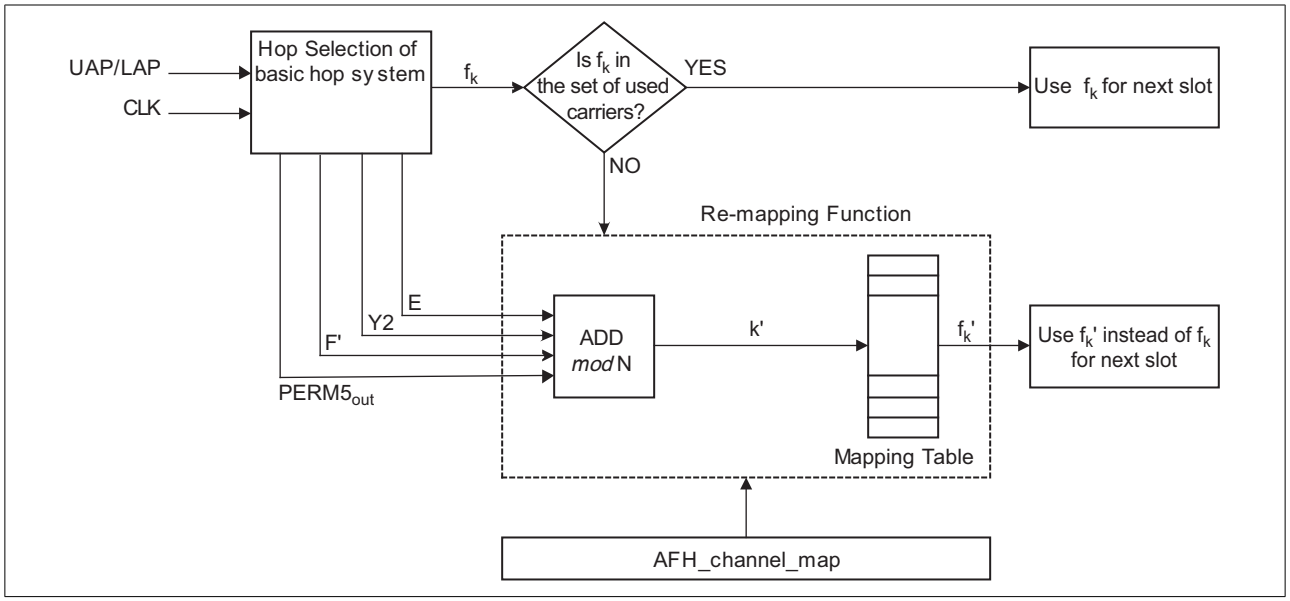



*Figure 2.20: Block diagram of adaptive hop selection mechanism*

When an unused RF channel is generated by the basic hop selection mechanism, it is re-mapped to the set of *used* RF channels as follows. A new index *k'* ∈ {0, 1,..., *N*-1} is calculated using some of the parameters from the basic hop selection kernel:

$$k' = (PERM5_{out} + E + F' + Y2) \bmod N$$

where *F'* is defined in Table 2.2. The index *k'* is then used to select the re-mapped channel from a mapping table that contains all of the even *used* RF channels in ascending order followed by all the odd *used* RF channels in ascending order (i.e., the mapping table of Figure 2.16 with all the *unused* RF channels removed).

### 2.6.4 Control word

The control word of the kernel is controlled by the overall control signals X, Y1, Y2, A to F, and F' as illustrated in Figure 2.16 and Figure 2.20. During paging and inquiry, the inputs A to E use the address values as given in the corresponding columns of Table 2.2. In addition, the inputs X, Y1 and Y2 are used. The F and F' inputs are unused. The clock bits $CLK_{6-2}$ (i.e., input X) specify the phase within the length 32 sequence. $CLK_1$ (i.e., inputs Y1 and Y2) is used to select between TX and RX. The address inputs determine the sequence order within segments. The final mapping onto the hop frequencies is determined by the register contents.

During the Connection state (see Section 8.5), the inputs A, C and D shall be derived from the address bits being bit-wise XORed with the clock bits as shown in the "Connection state" column of Table 2.2 (the two most significant bits, MSBs, are XORed together, the two second MSBs are XORed together, etc.).

| | **(a) Page scan**<br>**(b) Generalized Interlaced Page Scan**<br>**(c) Inquiry scan**<br>**(d) Generalized Interlaced Inquiry Scan** | **(e) Page**<br>**(f) Inquiry** | **(g) Central page response**<br>**(h) Peripheral page response**<br>**(k) Inquiry response** | **Connection state** |
|---|---|---|---|---|
| X | (a) $CLKN_{16-12}$<br>(b) ($CLKN_{16-12}$ + interlace offset) *mod* 32<br>(c) $Xir_{4-0}$<br>(d) ($Xir_{4-0}$ + interlace offset) *mod* 32 | (e) $Xp_{4-0}$<br>(f) $Xi_{4-0}$ | (g) $Xprc_{4-0}$<br>(h) $Xprp_{4-0}$<br>(k) $Xir_{4-0}$ | $CLK_{6-2}$ |
| Y1 | 0 | (e) $CLKE_1$<br>(f) $CLKN_1$ | (g) $CLKE_1$<br>(h) $CLKN_1$<br>(k) 1 | $CLK_1$ |
| Y2 | 32 × Y1 | | | |
| A | $A_{27-23}$ | | | $A_{27-23} \oplus CLK_{25-21}$ |
| B | $A_{22-19}$ | | | |
| C | $A_{8,6,4,2,0}$ | | | $A_{8,6,4,2,0} \oplus CLK_{20-16}$ |
| D | $A_{18-10}$ | | | $A_{18-10} \oplus CLK_{15-7}$ |
| E | $A_{13,11,9,7,5,3,1}$ | | | |
| F | 0 | | | 16 × $CLK_{27-7}$ *mod* 79 |
| F’ | n/a | | | 16 × $CLK_{27-7}$ *mod N* |

*Table 2.2: Control for hop system*

The five X input bits vary depending on the current state of the device. In the Page Scan and Inquiry Scan substates, the native clock (CLKN) shall be used. In Connection state the Central's clock (CLK) shall be used as input. The situation is somewhat more complicated for the other states.

The address bits $A_0$ to $A_{27}$ depend on the sequence selection input as specified in Table 2.3.

| **Sequence** | **$A_{23–0}$** | **$A_{27–24}$** |
|---|---|---|
| Page scan<br>Page<br>Central page response<br>Peripheral page response | LAP of the device being paged | $UAP_{3–0}$ of the device being paged |
| Inquiry scan<br>Inquiry<br>Inquiry response | GIAC (0x9E8B33) | DCI (0x00) |

| Sequence | $A_{23–0}$ | $A_{27–24}$ |
|---|---|---|
| Basic channel<br>Adapted channel | LAP of the Central | $UAP_{3–0}$ of the Central |

*Table 2.3: Address bits for each sequence selection input*

#### 2.6.4.1 Page scan and inquiry scan hopping sequences

For the transmitted access code and in the receiver correlator, the appropriate GIAC or DIAC shall be used. The application decides which inquiry access code to use depending on the purpose of the inquiry.

#### 2.6.4.2 Page hopping sequence

When the sequence selection input is set to page, the paging device shall start using the **A**-train, i.e., $(f(k-8), \ldots, f(k), \ldots, f(k+7))$ , where *f*(*k*) is the source's estimate of the current receiver frequency in the paged device. The index *k* is a function of all the inputs in Figure 2.16. There are 32 possible paging frequencies within each 1.28 second interval. Half of these frequencies belong to the **A**-train, the rest (i.e., $(f(k+8), \ldots, f(k+15), f(k-16), \ldots, f(k-9))$ ) belong to the **B**-train. In order to achieve the -8 offset of the **A**-train, a constant of 24 shall be added to the clock bits (which is equivalent to -8 due to the *mod* 32 operation). The **B**-train is obtained by setting the offset to 8. In the case where slots to receive the first Peripheral response are periodically not available, an additional offset ($k_{nudge}$) is added to the clock bits in order to shift the train by an integer multiple of 1.25 ms duration. A cyclic shift of the order within the trains is also necessary in order to avoid a possible repetitive mismatch between the paging and scanning devices. Thus,

$$Xp = \left[\mathrm{CLKE}_{16-12} + k_{offset} + k_{nudge} + (\mathrm{CLKE}_{4-2,0} - \mathrm{CLKE}_{16-12} + 32) \ \ mod\ 16\right] \ \ mod\ 32, \quad \text{(EQ 2)}$$

where

$$k_{offset} = \begin{cases} 24 & \text{A-train,} \\ 8 & \text{B-train.} \end{cases} \quad \text{(EQ 3)}$$

$$k_{nudge} = \begin{cases} 0 & \text{During } 1^{st} \ \ 2\times N_{\text{page}} \ \ repetitions \\ \text{Even} & \text{During all other repetitions.} \end{cases} \quad \text{(EQ 4)}$$

Alternatively, each switch between the **A-** and **B-**trains may be accomplished by adding 16 to the current value of $k_{offset}$ (originally initialized with 24).

#### 2.6.4.3 Peripheral page response hopping sequence

When the sequence selection input is set to *Peripheral page response,* in order to eliminate the possibility of losing the link due to discrepancies of the native clock CLKN

and the Central's clock estimate CLKE, the five bits $CLKN_{16\text{-}12}$ shall be frozen at their current value. The value shall be frozen at the content it has in the slot where the recipient's access code is detected. The native clock shall *not* be stopped; it is merely the values of the bits used for creating the X-input that are kept fixed for a while. A frozen value is denoted by an asterisk (*) in the discussion below.

For each response slot the paged device shall use an X-input value one larger (*mod* 32) than in the preceding response slot. However, the first response shall be made with the X-input kept at the same value as it was when the access code was recognized. Let *N* be a counter starting at zero. Then, the X-input in the (*N* + 1) -th response slot (the first response slot being the one immediately following the page slot now responding to) of the Peripheral Page Response substate is:

$$Xprp = [\mathrm{CLKN}^*_{16-12} + N] \ \ mod\ 32, \qquad \text{(EQ 5)}$$

The counter *N* shall be set to zero in the slot where the Peripheral acknowledges the page (see Figure 8.3 and Figure 8.4). Then, the value of *N* shall be increased by one each time $CLKN_1$ is set to zero, which corresponds to the start of a Central TX slot. The X-input shall be constructed this way until the first **FHS** packet is received *and* the immediately following response packet has been transmitted. After this the Peripheral shall enter the Connection state using the parameters received in the **FHS** packet.

#### 2.6.4.4 Central page response hopping sequence

When the sequence selection input is set to *Central page response*, the Central shall freeze its estimate of the Peripheral's clock to the value that triggered a response from the paged device. It is equivalent to using the values of the clock estimate when receiving the Peripheral's response (since only $CLKE_1$ will differ from the corresponding page transmission). Thus, the values are frozen when the Peripheral's **ID** packet is received. In addition to the clock bits used, the current values of $k_{offset}$ and $k_{nudge}$ shall also be frozen. The Central shall adjust its X-input in the same way the paged device does, i.e., by incrementing this value by one for each time $CLKE_1$ is set to zero. The first increment shall be done before sending the **FHS** packet to the paged device. Let *N* be a counter starting at one. The rule for forming the X-input is:

$$\begin{aligned} Xprc = [&\mathrm{CLKE}^*_{16-12} + \mathrm{k}^*_{offset} + \mathrm{k}^*_{nudge} + \\ &(\mathrm{CLKE}^*_{4-2,0} - \mathrm{CLKE}^*_{16-12}) \ \ mod\ 16 + N] \ \ mod\ 32, \end{aligned} \qquad \text{(EQ 6)}$$

The value of *N* shall be increased each time $CLKE_1$ is set to zero, which corresponds to the start of a Central TX slot.

#### 2.6.4.5 Inquiry hopping sequence

When the sequence selection input is set to *inquiry*, the X-input is similar to that used in the page hopping sequence. Since no particular device is addressed, the native clock

CLKN of the inquirer shall be used. Moreover, which of the two train offsets to start with is of no real concern in this state. Consequently,

$$Xi = [\mathrm{CLKN}_{16-12} + \mathrm{k}_{offset} + \mathrm{k}_{nudge} + (\mathrm{CLKN}_{4-2,0} - \mathrm{CLKN}_{16-12}) \ mod\ 16] \ \ mod\ 32 \quad \text{(EQ 7)}$$

where $k_{offset}$ and $k_{nudge}$ are defined by (EQ 3) and (EQ 4).

The initial choice of $k_{offset}$ is arbitrary.

#### 2.6.4.6 Inquiry response hopping sequence

The inquiry response hopping sequence is similar to the Peripheral page response hopping sequence with respect to the X-input. The clock input shall not be frozen, thus the following equation applies:

$$Xir = [\mathrm{CLKN}_{16-12} + N] \ \ mod\ \ 32, \quad \text{(EQ 8)}$$

Furthermore, the counter *N* is increased not on $\mathrm{CLKE}_1$ basis, but rather after each **FHS** packet has been transmitted in response to the inquiry. There is no restriction on the initial value of *N* as it is independent of the corresponding value in the inquiring unit.

The $X_{ir}$ value used for the extended inquiry response packet shall be the same $X_{ir}$ value as calculated for the immediately preceding FHS packet.

#### 2.6.4.7 Basic and adapted channel hopping sequence

In the basic and adapted channel hopping sequences, the clock bits to use in the basic or adapted hopping sequence generation shall always be derived from the Central's clock, CLK.

#### 2.6.4.8 Synchronization train RF channels

The synchronization train and synchronization scan use RF channels from the set of three fixed RF channels with indices 0, 24, and 78.

## 2.7 Synchronization scan physical channel

The synchronization scan physical channel enables devices to receive synchronization train packets.

### 2.7.1 Hopping characteristics

When a device enters the Synchronization Scan substate, it shall scan the synchronization train RF channels using the timing defined in Section 2.7.3. Each individual scan window shall use a different RF channel than the previous two scan windows.

The synchronization scan physical channel shall use all of the synchronization train RF channels defined in Section 2.6.4.8.

### 2.7.2 Synchronization Train procedure timing

The Central shall use the synchronization train procedure only when Connectionless Peripheral Broadcast mode is enabled or during the Coarse Clock Adjustment Recovery Mode (Section 8.6.10.2); these can happen concurrently. During the synchronization train procedure, the Central shall attempt to transmit synchronization train packets on all of the RF channels specified in Section 2.6.4.8. The transmission of synchronization train packets on each RF channel is independent of the transmission on other RF channels. For each RF channel, the Central shall:

1. Establish synchronization train events that are separated by $T_{Sync_Train_Interval}$ slots. During Coarse Clock Adjustment Recovery Mode the value of $T_{Sync_Train_Interval}$ shall be 32. At all other times, it shall be the value selected by the Controller from a range provided by the Host ([Vol 4] Part E, Section 7.3.90).
2. Attempt to send a synchronization train packet between each pair of synchronization train events.
3. In the absence of scheduling conflicts, delay the start of the synchronization train packet transmission by $T_{Sync_Train_Delay}$ slots from the synchronization train event, where $T_{Sync_Train_Delay}$ is a (pseudo-)random number between 0 and $T_{Sync_Train_Delay_Max}$. Each value of $T_{Sync_Train_Delay}$ shall be an even integer. $T_{Sync_Train_Delay_Max}$ shall equal 4 slots during Coarse Clock Adjustment Recovery Mode and 16 slots at all other times. Each synchronization packet transmission shall start at the beginning of a time slot where CLK[1:0]=0b00.
4. If the transmission of the synchronization train packet conflicts with the timing of higher priority packets, the actual delay may be adjusted to avoid the conflict. The actual delay should stay within the range 0 to $T_{Sync_Train_Delay_Max}$ and shall not be greater than or equal to $T_{Sync_Train_Interval}$. If it is not possible to transmit the complete packet before the next synchronization train event, it shall not be transmitted.

   Increasing the delay beyond the recommended range (0 to $T_{Sync_Train_Delay_Max}$) increases the chance that the scanning device will miss the packet.
5. There shall be no more than one synchronization train packet transmitted between any two consecutive synchronization train events.
6. The actual delays used shall not all be the same for any three consecutive synchronization train packets (though any two may be).

Subject to the above requirements, synchronization train events on different RF channels may be managed separately or may be co-ordinated using the same value of $T_{Sync_Train_Delay}$.

Figure 2.21 shows the timing relationship of consecutive synchronization train packets on a single RF channel.

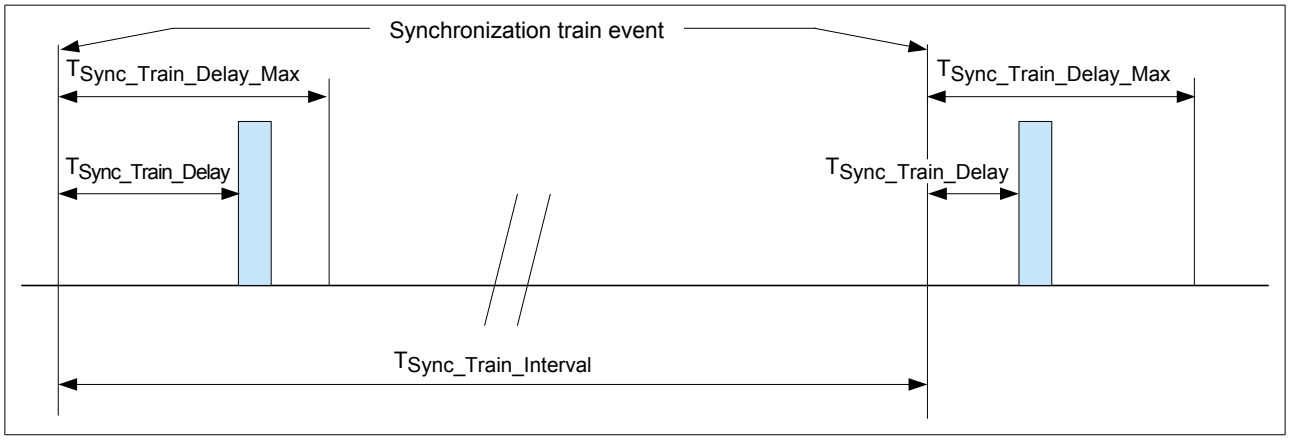



*Figure 2.21: Synchronization train packet timing on a single channel*

### 2.7.3 Synchronization Scan procedure timing

During the Synchronization Scan procedure, a device performs synchronization scans on the synchronization train RF channels. During each scan window, the device listens for the duration of $T_{Sync_Scan_Window}$. The RF channel for each scan window shall be selected as specified in Section 2.7.1. Each scan window should be continuous and not interrupted by other activities. The interval between the start of consecutive scan windows shall be equal to $T_{Sync_Scan_Interval}$. The values for $T_{Sync_Scan_Window}$ and $T_{Sync_Scan_Interval}$ shall be chosen as follows:

- During Connectionless Peripheral Broadcast, refer to [Vol 4] Part E, Section 7.1.52.
- During Coarse Clock Adjustment Recovery Mode, the values chosen are implementation specific.

This timing relationship is shown in Figure 2.22.

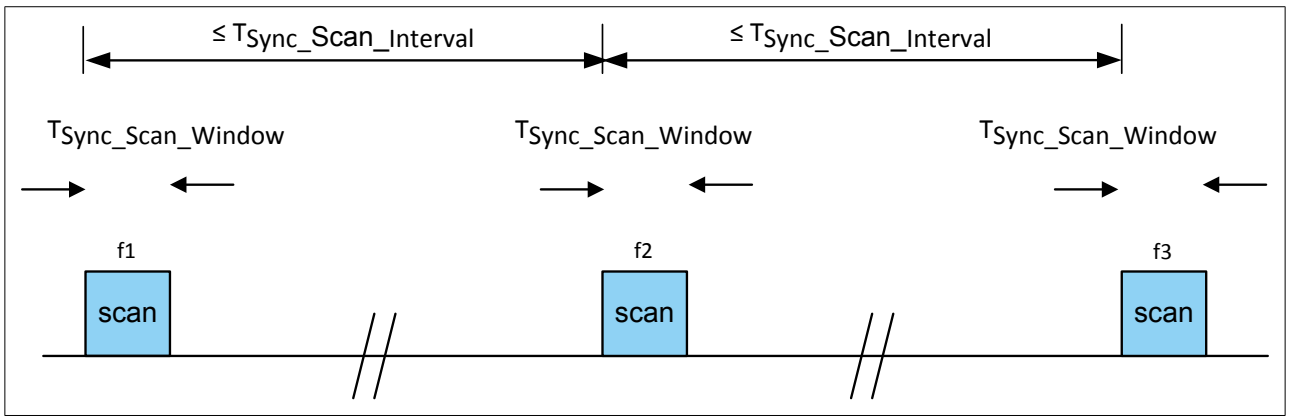



*Figure 2.22: Synchronization scan timing*

# 3 PHYSICAL LINKS

A physical link represents a Baseband connection between devices. A physical link is always associated with exactly one physical channel. Physical links have common properties that apply to all logical transports on the physical link.

Other than the Connectionless Peripheral Broadcast physical link, common properties of physical links are:

- Power control (see [Vol 2] Part C, Section 4.1.3)
- Link supervision (see Section 3.1 and [Vol 2] Part C, Section 4.1.6)
- Encryption (see [Vol 2] Part H, Section 4 and [Vol 2] Part C, Section 4.2.5)
- Channel quality-driven data rate change (see [Vol 2] Part C, Section 4.1.7)
- Multi-slot packet control (see [Vol 2] Part C, Section 4.1.10)

and for physical links on the adapted piconet physical channel:

- AFH channel map (see Section 2.3 and [Vol 2] Part C, Section 4.1.4)

The Connectionless Peripheral Broadcast physical link is associated with the BR/EDR adapted piconet physical channel, a single logical transport (the CPB logical transport), and does not support the Link Manager protocol. For information on link supervision on Connectionless Peripheral Broadcast physical links, see Section 3.2. Multi-slot packets on Connectionless Peripheral Broadcast physical links are controlled by the Host and specified at the profile level.

## 3.1 Link supervision for active physical links

A connection can break down due to various reasons such as a device moving out of range, encountering severe interference or a power failure condition. Since this can happen without any prior warning, it is important to monitor the link on both the Central and the Peripheral side to avoid possible collisions when the logical transport address (see Section 4.2) is reassigned to another Peripheral.

To be able to detect link loss, both the Central and the Peripheral shall use a link supervision timer, $T_{supervision}$. Upon reception of a valid packet header with one of the Peripheral's addresses (see Section 4.2) on the physical link, the timer shall be reset. If at any time in Connection state, the timer reaches the *supervisionTO* value, the connection shall be considered disconnected. The same link supervision timer shall be used for SCO, eSCO, and ACL logical transports.

The timeout period, *supervisionTO*, is negotiated by the Link Manager. Its value shall be chosen so that the supervision timeout will be longer than hold and sniff periods.

## 3.2 Link supervision for Connectionless Peripheral Broadcast physical links

For Connectionless Peripheral Broadcast physical links, only the Receiver side monitors the link. To detect link loss, the Receiver shall use a link supervision timer, $T_{CPB_Supervision}$. Each Connectionless Peripheral Broadcast Receiver shall reset the timer upon reception of a Connectionless Peripheral Broadcast packet with a valid packet header. If at any time in Connectionless Peripheral Broadcast mode of the Connection state, the timer reaches the *CPB_supervisionTO* value, the connection shall be considered disconnected.

For each Receiver, the timeout period, *CPB_supervisionTO*, can be provided by the Host (see Section B.1.7).

## 3.3 Authenticated payload timeout for active links

For active physical links, when encryption is enabled and AES-CCM is used, a device monitors the time between receiving packets from the remote device containing a MIC. To ensure the integrity of the link, an Authenticated Payload timer, $T_{Authenticated_Payload}$, is used. Each device shall reset the timer upon reception of a packet with a valid MIC. The timer shall not be reset upon the reception of a retransmitted packet. If at any time in the Connection state, the timer reaches the *authenticatedPayloadTO* value, the Host shall be notified. The timeout period, *authenticatedPayloadTO*, shall be provided by the Host.

The device shall reset the timer each time after the Host is notified. The Controller shall reset the timer when the Host writes the *authenticatedPayloadTO* value.

# 4 LOGICAL TRANSPORTS

## 4.1 General

Between Central and Peripheral(s), different types of logical transports may be established. Five logical transports have been defined:

- Synchronous Connection-Oriented (SCO) logical transport
- Extended Synchronous Connection-Oriented (eSCO) logical transport
- Asynchronous Connection-Oriented (ACL) logical transport
- Active Peripheral Broadcast (APB) logical transport
- Connectionless Peripheral Broadcast (CPB) logical transport.

The synchronous logical transports are point-to-point logical transports between a Central and a single Peripheral in the piconet. The synchronous logical transports typically support time-bounded information like voice or general synchronous data. The Central maintains the synchronous logical transports by using reserved slots at regular intervals. In addition to the reserved slots the eSCO logical transport may have a retransmission window after the reserved slots.

The ACL logical transport is also a point-to-point logical transport between the Central and a Peripheral. In the slots not reserved for synchronous logical transport(s), the Central can establish an ACL logical transport on a per-slot basis to any Peripheral, including the Peripheral(s) already engaged in a synchronous logical transport.

The APB logical transport is used by a Central to communicate with active Peripherals.

The CPB logical transport is used by a Central to send profile broadcast data to zero or more Peripherals.

## 4.2 Logical transport address (LT_ADDR)

Each Peripheral active in a piconet is assigned a primary 3-bit logical transport address (LT_ADDR). The all-zero LT_ADDR is reserved for APB broadcast messages. The CPB logical transport uses a single non-zero LT_ADDR. The Central does not have an LT_ADDR. A Central's timing relative to the Peripherals' distinguishes it from the Peripherals. A secondary LT_ADDR is assigned to the Peripheral for each eSCO logical transport in use in the piconet. The secondary LT_ADDR shall not be 0. Only eSCO traffic (i.e. NULL, POLL, and one of the EV packet types as negotiated at eSCO logical transport setup) may be sent on these LT_ADDRs. ACL traffic (including LMP) shall always be sent on the primary LT_ADDR. A Peripheral shall only accept packets with

matching primary or secondary LT_ADDR and broadcast packets. The LT_ADDR is carried in the packet header (see Section 6.4). The LT_ADDR shall only be valid for as long as a Peripheral is connected. As soon as it is disconnected, the Peripheral shall lose all of its LT_ADDRs.

The primary LT_ADDR shall be assigned by the Central to the Peripheral when the Peripheral is activated. This is either at connection establishment or role switch, when the primary LT_ADDR is carried in the **FHS** payload.

At any given time an LT_ADDR (other than the special case of the all-zero LT_ADDR) is either unused or is used for exactly one of the three purposes of the primary address for a Peripheral, a secondary address for eSCO traffic, or for a CPB logical transport. Therefore allocating a secondary LT_ADDR for an eSCO logical transport, or reserving an LT_ADDR for the CPB logical transport, reduces the maximum number of active Peripherals possible in the piconet.

## 4.3 Synchronous logical transports

The first type of synchronous logical transport, the SCO logical transport, is a symmetric, point-to-point transport between the Central and a specific Peripheral. The SCO logical transport reserves slots and can therefore be considered as a circuit-switched connection between the Central and the Peripheral. The Central may support up to three SCO links to the same Peripheral or to different Peripherals. A Peripheral may support up to three SCO links from the same Central, or two SCO links if the links originate from different Centrals. SCO packets are never retransmitted.

The second type of synchronous logical transport, the eSCO logical transport, is a point-to-point logical transport between the Central and a specific Peripheral. eSCO logical transports may be symmetric or asymmetric. Similar to SCO, eSCO reserves slots and can therefore be considered a circuit-switched connection between the Central and the Peripheral. In addition to the reserved slots, eSCO supports a retransmission window immediately following the reserved slots. Together, the reserved slots and the retransmission window form the complete eSCO window.

## 4.4 Asynchronous logical transport

In the slots not reserved for synchronous logical transports, the Central may exchange packets with any Peripheral on a per-slot basis. The ACL logical transport provides a packet-switched connection between the Central and all active Peripherals participating in the piconet. Both asynchronous and isochronous services are supported. Only a single ACL logical transport shall exist between any two devices. For most ACL packets, packet retransmission is applied to assure data integrity.

ACL packets not addressed to a specific Peripheral (LT_ADDR=0) are considered as broadcast packets and should be received by every Peripheral except Peripherals with

only a CPB logical transport. If there is no data to be sent on the ACL logical transport and no polling is required, no transmission is required.

## 4.5 Transmit/receive routines

This section describes the way to use the packets as defined in Section 6 in order to support the traffic on the ACL, SCO and eSCO logical transports. Both single-Peripheral and multi-Peripheral configurations are considered. In addition, the use of buffers for the TX and RX routines are described.

### 4.5.1 TX routine

The TX routine is carried out separately for each asynchronous and synchronous logical transport. Figure 4.1 shows the asynchronous and synchronous buffers as used in the TX routine. In this figure, only a single TX asynchronous buffer and a single TX synchronous buffer are shown. In the Central, there is a separate TX asynchronous buffer for each Peripheral. In addition there can be one or more TX synchronous buffers for each synchronous Peripheral (different SCO or eSCO logical transports could either reuse the same TX synchronous buffer, or each have their own TX synchronous buffer). Each TX buffer consists of two FIFO registers: one **current** register which can be accessed and read by the Link Controller in order to compose the packets, and one **next** register that can be accessed by the Baseband Resource Manager to load new information. The positions of the switches S1 and S2 determine which register is current and which register is next; the switches are controlled by the Link Controller. The switches at the input and the output of the FIFO registers can never be connected to the same register simultaneously.

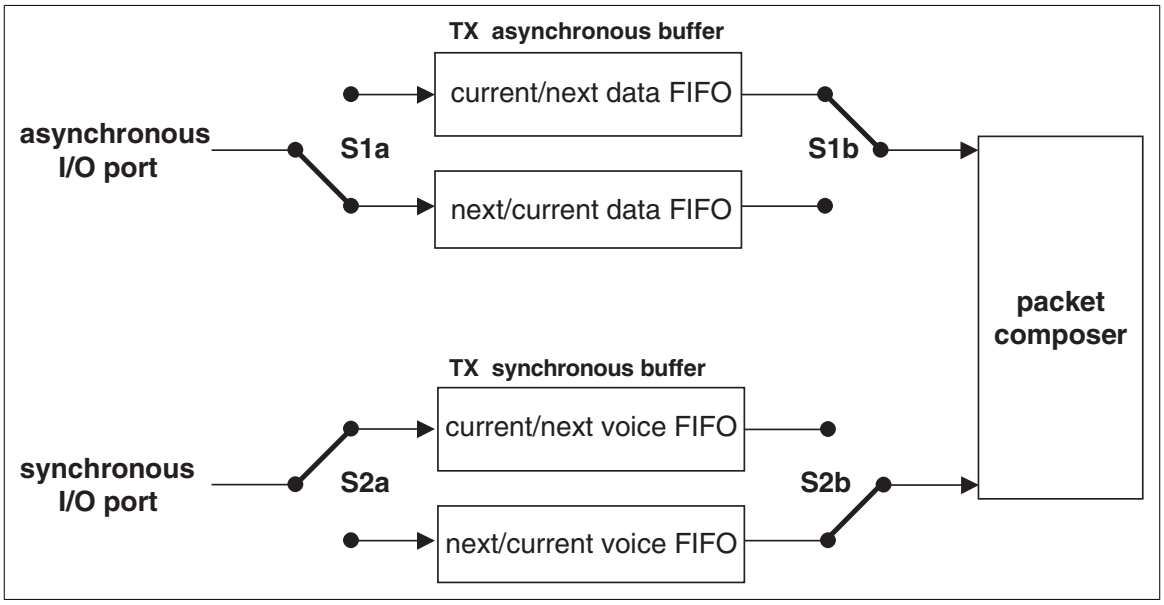



*Figure 4.1: Functional diagram of TX buffering*

Of the packets common to the ACL and SCO logical transports (**NULL**, **POLL and DM1**), only the **DM1** packet carries a payload that is exchanged between the Link Controller and the Link Manager; this packet makes use of the asynchronous buffer. All ACL packets make use of the asynchronous buffer. All SCO and eSCO packets make use of the synchronous buffer except for the **DV** packet where the synchronous data part is handled by the synchronous buffer and the data part is handled by the asynchronous buffer. In the next sections, the operation for ACL traffic, SCO traffic, eSCO traffic, and combined data-voice traffic on the SCO logical transport are described.

#### 4.5.1.1 ACL traffic

In the case of asynchronous data only the TX ACL buffer in Figure 4.1 has to be considered. In this case, only packet types **DM** or **DH** are used, and these can have different lengths. The length is indicated in the payload header. The selection of **DM** or **DH** packets should depend on the quality of the link. See [Vol 2] Part C, Section 4.1.7.

The default packet type in pure data traffic is **NULL** (see Section 6.5.1.2). This means that, if there is no data to be sent (the data traffic is asynchronous, and therefore pauses occur in which no data is available) or no Peripherals need to be polled, **NULL** packets are sent instead – in order to send link control information to the other device (e.g. ACK/STOP information for received data). When no link control information is available (either no need to acknowledge and/or no need to stop the RX flow) no packet is sent at all.

The TX routine works as follows. The Baseband Resource Manager loads new data information in the register to which the switch S1a points. Next, it gives a command to the Link Controller, which forces the switch S1 to change (both S1a and S1b switch synchronously). When the payload needs to be sent, the packet composer reads the current register and, depending on the packet type, builds a payload which is appended to the channel access code and the header and is subsequently transmitted. In the response packet (which arrives in the following RX slot if it concerned a Central transmission, or may be postponed until some later RX slot if it concerned a Peripheral transmission), the result of the transmission is reported back. In case of an ACK, the switch S1 changes position; if a NAK (explicit or implicit) is received instead, the switch S1 will not change position. In that case, the same payload is retransmitted at the next TX occasion.

As long as the Baseband Resource Manager keeps loading the registers with new information, the Link Controller will automatically transmit the payload; in addition, retransmissions are performed automatically in case of errors. The Link Controller will send **NULL** or nothing when no new data is loaded. If no new data has been loaded in the **next** register, during the last transmission, the packet composer will be pointing to an empty register after the last transmission has been acknowledged and the **next**

register becomes the **current** register. If new data is loaded in the **next** register, a Flush command is required to switch the S1 switch to the proper register. As long as the Baseband Resource Manager keeps loading the data and type registers before each TX slot, the data is automatically processed by the Link Controller since the S1 switch is controlled by the ACK information received in response. However, if the traffic from the Baseband Resource Manager is interrupted once and a default packet is sent instead, a Flush command is necessary to continue the flow in the Link Controller.

The Flush command may also be used in case of time-bounded (isochronous) data. In case of a bad link, many retransmissions are necessary. In certain applications, the data is time-bounded: if a payload is retransmitted all the time because of link errors, it may become outdated, and the system might decide to continue with more recent data instead and skip the payload that does not come through. This is accomplished by the Flush command as well. With the **flush**, the switch S1 is forced to change and the Link Controller is forced to consider the next data payload and overrules the ACK control. Any ACL type of packet can be used to send data or link control information to any other ACL Peripheral.

#### 4.5.1.2 SCO traffic

On the SCO logical transport only **HV** and **DV** packet types are used, See Section 6.5.2. The synchronous port may continuously load the **next** register in the synchronous buffer. The S2 switches are changed according to the $T_{sco}$ interval. This $T_{sco}$ interval is negotiated between the Central and the Peripheral at the time the SCO logical transport is established.

For each new SCO slot, the packet composer reads the **current** register after which the S2 switch is changed. If the SCO slot has to be used to send control information with high priority concerning a control packet between the Central and the SCO Peripheral, or a control packet between the Central and any other Peripheral, the packet composer will discard the SCO information and use the control information instead. This control information shall be sent in a DM1 packet. Data or link control information may also be exchanged between the Central and the SCO Peripheral by using the **DV** or **DM1** packets.

#### 4.5.1.3 Mixed data/voice traffic

In Section 6.5.2, a **DV** packet has been defined that can support both data and voice simultaneously on a single SCO logical transport. When the TYPE is **DV**, the Link Controller reads the data register to fill the data field and the voice register to fill the voice field. Thereafter, the switch S2 is changed. However, the position of S1 depends on the result of the transmission as on the ACL logical transport: only if an ACK has been received will the S1 switch change its position. In each **DV** packet, the voice information is new, but the data information might be retransmitted if the previous transmission failed. If there is no data to be sent, the SCO logical transport will

automatically change from **DV** packet type to the current **HV** packet type used before the mixed data/voice transmission.

A Flush command is necessary when the data stream has been interrupted and new data has arrived.

Combined data-voice transmission can also be accomplished by using a separate ACL logical transport in addition to the SCO logical transport(s) if channel capacity permits this.

#### 4.5.1.4 eSCO traffic

On the eSCO logical transport only **EV**, **POLL** and **NULL** packet types are used, see Section 6.5.3. The synchronous port may continuously load the next register in the synchronous buffer. The S2 switches are changed according to the $T_{eSCO}$ interval. This $T_{eSCO}$ interval is negotiated between the Central and the Peripheral at the time the eSCO logical transport is established.

For each new eSCO slot, the packet composer reads the current register after which the S2 switch is changed. If the eSCO slot has to be used to send control information with high priority concerning a control packet between the Central and the eSCO Peripheral, or an ACL packet between the Central and any other Peripheral, the packet composer will discard the eSCO information and use the control information instead. Control information to the eSCO Peripheral is sent in a DM1 packet on the primary LT_ADDR.

#### 4.5.1.5 Default packet types

On the ACL links, the default type is always **NULL** both for the Central and the Peripheral. This means that if no user information needs to be sent, either a **NULL** packet is sent if there is **ACK** or **STOP** information, or no packet is sent at all. The **NULL** packet can be used by the Central to allocate the next Peripheral-to-Central slot to a certain Peripheral (namely the one addressed). However, the Peripheral is not forced to respond to the **NULL** packet from the Central. If the Central requires a response, it sends a **POLL** packet.

The SCO and eSCO packet types are negotiated at the LM level when the SCO or eSCO logical transport is established. The agreed packet type is also the default packet type for the reserved SCO or eSCO slots.

### 4.5.2 RX routine

The RX routine is carried out separately for the ACL logical transport and the synchronous logical transports. However, in contrast to the Central TX asynchronous buffer, a single RX buffer is shared among all Peripherals. For the synchronous buffer, how the different synchronous logical transports are distinguished depends on whether

extra synchronous buffers are required or not. Figure 4.2 shows the asynchronous and synchronous buffers as used in the RX routine. The RX asynchronous buffer consists of two FIFO registers: one register that can be accessed and loaded by the Link Controller with the payload of the latest RX packet, and one register that can be accessed by the Baseband Resource Manager to read the previous payload. The RX synchronous buffer also consists of two FIFO registers: one register which is filled with newly arrived voice information, and one register which can be read by the voice processing unit.

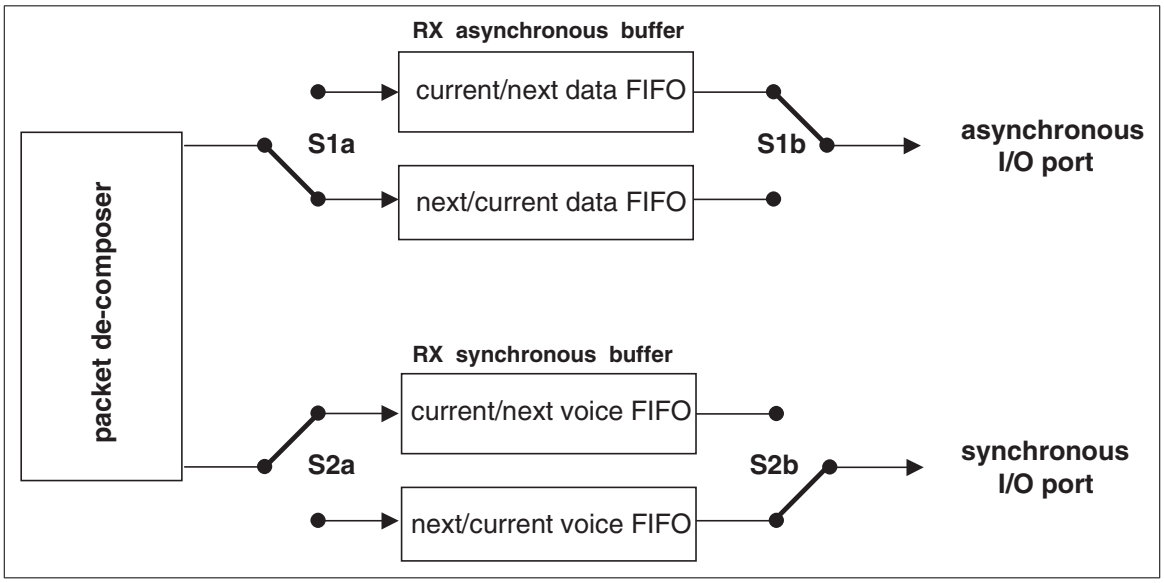



*Figure 4.2: Functional diagram of RX buffering*

Since the TYPE indication in the header (see Section 6.4.2) of the received packet indicates whether the payload contains data and/or voice, the packet de-composer can automatically direct the traffic to the proper buffers. The switch S1 changes every time the Baseband Resource Manager reads the old register. If the next payload arrives before the RX register is emptied, a STOP indication is included in the packet header of the next TX packet that is returned. The STOP indication is removed again as soon as the RX register is emptied. The SEQN field is checked before a new ACL payload is stored into the asynchronous register (flush indication in LLID and broadcast messages influence the interpretation of the SEQN field see Section 7.6).

The S2 switch is changed every $T_{SCO}$ or $T_{eSCO}$ for SCO and eSCO respectively. If, due to errors in the header, no new synchronous payload arrives, the switch still changes. The synchronous data processing unit then processes the synchronous data to account for the missing parts.

### 4.5.3 Flow control

Since the RX ACL buffer can be full while a new payload arrives, flow control is required. The header field FLOW in the return TX packet may use STOP or GO in order to control the transmission of new data.

#### 4.5.3.1 Destination control

As long as data cannot be received, a STOP indication shall be transmitted which is automatically inserted by the Link Controller into the header of the return packet. STOP shall be returned as long as the RX ACL buffer is not emptied by the Baseband Resource Manager. When new data can be accepted again, the GO indication shall be returned. GO shall be the default value. All packet types not including data can still be received. Voice communication for example is not affected by the flow control. Although a device can-not receive new information, it may still continue to transmit information: the flow control shall be separate for each direction.

#### 4.5.3.2 Source control

On the reception of a STOP signal, the Link Controller shall automatically switch to the default packet type. The ACL packet transmitted just before the reception of the STOP indication shall be kept until a GO signal is received. It may be retransmitted as soon as a GO indication is received. Only default packets shall be sent as long as the STOP indication is received. When no packet is received, GO shall be assumed implicitly. The default packets contain link control information (in the header) for the receive direction (which may still be open) and may contain synchronous data (**HV** or **EV** packets). When a GO indication is received, the Link Controller may resume transmitting the data that is present in the TX ACL buffers.

In a multi-Peripheral configuration, only the transmission to the Peripheral that issued the STOP signal shall be stalled. This means that the Central shall only stop transmission from the TX ACL buffer corresponding to the Peripheral that momentarily cannot accept data.

## 4.6 Active Peripheral broadcast transport

The Active Peripheral Broadcast logical transport is used to transport L2CAP user traffic and certain LMP traffic to all devices in the piconet that are currently connected to the piconet physical channel that is used by the APB. There is no acknowledgment protocol and the traffic is uni-directional from the piconet Central to the Peripherals.

The APB logical transport is unreliable. To improve reliability somewhat each packet is transmitted a number of times. An identical sequence number is used to assist with filtering retransmissions at the Peripheral.

The APB logical transport is identified by the reserved, all-zero, LT_ADDR. Packets on the APB logical transport may be sent by the Central at any time.

## 4.7 [This section is no longer used]

## 4.8 Connectionless Peripheral Broadcast logical transport

The CPB logical transport is used to transport profile broadcast data from a Transmitter (Central) to multiple Receivers (Peripherals). There is no acknowledgment scheme and the traffic is unidirectional.

Note: The CPB logical transport is not used for L2CAP connection-oriented channels, L2CAP control signaling, or LMP control signaling.

The CPB logical transport is unreliable. To improve reliability each packet payload may be transmitted a number of times.

# 5 LOGICAL LINKS

The following logical links are defined:

- Link Control (LC)
- ACL Control (ACL-C and APB-C)
- User Asynchronous/Isochronous (ACL-U and APB-U)
- User Synchronous (SCO-S)
- User Extended Synchronous (eSCO-S)
- Profile Broadcast Data (PBD)

The control logical link LC is used at the link control level. The ACL-C and APB-C logical links are used at the link manager level. The ACL-U and APB-U logical links are used to carry either asynchronous or isochronous user information. The SCO-S, and eSCO-S logical links are used to carry synchronous user information. The PBD logical link is used to carry profile broadcast data. The LC logical link is carried in the packet header, all other logical links are carried in the packet payload. The ACL-C, ACL-U, APB-C, and APB-U logical links are indicated in the logical link ID, LLID, field in the payload header. The SCO-S and eSCO-S logical links are carried by the synchronous logical transports only; the ACL-U link is normally carried by the ACL logical transport; however, it may also be carried by the data in the DV packet on the SCO logical transport. The ACL-C link may be carried either by the SCO or the ACL logical transport. The APB-C and APB-U links are carried by the APB logical transport. The PBD logical link is carried by the CPB logical transport.

## 5.1 Link Control logical link (LC)

The LC control logical link shall be mapped onto the packet header. This logical link carries low level link control information like ARQ, flow control, and payload characterization. The LC logical link is carried in every packet except in the **ID** packet which does not have a packet header.

## 5.2 ACL Control logical links (ACL-C and APB-C)

The ACL-C and APB-C logical links shall carry control information exchanged between the link managers of the Central and the Peripheral(s). The ACL-C logical link shall use DM1 or DV packets. DV packets shall only be used on the ACL-C link if the ACL-C message is less than or equal to 9 bytes and an HV1 synchronous logical transport is in use. The APB-C logical link shall use DM1 packets. The ACL-C and APB-C logical links are indicated by the LLID code 0b11 in the payload header.

## 5.3 User asynchronous/isochronous logical links (ACL-U and APB-U)

The ACL-U and APB-U logical links shall carry L2CAP asynchronous and isochronous user data. These messages may be transmitted in one or more Baseband packets. For fragmented messages, the start packet shall use an LLID code of 0b10 in the payload header. Remaining continuation packets shall use LLID code 0b01. If there is no fragmentation, all packets shall use the LLID start code 0b10.

For each logical link, the Controller shall transmit data over the air in the same order that it is received from the Host. The boundaries between packets over the air for a specific L2CAP PDU may be different from the boundaries in the data provided by the Host. Each new L2CAP PDU shall start a new packet over the air. (See [Vol 3] Part A, Section 7.2.1 for related requirements in L2CAP.)

For each logical link, the Controller shall transmit the data received over the air to the Host (whether over HCI or otherwise) in the same order that it was received. The boundaries in the data sent to the Host for a specific L2CAP PDU may be different from the boundaries between packets received over the air. The Controller shall retain boundaries between L2CAP PDUs. Data from different logical links may be interleaved. (See [Vol 3] Part A, Section 7.2.2 for related requirements in L2CAP.)

### 5.3.1 Pausing the ACL-U logical link

When paused by the LM, the Link Controller transmits the current packet with ACL-U information, if any, until an ACK is received. While the ACL-U logical link is paused, the Link Controller shall not transmit any (more) packets with ACL-U logical link information.

When the ACL-U logical link is resumed by the LM, the Link Controller may resume transmitting packets with ACL-U information.

## 5.4 User synchronous data logical link (SCO-S)

The SCO-S logical link carries transparent synchronous user data. This logical link is carried over the synchronous logical transport SCO.

## 5.5 User extended synchronous data logical link (eSCO-S)

The eSCO-S logical link also carries transparent synchronous user data. This logical link is carried over the extended synchronous logical transport eSCO.

## 5.6 Logical link priorities

The ACL-C logical link shall have a higher priority than the ACL-U logical link when scheduling traffic on the shared ACL logical transport, except in the case when

retransmissions of unacknowledged ACL packets shall be given priority over traffic on the ACL-C logical link. The APB-C logical link shall have a higher priority than the APB-U logical link when scheduling traffic on the shared APB logical transport. The ACL-C logical link should also have priority over traffic on the SCO-S and eSCO-S logical links but opportunities for interleaving the logical links should be taken. The ACL-C, SCO-S, and eSCO-S logical links should have priority over traffic on the PBD logical link.

## 5.7 Profile broadcast data logical link

The PBD logical link carries profile broadcast data. Messages shall not be fragmented and shall always use LLID start code 0b10.

# 6 PACKETS

Bluetooth devices shall use the packets as defined in the following sections.

## 6.1 General format

### 6.1.1 Basic Rate

The general packet format of Basic Rate packets is shown in Figure 6.1. Each packet consists of 3 entities: the access code, the header, and the payload. In the figure, the number of bits per entity is indicated.

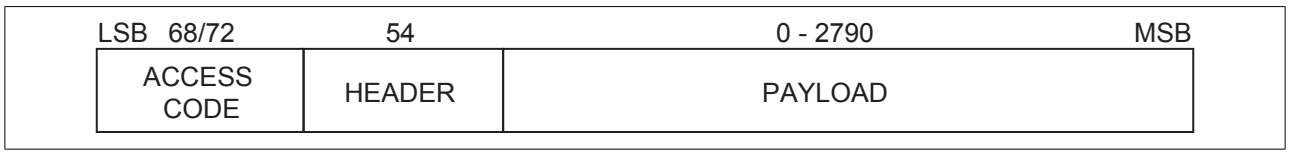



*Figure 6.1: General Basic Rate packet format*

The access code is 72 or 68 bits and the header is 54 bits. The payload ranges from zero to a maximum of 2790 bits. Different packet types have been defined. A packet may consist of:

- the shortened access code only (see Section 6.5.1.1)
- the access code and the packet header
- the access code, the packet header and the payload.

### 6.1.2 Enhanced Data Rate

The general format of Enhanced Data Rate packets is shown in Figure 6.2. The access code and packet header are identical in format and modulation to Basic Rate packets. Enhanced Data Rate packets have a guard time and synchronization sequence following the packet header. Following the payload are two trailer symbols. The guard time, synchronization sequence and trailer are defined in Section 6.6.

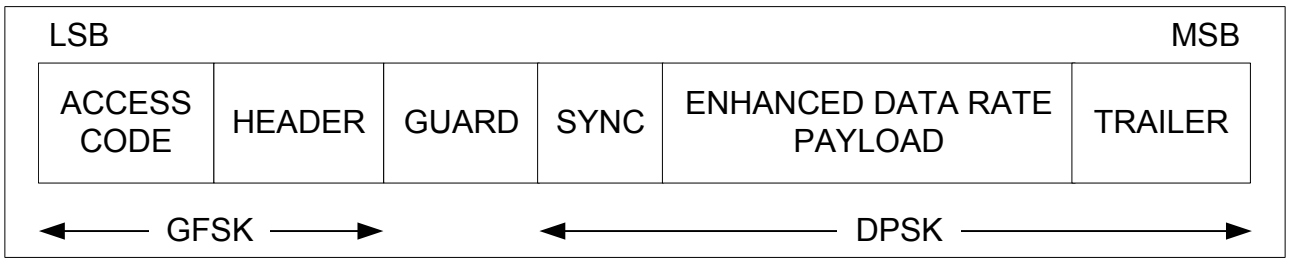



*Figure 6.2: General Enhanced Data Rate packet format*

## 6.2 Bit ordering

The bit ordering when defining packets and messages in the *Baseband Specification*, follows the little-endian format. The following rules apply:

- The *least significant bit* (LSB) corresponds to $b_0$;
- The LSB is the first bit sent over the air;
- In illustrations, the LSB is shown on the left side;

Furthermore, data fields generated internally at Baseband level, such as the packet header fields and payload header length, shall be transmitted with the LSB first. For instance, a 3-bit parameter X=3 is sent as:

$b_0 b_1 b_2 = 110$

over the air where 1 is sent first and 0 is sent last.

## 6.3 Access code

Every packet starts with an access code. If a packet header follows, the access code is 72 bits long, otherwise the access code is 68 bits long and is known as a shortened access code. The shortened access code does not contain a trailer. This access code is used for synchronization, DC offset compensation and identification. The access code identifies all packets exchanged on a physical channel: all packets sent in the same physical channel are preceded by the same access code. In the receiver of the device, a sliding correlator correlates against the access code and triggers when a threshold is exceeded. This trigger signal is used to determine the receive timing.

The shortened access code is used in paging and inquiry. In this case, the access code itself is used as a signaling message and neither a header nor a payload is present.

The access code consists of a preamble, a sync word, and possibly a trailer, see Figure 6.3. For details see Section 6.3.1.

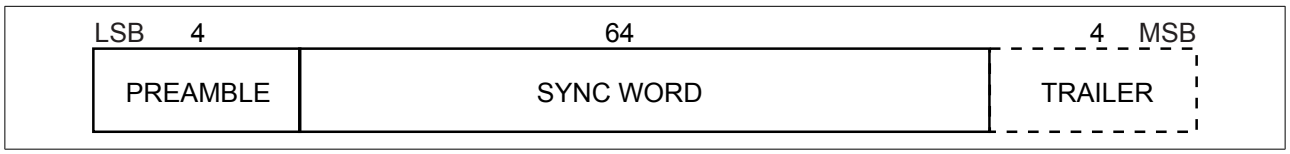



*Figure 6.3: Access code format*

### 6.3.1 Access code types

The different access code types use different Lower Address Parts (LAPs) to construct the sync word. The LAP field of the BD_ADDR is explained in Section 1.2. A summary of the different access code types is in Table 6.1.

| Code type | LAP | Code length | Comments |
|---|---|---|---|
| CAC | Central | 72 | See also Section 1.3 |
| DAC | Paged device | 68/72[1] | |
| GIAC | Reserved | 68/72[1] | |
| DIAC | Dedicated | 68/72[1] | |

*Table 6.1: Summary of access code types*

[1]length 72 is only used in combination with FHS packets

The CAC consists of a **preamble**, **sync word**, and **trailer** and its total length is 72 bits. When used as self-contained messages without a header, the DAC and IAC do not include the trailer bits and are of length 68 bits.

### 6.3.2 Preamble

The preamble is a fixed zero-one pattern of 4 symbols used to facilitate DC compensation. The sequence is either '1010' or '0101' (in transmission order), depending on whether the LSB of the following sync word is 1 or 0, respectively. The preamble is shown in Figure 6.4.

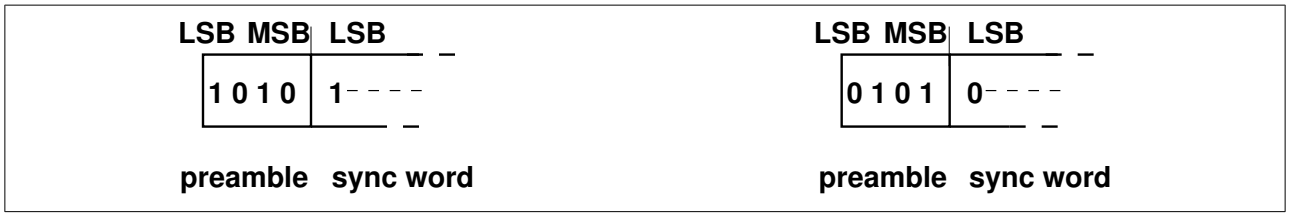



*Figure 6.4: Preamble*

### 6.3.3 Sync word

The sync word is a 64-bit code word derived from a 24 bit address (LAP); for the CAC the Central's LAP is used; for the GIAC and the DIAC, reserved, dedicated LAPs are used; for the DAC, the Peripheral LAP is used. The construction results in a large Hamming distance between sync words based on different LAPs. In addition, the good auto correlation properties of the sync word improve timing acquisition.

#### 6.3.3.1 Synchronization word definition

The sync words are based on a (64,30) expurgated block code with an overlay (bit-wise XOR) of a 64 bit full length pseudo-random noise (PN) sequence. The expurgated code results in a large Hamming distance ($d_{min}$ = 14) between sync words based on different

addresses. The PN sequence improves the auto correlation properties of the access code. The following steps describe how the sync word shall be generated:

1. Generate information sequence;
2. XOR this with the “information covering” part of the PN overlay sequence;
3. Generate the codeword;
4. XOR the codeword with all 64 bits of the PN overlay sequence;

The information sequence is generated by appending 6 bits to the 24 bit LAP (step 1). The appended bits are '001101' (in transmission order) if the MSB of the LAP equals 0. If the MSB of the LAP is 1 the appended bits are '110010'. The LAP MSB together with the appended bits constitute a length-seven Barker sequence. The purpose of including a Barker sequence is to further improve the auto correlation properties. In step 2 the information is pre-scrambled by XORing it with the bits $p_{34\ldots}p_{63}$ of the PN sequence (defined in Section 6.3.3.2). After generating the codeword (step 3), the complete PN sequence is XORed to the codeword (step 4). This step de-scrambles the information part of the codeword. At the same time the parity bits of the codeword are scrambled. Consequently, the original LAP and Barker sequence are ensured a role as a part of the access code sync word, and the cyclic properties of the underlying code is removed. The principle is depicted in Figure 6.5.

In the following discussion, binary sequences will be denoted by their corresponding D-transform (in which $D^i$ represents a delay of $i$ time units). Let $p'(D) = p'_0 + p'_1 D + \ldots + p'_{62} D^{62}$ be the 63 bit PN sequence, where $p'_0$ is the first bit (LSB) leaving the PRNG (see Figure 6.6), and, $p'_{62}$ is the last bit (MSB). To obtain 64 bits, an extra zero is appended at the *end* of this sequence (thus, $p'(D)$ is unchanged). For notational convenience, the reciprocal of this extended polynomial, $p(D) = D^{63} p'(1/D)$, will be used in the following discussion. This is the sequence $p'(D)$ in reverse order. We denote the 24 bit lower address part (LAP) of the Bluetooth Device Address by $a(D) = a_0 + a_1 D + \ldots + a_{23} D^{23}$ ($a_0$ is the LSB of the Bluetooth Device Address).

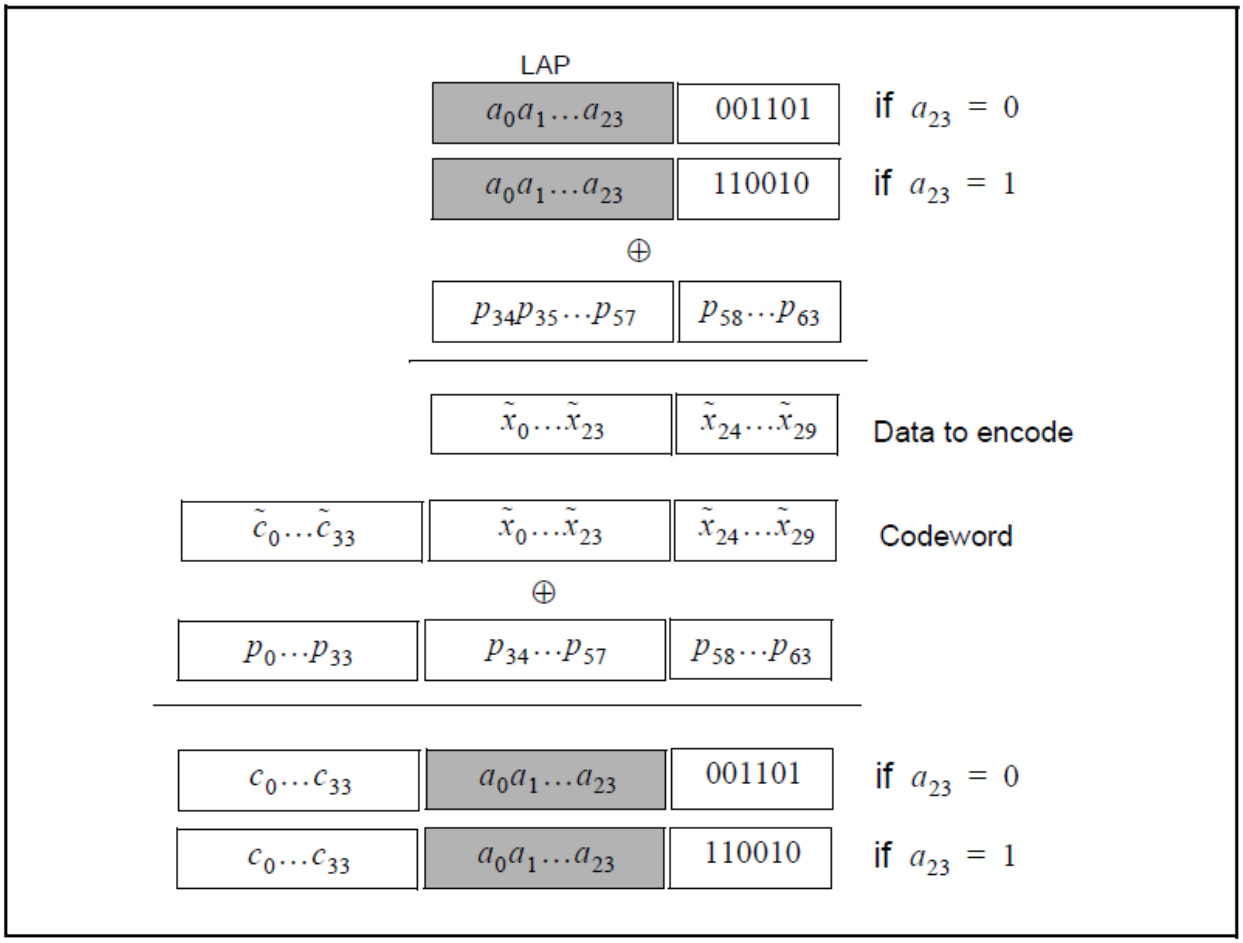



*Figure 6.5: Construction of the sync word*

The (64,30) block code generator polynomial is denoted $g(D) = (1 + D)g'(D)$, where $g'(D)$ is the generator polynomial 0x37CD0EB67 of a primitive binary (63,30) BCH code. Thus, *g(D)* is

$$g(D) = 0\text{x}585713\text{DA9}, \quad \text{(EQ 9)}$$

where the left-most bit corresponds to the high-order ($g_{34}$) coefficient. The DC-free four bit sequences '0101' and '1010' (in transmission order) can be written

$$\begin{cases} F_0(D) = D + D^3, \\ F_1(D) = 1 + D^2, \end{cases} \quad \text{(EQ 10)}$$

respectively. Furthermore,

$$\begin{cases} B_0(D) = D^2 + D^3 + D^5, \\ B_1(D) = 1 + D + D^4, \end{cases} \quad \text{(EQ 11)}$$

which are used to create the length seven Barker sequences. Then, the access code shall be generated by the following procedure:

1. Format the 30 information bits to encode:

$$x(D) = a(D) + D^{24}B_{a_{23}}(D).$$

2. Add the information covering part of the PN overlay sequence:

$$\tilde{x}(D) = x(D) + p_{34} + p_{35}D + \ldots + p_{63}D^{29}.$$

3. Generate parity bits of the (64,30) expurgated block code:[1]

$$\tilde{c}(D) = D^{34}\tilde{x}(D) \ \ mod \ \ g(D).$$

4. Create the codeword:

$$\tilde{s}(D) = D^{34}\tilde{x}(D) + \tilde{c}(D).$$

5. Add the PN sequence:

$$s(D) = \tilde{s}(D) + p(D).$$

6. Append the (DC-free) preamble and trailer:

$$y(D) = F_{c_0}(D) + D^4 s(D) + D^{68}F_{a_{23}}(D).$$

#### 6.3.3.2 Pseudo-random noise sequence generation

To generate the PN sequence the primitive polynomial *h(D)* = 1 + *D* + $D^3$ + $D^4$ + $D^6$ shall be used. The LFSR and its starting state are shown in Figure 6.6. The PN sequence generated (including the extra terminating zero) becomes 0x83848D96BBCC54FC. The LFSR output starts with the left-most bit of this PN sequence. This corresponds to $p'(D)$ of the previous section. Thus, using the reciprocal $p'(D)$ as overlay gives the 64 bit sequence:

$$p = '3F2A33DD69B121C1' \qquad \text{(EQ 12)}$$

(in transmission order) where the left-most bit is $p_0$ = 0 (there are two initial zeros in the binary representation of the hexadecimal digit 3), and $p_{63}$ = 1 is the right-most bit.

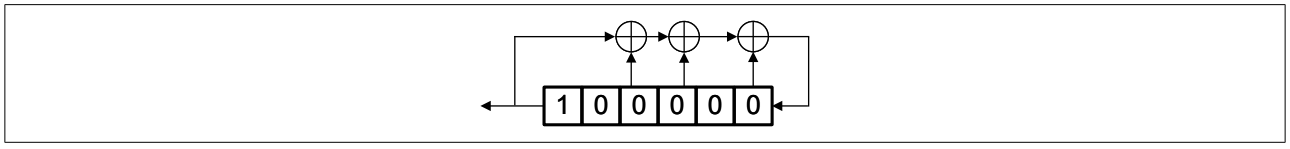



*Figure 6.6: LFSR and the starting state to generate $p'(D)$*

[1] *x*(*D*) *mod y*(*D*) denotes the remainder when *x*(*D*) is divided by *y*(*D*).

### 6.3.4 Trailer

The trailer is appended to the sync word as soon as the packet header follows the access code. This is typically the case with the CAC, but the trailer is also used in the DAC and IAC when these codes are used in FHS packets exchanged during page response and inquiry response.

The trailer is a fixed zero-one pattern of four symbols. The trailer together with the three MSBs of the syncword form a 7-bit pattern of alternating ones and zeroes which can be used for extended DC compensation. The trailer sequence is either ‘1010’ or ‘0101’ (in transmission order) depending on whether the MSB of the sync word is 0 or 1, respectively. The choice of trailer is illustrated in Figure 6.7.

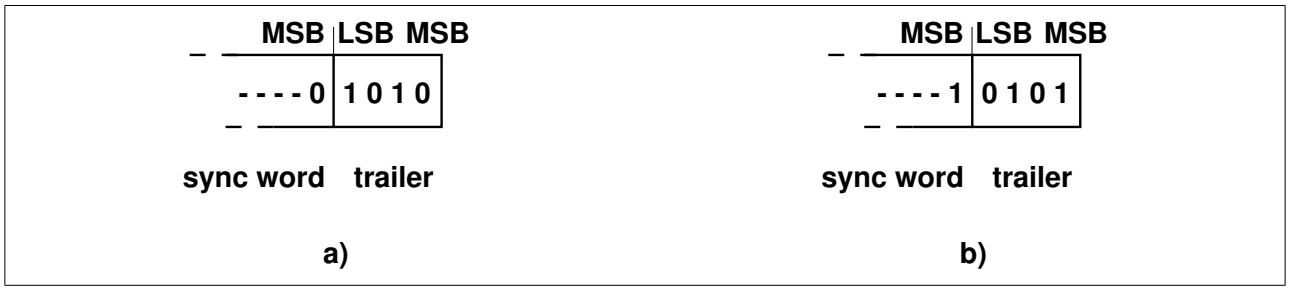



*Figure 6.7: Trailer in CAC when MSB of sync word is 0 (a), and when MSB of sync word is 1 (b)*

## 6.4 Packet header

The header contains link control (LC) information and consists of 6 fields:

- LT_ADDR: 3-bit logical transport address
- TYPE: 4-bit type code
- FLOW: 1-bit flow control
- ARQN: 1-bit acknowledge indication
- SEQN: 1-bit sequence number
- HEC: 8-bit header error check

The total header, including the HEC, consists of 18 bits, see Figure 6.8, and is encoded with a rate 1/3 FEC (not shown but described in Section 7.4) resulting in a 54-bit header. The LT_ADDR and TYPE fields shall be sent LSB first.

*Figure 6.8: Header format*

### 6.4.1 LT_ADDR

The 3-bit LT_ADDR field contains the logical transport address for the packet (see Section 4.2). This field indicates the destination Peripheral (or Peripherals in the case of a broadcast) for a packet in a Central-to-Peripheral transmission slot and indicates the source Peripheral for a Peripheral-to-Central transmission slot.

### 6.4.2 TYPE

Sixteen different types of packets can be distinguished. The 4-bit TYPE code specifies which packet type is used. The interpretation of the TYPE code depends on the logical transport address in the packet. First, it shall be determined whether the packet is sent on a SCO logical transport, an eSCO logical transport, an ACL logical transport, or a CPB logical transport. Second, it shall be determined whether Enhanced Data Rate has been enabled for the logical transport (ACL or eSCO) indicated by LT_ADDR. It can then be determined which type of SCO packet, eSCO packet, or ACL packet has been received. The TYPE code determines how many slots the current packet will occupy (see the slot occupancy column in Table 6.2). This allows the non-addressed receivers to refrain from listening to the channel for the duration of the remaining slots. In Section 6.5, each packet type is described in more detail.

### 6.4.3 FLOW

The FLOW bit is used for flow control of packets over the ACL logical transport. When the RX buffer for the ACL logical transport in the recipient is full, a STOP indication (FLOW=0) shall be returned to stop the other device from transmitting data temporarily. The STOP signal only affects ACL packets. Packets including only link control information (POLL and NULL packets), SCO packets or eSCO packets can still be received. When the RX buffer can accept data, a GO indication (FLOW=1) shall be returned. When no packet is received, or the received header is in error, a GO shall be assumed implicitly. In this case, the Peripheral can receive a new packet with CRC although its RX buffer is still not emptied. The Peripheral shall then return a NAK in response to this packet even if the packet passed the CRC check.

The FLOW bit is not used on the eSCO logical transport and shall be set to one on transmission and ignored upon receipt. The FLOW bit is reserved for future use on the CPB logical transport.

### 6.4.4 ARQN

The 1-bit acknowledgment indication ARQN is used to inform the source of a successful transfer of payload data with CRC, and can be positive acknowledge ACK or negative acknowledge NAK. See Section 7.6 for initialization and usage of this bit.

The ARQN bit is reserved for future use on the CPB logical transport.

### 6.4.5 SEQN

The SEQN bit provides a sequential numbering scheme to order the data packet stream. See Section 7.6.2 for initialization and usage of the SEQN bit. For Active Peripheral Broadcast packets, a modified sequencing method is used, see Section 7.6.5.

The SEQN bit is reserved for future use on the CPB logical transport.

### 6.4.6 HEC

Each header has a header-error-check to check the header integrity. The HEC is an 8-bit word (generation of the HEC is specified in Section 7.1.1). Before generating the HEC, the HEC generator is initialized with an 8-bit value. For FHS packets sent in Central Page Response substate, the Peripheral upper address part (UAP) shall be used. For FHS packets and extended inquiry response packets sent in Inquiry Response substate, the default check initialization (DCI, see Section 1.2.1) shall be used. In all other cases, the UAP of the Central shall be used.

After the initialization, a HEC shall be calculated for the 10 header bits. Before checking the HEC, the receiver shall initialize the HEC check circuitry with the proper 8-bit UAP (or DCI). If the HEC does not check, the entire packet shall be discarded. More information can be found in Section 7.1.

## 6.5 Packet types

The packets used on the piconet are related to the logical transports on which they are used. Four logical transports with distinct packet types are defined (see Section 4):

- SCO logical transport
- eSCO logical transport
- ACL logical transport
- CPB logical transport.

To indicate the different packets on a logical transport, the 4-bit TYPE code is used. The packet types are divided into four segments. The first segment is reserved for control packets. All control packets occupy a single time slot. The second segment is reserved for packets occupying a single time slot. The third segment is reserved for packets occupying three time slots. The fourth segment is reserved for packets occupying five time slots. The slot occupancy is reflected in the segmentation and can directly be derived from the type code. Table 6.2 summarizes the packets defined for the SCO, eSCO, ACL, and CPB logical transport types; a dash means the value is reserved for future use.

All packet types with a payload shall use GFSK modulation unless specified otherwise in the following sections.

ACL logical transports Enhanced Data Rate packet types are explicitly selected via LMP using the *packet_type_table* (ptt) parameter. eSCO Enhanced Data Rate packet types are selected when the eSCO logical transport is established. Enhanced Data Rate packet types for the CPB logical transport are selected when the CPB logical transport is established.

| Segment | TYPE code $b_3b_2b_1b_0$ | Slot occu-pancy | SCO logical transport (1 Mb/s) | eSCO logical transport (1 Mb/s) | eSCO logical transport (2-3 Mb/s) | ACL logical transport (1 Mb/s) ptt=0 | ACL logical transport (2-3 Mb/s) ptt=1 | CPB logical transport (1 Mb/s) | CPB logical transport (2-3 Mb/s) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0000 | 1 | NULL | NULL | NULL | NULL | NULL | NULL | NULL |
| | 0001 | 1 | POLL | POLL | POLL | POLL | POLL | — | — |
| | 0010 | 1 | FHS | — | — | FHS | FHS | — | — |
| | 0011 | 1 | DM1 | — | — | DM1 | DM1 | DM1 | DM1 |
| 2 | 0100 | 1 | — | — | — | DH1 | 2-DH1 | DH1 | 2-DH1 |
| | 0101 | 1 | HV1 | — | — | — | — | — | — |
| | 0110 | 1 | HV2 | — | 2-EV3 | — | — | — | — |
| | 0111 | 1 | HV3 | EV3 | 3-EV3 | — | — | — | — |
| | 1000 | 1 | DV | — | — | — | 3-DH1 | — | 3-DH1 |
| | 1001 | 1 | — | — | — | AUX1 | AUX1 | — | — |
| 3 | 1010 | 3 | — | — | — | DM3 | 2-DH3 | DM3 | 2-DH3 |
| | 1011 | 3 | — | — | — | DH3 | 3-DH3 | DH3 | 3-DH3 |
| | 1100 | 3 | — | EV4 | 2-EV5 | — | — | — | — |
| | 1101 | 3 | — | EV5 | 3-EV5 | — | — | — | — |
| 4 | 1110 | 5 | — | — | — | DM5 | 2-DH5 | DM5 | 2-DH5 |
| | 1111 | 5 | — | — | — | DH5 | 3-DH5 | DH5 | 3-DH5 |

*Table 6.2: Packets defined for synchronous, asynchronous, and CPB logical transport types*

### 6.5.1 Common packet types

There are five common kinds of packets. In addition to the types listed in segment 1 of Table 6.2, the ID packet is also a common packet type but is not listed in segment 1 because it does not have a packet header.

#### 6.5.1.1 ID packet

The identity or ID packet consists of the device access code (DAC) or inquiry access code (IAC). It has a fixed length of 68 bits. It is a very robust packet since the receiver uses a bit correlator to match the received packet to the known bit sequence of the ID packet.

The ID packet shall only be used where explicitly stated in paging (see Section 8.3), inquiry (see Section 8.4), and role switch (see Section 8.6.5). It shall not be used where any other packet type is permitted.

#### 6.5.1.2 NULL packet

The NULL packet has no payload and consists of the channel access code and packet header only. Its total (fixed) length is 126 bits. The NULL packet may be used to return link information to the source regarding the success of the previous transmission (ARQN), or the status of the RX buffer (FLOW). The NULL packet does not require acknowledgment.

#### 6.5.1.3 POLL packet

The POLL packet is very similar to the NULL packet. It does not have a payload. In contrast to the NULL packet, it requires a confirmation from the recipient. It is not a part of the ARQ scheme. The POLL packet does not affect the ARQN and SEQN fields. Upon reception of a POLL packet the Peripheral shall respond with a packet even when the Peripheral does not have any information to send unless the Peripheral has scatternet commitments in that timeslot. This return packet is an implicit acknowledgment of the POLL packet. This packet can be used by the Central in a piconet to poll the Peripherals. Peripherals shall not transmit the POLL packet. POLL packets shall not be sent on a Connectionless Peripheral Broadcast logical transport.

#### 6.5.1.4 FHS packet

The FHS packet is a special control packet containing, among other things, the Bluetooth Device Address and the clock of the sender. The payload contains 144 information bits plus a 16-bit CRC code. The payload is coded with a rate 2/3 FEC with a gross payload length of 240 bits.

Figure 6.9 illustrates the format and contents of the FHS payload. The FHS packet is used in Central page response, inquiry response and in role switch.

The FHS packet is not encrypted. No payload header or MIC is present.

The FHS packet contains real-time clock information. This clock information shall be updated before each retransmission. The retransmission of the FHS payload is different than retransmissions of ordinary data payloads where the same payload is used for each retransmission. The FHS packet is used for frequency hop synchronization before the piconet channel has been established, or when an existing piconet changes to a new piconet. However, the FHS packet is not used by Connectionless Peripheral Broadcast Receivers for frequency hop synchronization with Connectionless Peripheral Broadcast Transmitters.

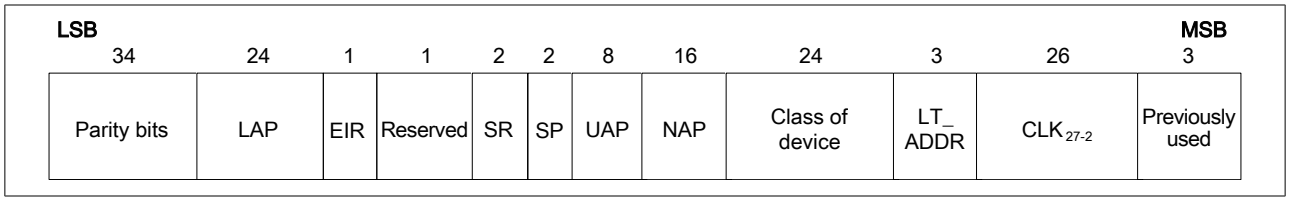



*Figure 6.9: Format of the FHS payload*

Each field is described in more detail below:

| | |
|---|---|
| **Parity bits** | This 34-bit field contains the parity bits that form the first part of the sync word of the access code of the device that sends the FHS packet. These bits are derived from the LAP as described in Section 1.2. |
| **LAP** | This 24-bit field shall contain the lower address part of the device that sends the FHS packet. |
| **EIR** | This bit shall indicate that an extended inquiry response packet may follow. See Section 8.4.3. |
| **Reserved** | This 1-bit field is reserved for future use. |
| **SR** | This 2-bit field is the page scan repetition mode field and indicates the interval between two consecutive page scan windows, see also Table 6.4 and Table 8.1 |
| **SP** | This 2-bit field shall be set to 0b10. |
| **UAP** | This 8-bit field shall contain the upper address part of the device that sends the FHS packet. |
| **NAP** | This 16-bit field shall contain the non-significant address part of the device that sends the FHS packet (see also Section 1.2 for LAP, UAP, and NAP). |
| **Class of Device** | This 24-bit field shall contain the Class of Device of the device that sends the FHS packet. The field is defined in Assigned Numbers. |

| | |
|---|---|
| **LT_ADDR** | This 3-bit field shall contain the logical transport address the recipient shall use if the FHS packet is used at connection setup or role switch. A Peripheral responding to a Central or a device responding to an inquiry request message shall include an all-zero LT_ADDR field if it sends the FHS packet. |
| **$CLK_{27-2}$** | This 26-bit field shall contain the value of the native clock of the device that sends the FHS packet, sampled at the beginning of the transmission of the access code of this FHS packet. This clock value has a resolution of 1.25 ms (two-slot interval). For each new transmission, this field is updated so that it accurately reflects the real-time clock value. |

*Table 6.3: Description of the FHS payload*

The device sending the FHS shall set the SR bits according to Table 6.4.

| SR bit format $b_1b_0$ | SR (page scan repetition) mode |
|---|---|
| 00 | R0 |
| 01 | R1 |
| 10 | R2 |
| 11 | Reserved for future use |

*Table 6.4: Contents of SR field*

The LAP, UAP, and NAP together form the 48-bit Bluetooth Device Address of the device that sends the FHS packet. Using the parity bits and the LAP, the recipient can directly construct the channel access code of the sender of the FHS packet.

When initializing the HEC and CRC for the FHS packet of inquiry response, the UAP shall be the DCI.

#### 6.5.1.5 DM1 packet

DM1 is part of segment 1 in order to support control messages in any logical transport that allows the DM1 packet (see Table 6.2). However, it may also carry regular user data. Since the DM1 packet can be regarded as an ACL packet, it will be discussed in Section 6.5.4.

### 6.5.2 SCO packets

HV and DV packets are used on the synchronous SCO logical transport. The HV packets do not include a MIC or CRC and shall not be retransmitted. DV packets include a CRC on the data section, but not on the synchronous data section. DV packets do not include a MIC. The data section of DV packets shall be retransmitted. SCO packets may be routed to the synchronous I/O port. Four packets are allowed on the SCO logical transport: HV1, HV2, HV3 and DV. These packets are typically used for 64 kb/s speech transmission but may be used for transparent synchronous data.

SCO packets shall only be encrypted with E0 when E0 is enabled. SCO packets shall not be sent when AES-CCM Encryption is enabled.

#### 6.5.2.1 HV1 packet

The **HV1** packet has 10 information bytes. The bytes are protected with a rate 1/3 FEC. No MIC is present. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

#### 6.5.2.2 HV2 packet

The **HV2** packet has 20 information bytes. The bytes are protected with a rate 2/3 FEC. No MIC is present. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

#### 6.5.2.3 HV3 packet

The **HV3** packet has 30 information bytes. The bytes are not protected by FEC. No MIC is present. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

#### 6.5.2.4 DV packet

The DV packet is a combined data - voice packet. The DV packet shall only be used in place of an HV1 packet. The payload is divided into a voice field of 80 bits and a data field containing up to 150 bits, see Figure 6.10. The voice field is not protected by FEC. The data field has between 1 and 10 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. No MIC is present. The data field (including the CRC) is encoded with a rate 2/3 FEC. Since the **DV** packet has to be sent at regular intervals due to its synchronous contents, it is listed under the SCO packet types. The voice and data fields shall be treated separately. The voice field shall be handled in the same way as normal SCO data and shall never be retransmitted; that is, the voice field is always new. The data field is checked for errors and shall be retransmitted if necessary. When the asynchronous data field in the DV packet has not been acknowledged before the SCO logical transport is terminated, the asynchronous data field shall be retransmitted in a DM1 packet.

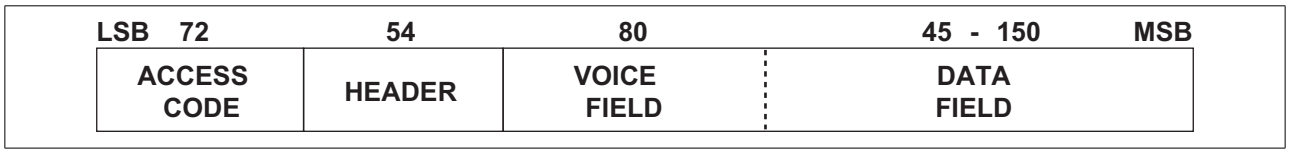



*Figure 6.10: DV packet format*

### 6.5.3 eSCO packets

EV packets are used on the synchronous eSCO logical transport. The packets include a CRC and retransmission may be applied if no acknowledgment of proper reception

is received within the retransmission window. No MIC is present. eSCO packets may be routed to the synchronous I/O port. Three eSCO packet types (EV3, EV4, EV5) are defined for Basic Rate operation and four additional eSCO packet types (2-EV3, 3-EV3, 2-EV5, 3-EV5) for Enhanced Data Rate operation. The eSCO packets may be used for 64 kb/s speech transmission as well as transparent data at 64 kb/s and other rates.

#### 6.5.3.1 EV3 packet

The **EV3** packet has between 1 and 30 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The EV3 packet may cover up to a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.2 EV4 packet

The **EV4** packet has between 1 and 120 information bytes plus a 16-bit CRC code. No MIC is present. The EV4 packet may cover to up three time slots. The information plus CRC bits are coded with a rate 2/3 FEC. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.3 EV5 packet

The **EV5** packet has between 1 and 180 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.4 2-EV3 packet

The **2-EV3** packet is similar to the EV3 packet except that the payload is modulated using π/4-DQPSK. It has between 1 and 60 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The 2-EV3 packet covers a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.5 2-EV5 packet

The **2-EV5** packet is similar to the EV5 packet except that the payload is modulated using π/4-DQPSK. It has between 1 and 360 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The 2-EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.6 3-EV3 packet

The **3-EV3** packet is similar to the EV3 packet except that the payload is modulated using 8DPSK. It has between 1 and 90 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The 3-EV3 packet covers a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

#### 6.5.3.7 3-EV5 packet

The **3-EV5** packet is similar to the EV5 packet except that the payload is modulated using 8DPSK. It has between 1 and 540 information bytes plus a 16-bit CRC code. No MIC is present. The bytes are not protected by FEC. The 3-EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

### 6.5.4 ACL packets

ACL packets are used on the asynchronous logical transport and the CPB logical transport. The information carried may be user data for either logical transport or control data for the asynchronous logical transport.

Six packet types are defined for Basic Rate operation: DM1, DH1, DM3, DH3, DM5, and DH5. Six additional packets are defined for Enhanced Data Rate operation: 2-DH1, 3-DH1, 2-DH3, 3-DH3, 2-DH5 and 3-DH5. The AUX1 packet is also defined for test purposes.

The length indicator in the payload header specifies the number of user bytes (excluding payload header, MIC and the CRC code).

#### 6.5.4.1 DM1 packet

The DM1 packet carries data information only. The payload has between 1 and 18 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The DM1 packet occupies a single time slot. The information bits, MIC bits (when present), plus CRC bits are coded with a rate 2/3 FEC. The payload header in the DM1 packet is 1 byte long, see Figure 6.12.

#### 6.5.4.2 DH1 packet

This packet is similar to the DM1 packet, except that the information in the payload is not FEC encoded. As a result, the DH1 packet has between 1 and 28 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The DH1 packet occupies a single time slot.

#### 6.5.4.3 DM3 packet

The DM3 packet may occupy up to three time slots. The payload has between 2 and 123 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The information bits, MIC bits (when present), plus CRC bits are coded with a rate 2/3 FEC. The payload header in the DM3 packet is 2 bytes long, see Figure 6.13.

#### 6.5.4.4 DH3 packet

This packet is similar to the DM3 packet, except that the information in the payload is not FEC encoded. As a result, the DH3 packet has between 2 and 185 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The DH3 packet may occupy up to three time slots.

#### 6.5.4.5 DM5 packet

The DM5 packet may occupy up to five time slots. The payload has between 2 and 226 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The payload header in the DM5 packet is 2 bytes long. The information bits, MIC bits (when present), plus CRC bits are coded with a rate 2/3 FEC.

#### 6.5.4.6 DH5 packet

This packet is similar to the DM5 packet, except that the information in the payload is not FEC encoded. As a result, the DH5 packet has between 2 and 341 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The DH5 packet may occupy up to five time slots.

#### 6.5.4.7 AUX1 packet

This packet resembles a DH1 packet but has no MIC or CRC code. The AUX1 packet has between 1 and 30 information bytes (including the 1-byte payload header). The AUX1 packet occupies a single time slot. The AUX1 packet shall only be used in test mode (see [Vol 3] Part D). The Link Controller shall discard any AUX1 packet received in any other circumstances.

#### 6.5.4.8 2-DH1 packet

This packet is similar to the DH1 packet except that the payload is modulated using π/4-DQPSK. The 2-DH1 packet has between 2 and 56 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled.The 2-DH1 packet occupies a single time slot.

#### 6.5.4.9 2-DH3 packet

This packet is similar to the DH3 packet except that the payload is modulated using π/4-DQPSK. The 2-DH3 packet has between 2 and 369 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The 2-DH3 packet may occupy up to three time slots.

#### 6.5.4.10 2-DH5 packet

This packet is similar to the DH5 packet except that the payload is modulated using π/4-DQPSK. The 2-DH5 packet has between 2 and 681 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The 2-DH5 packet may occupy up to five time slots.

#### 6.5.4.11 3-DH1 packet

This packet is similar to the DH1 packet except that the payload is modulated using 8DPSK. The 3-DH1 packet has between 2 and 85 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The 3-DH1 packet occupies a single time slot.

#### 6.5.4.12 3-DH3 packet

This packet is similar to the DH3 packet except that the payload is modulated using 8DPSK. The 3-DH3 packet has between 2 and 554 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The 3-DH3 packet may occupy up to three time slots.

#### 6.5.4.13 3-DH5 packet

This packet is similar to the DH5 packet except that the payload is modulated using 8DPSK. The 3-DH5 packet has between 2 and 1023 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. A 32-bit MIC is present only when encryption with AES-CCM is enabled. The 3-DH5 packet may occupy up to five time slots.

## 6.6 Payload format

In the payload, two fields are distinguished: the synchronous data field and the asynchronous data field. The ACL packets only have the asynchronous data field and the SCO and eSCO packets only have the synchronous data field – with the exception of the DV packets which have both.

### 6.6.1 Synchronous data field

In SCO, which is only supported in Basic Rate mode, the synchronous data field has a fixed length and consists only of the synchronous data body portion. No payload header is present.

In Basic Rate eSCO, the synchronous data field consists of two segments: a synchronous data body and a CRC code. No payload header is present.

In Enhanced Data Rate eSCO, the synchronous data field consists of five segments: a guard time, a synchronization sequence, a synchronous data body, a CRC code and a trailer. No payload header is present.

1. **Enhanced Data Rate guard time**

   For Enhanced Data Rate packets the guard time is defined as the period starting at the end of the last GFSK symbol of the header and ending at the start of the reference symbol of the synchronization sequence. The length of the guard time shall be between 4.75 µs and 5.25 µs.

2. **Enhanced Data Rate synchronization sequence**

   For Enhanced Data Rate packets the symbol timing at the start of the synchronization sequence shall be within ±¼ µs of the symbol timing of the last GFSK symbol of the packet header. The length of the synchronization sequence is 11 µs (11 DPSK symbols) and consists of a reference symbol (with arbitrary phase) followed by ten DPSK symbols.

   The phase changes between the DPSK symbols (shown in Figure 6.11) shall be

$$\{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}\} = \{3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, -3\pi/4, 3\pi/4, 3\pi/4, 3\pi/4\} \quad \text{(EQ 13)}$$

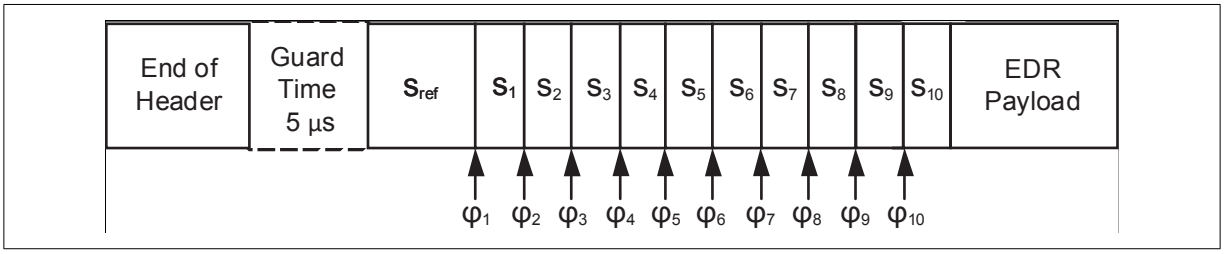



*Figure 6.11: Synchronization sequence*

Sref is the reference symbol. $\varphi_1$ is the phase change between the reference symbol and the first DPSK symbol $S_1$. $\varphi_k$ is the phase change between the k-1$^{th}$ symbol $S_{k-1}$ and the k$^{th}$ symbol $S_k$

Note: The synchronization sequence can be generated using the modulator by pre-pending the data with bits that generate the synchronization sequence.

For π/4-DQPSK, the bit sequence used to generate the synchronization sequence is 0,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,0,1,0,1.

For 8DPSK, the bit sequence used to generate the synchronization sequence is 0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,0,0,1,0,0,1,0.

3. **Synchronous data body**

   For HV and DV packets, the synchronous data body length is fixed. For EV packets, the synchronous data body length is negotiated during the LMP eSCO setup. Once negotiated, the synchronous data body length remains constant unless re-negotiated. The synchronous data body length may be different for each direction of the eSCO logical transport.

4. **CRC code**

   The 16-bit CRC in the payload is generated as specified in Section 7.1. The 8-bit UAP of the Central is used to initialize the CRC generator.

   Only the Synchronous data body segment is used to generate the CRC code.

5. **Enhanced Data Rate trailer**

   For Enhanced Data Rate packets, two trailer symbols shall be added to the end of the payload. The trailer bits shall be all zero, i.e. {00, 00} for the π/4-DQPSK and {000, 000} for the 8DPSK.

### 6.6.2 Asynchronous data field

Basic rate ACL packets have an asynchronous data field consisting of two, three or four segments: a payload header, a payload body, possibly a MIC, and possibly a CRC code.

Enhanced Data Rate ACL packets have an asynchronous data field consisting of six or seven segments: a guard time, a synchronization sequence, a payload header, a payload body, possibly a MIC, a CRC and a trailer.

1. **Enhanced Data Rate guard time**

   This is the same as defined for the Synchronous data field in Section 6.6.1.

2. **Enhanced Data Rate synchronization sequence**

   This is the same as defined for the Synchronous data field in Section 6.6.1.

3. **Payload header**

   The payload header is one or two bytes long. Basic rate packets in segments one and two have a 1-byte payload header; Basic Rate packets in segments three

and four and all Enhanced Data Rate packets have a 2-byte payload header. The payload header specifies the logical link (2-bit LLID indication), controls the flow on the logical channels (1-bit FLOW indication), and has a payload length indicator (5 bits and 10 bits for 1-byte and 2-byte payload headers, respectively). In the case of a 2-byte payload header, the length indicator is extended by five bits into the next byte. The remaining three bits of the second byte are reserved for future use. The formats of the 1-byte and 2-byte payload headers are shown in Figure 6.12 and Figure 6.13.

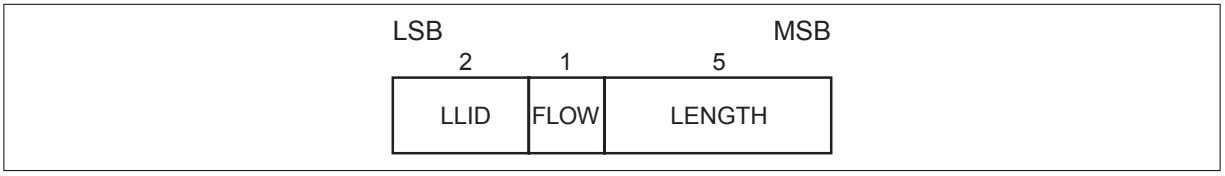



*Figure 6.12: Payload header format for Basic Rate single-slot ACL packets*

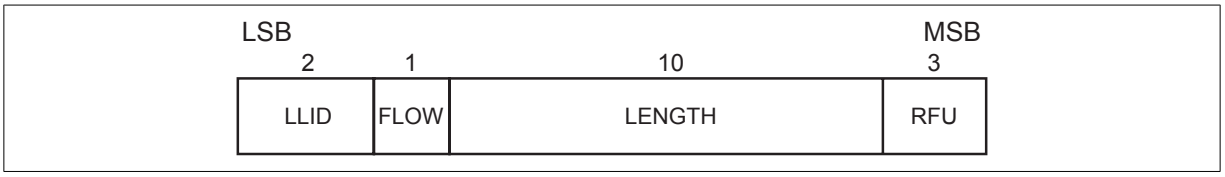



*Figure 6.13: Payload header format for multi-slot ACL packets and all EDR ACL packets*

The LLID field shall be transmitted first, the length field last. In Table 6.5, more details about the contents of the LLID field are listed.

| LLID | Logical Link | Information |
|---|---|---|
| 0b00 | | Reserved for future use |
| 0b01 | ACL-U and APB-U | Continuation fragment of an L2CAP message |
| 0b10 | ACL-U and APB-U | Start of an L2CAP message or no fragmentation |
| | PBD | Profile broadcast data, no fragmentation |
| 0b11 | ACL-C and APB-C | LMP message |

*Table 6.5: Logical Link LLID field contents*

An L2CAP message may be fragmented into several packets. Code 0b10 shall be used for an ACL-U or APB-U packet carrying the first fragment of such a message; code 0b01 shall be used for continuing fragments. The first fragment of an L2CAP message sent over HCI is identified by having a Packet_Boundary_Flag value of 0b00 or 0b10 both of which are mapped to LLID code 0b10. If there is no fragmentation, code 0b10 shall be used for every packet. ACL packets used over the PBD logical link do not use fragmentation. For ACL packets used over the

PBD logical link, LLID code 0b10 shall be used by the transmitter. Any ACL packet used over the PBD logical link with LLID not equal to 0b10 shall be ignored by the receiver. Code 0b11 shall be used for LMP messages. Code 0b00 is reserved for future use.

The flow indicator in the payload is used to control the flow at the L2CAP level. It is used to control the flow per logical link. FLOW=1 means flow-on (GO) and FLOW=0 means flow-off (STOP). After a new connection has been established the flow indicator shall be set to GO. When a device receives a payload header with the flow bit set to STOP, it shall stop the transmission of ACL-U packets before an additional amount of payload data is sent. This amount is defined as the flow control lag, expressed as a number of bytes. The shorter the flow control lag, the less buffering the other device must dedicate to this function. The flow control lag shall not exceed 1792 bytes (7 × 256 bytes). In order to allow devices to optimize the selection of packet length and buffer space, the flow control lag of a given implementation shall be provided in the LMP_FEATURES_RES message.

If a packet containing the payload flow bit of STOP is received, with a valid packet header but bad payload, the payload flow control bit shall be ignored. The Baseband acknowledgment contained in the packet header will be received and a further ACL packet may be transmitted. Each occurrence of this situation allows a further ACL packet to be sent in spite of the flow control request being sent via the payload header flow control bit. It is recommended that devices that use the payload header flow bit should ensure that no further ACL packets are sent until the payload flow bit has been correctly received. This can be accomplished by simultaneously turning on the flow bit in the packet header and keeping it on until an ACK is received back (ARQN=1). This will typically be only one round trip time.

The Baseband Resource Manager is responsible for setting and processing the flow bit in the payload header. Real-time flow control shall be carried out at the packet level by the Link Controller via the flow bit in the packet header (see Section 6.4.3). With the payload flow bit, traffic from the remote end can be controlled. It is allowed to generate and send an ACL packet with payload length zero irrespective of flow status. L2CAP start-fragment and continue-fragment indications (LLID=10 and LLID=01) also retain their meaning when the payload length is equal to zero (i.e. an empty start-fragment shall not be sent in the middle of an on-going ACL-U or APB-U packet transmission). It is always allowed to send an ACL packet with length=0 and LLID=01. The payload flow bit has its own meaning for each logical link, see Table 6.6. On the ACL-C, APB-C, and APB-U logical links, no flow control is applied and the payload FLOW bit shall always be set to one. On the PBD logical link, no flow control is applied and the payload FLOW bit shall always be set to zero.

| LLID code $b_1b_0$ | Logical Link | Usage and semantics of the ACL payload header FLOW bit |
|---|---|---|
| 00 | | Reserved for future use. |
| 01 | ACL-U | Flow control of the ACL-U channel (L2CAP messages). |
| | APB-U | Always set FLOW=1 on transmission and ignore the bit on reception. |
| | PBD | Reserved for future use. |
| 10 | ACL-U | Flow control of the ACL-U channel (L2CAP messages). |
| | APB-U | Always set FLOW=1 on transmission and ignore the bit on reception. |
| | PBD | Always set FLOW=0 on transmission and ignore the bit on reception. |
| 11 | ACL-C and APB-C | Always set FLOW=1 on transmission and ignore the bit on reception. |

*Table 6.6: Use of payload header flow bit on the Logical Links*

The length indicator shall be set to the number of bytes (i.e. 8-bit words) in the payload excluding the payload header, MIC, and the CRC code; i.e. the payload body only. With reference to Figure 6.12 and Figure 6.13, the MSB of the length field in a 1-byte header is the last (right-most) bit in the payload header; the MSB of the length field in a 2-byte header is the fourth bit to the left from the right-most end of the second byte in the payload header.

4. **Payload body**

   The payload body includes the user information and determines the effective user throughput. The length of the payload body is indicated in the length field of the payload header.

5. **Message Integrity Check**

   The 32-bit Message Integrity Check (MIC) in the payload is generated as specified in [Vol 2] Part H, Section 9.2.

6. **CRC code generation**

   The 16-bit cyclic redundancy check code in the payload is generated as specified in Section 7.1. Before determining the CRC code, an 8-bit value is used to initialize the CRC generator. For the CRC code in the FHS packets sent in Central Page Response substate, the UAP of the Peripheral is used. For the FHS packet and the extended inquiry response packet sent in Inquiry Response substate, the DCI (see Section 1.2.1) is used. For all other packets, the UAP of the Central is used.

   Only the Payload header, Payload body, and MIC segments (if present) are used to generate the CRC code.

7. **Enhanced Data Rate trailer**

   This is the same as defined for the Synchronous data field in Section 6.6.1.

## 6.7 Packet summary

A summary of the packets and their characteristics is shown in Table 6.7, Table 6.8, and Table 6.9. The payload represents the packet payload excluding FEC, CRC, and payload header.

| Type | Payload (bytes) | FEC | MIC | CRC | Symmetric Max. Rate | Asymmetric Max. Rate |
|---|---|---|---|---|---|---|
| ID | N/A | N/A | N/A | N/A | N/A | N/A |
| NULL | N/A | N/A | N/A | N/A | N/A | N/A |
| POLL | N/A | N/A | N/A | N/A | N/A | N/A |
| FHS | 18 | 2/3 | N/A | Yes | N/A | N/A |

*Table 6.7: Link control packets*

| Type | Payload Header (bytes) | User Payload (bytes) | FEC | MIC | CRC | Symmetric Max. Rate (kb/s) | Asymmetric Max. Rate (kb/s) | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | Forward | Reverse |
| DM1 | 1 | 0-17 | 2/3 | C.1 | Yes | 108.8 | 108.8 | 108.8 |
| DH1 | 1 | 0-27 | No | C.1 | Yes | 172.8 | 172.8 | 172.8 |
| DM3 | 2 | 0-121 | 2/3 | C.1 | Yes | 258.1 | 387.2 | 54.4 |
| DH3 | 2 | 0-183 | No | C.1 | Yes | 390.4 | 585.6 | 86.4 |
| DM5 | 2 | 0-224 | 2/3 | C.1 | Yes | 286.7 | 477.8 | 36.3 |
| DH5 | 2 | 0-339 | No | C.1 | Yes | 433.9 | 723.2 | 57.6 |
| 2-DH1 | 2 | 0-54 | No | C.1 | Yes | 345.6 | 345.6 | 345.6 |
| 2-DH3 | 2 | 0-367 | No | C.1 | Yes | 782.9 | 1174.4 | 172.8 |
| 2-DH5 | 2 | 0-679 | No | C.1 | Yes | 869.1 | 1448.5 | 115.2 |
| 3-DH1 | 2 | 0-83 | No | C.1 | Yes | 531.2 | 531.2 | 531.2 |
| 3-DH3 | 2 | 0-552 | No | C.1 | Yes | 1177.6 | 1766.4 | 265.6 |
| 3-DH5 | 2 | 0-1021 | No | C.1 | Yes | 1306.9 | 2178.1 | 177.1 |
| C.1: Mandatory when encryption with AES-CCM enabled, else excluded | | | | | | | | |

*Table 6.8: ACL packets.*

| Type | Payload Header (bytes) | User Payload (bytes) | FEC | MIC | CRC | Symmetric Max. Rate (kb/s) |
|---|---|---|---|---|---|---|
| HV1 | N/A | 10 | 1/3 | No | No | 64.0 |
| HV2 | N/A | 20 | 2/3 | No | No | 64.0 |
| HV3 | N/A | 30 | no | No | No | 64.0 |
| DV[1] | 1 D | 10+(0-9) D | 2/3 D | No | Yes D | 64.0+57.6 D |
| EV3 | N/A | 1-30 | No | No | Yes | 96 |
| EV4 | N/A | 1-120 | 2/3 | No | Yes | 192 |
| EV5 | N/A | 1-180 | No | No | Yes | 288 |
| 2-EV3 | N/A | 1-60 | No | No | Yes | 192 |
| 2-EV5 | N/A | 1-360 | No | No | Yes | 576 |
| 3-EV3 | N/A | 1-90 | No | No | Yes | 288 |
| 3-EV5 | N/A | 1-540 | No | No | Yes | 864 |

*Table 6.9: Synchronous packets*

[1]Items followed by ‘D’ relate to data field only.

# 7 BIT STREAM PROCESSING

Bluetooth devices shall use the bit stream processing schemes as defined in the following sections.

Before the payload is sent over the air interface, several bit manipulations are performed in the transmitter to increase reliability and security. An HEC is added to the packet header, the header bits are scrambled with a whitening word, and FEC coding is applied. In the receiver, the inverse processes are carried out. Figure 7.1 shows the processes carried out for the packet header both at the transmit and the receive side. All header bit processes are mandatory except that whitening and de-whitening shall not be performed on synchronization train packets. In Figure 7.1 processes not performed on all packet types are indicated by dashed blocks.

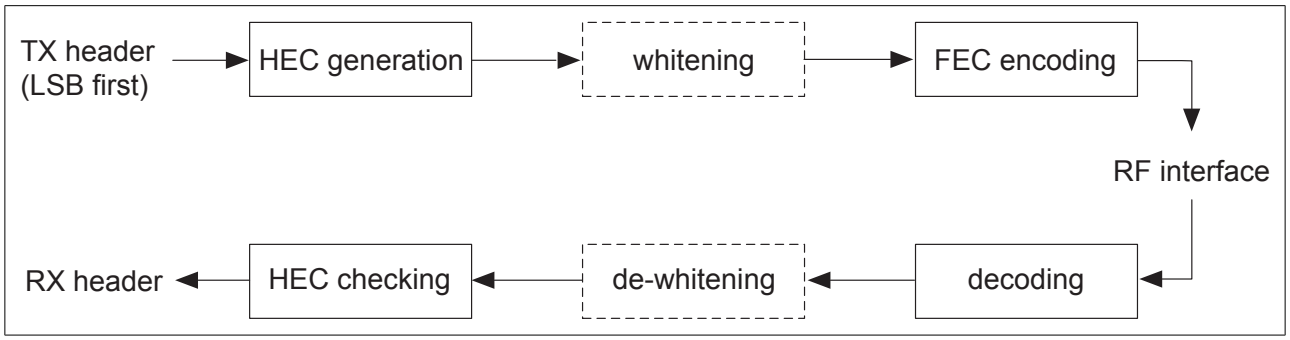



*Figure 7.1: Header bit processes*

Figure 7.2 shows the processes that may be carried out on the payload. In addition to the processes defined for the packet header, encryption can be applied on the payload. Whitening and de-whitening, as explained in Section 7.2, are mandatory for every payload except synchronization train payloads, where they are forbidden. All other processes are optional and depend on the packet type (see Section 6.6) and whether encryption is enabled. In Figure 7.2, optional processes are indicated by dashed blocks. When E0 encryption is used, the entire payload shall be encrypted. When AES-CCM encryption is used, only the payload body and MIC shall be encrypted; the payload header and CRC shall not be encrypted.

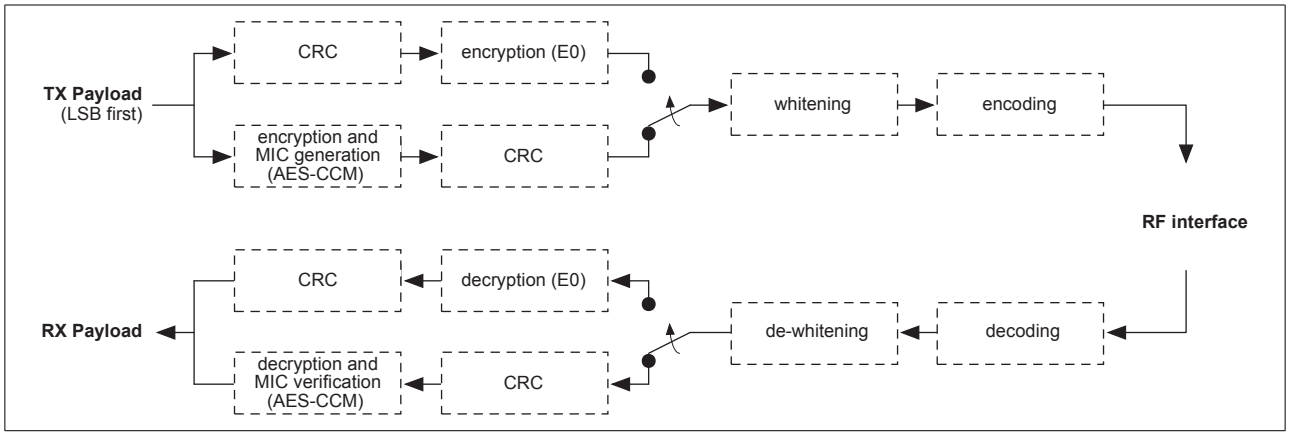



*Figure 7.2: Payload bit processes*

## 7.1 Error checking

The packet can be checked for errors or wrong delivery using the channel access code, the HEC in the header, and the CRC in the payload. At packet reception, the access code is checked first. Since the 64-bit sync word in the channel access code is derived from the 24-bit Central's LAP, this checks if the LAP is correct, and prevents the receiver from accepting a packet of another piconet (provided the LAP field of the Central's BD_ADDR is different).

The HEC and CRC computations are initialized as follows:

- In the Inquiry Response substate, the DCI value shall be used for the FHS and Extended Inquiry Response packet.
- In the Central Page Response substate, the UAP of the Peripheral shall be used in the FHS packet.
- In the Connection state, the UAP of the Central shall be used. Even though the access code may be the same for two piconets the different UAP values will typically cause the HEC and CRC to fail.
- During Role Switch starting from the TDD switch, the UAP of the new Peripheral shall be used for the FHS packet.

The generation and check of the HEC and CRC are summarized in Figure 7.5 and Figure 7.8. Before calculating the HEC or CRC, the shift registers in the HEC/CRC generators shall be initialized with the 8-bit UAP (or DCI) value. Then the header and payload information shall be shifted into the HEC and CRC generators, respectively (with the LSB first).

A packet has a valid header if the access code is correct for the piconet and the HEC has the correct value, irrespective of the values of the other header fields.

### 7.1.1 HEC generation

The HEC generating LFSR is depicted in Figure 7.3. The generator polynomial is $g(D) = (D + 1)(D^7 + D^4 + D^3 + D^2 + 1) = D^8 + D^7 + D^5 + D^2 + D + 1$. Initially this circuit shall be pre-loaded with the 8-bit UAP such that the LSB of the UAP (denoted $UAP_0$) goes to the left-most shift register element, and, $UAP_7$ goes to the right-most element. The initial state of the HEC LFSR is depicted in Figure 7.4. Then the data shall be shifted in with the switch S set in position 1. When the last data bit has been clocked into the LFSR, the switch S shall be set in position 2, and, the HEC can be read out from the register. The LFSR bits shall be read out from right to left (i.e., the bit in position 7 is the first to be transmitted, followed by the bit in position 6, etc.).

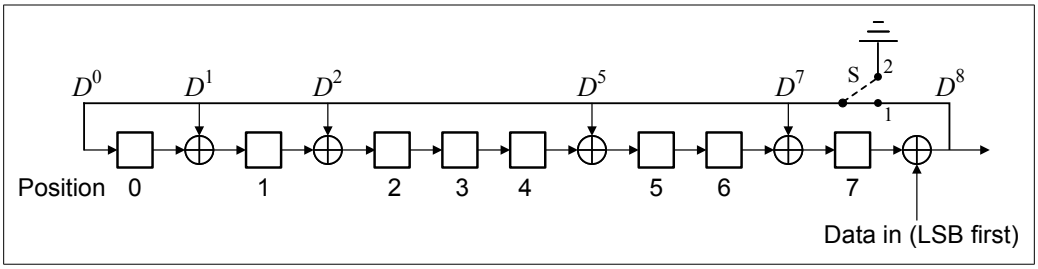



*Figure 7.3: LFSR circuit generating the HEC*

| **Position:** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| **LFSR:** | UAP 0 | UAP 1 | UAP 2 | UAP 3 | UAP 4 | UAP 5 | UAP 6 | UAP 7 |

*Figure 7.4: Initial state of the HEC generating circuit*

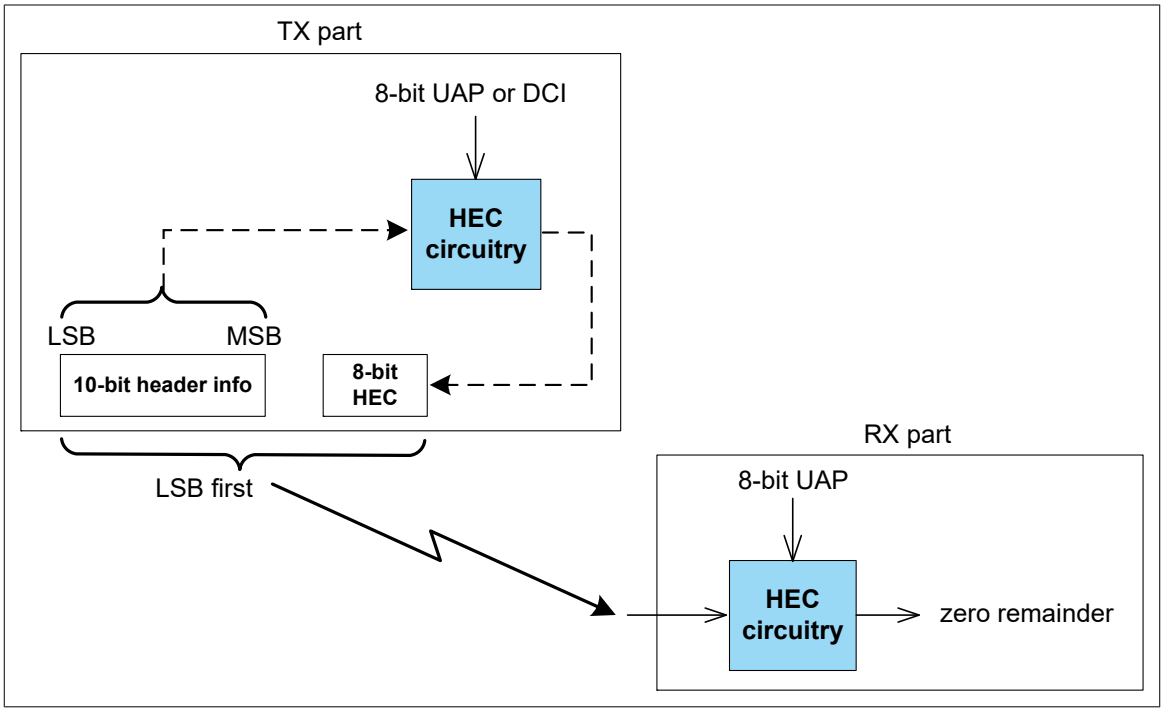



*Figure 7.5: HEC generation and checking*

### 7.1.2 CRC generation

The 16 bit LFSR for the CRC is constructed similarly to the HEC using the CRC-CCITT generator polynomial $g(D) = D^{16} + D^{12} + D^5 + 1$ (i.e., 0x11021) (see Figure 7.6). For this case, the 8 left-most bits shall be initially loaded with the 8-bit UAP ($UAP_0$ to the left and $UAP_7$ to the right) while the 8 right-most bits shall be reset to zero. The initial state of the 16 bit LFSR is specified in Figure 7.7. The switch S shall be set in position 1 while the data is shifted in. After the last bit has entered the LFSR, the switch shall be set in position 2, and, the register's contents shall be transmitted, from right to left (i.e., starting with position 15, then position 14, etc.).

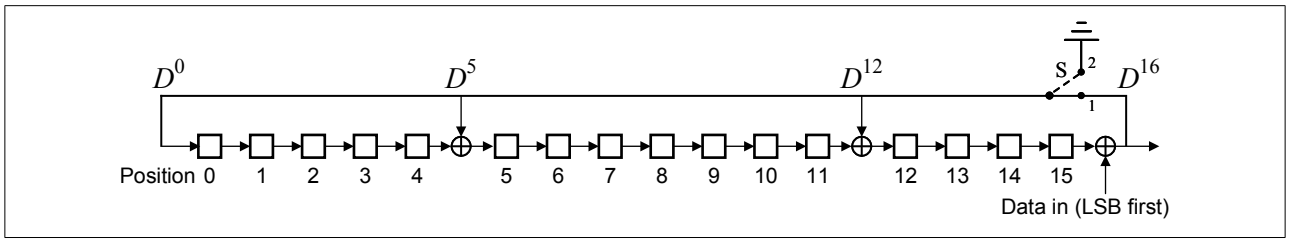



*Figure 7.6: The LFSR circuit generating the CRC*

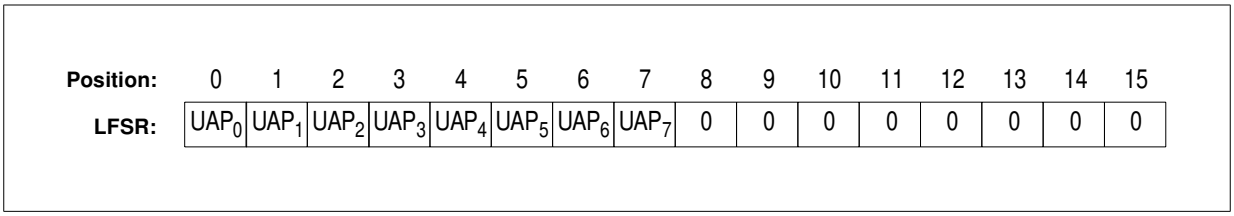



*Figure 7.7: Initial state of the CRC generating circuit*

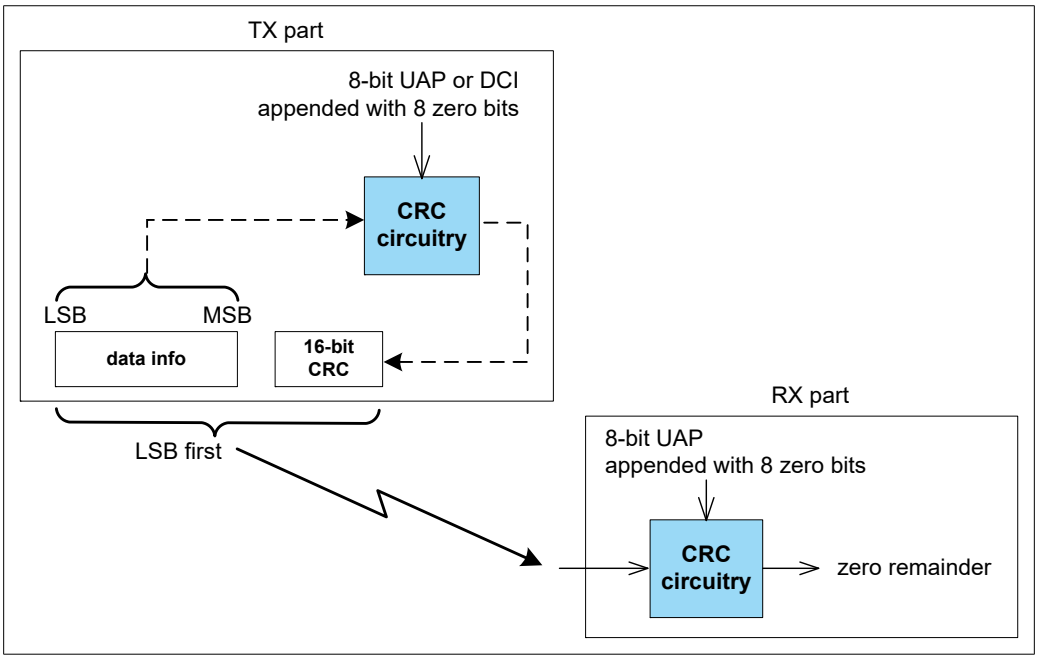



*Figure 7.8: CRC generation and checking*

## 7.2 Data whitening

Before transmission of all packets both the header and the payload shall be scrambled with a data whitening word in order to randomize the data from highly redundant patterns and to minimize DC bias in the packet. The scrambling shall be performed prior to the FEC encoding. As an exception, Synchronization train packets shall not be scrambled with a data whitening word.

At the receiver, the received data of scrambled packets shall be descrambled using the same whitening word generated in the recipient. The descrambling shall be performed after FEC decoding.

The whitening word is generated with the polynomial $g(D) = D^7 + D^4 + 1$ (i.e., 0x91) and shall be subsequently XORed with the header and the payload. The whitening word is generated with the linear feedback shift register shown in Figure 7.9. Before each transmission, the shift register shall be initialized with a portion of the Central's clock, $CLK_{6-1}$, extended with an MSB of value one. This initialization shall be carried out with

$CLK_1$ written to position 0, $CLK_2$ written to position 1, etc. Exceptions are the FHS packet sent during inquiry response or Central page response, and the extended inquiry response packet sent during inquiry response, where initialization of the whitening register shall be carried out differently. Instead of the Central's clock, the X-input used in the inquiry or page response (depending on current state) routine shall be used, see Table 2.2. The 5-bit value shall be extended with two MSBs of value 1. During register initialization, the LSB of X (i.e., $X_0$) shall be written to position 0, $X_1$ shall be written to position 1, etc.

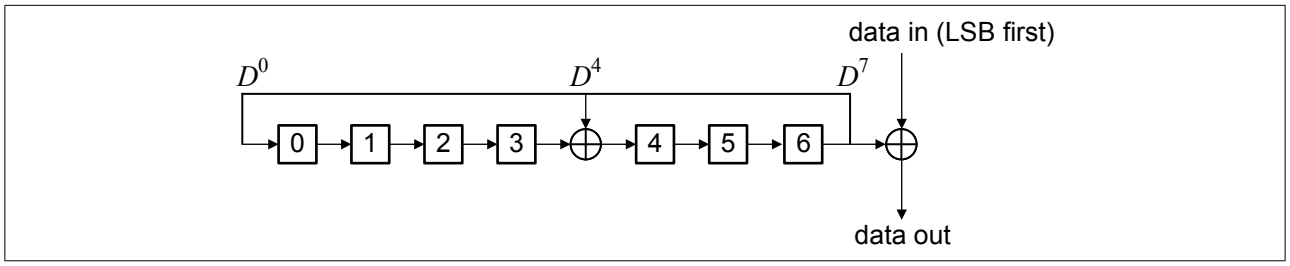



*Figure 7.9: Data whitening LFSR*

After initialization, the packet header and the payload (including the CRC) are whitened. The payload whitening shall continue from the state the whitening LFSR had at the end of HEC. There shall be no re-initialization of the shift register between packet header and payload. The first bit of the “data in” sequence shall be the LSB of the packet header.

For Enhanced Data Rate packets, whitening shall not be applied to the guard, synchronization and trailer portions of the Enhanced Data Rate packets. During the periods where whitening is not applied the LFSR shall be paused.

## 7.3 Error correction

There are three error correction schemes defined for Bluetooth:

- 1/3 rate FEC
- 2/3 rate FEC
- ARQ scheme for the data

The purpose of the FEC scheme on the data payload is to reduce the number of retransmissions. However, in a reasonably error-free environment, FEC gives unnecessary overhead that reduces the throughput. Therefore, the packet definitions given in Section 6 have been kept flexible to use FEC in the payload or not, resulting in the **DM** and **DH** packets for the ACL logical transport, **HV** packets for the SCO logical transport, and **EV** packets for the eSCO logical transport. The packet header is always protected by a 1/3 rate FEC since it contains valuable link information and is designed to withstand more bit errors.

Correction measures to mask errors in the voice decoder are not included in this section. This matter is discussed in Section 9.3.

## 7.4 FEC code: rate 1/3

A simple 3-times repetition FEC code is used for the header. The repetition code is implemented by repeating each bit three times, see the illustration in Figure 7.10. The 3-times repetition code is used for the entire header, as well as for the synchronous data field in the **HV1** packet.

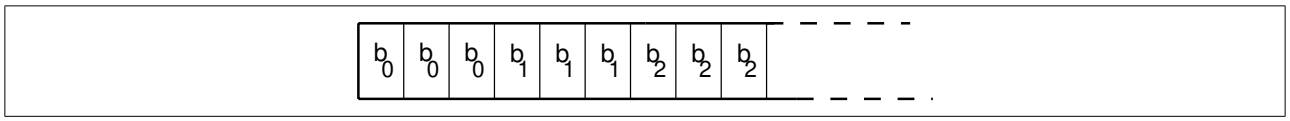



*Figure 7.10: Bit-repetition encoding scheme*

## 7.5 FEC code: rate 2/3

The other FEC scheme is a (15,10) shortened Hamming code. The generator polynomial is $g(D) = (D + 1)(D^4 + D + 1)$. This corresponds to 0x35. The LFSR generating this code is depicted in Figure 7.11. Initially all register elements are set to zero. The 10 information bits are sequentially fed into the LFSR with the switches S1 and S2 set in position 1. Then, after the final input bit, the switches S1 and S2 are set in position 2, and the five parity bits are shifted out. The parity bits are appended to the information bits. Subsequently, each block of 10 information bits is encoded into a 15 bit codeword. This code can correct all single errors and detect all double errors in each codeword. This 2/3 rate FEC is used in the **DM** packets, in the data field of the **DV** packet, in the **FHS** packet, in the **HV2** packet, and in the **EV4** packet. Since the encoder operates with information segments of length 10, tail bits with value zero shall be appended after the CRC bits to bring the total number of bits equal to a multiple of 10. The number of tail bits to append shall be the least possible that achieves this (i.e., in the interval 0...9). These tail bits are not included in the payload length indicator for ACL packets or in the payload length field of the eSCO setup LMP command.

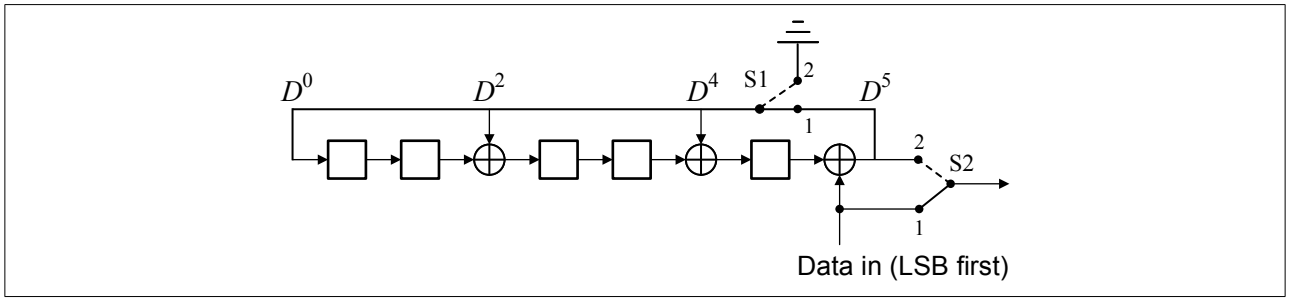



*Figure 7.11: LFSR generating the (15,10) shortened Hamming code*

## 7.6 ARQ scheme

With an automatic repeat request scheme, DM packets, DH packets, the data field of DV packets, and EV packets shall be transmitted until acknowledgment of a successful reception is returned by the destination (or timeout is exceeded). The acknowledgment information shall be included in the header of the return packet. The ARQ scheme is only used on the payload in the packet and only on packets that have a CRC. The packet header and the synchronous data payload of HV and DV packets are not protected by the ARQ scheme.

### 7.6.1 Unnumbered ARQ

Bluetooth uses a fast, unnumbered acknowledgment scheme. An ACK (ARQN=1) or a NAK (ARQN=0) is returned in response to the receipt of a previously received packet. The Peripheral shall respond in the Peripheral-to-Central slot directly following the Central-to-Peripheral slot unless the Peripheral has scatternet commitments in that timeslot; the Central shall respond at the next event addressing the same Peripheral (the Central may have addressed other Peripherals between the last received packet from the considered Peripheral and the Central's response to this packet). A packet reception is successful if the HEC passes and, when present, the MIC and CRC pass.

In the first POLL packet at the start of a new connection (as a result of a page, page scan, or role switch) the Central shall initialize the ARQN bit to NAK. The response packet sent by the Peripheral shall also have the ARQN bit set to NAK. The subsequent packets shall use the following rules. The initial value of the Central's eSCO ARQN at link set-up shall be NAK.

The ARQN bit shall only be affected by data packets containing CRC and empty slots. As shown in Figure 7.12, Figure 7.13, and Figure 7.14, upon successful reception of a CRC packet, the ARQN bit shall be set to ACK. If, in any receive slot in the Peripheral, or, in a receive slot in the Central following transmission of a packet, one of these events applies:

1. no access code is detected
2. the HEC fails
3. the CRC fails
4. the MIC fails.

then the ARQN bit shall be set to NAK except in the conditions below:

- In eSCO the ARQN bit may be set to ACK even when the CRC on an EV packet has failed thus enabling delivery of erroneous packets.
- For ACL packets, if a CRC packet with a correct header has the same SEQN as the previously received CRC packet, the ARQN bit shall be set to ACK and the payload shall be ignored without checking the CRC.

Packets that have correct HEC but that are addressed to other Peripherals, or packets other than DH, DM, DV or EV packets, shall not affect the ARQN bit, except as noted in Section 7.6.2.2. In these cases the ARQN bit shall be left as it was prior to reception of the packet. For eSCO packets, the SEQN shall not be used when determining the ARQN. If an eSCO packet has been received successfully within the eSCO window subsequent receptions within the eSCO window shall be ignored. At the end of the eSCO window, the Central's ARQN shall be retained for the first Central-to-Peripheral transmission in the next eSCO window.

The ARQN bit in the FHS packet is not meaningful. Contents of the ARQN bit in the FHS packet shall not be checked.

The ARQN bit in the extended inquiry response packet is reserved for future use.

Broadcast packets shall be checked on errors using the CRC, but no ARQ scheme shall be applied. Broadcast packets shall never be acknowledged.

In Figure 7.12, TX refers to the next transmission that the device makes, which might not be in the next slot. NO TX indicates that the device is not allowed to transmit in the next slot.

In Figure 7.14, "Packet OK" means that the packet has an allowed TYPE for the connection and the CRC is valid.

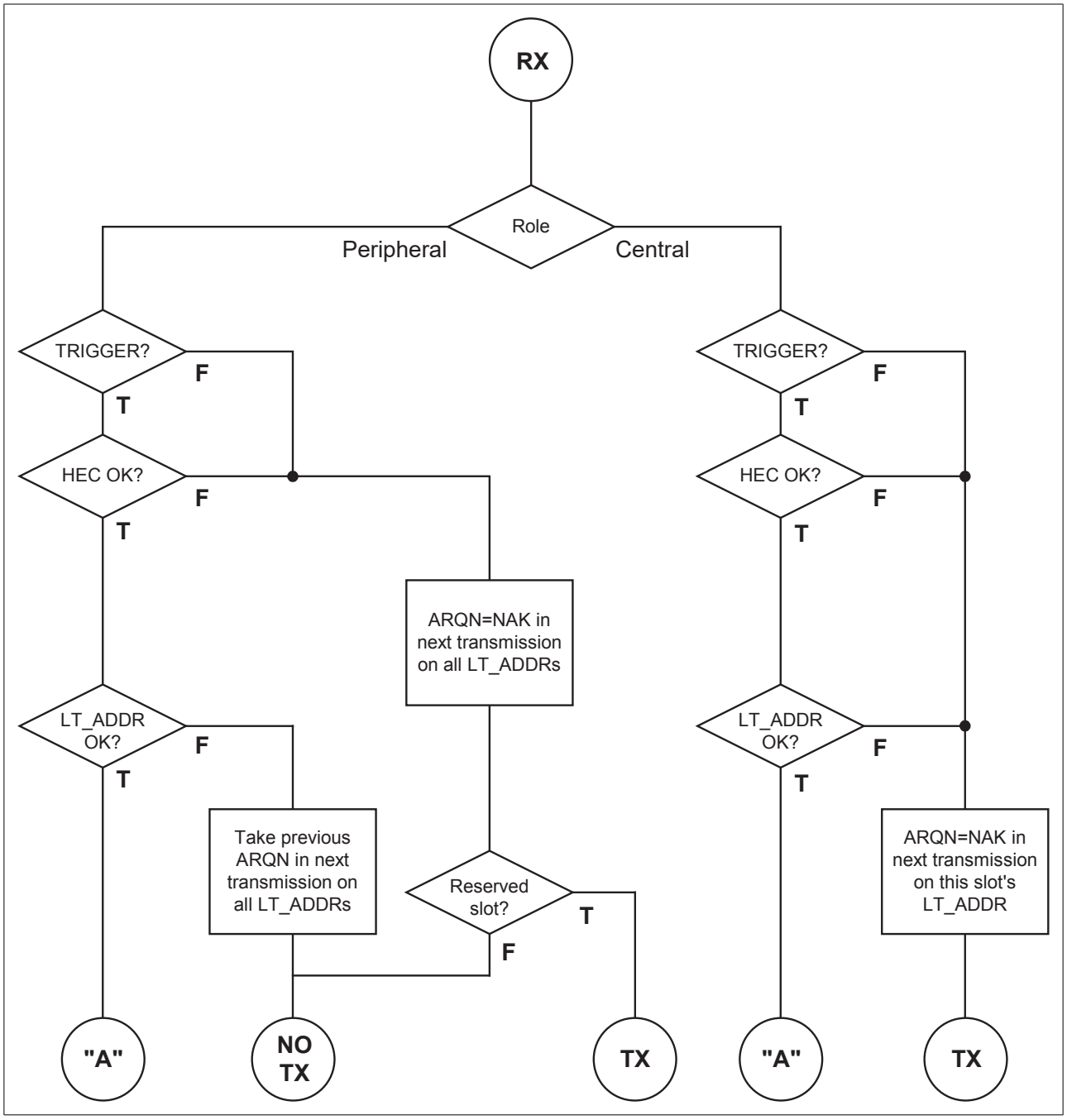



*Figure 7.12: Stage 1 of the receive protocol for determining the ARQN bit*

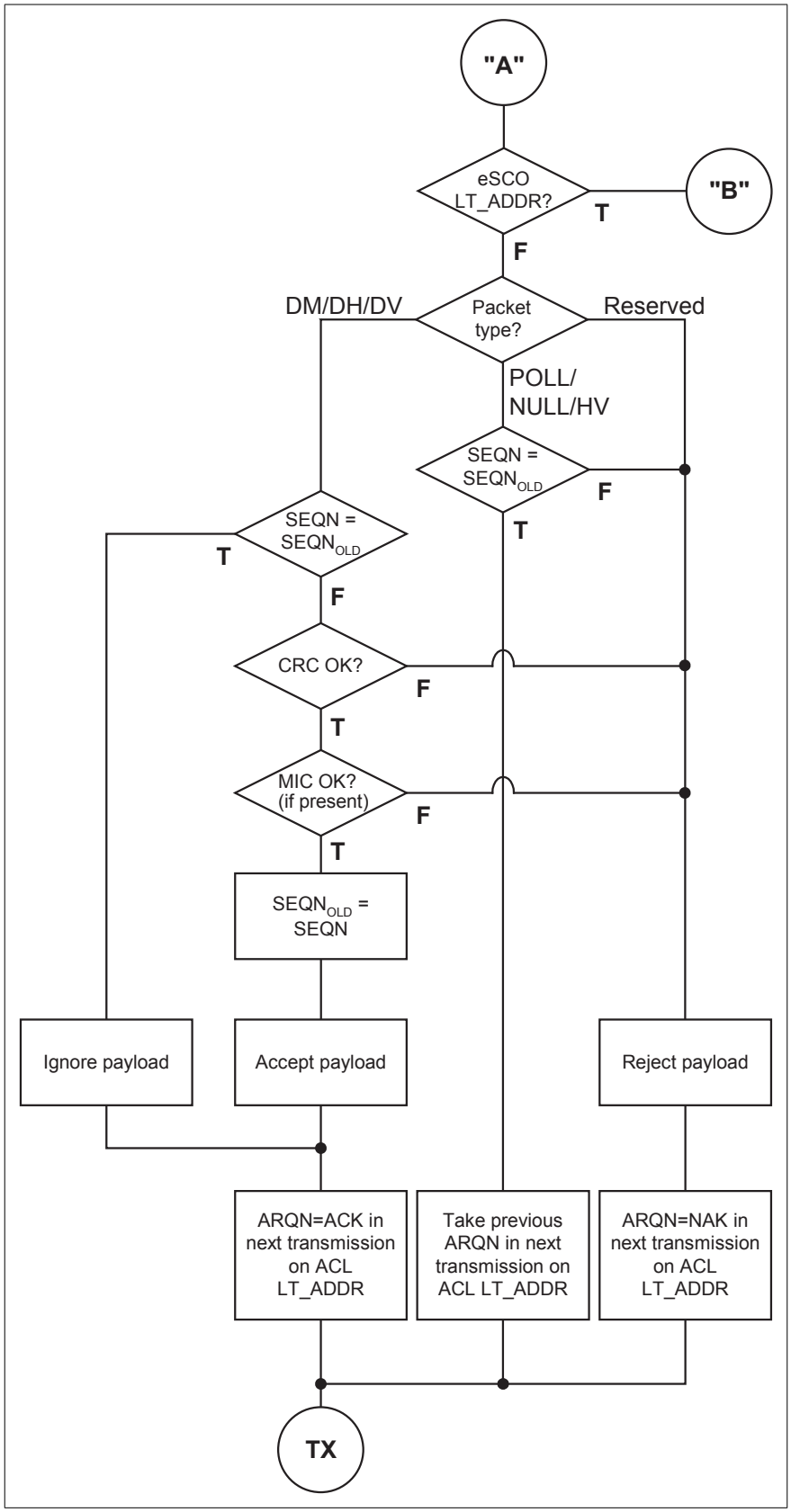



*Figure 7.13: Stage 2 (ACL) of the receive protocol for determining the ARQN bit*

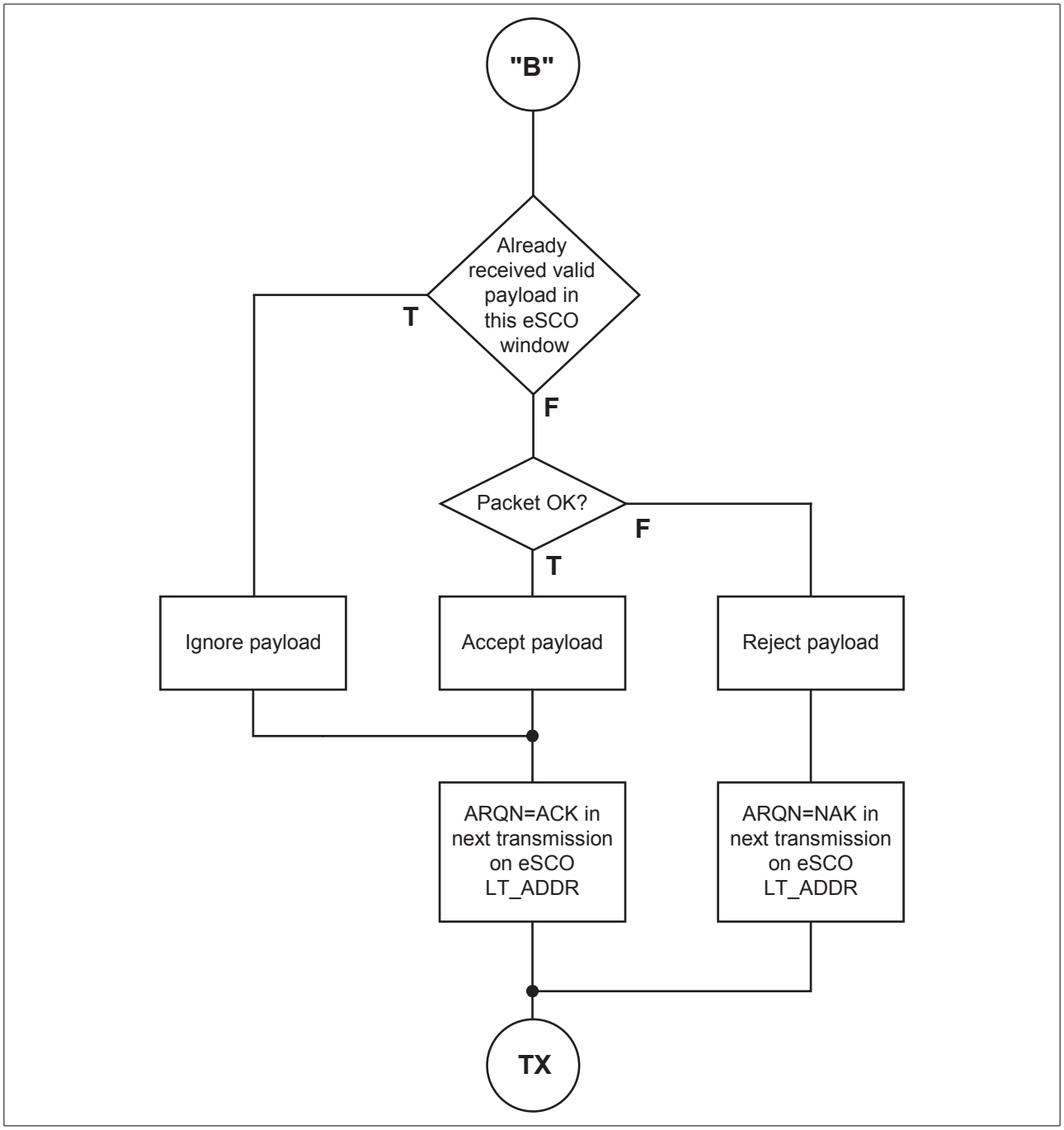



*Figure 7.14: Stage 2 (eSCO) of the receive protocol for determining the ARQN bit*

### 7.6.2 Retransmit filtering

The data payload shall be transmitted until a positive acknowledgment is received or a timeout is exceeded. A retransmission shall be carried out either because the packet transmission itself failed, or because the acknowledgment transmitted in the return packet failed (which has a lower failure probability since the header is more heavily coded). In the latter case, the destination keeps receiving the same payload over and

over again. In order to filter out the retransmissions in the destination, the SEQN bit is present in the header. Normally, this bit is alternated for every new CRC data payload transmission. In case of a retransmission, this bit shall not be changed so the destination can compare the SEQN bit with the previous SEQN value. If different, a new data payload has arrived; otherwise it is the same data payload and may be ignored. Only new data payloads shall be transferred to the Baseband Resource Manager.

Note: CRC data payloads can be carried only by **DM**, **DH**, **DV** or **EV** packets.

#### 7.6.2.1 Initialization of SEQN at start of new connection

The SEQN bit of the first CRC data packet at the start of a connection (as a result of page, page scan, or role switch) on both the Central and the Peripheral sides shall be set to 1. The subsequent packets shall use the rules in the following sections.

#### 7.6.2.2 ACL and SCO retransmit filtering

The SEQN bit shall only be affected by the CRC data packets as shown in Figure 7.15. It shall be inverted every time a new CRC data packet is sent. The CRC data packet shall be retransmitted with the same SEQN number until an ACK is received or the packet is flushed. When an ACK is received, a new payload may be sent and on that transmission the SEQN bit shall be inverted. If a device decides to flush (see Section 7.6.3), and it has not received an acknowledgment for the current packet, it shall replace the current packet with an ACL-U continuation packet with the same sequence number as the current packet and length zero. If it replaces the current packet in this way it shall not move on to transmit the next packet until it has received an ACK.

If the Peripheral receives a packet other than DH, DM, DV or EV with the SEQN bit inverted from that in the last header successfully received on the same LT_ADDR, it shall set the ARQN bit to NAK until a DH, DM, DV or EV packet is successfully received.

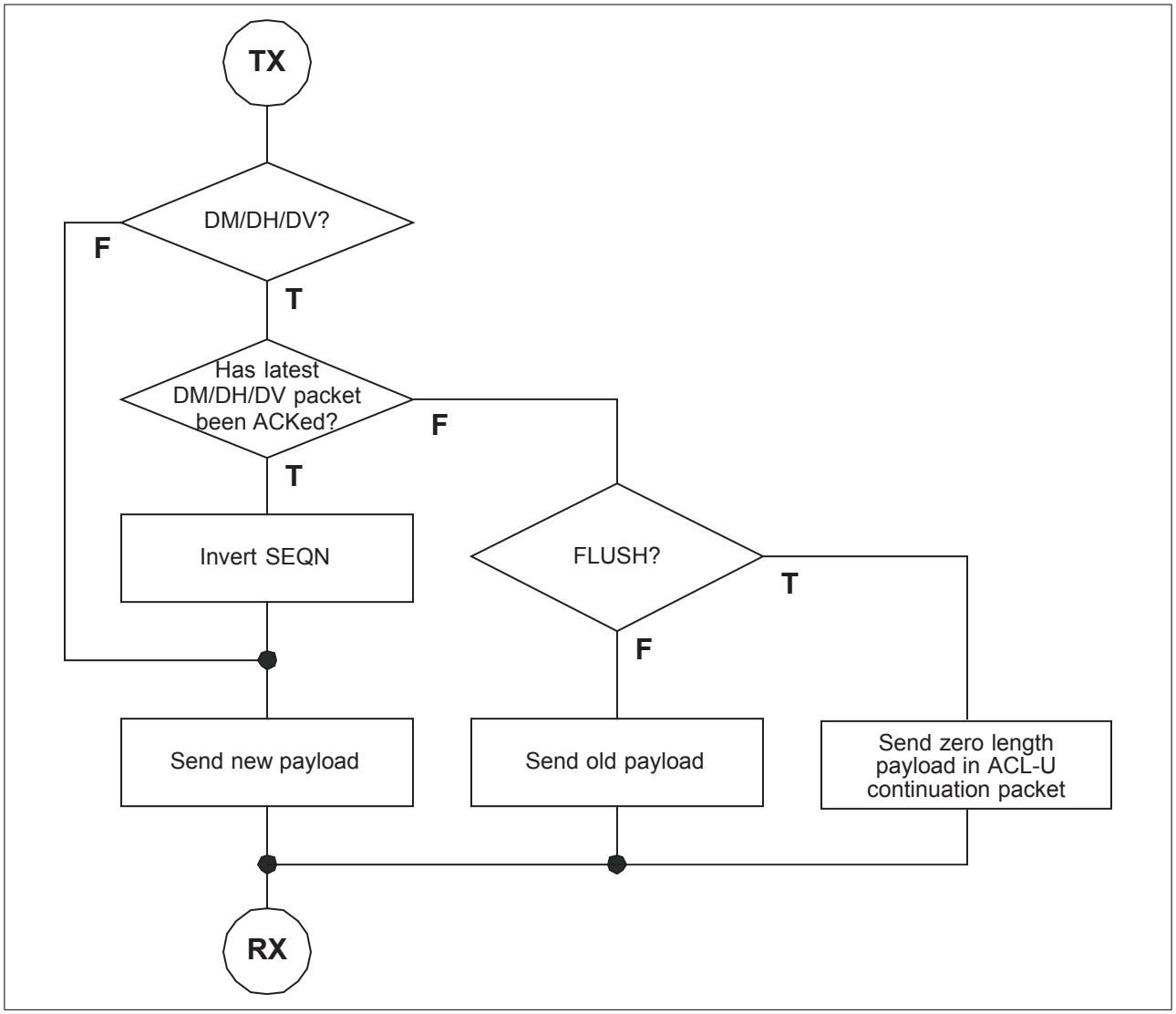



*Figure 7.15: Transmit filtering for packets with CRC*

#### 7.6.2.3 eSCO retransmit filtering

In eSCO, the SEQN bit shall be toggled every eSCO window. The value shall be constant for the duration of the eSCO window. The initial value of SEQN shall be zero.

For a given eSCO window the SEQN value shall be constant.

#### 7.6.2.4 FHS retransmit filtering

The SEQN bit in the FHS packet is not meaningful. This bit may be set to any value. Contents of the SEQN bit in the FHS packet shall not be checked.

#### 7.6.2.5 Extended inquiry response retransmit filtering

The SEQN bit in the extended inquiry response packet is reserved for future use.

#### 7.6.2.6 Packets without CRC retransmit filtering

During transmission of packets without a CRC the SEQN bit shall remain the same as it was in the previous packet.

### 7.6.3 Flushing payloads

On ACL logical transports, the ARQ scheme can cause variable delay in the traffic flow since retransmissions are inserted to assure error-free data transfer. For certain communication links, only a limited amount of delay is allowed: retransmissions are allowed up to a certain limit at which the current payload shall be ignored. This data transfer is indicated as **isochronous traffic**. This means that the retransmit process must be overruled in order to continue with the next data payload. Aborting the retransmit scheme is accomplished by *flushing* the old data and forcing the Link Controller to take the next data instead.

Flushing results in loss of remaining portions of an L2CAP message. Therefore, the packet following the flush shall have a start packet indication of LLID = 10 in the payload header for the next L2CAP message. This informs the destination of the flush (see Section 6.6). Flushing will not necessarily result in a change in the SEQN bit value, see the previous section.

The Flush Timeout defines a maximum period after which all segments of the ACL-U packet with a Packet_Boundary_Flag value of 10 are flushed from the Controller buffer. The Flush Timeout shall start when the First segment of the ACL-U packet is stored in the Controller buffer. If the First segment of an ACL-U packet has a Packet_Boundary_Flag value of 00, it is non-automatically-flushable and shall not cause the Flush Timeout to start. After the Flush timeout has expired the Link Controller may continue transmissions according to the procedure described in Section 7.6.2.2, however the Baseband Resource Manager shall not continue the transmission of the ACL-U packet to the Link Controller. If the Baseband Resource Manager has further segments of the packet queued for transmission to the Link Controller it shall delete the remaining segments of the ACL-U packet from the queue. In case the complete ACL-U packet was not stored in the Controller buffer yet, any Continuation segments, received for the ACL logical transport, shall be flushed, until a First segment is received. When the complete ACL-U packet has been flushed, the Link Manager shall continue transmission of the next ACL-U packet for the ACL logical transport. The default Flush Timeout shall be infinite, i.e. re-transmissions are carried out until physical link loss occurs. This is also referred to as a 'reliable channel.' All devices shall support the default Flush Timeout. Reliable data shall be sent over a channel with a finite flush timeout by marking reliable packets as non-automatically-flushable.

On eSCO logical transports, packets shall be automatically flushed at the end of the eSCO window.

Connectionless Peripheral Broadcast packets are transmitted at each scheduled Connectionless Peripheral Broadcast Instant. If no new data is available, the Connectionless Peripheral Broadcast Transmitter shall transmit a packet with the last available data in the payload.

### 7.6.4 Multi-Peripheral considerations

In a piconet with multiple logical transports, the Central shall carry out the ARQ protocol independently on each logical transport.

### 7.6.5 Active Peripheral Broadcast packets

APB broadcast packets are packets transmitted by the Central to all the Peripherals simultaneously (see Section 8.6.4) If multiple hop sequences are being used each transmission may only be received by some of the Peripherals. In this case the Central shall repeat the transmission on each hop sequence. An APB broadcast packet shall be indicated by the all-zero LT_ADDR.

Note: The FHS packet and the extended inquiry response packet are the only packets which can have an all-zero LT_ADDR but are not broadcast packets.

Broadcast packets shall not be acknowledged (at least not at the LC level), hence each broadcast packet is transmitted at least a fixed number of times. An APB broadcast packet should be transmitted $N_{BC}$ times before the next broadcast packet of the same broadcast message is transmitted (see Figure 7.16), but may be transmitted $N_{BC}$ + 1 times, so $N_{BC}$=1 means that each broadcast packet should be sent only once but may be sent twice. However, time-critical broadcast information may abort the ongoing broadcast train. In addition, an LMP packet sent on the APB-C logical link is always a complete message for these purposes and is always sent exactly once; the LMP specification may provide requirements for whether and when to send another message with the same or related contents.

If multiple hop sequences are being used then the Central may transmit on the different hop sequences in any order, providing that transmission of a new broadcast packet shall not be started until all transmissions of any previous broadcast packet have completed on all hop sequences. The transmission of a single broadcast packet may be interleaved among the hop sequences to minimize the total time to broadcast a packet. The Central has the option of transmitting only $N_{BC}$ times on channels common to all hop sequences.

APB broadcast packets with a CRC shall have their own sequence number. The SEQN of the first broadcast packet with a CRC shall be set to SEQN = 1 by the Central and shall be inverted for each new broadcast packet with CRC thereafter. APB broadcast packets without a CRC have no influence on the sequence number. The Peripheral shall accept the SEQN of the first broadcast packet it receives in a connection and

shall check for change in SEQN for subsequent broadcast packets. Since there is no acknowledgment of broadcast messages and there is no end packet indication, it is important to receive the start packets correctly. To ensure this, repetitions of the broadcast packets that are L2CAP start packets and LMP packets shall not be filtered out. These packets shall be indicated by LLID=1X in the payload header as explained in Table 6.5. Only repetitions of the L2CAP continuation packets shall be filtered out.

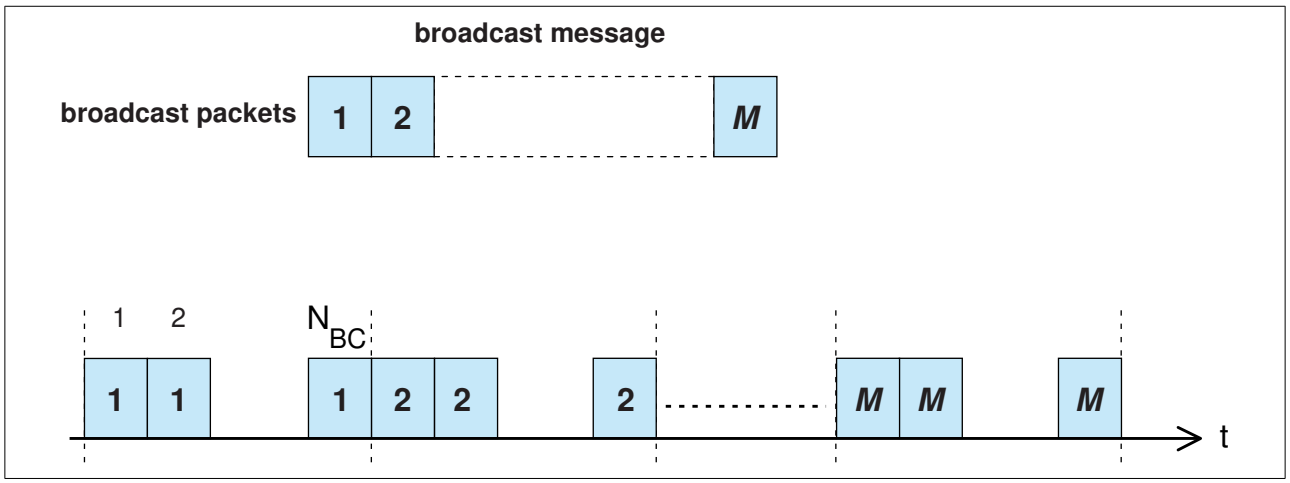



*Figure 7.16: Broadcast repetition scheme*

## 7.7 Erroneous synchronous data reporting

Erroneous data reporting may be enabled for synchronous links. When enabled, synchronous data shall be processed as follows:

- If, during an (e)SCO interval, an eSCO packet was received with valid HEC, an allowed TYPE for the connection, and valid CRC, or a SCO packet was received with valid HEC, the received payload data shall be sent to the upper-edge of the Controller with a "good data" indication.
- If, during an eSCO interval, eSCO packets with valid HEC were received, but none of them had an allowed TYPE for the connection and a valid CRC, the best known data available (e.g., data derived from the received data or the actual payload data that was received with a CRC error) shall be sent to the upper-edge of the Controller with a "data with possible errors" indication.
- If, during an (e)SCO interval, no SCO or eSCO packet with valid HEC has been received, a "lost data" indication shall be sent to the upper-edge of the Controller.

## 7.8 Message Integrity Check

When encryption with AES-CCM is enabled, a Message Integrity Check (MIC) is added to the payload before the CRC for packets that are defined to include the MIC. The MIC is sent Most Significant Octet (MSO) first.

# 8 LINK CONTROLLER OPERATION

This section describes how a piconet is established and how devices can be added to and released from the piconet. Several states of operation of the devices are defined to support these functions. In addition, the operation of several piconets with one or more common members, the scatternet, is discussed.

## 8.1 Overview of states

Figure 8.1 shows a state diagram illustrating the different states used in the Link Controller. There are two major states: Standby and Connection; in addition, there are nine substates, Page, Page Scan, Inquiry, Inquiry Scan, Synchronization Train, Synchronization Scan, Central Page Response, Peripheral Page Response, and Inquiry Response (the Central Page Response, Peripheral Page Response, and Inquiry Response substates are not shown in the simplified figure below). The substates are interim states that are used to establish connections and enable device discovery. To move from one state or substate to another, either commands from the link manager are used, or internal signals in the Link Controller are used (such as the trigger signal from the correlator and the timeout signals).

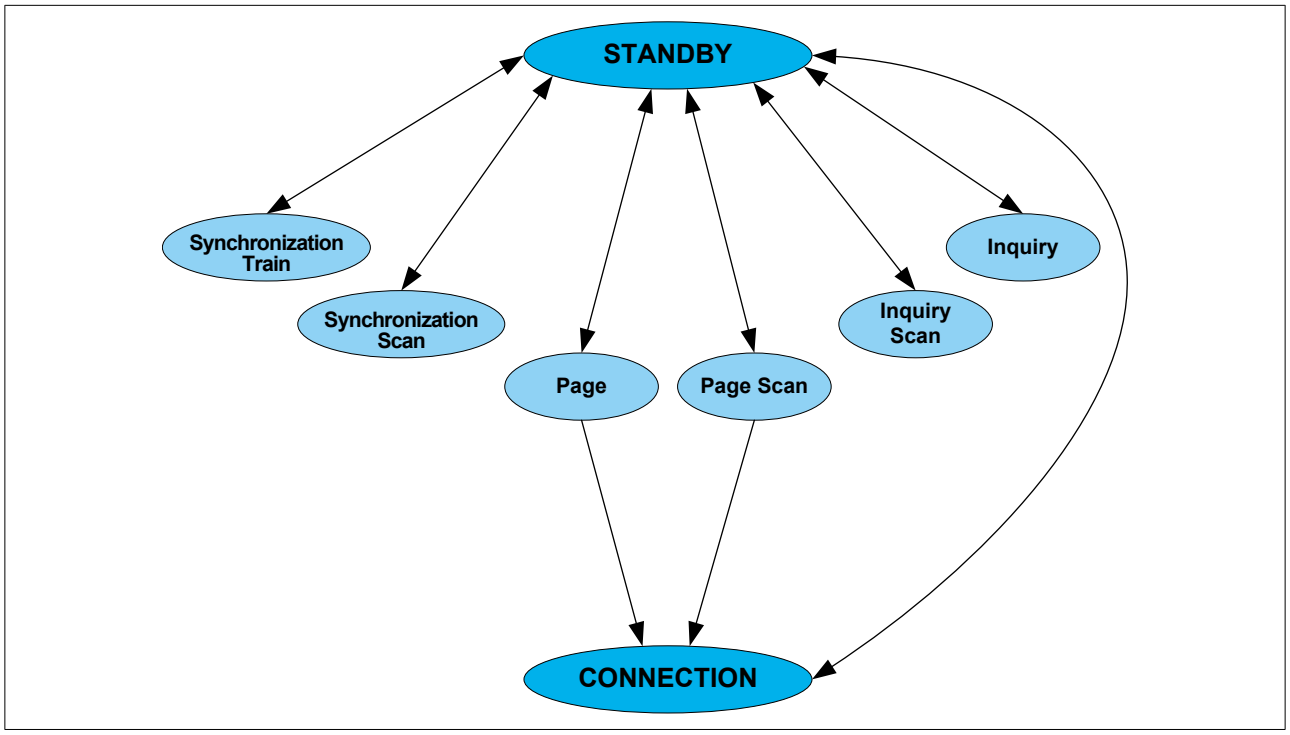



*Figure 8.1: State diagram of Link Controller*

## 8.2 Standby state

The Standby state is the default state in the device. In this state, the device may be in a low-power mode. Only the native clock is running with a worst case accuracy of ±250 ppm.

The Controller may leave the Standby state to scan for page or inquiry messages, or to page or inquiry itself.

## 8.3 Connection establishment substates

In order to establish new connections the paging procedure or the synchronization scan procedure is used. Only the Bluetooth Device Address is required to set up a connection using the paging procedure. Knowledge about the clock, obtained from the inquiry procedure (see Section 8.4) or from a previous connection with this device, and the page scanning mode of the other device will accelerate the paging procedure. A device that establishes a connection using a page procedure will automatically become the Central of the connection. A device that establishes a connection using a synchronization scan procedure does so in two steps. First, from within the Synchronization Scan substate, the BR/EDR Controller follows the synchronization scan procedure to collect clock, packet timing, and AFH channel map information from the Central and then transitions to the Standby state. Second, as commanded by the Host, the BR/EDR Controller transitions from the Standby state to the Connectionless Peripheral Broadcast mode of the Connection state.

The truncated page procedure is a modification of the paging procedure and is used to deliberately generate a page response timeout on the Peripheral.

### 8.3.1 Page Scan substate

In the Page Scan substate, a device may be configured to use either the standard or generalized interlaced scanning procedure. During a standard scan, a device listens for the duration of the scan window $T_{w_page_scan}$ (11.25 ms default, see [Vol 4] Part E, Section 7.3.20), while the generalized interlaced scan is performed as two back to back scans of $T_{w_page_scan}$. If the scan interval is not at least twice the scan window, then generalized interlaced scan shall not be used. During each scan window, the device shall listen at a single hop frequency, its correlator matched to its device access code (DAC). The scan window shall be long enough to completely scan 16 page frequencies.

When a device enters the Page Scan substate, it shall select the scan frequency according to the page hopping sequence determined by the device's Bluetooth Device Address, see Section 2.6.4.1. The phase in the sequence shall be determined by $CLKN_{16\text{-}12}$ of the device's native clock; that is, every 1.28 s a different frequency is selected.

In the case of a standard scan, if the correlator exceeds the trigger threshold during the page scan, the device shall enter the Peripheral Page Response substate described in Section 8.3.3.1. The scanning device may also use generalized interlaced scan. In this case, if the correlator does not exceed the trigger threshold during the first scan it shall scan a second time using the phase in the sequence determined by [$CLKN_{16-12}$ + interlace_offset] *mod* 32. If on this second scan the correlator exceeds the trigger threshold the device shall enter the Peripheral Page Response substate using [$CLKN_{16-12}$ + interlace_offset] *mod* 32 as the frozen CLKN* in the calculation for Xprp (see Section 2.6.4.3 for details). If the correlator does not exceed the trigger threshold during a scan in normal mode or during the second scan in interlaced scan mode it shall return to either the Standby or Connection state.

The interlace_offset value ranges from 0 to 31. The value 16 should be used unless the pattern of slots that are not available for scanning implies a different value should be used.

The Page Scan substate can be entered from the Standby state or the Connection state. In the Standby state, no connection has been established and the device can use all the capacity to carry out the page scan. Before entering the Page Scan substate from the Connection state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL connections in Hold mode (see Section 8.8) or Sniff mode (see Section 8.7). Synchronous connections should not be interrupted by the page scan, although eSCO retransmissions should be paused during the scan. The page scan may be interrupted by the reserved synchronous slots which should have higher priority than the page scan. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window shall be increased to minimize the setup delay. If one SCO logical transport is present using **HV3** packets and $T_{SCO}$=6 slots or one eSCO logical transport is present using **EV3** packets and $T_{eSCO}$=6 slots, a total scan window $T_{w_page_scan}$ of at least 36 slots (22.5 ms) is recommended; if two SCO links are present using **HV3** packets and $T_{SCO}$=6 slots or two eSCO links are present using **EV3** packets and $T_{eSCO}$=6 slots, a total scan window of at least 54 slots (33.75 ms) is recommended.

The scan interval $T_{page_scan}$ is defined as the interval between the beginnings of two consecutive page scans. A distinction is made between the case where the scan interval is equal to the scan window $T_{w_page_scan}$ (continuous scan), the scan interval is maximal 1.28 s, or the scan interval is maximal 2.56 s. These three cases shall determine the behavior of the paging device; that is, whether the paging device shall use R0, R1 or R2, see also Section 8.3.2. Table 8.1 illustrates the relationship between $T_{page_scan}$ and modes R0, R1 and R2. Although scanning in the R0 mode is continuous, the scanning may be interrupted for example by reserved synchronous slots. The scan interval information (also known as the page scan repetition mode) is included in the SR field in the FHS packet.

| SR mode | $T_{page_scan}$ |
|---|---|
| R0 | ≤ 1.28 s and = $T_{w_page_scan}$ |
| R1 | ≤ 1.28 s |
| R2 | ≤ 2.56 s |

*Table 8.1: Relationship between scan interval and page scan repetition modes R0, R1, and R2*

### 8.3.2 Page substate

The Page substate is used by the Central (source) to activate and connect to a Peripheral (destination) in the Page Scan substate. The Central tries to coincide with the Peripheral's scan activity by repeatedly transmitting the paging message consisting of the Peripheral's device access code (DAC) in different hop channels. Since the Bluetooth clocks of the Central and the Peripheral are not synchronized, the Central does not know exactly when the Peripheral wakes up and on which hop frequency. Therefore, it transmits a train of identical page messages at different hop frequencies and listens in between the transmit intervals until it receives a response from the Peripheral.

The page procedure in the Central consists of a number of steps. First, the Host communicates the BD_ADDR of the Peripheral to the Controller. This BD_ADDR shall be used by the Central to determine the page hopping sequence; see Section 2.6.4.2. If the BD_ADDR of the Peripheral is identical to the BD_ADDR of the Central, then the Central's Controller shall reject the page procedure. For the phase in the sequence, the Central shall use an estimate of the Peripheral's clock. For example, this estimate can be derived from timing information that was exchanged during the last encounter with this particular device (which could have acted as a Central at that time), or from an inquiry procedure. With this estimate CLKE of the Peripheral's Bluetooth clock, the Central can predict on which hop channel the Peripheral starts page scanning.

The estimate of the Bluetooth clock in the Peripheral can be completely wrong. Although the Central and the Peripheral use the same hopping sequence, they use different phases in the sequence and might never select the same frequency. To compensate for the clock drifts, the Central shall send its page message during a short time interval on a number of wake-up frequencies. It shall transmit also on hop frequencies just before and after the current, predicted hop frequency. During each TX slot, the Central shall sequentially transmit on two different hop frequencies. In the following RX slot, the receiver shall listen sequentially to two corresponding RX hops for an ID packet. The RX hops shall be selected according to the page response hopping sequence. The page response hopping sequence is strictly related to the page hopping sequence: for each page hop there is a corresponding page response hop. The RX/TX timing in the Page substate is described in Section 2.2.5, see also Figure 2.7. In the next TX slot, the Central shall transmit on two hop frequencies different from the former ones.

Note: The hop rate is increased to 3200 hops per second.

With the increased hopping rate as described above, the transmitter can cover 16 different hop frequencies in 16 slots or 10 ms. The page hopping sequence is divided over two paging trains **A** and **B** of 16 frequencies. Train **A** includes the 16 hop frequencies surrounding the current, predicted hop frequency *f(k)*, where k is determined by the clock estimate $CLKE_{16-12}$. The first train consists of hops

f(k-8), f(k-7),...,f(k),...,f(k+7)

When the difference between the Bluetooth clocks of the Central and the Peripheral is between -8x1.28 s and +7x1.28 s, one of the frequencies used by the Central will be the hop frequency the Peripheral will listen to. Since the Central does not know when the Peripheral will enter the Page Scan substate, the Central has to repeat this train **A** $N_{page}$ times or until a response is obtained, whichever is shorter. If the Peripheral scan interval corresponds to R1, the repetition number is at least 128; if the Peripheral scan interval corresponds to R2 or if the Central has not previously read the Peripheral's SR (page scan repetition) mode, the repetition number is at least 256. If the Central has not previously read the Peripheral's SR mode it shall use $N_{page} \geq 256$.

Note: $CLKE_{16-12}$ changes every 1.28 s; therefore, every 1.28 s, the trains will include different frequencies of the page hopping set.

When the difference between the Bluetooth clocks of the Central and the Peripheral is less than -8x1.28 s or larger than +7x1.28 s, the remaining 16 hops are used to form the new 10 ms train **B.** The second train consists of hops

f(k-16), f(k-15),...,f(k-9),f(k+8)...,f(k+15)

Train **B** shall be repeated for $N_{page}$ times. If no response is obtained, $k_{nudge}$ may be updated in the case where slots to receive are periodically not available and train **A** shall be tried again $N_{page}$ times. Alternate use of train **A** and train **B** and updates of $k_{nudge}$ shall be continued until a response is received or the timeout *pageTO* + *extended_pageTO* is exceeded. If a response is returned by the Peripheral, the Central enters the Central Page Response substate.

The Page substate may be entered from the Standby state or the Connection state. In the Standby state, no connection has been established and the device can use all the capacity to carry out the page. Before entering the Page substate from the Connection state, the device should free as much capacity as possible for paging. To ensure this, it is recommended that the ACL connections are put on hold. However, the synchronous connections shall not be disturbed by the page. This means that the page will be interrupted by the reserved SCO and eSCO slots which have higher priority than the page. In order to obtain as much capacity for paging, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO or eSCO

links are present, the repetition number $N_{page}$ of a single train shall be increased, see Table 8.2. Here it has been assumed that the **HV3** packet are used with an interval $T_{SCO}$=6 slots or **EV3** packets are used with an interval of $T_{eSCO}$=6 slots, which would correspond to a 64 kb/s synchronous link.

| SR mode | no synchronous link | one synchronous link (HV3) | two synchronous links (HV3) |
|---|---|---|---|
| R0 | $N_{page}$≥1 | $N_{page}$≥2 | $N_{page}$≥3 |
| R1 | $N_{page}$≥128 | $N_{page}$≥256 | $N_{page}$≥384 |
| R2 | $N_{page}$≥256 | $N_{page}$≥512 | $N_{page}$≥768 |

*Table 8.2: Relationship between train repetition and paging modes R0, R1 and R2 when synchronous links are present*

The construction of the page train shall be independent of the presence of synchronous links; that is, synchronous packets are sent on the reserved slots but shall not affect the hop frequencies used in the unreserved slots, see Figure 8.2.

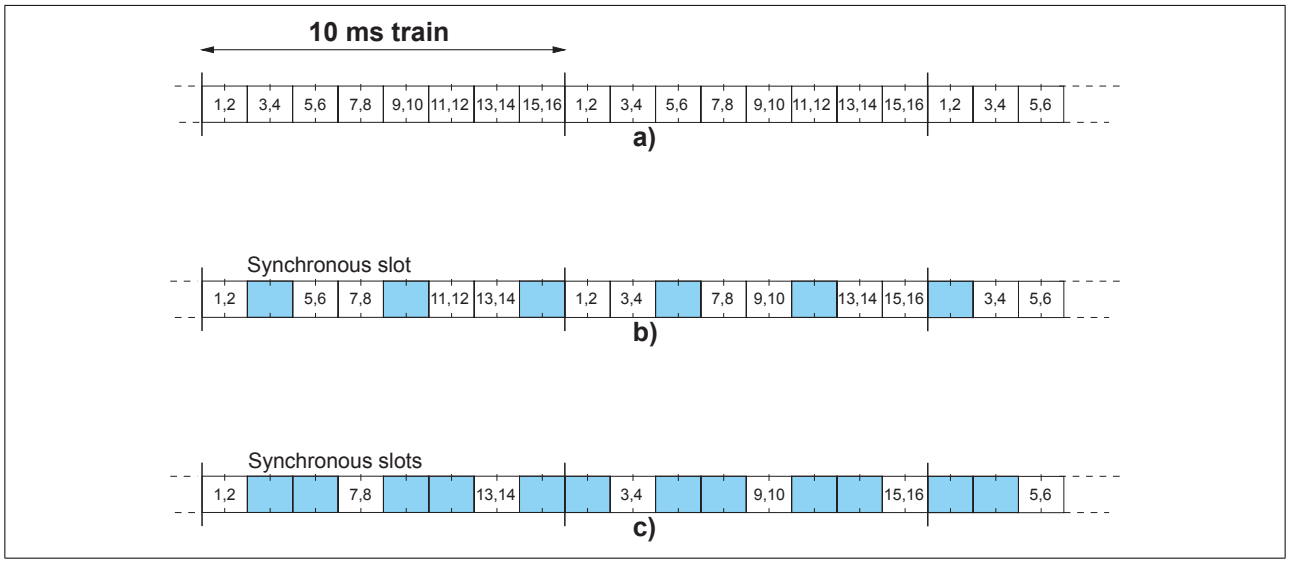



*Figure 8.2: Conventional page (a), page while one synchronous link present (b), page while two synchronous links present (c)*

### 8.3.3 Page response substates

When a page message is successfully received by the Peripheral, there is a coarse frequency hopping synchronization between the Central and the Peripheral. Both the Central and the Peripheral enter a response substate to exchange vital information to continue the connection setup. It is important for the piconet connection that both devices shall use the same channel access code, use the same channel hopping sequence, and their clocks are synchronized. These parameters shall be derived from the Central. The device that initializes the connection (starts paging) is defined as

the Central (which is thus only valid during the time the piconet exists). The channel access code and channel hopping sequence shall be derived from the Bluetooth Device Address (BD_ADDR) of the Central. The timing shall be determined by the Central's clock. An offset shall be added to the Peripheral's native clock to temporarily synchronize the Peripheral's clock to the Central's clock. At start-up, the Central's parameters are transmitted from the Central to the Peripheral. The messaging between the Central and the Peripheral at start-up is specified in this section.

If Central and Peripheral are already connected (irrespective of roles) then the Peripheral shall either ignore the page or shall immediately close the existing connection without sending any further packets relating to that connection and then continue with the page response procedure.

If the BD_ADDR of the Central is identical to the BD_ADDR of the Peripheral, then the Peripheral's Controller shall ignore the page and end the page response procedure.

The initial messaging between Central and Peripheral is shown in Table 8.3 and in Figure 8.3 and Figure 8.4. In those two figures frequencies f (k), f(k+1), etc. are the frequencies of the page hopping sequence determined by the Peripheral's BD_ADDR. The frequencies f'(k), f'(k+1), etc. are the corresponding page_response frequencies (Peripheral-to-Central). The frequencies g(m) belong to the basic channel hopping sequence.

| Step | Message | Packet Type | Direction | Hopping Sequence | Access Code and Clock |
|---|---|---|---|---|---|
| 1 | Page | ID | Central to Peripheral | Page | Peripheral |
| 2 | First Peripheral page response | ID | Peripheral to Central | Page response | Peripheral |
| 3 | Central page re-sponse | FHS | Central to Peripheral | Page | Peripheral |
| 4 | Second Peripheral page response | ID | Peripheral to Central | Page response | Peripheral |
| 5 | 1st packet Central | POLL | Central to Peripheral | Channel | Central |
| 6 | 1st packet Peripheral | NULL/DM 1/DH1 | Peripheral to Central | Channel | Central |

*Table 8.3: Initial messaging during start-up*

In step 1 (see Table 8.3), the Central is in Page substate and the Peripheral in the Page Scan substate. Assume in this step that the page message sent by the Central reaches the Peripheral. On receiving the page message, the Peripheral enters the Peripheral Page Response substate in step 2. The Central waits for a reply from the Peripheral and when this arrives in step 2, it shall enter the Central Page Response substate in step 3.

Note: During the initial message exchange, all parameters are derived from the Peripheral's device address, and that only the page hopping and page response hopping sequences are used (are also derived from the Peripheral's device address). When the Central and Peripheral enter the response states, their clock input to the page and page response hop selection is frozen as is described in Section 2.6.4.3.

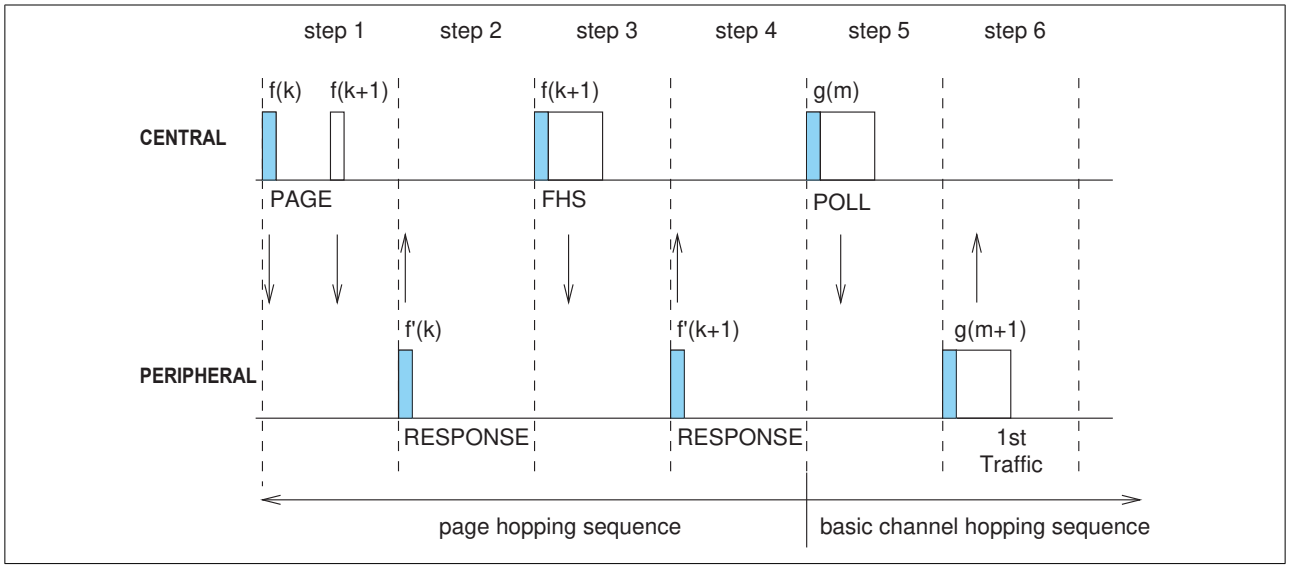



*Figure 8.3: Messaging at initial connection when Peripheral responds to first page message*

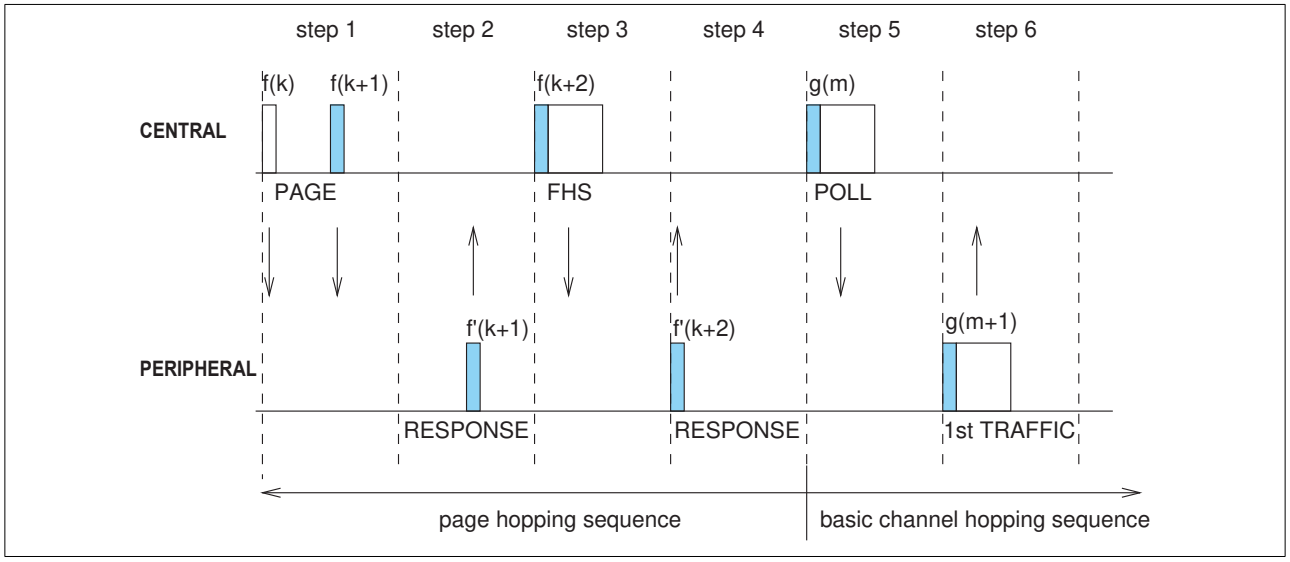



*Figure 8.4: Messaging at initial connection when Peripheral responds to second page message*

In the case of the truncated page procedure, the messaging stops after step 2 in the sequence shown in Table 8.3. This procedure is illustrated in Figure 8.5 and Figure 8.6.

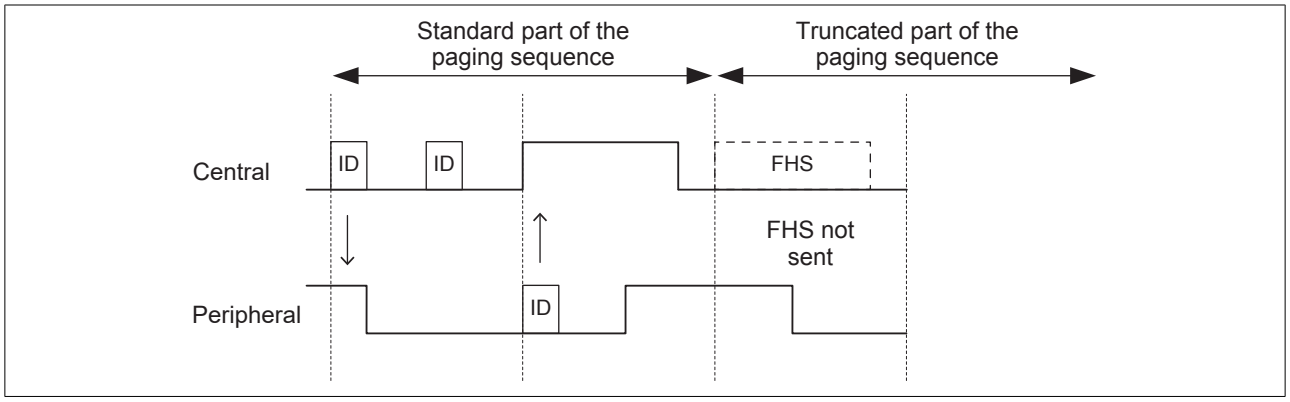



*Figure 8.5: First half slot truncated page*

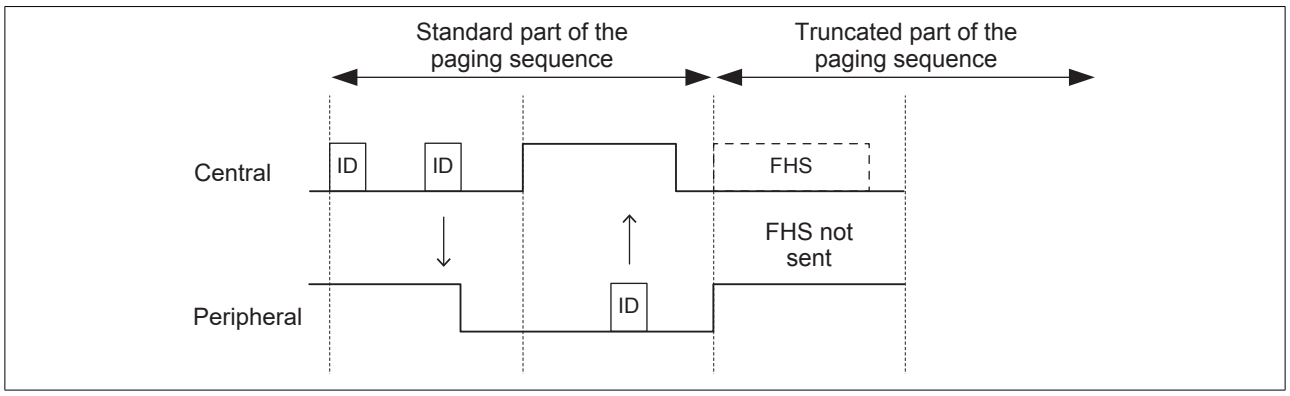



*Figure 8.6: Second half slot truncated page*

#### 8.3.3.1 Peripheral Page Response substate

After having received the page message in step 1, the Peripheral shall transmit a Peripheral page response message (the Peripheral's device access code) in step 2. This response message shall be the Peripheral's device access code. The Peripheral shall transmit this response 625 µs after the beginning of the received page message and at the response hop frequency that corresponds to the hop frequency in which the page message was received. The Peripheral's transmission is therefore time aligned to the Central's transmission. During initial messaging, the Peripheral shall still use the page response hopping sequence to return information to the Central. The clock input $CLKN_{16-12}$ shall be frozen at the value it had at the time the page message was received.

After having sent the response message, the Peripheral's receiver shall be activated 312.5 µs after the start of the response message and shall await the arrival of an **FHS** packet.

Note: An **FHS** packet can arrive 312.5 µs after the arrival of the page message as shown in Figure 8.4, and not after 625 µs as is usually the case in the piconet physical channel RX/TX timing (more details about the timing can be found in Section 2.4.4).

If the setup fails before the Connection state has been reached, the following procedure shall be carried out. The Peripheral shall listen as long as no **FHS** packet is received until *pagerespTO* is exceeded. Every 1.25 ms, however, it shall select the next Central-to-Peripheral hop frequency according to the page hop sequence. If nothing is received after *pagerespTO*, the Peripheral shall return back to the Page Scan substate for one scan period. Length of the scan period depends on the synchronous reserved slots present. If no page message is received during this additional scan period, the Peripheral shall resume scanning at its regular scan interval and return to the state it was in prior to the first page scan state.

If an **FHS** packet is received by the Peripheral in the Peripheral Page Response substate, the Peripheral shall return a Peripheral page response message in step 4 to acknowledge reception of the **FHS** packet. This response shall use the page response hopping sequence. The transmission of the Peripheral page response packet is based on the reception of the **FHS** packet. Then the Peripheral shall change to the Central's channel access code and clock as received from the **FHS** packet. Only the 26 MSBs of the Central's clock are transferred: the timing shall be such that $CLK_1$ and $CLK_0$ are both zero at the time the **FHS** packet was received as the Central transmits in even slots only. The offset between the Central's clock and the Peripheral's clock shall be determined from the Central's clock in the **FHS** packet and reported to the Peripheral's Baseband Resource Manager.

Finally, the Peripheral enters the Connection state in step 5. From then on, the Peripheral shall use the Central's clock and the Central's BD_ADDR to determine the basic channel hopping sequence and the channel access code. The Peripheral shall use the LT_ADDR in the **FHS** payload as the primary LT_ADDR in the Connection state. The connection mode shall start with a POLL packet transmitted by the Central. The Peripheral may respond with a NULL, DM1, or DH1 packet. If the POLL packet is not received by the Peripheral, or the response packet is not received by the Central, within *newconnectionTO* number of slots after FHS packet acknowledgment, the Central and the Peripheral shall return to Page and Page Scan substates, respectively. See Section 8.5.

#### 8.3.3.2 Central Page Response substate

When the Central has received a Peripheral page response message in step 2, and the Central is not executing the truncated page procedure, it shall enter the Central Response substate. It shall freeze the current clock input to the page hop selection scheme. The Central shall then transmit an **FHS** packet in step 3 containing the Central's real-time Bluetooth clock, the Central's BD_ADDR, the BCH parity bits, and the Class of Device. The **FHS** packet contains all information to construct the channel access code without requiring a mathematical derivation from the Central's Bluetooth Device Address. The LT_ADDR field in the packet header of FHS packets in the Central Page Response substate shall be set to all-zeros. The **FHS** packet shall be

transmitted at the beginning of the Central-to-Peripheral slot following the slot in which the Peripheral responded. The FHS packet shall carry the all-zero LT_ADDR. The TX timing of the **FHS** is not based on the reception of the response packet from the Peripheral. The **FHS** packet may therefore be sent 312.5 µs after the reception of the response packet like shown in Figure 8.4 and not 625 µs after the received packet as is usual in the piconet physical channel RX/TX timing, see also Section 2.4.4.

After the Central has sent its **FHS** packet, it shall wait for a second Peripheral page response message in step 4 acknowledging the reception of the **FHS** packet. This response shall be the Peripheral's device access code. If no response is received, the Central shall retransmit the **FHS** packet with an updated clock and still using the Peripheral's parameters. It shall retransmit the FHS packet with the clock updated each time until a second Peripheral page response message is received, or the timeout of *pagerespTO* is exceeded. In the latter case, the Central shall return to the Page substate and send an error message to the Baseband Resource Manager. During the retransmissions of the **FHS** packet, the Central shall use the page hopping sequence.

If the Peripheral's response is received, the Central shall change to using the Central's parameters, so it shall use the channel access code and the Central's clock. The lower clock bits $CLK_0$ and $CLK_1$ shall be reset to zero at the start of the **FHS** packet transmission and are not included in the **FHS** packet. Finally, the Central enters the Connection state in step 5. The Central BD_ADDR shall be used to change to a new hopping sequence, the *basic channel hopping sequence*. The basic channel hopping sequence uses all 79 hop channels in a pseudo-random fashion, see also Section 2.6.4.7. The Central shall now send its first traffic packet in a hop determined with the new (Central) parameters. This first packet shall be a POLL packet. See Section 8.5. This packet shall be sent within *newconnectionTO* number of slots after reception of the FHS packet acknowledgment. The Peripheral may respond with a NULL, DM1, or DH1 packet. If the POLL packet is not received by the Peripheral or the POLL packet response is not received by the Central within *newconnectionTO* number of slots, the Central and the Peripheral shall return to Page and Page Scan substates, respectively.

## 8.4 Device discovery substates

In order to discover other devices a device shall enter Inquiry substate. In this substate, it shall repeatedly transmit the inquiry message (ID packet, see Section 6.5.1.1) at different hop frequencies. The inquiry hop sequence is derived from the LAP of the GIAC. Thus, even when DIACs are used, the applied hopping sequence is generated from the GIAC LAP. A device that allows itself to be discovered, shall regularly enter the Inquiry Scan substate to respond to inquiry messages. The following sections describe the message exchange and contention resolution during inquiry response. The inquiry response is optional: a device is not forced to respond to an inquiry message.

During the Inquiry substate, the inquiring device collects the Bluetooth Device Addresses and clocks of all devices that respond to the inquiry message. In addition, the inquiring device also collects extended information (e.g. local name and supported services) from devices that respond with an extended inquiry response packet. It can then, if desired, make a connection to any one of the discovered devices by means of the previously described page procedure.

The inquiry message broadcast by the source does not contain any information about the source. However, it may indicate which class of devices should respond. There is one general inquiry access code (GIAC) to inquire for any device, and a number of dedicated inquiry access codes (DIAC) that only inquire for a certain type of device. The inquiry access codes are derived from reserved Bluetooth Device Addresses and are further described in Section 6.3.1.

### 8.4.1 Inquiry Scan substate

The Inquiry Scan substate is very similar to the Page Scan substate. However, instead of scanning for the device's device access code, the receiver shall scan for the inquiry access code long enough to completely scan for 16 inquiry frequencies. Two types of scans are defined: standard and generalized interlaced. In the case of a standard scan the length of this scan period is denoted $T_{w_inquiry_scan}$ (11.25 ms default, see [Vol 4] Part E, Section 7.3.22). The standard scan is performed at a single hop frequency as defined by $Xir_{4-0}$ (see Section 2.6.4.6). The generalized interlaced scan is performed as two back to back scans of $T_{w_inquiry_scan}$ where the first scan is on the normal hop frequency and the second scan is defined by [$Xir_{4-0}$ + interlace_offset] *mod* 32. If the scan interval is not at least twice the scan window then generalized interlaced scan shall not be used. The inquiry procedure uses 32 dedicated inquiry hop frequencies according to the inquiry hopping sequence. These frequencies are determined by the general inquiry address. The phase is determined by the native clock of the device carrying out the inquiry scan; the phase changes every 1.28 s.

The interlace_offset value ranges from 0 to 31. The value 16 should be used unless the pattern of slots that are not available for scanning implies a different value should be used.

Instead of, or in addition to, the general inquiry access code, the inquiring device may scan for one or more dedicated inquiry access codes. However, the scanning shall follow the inquiry scan hopping sequence determined by the general inquiry address. If an inquiry message is received during an inquiry wake-up period, the device shall enter the Inquiry Response substate.

The Inquiry Scan substate can be entered from the Standby state or the Connection state. In the Standby state, no connection has been established and the device can use all the capacity to carry out the inquiry scan. Before entering the Inquiry Scan

substate from the Connection state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL logical transports in Sniff mode or Hold mode. Synchronous logical transports are preferably not interrupted by the inquiry scan, although eSCO retransmissions should be paused during the scan. In this case, the inquiry scan may be interrupted by the reserved synchronous slots. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window, $T_{w\ inquiry\ scan}$, shall be increased to increase the probability to respond to an inquiry message. If one SCO logical transport is present using HV3 packets and $T_{SCO}$=6 slots or one eSCO logical transport is present using EV3 packets and $T_{eSCO}$=6 slots, a total scan window of at least 36 slots (22.5 ms) is recommended; if two SCO links are present using HV3 packets and $T_{SCO}$=6 slots or two eSCO links are present using EV3 packets and $T_{eSCO}$=6 slots, a total scan window of at least 54 slots (33.75 ms) is recommended.

The scan interval $T_{inquiry\ scan}$ is defined as the interval between two consecutive inquiry scans. The inquiry scan interval shall be less than or equal to 2.56 s.

### 8.4.2 Inquiry substate

The Inquiry substate is used to discover new devices. This substate is very similar to the Page substate; the TX/RX timing shall be the same as in paging, see Section 2.4.4 and Figure 2.7. The TX and RX frequencies shall follow the inquiry hopping sequence and the inquiry response hopping sequence, and are determined by the general inquiry access code and the native clock of the discovering device. In between inquiry transmissions, the receiver shall scan for inquiry response messages. When a response is received, the entire packet (an FHS packet) shall be read. If the EIR bit in the FHS packet is set to one, the device should try to receive the extended inquiry response packet 1250 µs after the start of the FHS packet. After this, the device shall continue with inquiry transmissions. The device in an Inquiry substate shall not acknowledge the inquiry response messages. If enabled by the Host (see [Vol 4] Part E, Section 7.3.50), the RSSI value of the inquiry response message shall be measured. It shall keep probing at different hop channels and in between listening for response packets. As in the Page substate, two 10 ms trains **A** and **B** are defined, splitting the 32 frequencies of the inquiry hopping sequence into two 16-hop parts. A single train shall be repeated for at least $N_{inquiry}$=256 times before a new train is used. In order to collect all responses in an error-free environment, at least three train switches must have taken place. As a result, the Inquiry substate may have to last for 10.24 s unless the inquirer collects enough responses and aborts the Inquiry substate earlier. If desired, the inquirer may also prolong the Inquiry substate to increase the probability of receiving all responses in an error-prone environment. When some receive slots are periodically not available, it may require up to 31 train switches to collect all responses in an error-free environment, depending on the pattern of the available receive slots. Before each switch to train **A**, $k_{nudge}$ may be updated. As a result, the Inquiry substate may have to last for 40.96 s. If an inquiry procedure is automatically initiated periodically (say a 10 s period every

minute), then the interval between two inquiry instances shall be determined randomly. This is done to avoid two devices synchronizing their inquiry procedures.

The Inquiry substate is continued until stopped by the Baseband Resource Manager (when it decides that it has sufficient number of responses), when a timeout has been reached (Inquiry_Length + Extended_Inquiry_Length), or by a command from the Host to cancel the inquiry procedure.

The Inquiry substate can be entered from the Standby state or the Connection state. In the Standby state, no connection has been established and the device can use all the capacity to carry out the inquiry. Before entering the Inquiry substate from the Connection state, the device should free as much capacity as possible for inquiry. To ensure this, it is recommended that the ACL logical transports are placed in Sniff mode or Hold mode. However, the reserved slots of synchronous logical transports shall not be disturbed by the inquiry. This means that the inquiry will be interrupted by the reserved SCO and eSCO slots which have higher priority than the inquiry. In order to obtain as much capacity as possible for inquiry, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO or eSCO links are present, the repetition number $N_{inquiry}$ shall be increased, see Table 8.4.

Here it has been assumed that **HV3** packets are used with an interval $T_{SCO}$=6 slots or **EV3** packets are used with an interval of $T_{eSCO}$=6 slots, which would correspond to a 64 kb/s synchronous link.

| | No synchronous links | One synchronous link (HV3) | Two synchronous links (HV3) |
|---|---|---|---|
| $N_{inquiry}$ | ≥ 256 | ≥ 512 | ≥ 768 |

*Table 8.4: Increase of train repetition when synchronous links are present*

If an extended inquiry response packet could not be received because of higher priority traffic, the reception failed due to HEC or CRC failure, or because the packet type is not supported by the device, the inquiry response shall be reported to higher layers as an extended inquiry response with all-zero data.

### 8.4.3 Inquiry Response substate

The response substate for inquiries differs completely from the Peripheral Page Response substate applied for pages. When the inquiry message is received in the Inquiry Scan substate, the recipient shall return an inquiry response (FHS) packet containing the recipient's device address (BD_ADDR) and other parameters. If the recipient has non-zero extended inquiry response data to send it shall return an extended inquiry response packet after the FHS packet.

The following protocol in the discoverable device's inquiry response shall be used. On the first inquiry message received in the Inquiry Scan substate the discoverable device

shall enter the Inquiry Response substate. If the discoverable device has non-zero extended inquiry response data to send, then it shall return an FHS packet with the EIR bit set to one to the inquiring device 625 µs after the inquiry message was received. It shall then return an extended inquiry response packet 1250 µs after the start of the FHS packet. If the discoverable device's extended inquiry response data is all zeroes, then the discoverable device shall only return an FHS packet with the EIR bit set to zero. A contention problem could arise when several devices are in close proximity to the inquiring device and all respond to an inquiry message at the same time. However, because every device has a free running clock it is highly unlikely that they all use the same phase of the inquiry hopping sequence. In order to avoid repeated collisions between devices that wake up in the same inquiry hop channel simultaneously, a device shall back-off for a random period of time. Thus, if the device receives an inquiry message and returns an FHS packet, then it shall generate a random number, RAND, between 0 and MAX_RAND. For scanning intervals ≥ 1.28 s MAX_RAND shall be 1023, however, for scanning intervals < 1.28 s MAX_RAND may be as small as 127. A profile that uses a special DIAC may choose to use a smaller MAX_RAND than 1023 even when the scanning interval is ≥ 1.28 s. The discoverable device shall return to the Connection or Standby state for the duration of at least RAND time slots. Before returning to the Connection and Standby state, the device may go through the Page Scan substate. After at least RAND slots, the device shall add an offset of 1 to the phase in the inquiry hop sequence (the phase has a 1.28 s resolution) and return to the Inquiry Scan substate again. If the discoverable device is triggered again, then it shall repeat the procedure using a new RAND. The offset to the clock accumulates each time an **FHS** packet is returned. During a probing window, a discoverable device may respond multiple times, but on different frequencies and at different times. Reserved synchronous slots should have priority over response packets; that is, if a response packet overlaps with a reserved synchronous slot, it shall not be sent but the next inquiry message is awaited. If a device has extended inquiry response data to send but the extended inquiry response packet overlaps with a reserved synchronous slot the FHS packet may be sent with the EIR bit set to zero.

The messaging during the inquiry routines is summarized in Table 8.5. In step 1, the inquiring device transmits an inquiry message using the inquiry access code and its own clock. The discoverable device responds with the **FHS** packet containing the discoverable device’s Bluetooth Device Address, native clock and other information. This **FHS** packet is returned at times that tend to be random. The **FHS** packet is not acknowledged in the inquiry routine, but it is retransmitted at other times and frequencies as long as the inquiring device is probing with inquiry messages. If the discoverable device has non-zero extended inquiry response data, then it sends an extended inquiry response packet to the inquiring device in step 3.

The extended inquiry response packet is an ACL packet with type DM1, DM3, DM5, DH1, DH3 or DH5. To minimize interference it is recommended to use the shortest

packet that fits the data. The packet shall be sent on the same frequency as the FHS packet, 1250 µs after the start of the FHS packet. In the packet header, LT_ADDR shall be set to zero. TYPE shall be one of DM1, DM3, DM5, DH1, DH3 or DH5. FLOW, ARQN and SEQN shall all be set to zero and ignored during receipt. The HEC LFSR shall be initialized with the same DCI (default check initialization) as for the FHS packet. In the payload header, LLID shall contain the value 10 (start of an L2CAP message or no fragmentation). FLOW shall be set to zero and ignored upon receipt. The length of the payload body (LENGTH) shall be smaller than or equal to 240 bytes. The CRC LFSR shall be initialized with the same DCI as for the FHS packet. The data whitening LFSR shall be initialized with the same value as for the FHS packet.

The payload data has two parts, a significant part followed by a non-significant part. The significant part contains a sequence of data structures as defined in [Vol 3] Part C, Section 8. The non-significant part contains all-zero octets.

The Baseband shall not change any octets in the significant part. When transmitting data, the non-significant part octets may be omitted from the payload.

A device shall store a single extended inquiry response packet. This packet shall be used with all IACs.

| Step | Message | Packet Type | Direction | Hopping Sequence | Access Code and Clock |
|---|---|---|---|---|---|
| 1 | Inquiry | ID | inquiring device to discoverable device | inquiry | inquiry |
| 2 | Inquiry response | FHS | discoverable device to inquiring device | inquiry response | inquiry |
| 3 | Extended Inquiry Response | DM1, DM3, DM5, DH1, DH3, DH5 | discoverable device to inquiring device | same frequency as step 2 | inquiry |

*Table 8.5: Messaging during inquiry routines*

## 8.5 Connection state

In the Connection state, the connection has been established and packets can be sent back and forth, with the exception of Connectionless Peripheral Broadcast mode, in which case packets are only sent from the Central. In both devices, the channel access code (derived from the Central's BD_ADDR), the Central's Bluetooth clock, and the AFH_channel_map are used. Connection state uses the basic or adapted channel hopping sequence.

There are two ways to enter the Connection state. A device can transition from the Page or Page Scan substates to the Connection state or directly from the Standby state to the Connectionless Peripheral Broadcast mode of the Connection state.

If a device enters from the Page or Page Scan substates, the Connection state starts with a POLL packet sent by the Central to verify the switch to the Central's timing and channel frequency hopping. The Peripheral may respond with a NULL, DM1, or DH1 packet. If the Peripheral does not receive the POLL packet or the Central does not receive the response packet for *newconnectionTO* number of slots, both devices shall return to Page or Page Scan substate.

The first information packets in the Connection state contain control messages that characterize the link and give more details regarding the devices. These messages are exchanged between the link managers of the devices. For example, they may define the SCO logical transport and the sniff parameters. Then the transfer of user information can start by alternately transmitting and receiving packets.

The Connection state is left through a Detach or Reset command. The Detach command is used if the link has been disconnected in the normal way; all configuration data in the Link Controller shall remain valid. The Reset command is a soft reset of the Link Controller. The functionality of the soft reset is described in [Vol 4] Part E, Section 7.3.2.

In the Connection state, if a device is not going to be nominally present on the channel at all times it may describe its unavailability by using Sniff mode or Hold mode (see Figure 8.7).

If a device enters the Connectionless Peripheral Broadcast mode of the Connection state, the Transmitter (Central) starts by sending packets on the CPB logical transport and the Receiver (Peripheral) starts by receiving packets sent on the CPB logical transport.

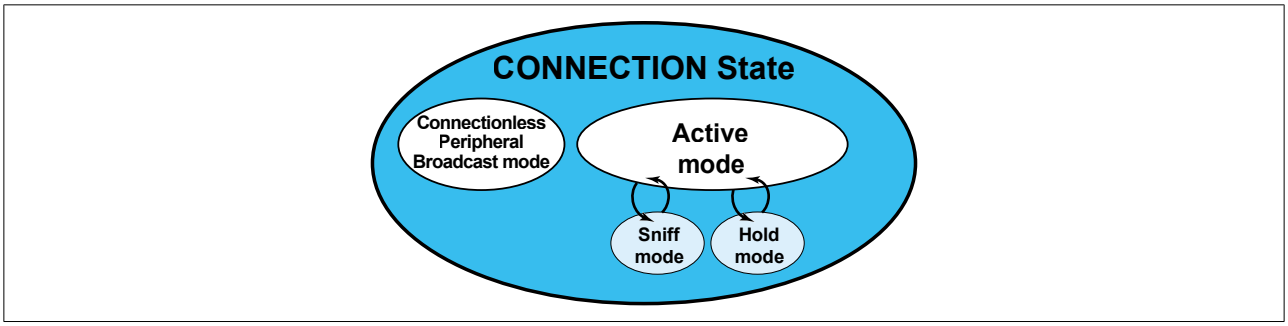



*Figure 8.7: Connection state*

## 8.6 Active mode

In the Active mode, both Central and Peripheral actively participate on the channel. Up to seven Peripherals may be in the Active mode at any given time in a piconet. The Central schedules the transmission based on traffic demands to and from the different Peripherals. In addition, it supports regular transmissions to keep Peripherals

synchronized to the channel. Peripherals in the Active mode listen in the Central-to-Peripheral slots for packets. These devices are known as *active Peripherals*. If an active Peripheral is not addressed, it may sleep until the next new Central transmission. Peripherals can derive the number of slots the Central has reserved for transmission from the TYPE field in the packet header; during this time, the non-addressed Peripherals do not have to listen on the Central-to-Peripheral slots. When a device is participating in multiple piconets it should listen in the Central-to-Peripheral slot for the current piconet. It is recommended that a device not be away from each piconet in which it is participating for more than $T_{poll}$ slots. A periodic Central transmission is required to keep the Peripherals synchronized to the channel. Since the Peripherals only need the channel access code to synchronize, any packet type can be used for this purpose.

Only the Peripheral that is addressed by one of its LT_ADDRs (primary or secondary) may return a packet in the next Peripheral-to-Central slot. If no valid packet header is received, the Peripheral may only respond in its reserved SCO or eSCO Peripheral-to-Central slot. In the case of a broadcast message, no Peripheral shall return a packet.

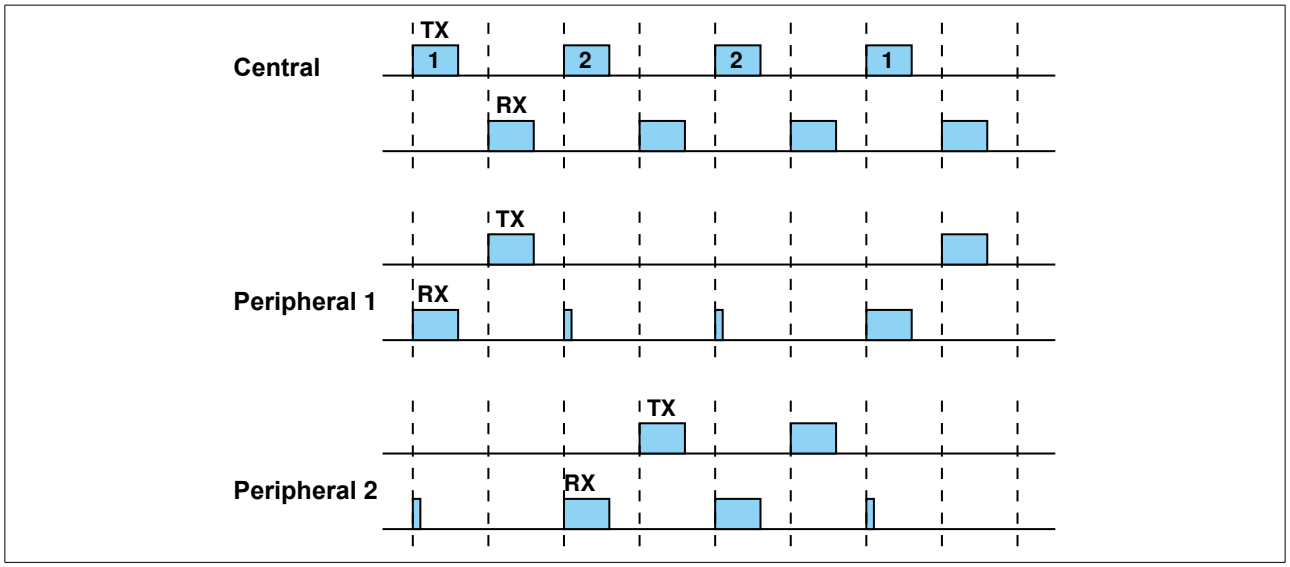



*Figure 8.8: RX/TX timing in multi-Peripheral configuration*

For ACL logical transports the mode selection may be left to real time packet type selections. The packet type table (ptt) in Section 6.5 allows the selection of Basic Rate or Enhanced Data Rate for each of the packet type codes, however; the DM1 packet is available in all packet type tables. ACL traffic over this given physical or logical link shall use the packet types in the given column of Table 6.2.

### 8.6.1 Polling in the Active mode

The Central always has full control over the piconet. Due to the TDD scheme, Peripherals can only communicate with the Central and not other Peripherals. In order

to avoid collisions on the ACL logical transport, a Peripheral is only allowed to transmit in the Peripheral-to-Central slot when addressed by the LT_ADDR in the packet header in the preceding Central-to-Peripheral slot. If the LT_ADDR in the preceding slot does not match, or a valid packet header was not received, the Peripheral shall not transmit.

The Central normally attempts to poll a Peripheral's ACL logical transport no less often than once every $T_{poll}$ slots. $T_{poll}$ is set by the Link Manager (see [Vol 2] Part C, Section 4.1.8).

The Peripheral's ACL logical transport may be polled with any packet type except for FHS and ID. For example, polling during SCO may use HV packets, since the Peripheral may respond to an HV packet with a DM1 packet (see Section 8.6.2).

### 8.6.2 SCO

The SCO logical transport shall be established by the Central sending a SCO setup message via the LM protocol. This message contains timing parameters including the SCO interval $T_{SCO}$ and the offset $D_{SCO}$ to specify the reserved slots.

In order to prevent clock wrap-around problems, an initialization flag in the SCO setup message indicates whether initialization procedure 1 or 2 is being used. The Peripheral shall apply the initialization method as indicated by the initialization flag. The Central shall use initialization 1 when the MSB of the Central's current clock ($CLK_{27}$) is 0; it shall use initialization 2 when the MSB of the Central's current clock ($CLK_{27}$) is 1. The Central-to-Peripheral SCO slots reserved by the Central and the Peripheral shall be initialized on the slots for which the clock satisfies the applicable equation:

$$\begin{aligned} &CLK_{27-1} \ mod \ \ T_{SCO} = D_{SCO} && \text{for initialization 1} \\ &\left(\overline{CLK_{27}}, CLK_{26-1}\right) \ mod \ \ T_{SCO} = D_{SCO} && \text{for initialization 2} \end{aligned}$$

The Peripheral-to-Central SCO slots shall directly follow the reserved Central-to-Peripheral SCO slots. After initialization, the clock value CLK(k+1) for the next Central-to-Peripheral SCO slot shall be derived by adding the fixed interval $T_{SCO}$ to the clock value of the current Central-to-Peripheral SCO slot:

$$CLK_{27-1}(k+1) = CLK_{27-1}(k) + T_{SCO}$$

The Central will send SCO packets to the Peripheral at regular intervals (the SCO interval $T_{SCO}$ counted in slots) in the reserved Central-to-Peripheral slots. An HV1 packet can carry 1.25 ms of speech at a 64 kb/s rate. An HV1 packet shall therefore be sent every two time slots ($T_{SCO}$=2). An HV2 packet can carry 2.5 ms of speech at a 64 kb/s rate. An HV2 packet shall therefore be sent every four time slots ($T_{SCO}$=4). An HV3 packet can carry 3.75 ms of speech at a 64 kb/s rate. An HV3 packet shall therefore be sent every six time slots ($T_{SCO}$=6).

The Peripheral is allowed to transmit in the slot reserved for its SCO logical transport unless a packet with the correct access code was received in the preceding slot and the LT_ADDR did not address this Peripheral (irrespective of whether or not the HEC was correct). If the correct LT_ADDR was received in the preceding slot but the HEC was incorrect, the Peripheral should not transmit in the reserved SCO slot.

Since the DM1 packet is recognized on the SCO logical transport, it may be sent during the SCO reserved slots if a valid packet header with the primary LT_ADDR is received in the preceding slot. DM1 packets sent during SCO reserved slots shall only be used to send ACL-C data.

The Peripheral shall not transmit anything other than an HV packet in a reserved SCO slot unless it decodes its own Peripheral address in the packet header of the packet in the preceding Central-to-Peripheral transmission slot.

### 8.6.3 eSCO

The eSCO logical transport is established by sending an eSCO setup message via the LM protocol. This message contains timing parameters including the eSCO interval $T_{eSCO}$ and the offset $D_{eSCO}$ to specify the reserved slots.

To enter eSCO, the Central or Peripheral shall send an eSCO setup message via the LM protocol. This message shall contain the eSCO interval $T_{eSCO}$ and an offset $D_{eSCO}$. In order to prevent clock wrap-around problems, an initialization flag in the eSCO setup message indicates whether initialization procedure 1 or 2 shall be used. The device sending the eSCO setup message shall use initialization 1 when the MSB of the Central's current clock ($CLK_{27}$) is 0; it shall use initialization 2 when the MSB of the Central's current clock ($CLK_{27}$) is 1. The device that receives the eSCO setup message shall apply the initialization method as indicated by the initialization flag. The Central-to-Peripheral eSCO slots reserved by the Central and the Peripheral shall be initialized on the slots for which the clock satisfies the applicable equation:

$$\begin{aligned} &CLK_{27-1} \; mod \;\; T_{eSCO} = D_{eSCO} && \text{for initialization 1} \\ &\left(\overline{CLK_{27}}, CLK_{26-1}\right) mod \;\; T_{eSCO} = D_{eSCO} && \text{for initialization 2} \end{aligned}$$

The Peripheral-to-Central eSCO slots shall directly follow the reserved Central-to-Peripheral eSCO slots. After initialization, the clock value CLK(k+1) for the next Central-to-Peripheral eSCO slot shall be found by adding the fixed interval $T_{eSCO}$ to the clock value of the current Central-to-Peripheral eSCO slot:

$$CLK_{27-1}(k+1) = CLK_{27-1}(k) + T_{eSCO}$$

When an eSCO logical transport is established, the Central shall assign an additional LT_ADDR to the Peripheral. This provides the eSCO logical transport with an ARQ scheme that is separate from that of the ACL logical transport. All traffic on a particular

eSCO logical transport, and only that eSCO traffic, is carried on the eSCO LT_ADDR. The eSCO ARQ scheme uses the ARQN bit in the packet header, and operates similarly to the ARQ scheme on ACL links.

The Central may send a packet in the reserved Central-to-Peripheral slot. The Peripheral may transmit on the eSCO LT_ADDR in the following slot either if it received a packet on the eSCO LT_ADDR in the previous slot, or if it did not receive a valid packet header in the previous slot. When the Central-to-Peripheral packet type is a three-slot packet, the Peripheral's transmit slot is the fourth slot of the eSCO reserved slots.

A Central shall transmit in an eSCO retransmission window on a given eSCO LT_ADDR only if it addressed that eSCO LT_ADDR or did not transmit any packet in the immediately preceding eSCO reserved slots. A Peripheral shall transmit in an eSCO retransmission window on a given eSCO LT_ADDR only if it received a valid packet header and that LT_ADDR on the eSCO channel in the previous Central-to-Peripheral transmission slot. The Central may transmit on any non-eSCO LT_ADDR in any Central-to-Peripheral transmission slot inside the eSCO retransmission window; if it addresses a Peripheral, that Peripheral may respond on the ACL channel as usual.

The Central shall only transmit on an eSCO LT_ADDR in the retransmission window if there are enough slots left for both the Central and Peripheral packets to complete in the retransmission window. If the Central is transmitting a NULL packet (with ARQN=ACK), then this requires one slot, otherwise it requires enough slots for the Central's negotiated eSCO packet type plus:

- if the Central's packet has ARQN=NAK, then enough slots for the Peripheral's negotiated eSCO packet type;
- if the Central's packet has ARQN=ACK, then one slot.

The Central may refrain from transmitting in any slot during the eSCO retransmission window. When there is no data to transmit from Central to Peripheral, either due to the traffic being unidirectional or due to the Central-to-Peripheral packet having been ACK'ed, the Central shall use the POLL packet. When the Central-to-Peripheral packet has been ACK'ed, and the Peripheral-to-Central packet has been correctly received, the Central shall not address the Peripheral on the eSCO LT_ADDR until the next eSCO reserved slot, with the exception that the Central may transmit a NULL packet with ARQN=ACK on the eSCO LT_ADDR. If the Central is transmitting on the eSCO LT_ADDR during a retransmission slot and there is no data to transmit from Peripheral to Central, either due to the traffic being unidirectional or due to the Peripheral-to-Central packet having been ACK'ed, the Peripheral shall use NULL packets. eSCO traffic should be given priority over ACL traffic in the retransmission window.

Figure 8.9 shows the eSCO window when single slot packets are used.

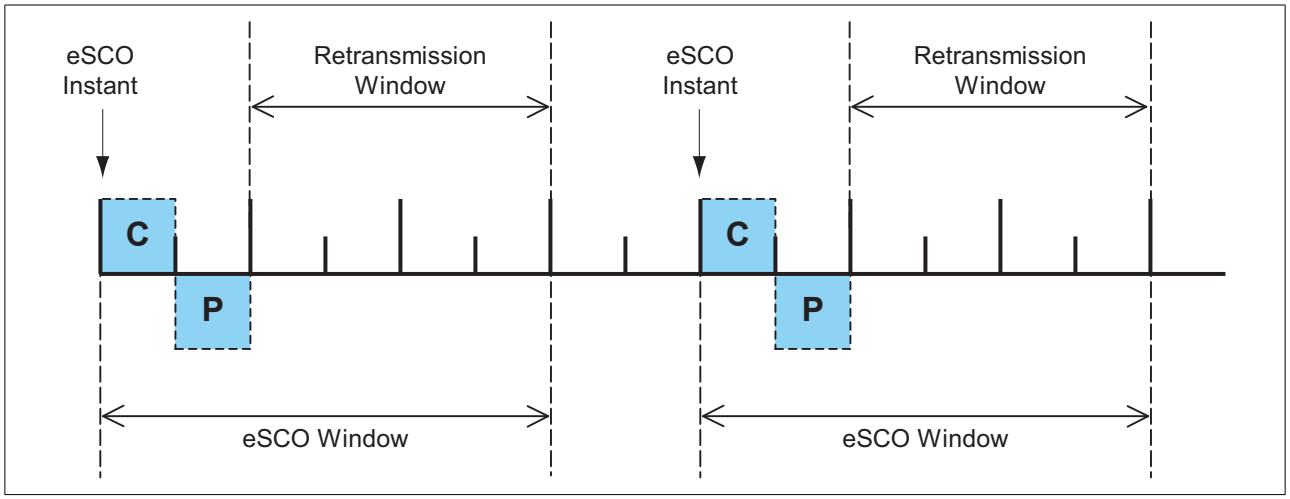



*Figure 8.9: eSCO window details for single-slot packets*

When multi-slot packets are used in either direction of the eSCO logical transport, the first transmission continues into the following slots. The retransmission window in this case starts the slot after the end of the Peripheral-to-Central packet, i.e. two, four or six slots immediately following the eSCO instant are reserved and should not be used for other traffic. Figure 8.10 shows the eSCO window when multi-slot packets are used in one direction and single-slot packets are used in the other direction.

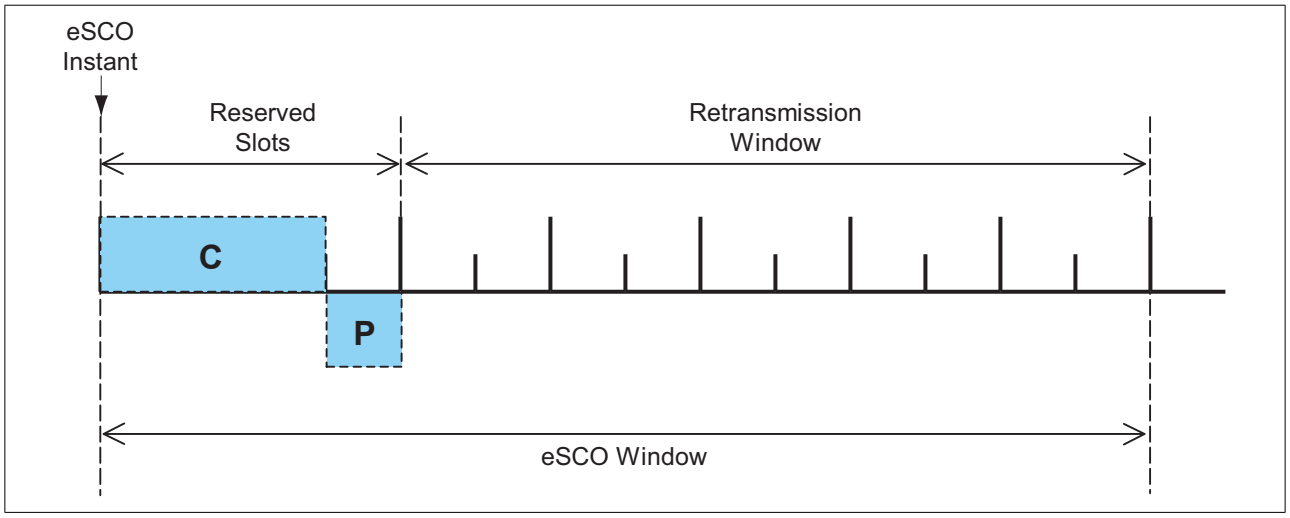



*Figure 8.10: eSCO window details for asymmetric traffic*

eSCO windows may overlap on the Central, but shall not overlap on an individual Peripheral.

In the reserved slots both Central and Peripheral shall only listen and transmit at their allocated slots at the first transmission time of each eSCO window. Intermittent lapses due to, for instance, time-critical signaling during connection establishment are allowed. Both Central and Peripheral may refrain from sending data and may use instead POLL

and NULL packets respectively. When the Central transmits a POLL packet instead of an eSCO 3-slot packet, or the Peripheral does not receive anything in the reserved Central-to-Peripheral transmission slot, it shall start any transmission in the same slot as if the Central had transmitted the negotiated packet type. For example, if the Central had negotiated an EV5 packet the Peripheral would transmit three slots later. If the Central does not receive a Peripheral transmission in response to an eSCO packet it causes an implicit NAK of the packet in question. The listening requirements for the Peripheral during the retransmission window are the same as for an active ACL logical transport.

### 8.6.4 Broadcast scheme

The Central of the piconet can broadcast messages to all Peripherals on the APB logical transport. An APB broadcast packet shall have an LT_ADDR set to all zero. If a new broadcast message carries APB-U data, it may be fragmented in the same way as ACL packets. Therefore it shall start with a packet carrying the start of L2CAP message indication (LLID=0b10) and may be followed by packets carrying the continuation of L2CAP message indication (LLID=0b01). If a new broadcast message carries APB-C data, it shall not be fragmented and shall consist of a single packet carrying the LMP message indication (LLID=0b11). In either case, the Central may carry out a number of retransmissions of each of these packets to increase the probability for error-free delivery; see also Section 7.6.5.

A broadcast packet shall never be acknowledged by the Baseband.

The Broadcast LT_ADDR shall use a ptt=0.

### 8.6.5 Role switch

There are several occasions when a role switch is used:

- A role switch is necessary in order to make a paging device a Peripheral when joining an existing piconet, since by definition the paging device is initially Central of a piconet involving the pager (Central) and the paged (Peripheral) device.
- A role switch is necessary in order for a Peripheral in an existing piconet to set up a new piconet with itself as Central and the original piconet Central as Peripheral. If the original piconet had more than one Peripheral, then this implies a double role for the original piconet Central; it becomes a Peripheral in the new piconet while still maintaining the original piconet as Central.

Prior to the role switch, encryption if present, shall be paused or disabled in the old piconet. A role switch shall not be performed if the physical link is in Sniff or Hold mode, or has any synchronous logical transports.

For the Central and Peripheral involved in the role switch, the switch results in a reversal of their TX and RX timing: a TDD switch. Additionally, since the piconet

parameters are derived from the Bluetooth Device Address and clock of the Central, a role switch inherently involves a redefinition of the piconet as well: a piconet switch. The new piconet's parameters shall be derived from the former Peripheral's device address and clock.

Assume device A is to become Central; device B was the former Central. Then there are two alternatives: either the Peripheral initiates the role switch or the Central initiates the role switch. These alternatives are described in Link Manager Protocol, [Vol 2] Part C, Section 4.4.2. The Baseband procedure is the same regardless of which alternative is used.

To begin the role switch, A and B shall perform a TDD switch using the former hopping scheme (still using the Bluetooth Device Address and clock of device B), so there is no piconet switch yet. The slot offset information sent by A shall not be used yet; it is used when both devices have switched to the new piconet and device B is positioning its correlation window. Device A now becomes the Central, device B the Peripheral.

At the moment of the TDD switch, both devices A and B shall start a timer with a time out of *newconnectionTO*. The timer shall be stopped in B as soon as it receives an FHS packet from A on the TDD-switched channel. The timer shall be stopped in A as soon as it receives an ID packet from B. If the *newconnectionTO* expires, A and B shall return to the old piconet timing and AFH state, taking their old roles of Peripheral and Central respectively. The FHS packet shall be sent by A using the "old" piconet parameters. The LT_ADDR in the FHS packet header shall be the former LT_ADDR used by device A. The LT_ADDR carried in the FHS payload shall be the new LT_ADDR intended for device B when operating on the new piconet. After the FHS acknowledgment, which is the ID packet and shall be sent by B on the old hopping sequence, both A and B shall use the new channel parameters of the new piconet as indicated by the FHS with the sequence selection set to basic channel hopping sequence. If the new Central A has physical links that are *AFH-enabled*, following the piconet switch the new Central is responsible for controlling the AFH operational mode of its new Peripheral B.

Since the old and new Centrals' clocks are synchronized, the clock information sent in the FHS payload shall indicate the new Central's clock at the beginning of the FHS packet transmission. Furthermore, the 1.25 ms resolution of the clock information given in the FHS packet is not sufficient for aligning the slot boundaries of the two piconets. The slot-offset information in the LMP message previously sent by device A shall be used to provide more accurate timing information. The slot offset indicates the delay between the start of the Central-to-Peripheral slots of the old and new piconet channels. This timing information ranges from 0 to 1249 µs with a resolution of 1 µs. It shall be used together with the clock information in the FHS packet to accurately position the correlation window when switching to the new Central's timing after acknowledgment of the FHS packet.

After reception of the FHS packet acknowledgment, the new Central A shall switch to its own timing with the sequence selection set to the basic channel hopping sequence and shall send a POLL packet to verify the switch. Both A and B shall start a new timer with a time out of *newconnectionTO* on FHS packet acknowledgment. The start of this timer shall be aligned with the beginning of the first Central TX slot boundary of the new piconet, following the FHS packet acknowledgment. B shall stop the timer when the POLL packet is received; A shall stop the timer when the POLL packet is acknowledged. The Peripheral B shall respond with a NULL, DM1, or DH1 packet to acknowledge the POLL. Any pending AFH_Instant shall be cancelled once the POLL packet has been received by the Peripheral. If no response is received, A shall re-send the POLL packet until *newconnectionTO* is reached. If this timer expires, both A and B shall return to the old piconet timing with the old roles: B as Central and A as Peripheral. Expiry of the timer shall also restore the state associated with AFH (including any pending AFH_Instant), Channel Quality Driven Data Rate (CQDDR, Link Manager Protocol [Vol 2] Part C, Section 4.1.7) and power control (Link Manager Protocol [Vol 2] Part C, Section 4.1.3). The role switch procedure may then restart from the TDD switch. Aligning the timer with TX boundaries of the new piconet ensures that no device returns to the old piconet timing in the middle of a Central RX slot.

The messaging during a successful role switch is summarized in Table 8.6.

| Step | Message or Packet Type | Direction | Central |
|---|---|---|---|
| 1 | LMP_ACCEPTED | Responder → Initiator | B |
| 2 | FHS | A → B | B |
| 3 | ID | B → A | B |
| 4 | POLL | A → B | A |
| 5 | NULL/DM1/DH1 | B → A | A |

*Table 8.6: Messaging during successful role switch*

After the role switch the ACL logical transport is reinitialized as if it were a new connection. For example, the SEQN of the first data packet containing a CRC on the new piconet channel shall be set according to the rules in Section 7.6.2.

### 8.6.6 Scatternet

Multiple piconets can cover the same area. Since each piconet has a different Central, the piconets hop independently, each with their own hopping sequence and phase as determined by the respective Central. In addition, the packets carried on the channels are preceded by different channel access codes as determined by the Centrals' device addresses. As more piconets are added, the probability of collisions increases; a graceful degradation of performance results as is common in frequency-hopping spread spectrum systems.

If multiple piconets cover the same area, a device can participate in two or more overlaying piconets by applying time multiplexing. To participate on the proper channel, it shall use the associated Central's device address and proper clock offset to obtain the correct phase. A device can act as a Peripheral in several piconets, but only as a Central in a single piconet: since two piconets with the same Central are synchronized and use the same hopping sequence, they are one and the same piconet. A group of piconets in which connections exist between different piconets is called a *scatternet*.

A Central or Peripheral can become a Peripheral in another piconet by being paged by the Central of this other piconet. On the other hand, a device participating in one piconet can page the Central or Peripheral of another piconet. Since the paging device always starts out as Central, a role switch is required if a Peripheral role is desired. This is described in Section 8.6.5.

#### 8.6.6.1 Inter-piconet communications

Time multiplexing must be used to switch between piconets. Devices may achieve the time multiplexing necessary to implement scatternet by using Sniff mode or by remaining in an active ACL connection. For an ACL connection in piconets where the device is a Peripheral in the Connection state, it may choose not to listen in every Central slot. In this case it should be recognized that the quality of service on this link can degrade abruptly if the Peripheral is not present enough to match up with the Central polling that Peripheral. Similarly, in piconets where the device is Central it may choose not to transmit in every Central slot. In this case it is important to honor $T_{poll}$ as much as possible. Devices in Sniff mode could have sufficient time to visit another piconet in between sniff slots. When the device is a Peripheral using Sniff mode and there are not sufficient idle slots, the device may choose to not listen to all Central transmission slots in the sniff_attempts period or during the subsequent sniff_timeout period. A Central is not required to transmit during sniff slots and therefore has flexibility for scatternet. If SCO or eSCO links are established, other piconets shall only be visited in the non-reserved slots in between reserved slots. This is only possible if there is a single SCO logical transport using HV3 packets or eSCO links where at least four slots remain in between the reserved slots. Since the multiple piconets are not synchronized, guard time must be left to account for misalignment. This means that only 2 slots can effectively be used to visit another piconet in between the HV3 packets.

Since the clocks of two Centrals of different piconets are not synchronized, a Peripheral participating in two piconets shall maintain two offsets that, added to its own native clock, create the two Centrals' clocks. Since the two Centrals' clocks drift independently, the Peripheral must regularly update the offsets in order to keep synchronization to both Centrals.

### 8.6.7 Hop sequence switching

Hop sequence adaptation is controlled by the Central and can be set to either *enabled* or *disabled*. Once enabled, hop sequence adaptation shall apply to all logical transports on a physical link. Once enabled, the Central may periodically update the set of *used* and *unused* channels as well as disable hop sequence adaptation on a physical link. When a Central has multiple physical links the state of each link is independent of all other physical links.

When hop sequence adaptation is enabled, the *sequence selection* hop selection kernel input is set to adapted channel hopping sequence and the *AFH_channel_map* input is set to the appropriate set of *used* and *unused* channels. Additionally, the *same channel* mechanism shall be used. When hop sequence adaptation is enabled with all channels *used* this is known as AHS(79).

When hop sequence adaptation is disabled, the *sequence selection* input of the hop selection kernel is set to *basic channel hopping sequence* (the *AFH_channel_map* input is unused in this case) and the *same channel* mechanism shall not be used.

The hop sequence adaptation state shall be changed when the Central sends the LMP_SET_AFH PDU and a Baseband acknowledgment is received. When the Baseband acknowledgment is received prior to the hop sequence switch instant, *AFH_Instant*, (see [Vol 2] Part C, Section 4.1.4) the hop sequence proceeds as shown in Figure 8.11.

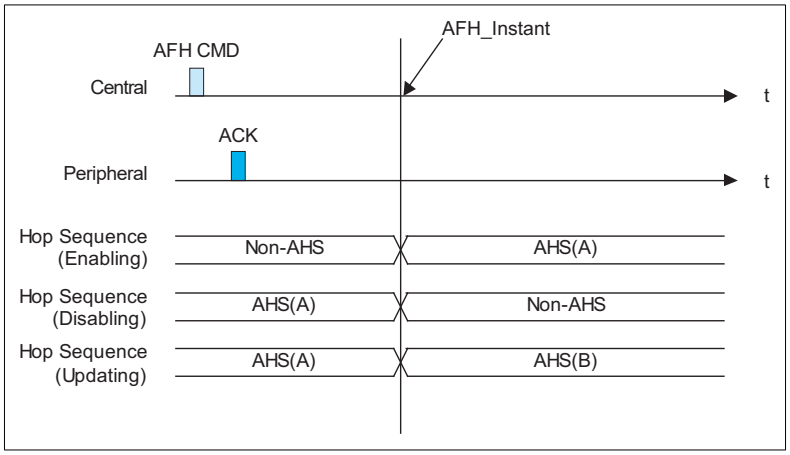



*Figure 8.11: Successful hop sequence switching*

When the Baseband acknowledgment is not received prior to the *AFH_Instant* the Central shall use a recovery hop sequence for the Peripheral(s) that did not respond with an acknowledgment (this may be because the Peripheral did not hear the Central's transmission or the Central did not hear the Peripheral's transmission). When hop sequence adaptation is being enabled or disabled the recovery sequence shall be the *AFH_channel_map* specified in the LMP_SET_AFH PDU. When the *AFH_channel_map*

is being updated the Central shall choose a recovery sequence that includes all of the RF channels marked as *used* in either the old or new *AFH_channel_map,* e.g. AHS(79). Once the Baseband acknowledgment is received the Central shall use the *AFH_channel_map* in the LMP_SET_AFH PDU starting with the next transmission to the Peripheral. See Figure 8.12.

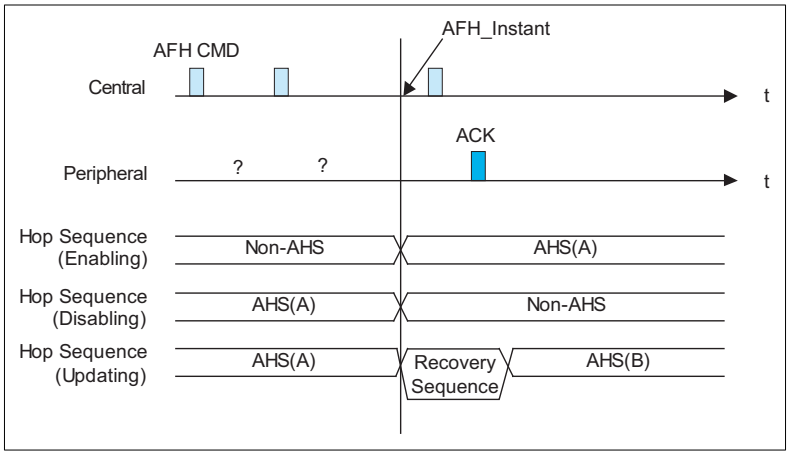



*Figure 8.12: Recovery hop sequences*

When the *AFH_Instant* occurs during a multi-slot packet transmitted by the Central, the Peripheral shall use the same hopping sequence parameters as the Central used at the start of the multi-slot packet. An example of this is shown in Figure 8.13. In this figure the basic channel hopping sequence is designated f. The first adapted channel hopping sequence is designated with f', and the second adapted channel hopping sequence is designated f''.

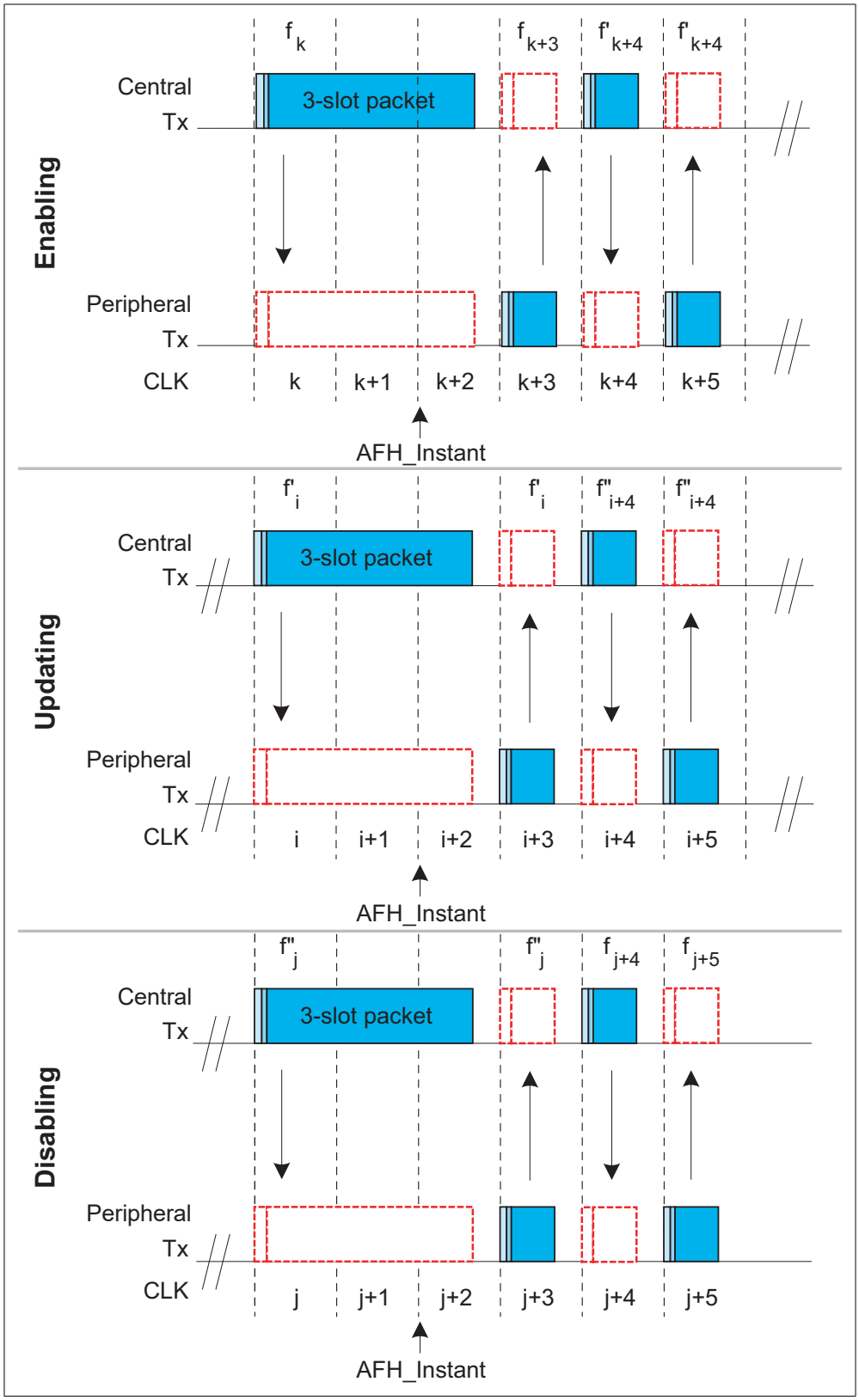



*Figure 8.13: AFH_Instant changes during multi-slot packets transmitted by the Central*

### 8.6.8 Channel classification and channel map selection

RF channels are classified as being *unknown, bad or good*. These classifications are determined individually by the Central and Peripherals based on local information (e.g. active or passive channel assessment methods or from the Host via HCI). Information received from other devices via LMP (for example, an *AFH_channel_map* from a Central or a channel classification report from a Peripheral) shall not be included in a device's channel classification.

The three possible channel classifications shall be as defined in Table 8.7.

| Classification | Definition |
|---|---|
| *unknown* | A channel shall be classified as *unknown* if the channel assessment measurements are insufficient to reliably classify the channel, and the channel is not marked as *bad* in the most recent HCI_ Set_AFH_Host_Channel_Classification command. |
| *bad* | A channel may be classified as *bad* if an ACL or synchronous throughput failure measure associated with it has exceeded a threshold (defined by the particular channel assessment algorithm employed).<br>A channel may also be classified as *bad* if an interference-level measure associated with it, determining the interference level that the link poses upon other systems in the vicinity, has exceeded a threshold (defined by the particular channel assessment algorithm employed).<br>A channel shall be classified as *bad* when it is marked as *bad* in the most recent HCI_ Set_AFH_Host_Channel_Classification command. |
| *good* | A channel shall be classified as *good* if it is not either *unknown* or *bad*. |

*Table 8.7: Channel classification descriptions*

A Central with AFH-enabled physical links shall determine an *AFH_channel_map* for each such link (two or more links may use the same map) based on any combination of the following information:

- Channel classification from local measurements (e.g. active or passive channel assessment in the Controller), if supported and enabled. The Host may enable or disable local measurements using the HCI_ Write_AFH_Channel_Assessment_Mode command, defined in the HCI Functional Specification [Vol 4] Part E, Section 7.3.54 if HCI is present.
- Channel classification information from the Host using the HCI_ Set_AFH_Host_Channel_Classification command, defined in the HCI Functional Specification [Vol 4] Part E, Section 7.3.46 if HCI is present. Channels classified as *bad* in the most recent *AFH_Host_Channel_Classification* shall be marked as *unused* in the *AFH_channel_map*.
- Channel classification reports received from Peripherals in *LMP_CHANNEL_CLASSIFICATION* PDUs, defined in the LMP Specification [Vol 2] Part C, Section 4.1.5.

The algorithm used by the Central to combine these information sources and generate the *AFH_channel_map* is not defined in the specification and will be implementation specific. At no time shall the number of channels used be less than $N_{min}$, defined in Section 2.3.1.

If a Central determines that all channels should be *used,* it may keep AFH operation enabled using an *AFH_channel_map* of 79 *used* channels, i.e., AHS(79).

For all devices that support the Synchronization Train substate (see Section 8.11.2), the RF channel indices used for the Synchronization Train (see Section 2.6.4.8) shall be marked as *unused* in the *AFH_channel_map* for all logical links.

### 8.6.9 Power management

Features are provided to allow low-power operation. These features are both at the microscopic level when handling the packets, and at the macroscopic level when using certain operation modes.

#### 8.6.9.1 Packet handling

In order to minimize power consumption, packet handling is minimized both at TX and RX sides. At the TX side, power is minimized by only sending useful data. This means that if only link control information needs to be exchanged, **NULL** packets may be used. No transmission is required if there is no link control information to be sent, or if the transmission would only involve a NAK (NAK is implicit on no reply). If there is data to be sent, the payload length shall be adapted in order to send only the valid data bytes. At the RX side, packet processing takes place in different steps. If no valid access code is found in the search window, the transceiver may return to sleep. If an access code is found, the receiver device shall start to process the packet header. If the HEC fails, the device may return to sleep after the packet header. A valid header indicates if a payload will follow and how many slots are involved.

#### 8.6.9.2 Slot occupancy

As was described in Section 6.5, the packet type indicates how many slots a packet may occupy. A Peripheral not addressed in the packet header may go to sleep for the remaining slots the packet occupies. This can be read from the TYPE code.

#### 8.6.9.3 Recommendations for low-power operation

The most common and flexible method for reducing power consumption is the use of Sniff mode. Hold mode can also be used by repeated negotiation of hold periods.

#### 8.6.9.4 Enhanced Data Rate

Enhanced Data Rate provides power saving because of the ability to send a given amount of data in either fewer packets or with the same (or similar) number of packets but with shorter payloads.

### 8.6.10 Piconet clock adjustment

A Central may adjust the piconet clock during the existence of a piconet using two mechanisms: Coarse Clock Adjustment and Clock Dragging. Both mechanisms may be used within the same piconet.

#### 8.6.10.1 Coarse clock adjustment

The Central carries out a coarse clock adjustment by selecting an adjustment instant (clk_adj_instant), which shall be a Central-to-Peripheral slot, and the amount of the adjustment, and then broadcasting these parameters to all Peripherals by sending LMP_CLK_ADJ (see [Vol 2] Part C, Section 4.1.14.1). The amount of the adjustment is specified as two values: clk_adj_slots, which is the change in the value of CLK[27:1] at the clk_adj_instant, and clk_adj_us, which is the number of microseconds between the start of a Central slot in the old clock ($CLK_{old}$) and in the new clock ($CLK_{new}$) domains.

The Central shall provide the opportunity for each Peripheral to acknowledge the adjustment with LMP_CLK_ADJ_ACK (see [Vol 2] Part C, Section 4.1.14.1). If the Central receives any other packet than LMP_CLK_ADJ_ACK with the current clk_adj_id, it should poll the Peripheral more times until the expected acknowledgment is received or the Peripheral responds with a NULL packet. If any Peripheral does not acknowledge the adjustment, the Central shall start the Coarse Clock Adjustment Recovery Mode (see Section 8.6.10.2).

When the clk_adj_instant occurs, the Central will add (clk_adj_slots × 625 + clk_adj_us) µs to time_base_offset and the Peripheral(s) will add the same amount to peripheral_clock_offset. If the clk_adj_instant occurs during a multi-slot packet the adjustment is delayed until the start of the following Central-to-Peripheral slot. If a role switch is successful prior to the clk_adj_instant, the pending coarse clock adjustment shall be discarded. The effect of the adjustment is that devices will use the new clock domain starting at clk_adj_instant.

Figure 8.14 shows an example of a positive clock adjustment. In this example the clk_adj_instant is set to slot 12 (which is in the old clock domain), clk_adj_slots is set to 6 and clk_adj_us is set to 400. Time is moved forward clk_adj_slots × 625 + clk_adj_us = 6 × 625 + 400 = 4150 µs. The initial slot in the new clock domain is at $CLK_{new}$[27:1] = clk_adj_instant + clk_adj_slots = 12 + 6 = 18. This is an even value so the first slot is a Central-to-Peripheral slot. A positive clk_adj_us means that the first slot is assumed to have started clk_adj_us before the clk_adj_instant, and hence

only 625 – clk_adj_us = 225 µs remains after the instant. Since transmissions cannot be started part way through a slot, this partial slot is unusable. The first complete slot after the clk_adj_instant is a Peripheral-to-Central slot which can be used, for example, for eSCO. If clk_adj_slots had been an odd number, the first complete slot would have been a Central-to-Peripheral slot. The first complete Central-to-Peripheral slot in this example happens at $CLK_{new}[27:1] = 20$.

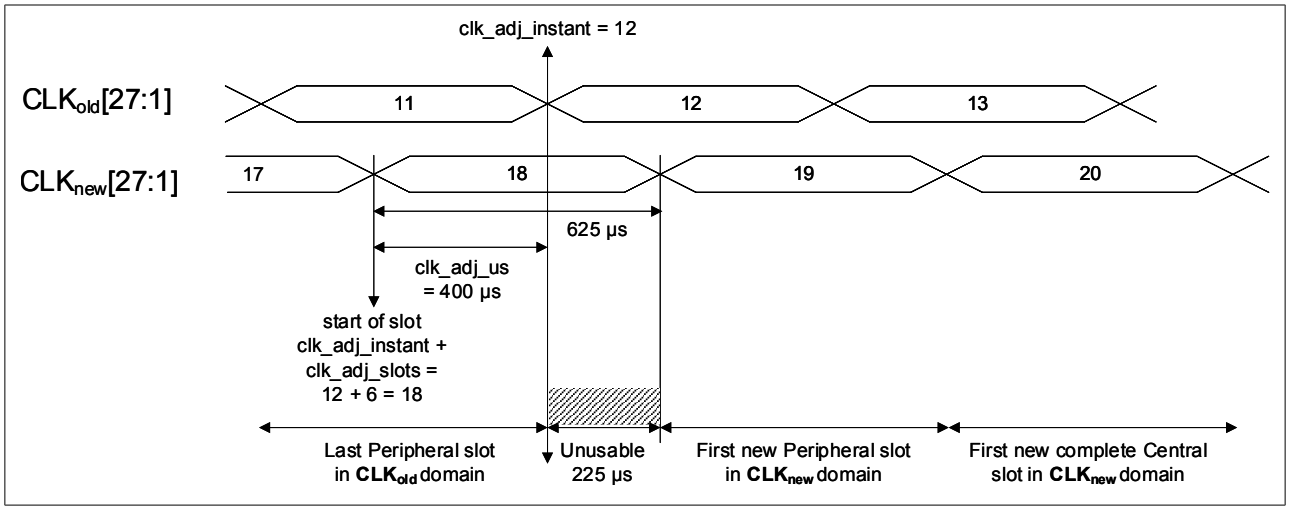



*Figure 8.14: Positive coarse clock adjustment*

Figure 8.15 shows an example of a negative clock adjustment. In this example the clk_adj_instant is again set to slot 12, clk_adj_slots is set to 0 and clk_adj_us is set to –400. Time is moved by clk_adj_slots × 625 + clk_adj_us = 0 × 625 + (–400) = –400 µs, which is 400 µs backwards. The initial slot in the $CLK_{new}$ domain is at $CLK_{new}[27:1]$ = clk_adj_instant + clk_adj_slots = 12 + 0 = 12. This is an even value so the first slot is a Central-to-Peripheral slot. A negative clk_adj_us means that the first slot in the $CLK_{new}$ domain starts clk_adj_us after clk_adj_instant. The time between the first slot in the $CLK_{new}$ domain and clk_adj_instant (in this case 400 µs) is effectively a continuation of the previous slot, even if its number changes, and hence is unusable. (This is the case whenever clk_adj_us is negative, even when clk_adj_slots is non-zero.)

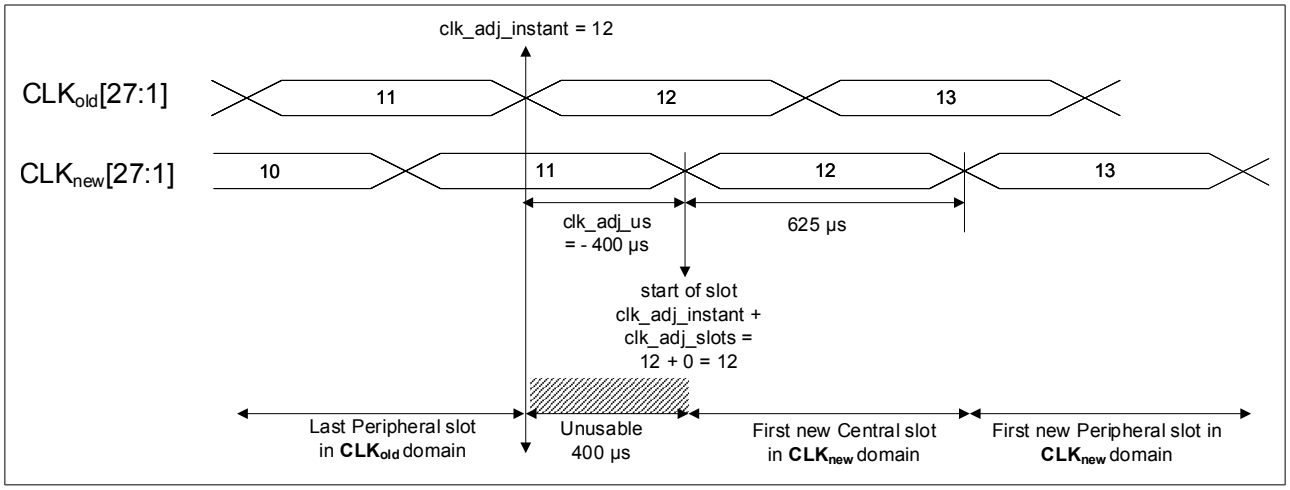



*Figure 8.15: Negative coarse clock adjustment*

If clk_adj_us is zero then the adjustment alters the slot numbering but not the actual times at which slots start. Therefore all slots remain available for use, though if clk_adj_slots is odd then there will be two consecutive Peripheral-to-Central slots.

### 8.6.10.2 Coarse Clock Adjustment Recovery mode

In Coarse Clock Adjustment Recovery Mode, the Central enters the Synchronization Train substate (see Section 8.11.2). The Central should remain in this substate and transmit synchronization train packets until all Peripherals have acknowledged the adjustment with the LMP_CLK_ADJ_ACK PDU, or link supervision timeout has expired for the last unresponsive Peripheral. The Central may cancel Coarse Clock Adjustment Recovery Mode to initiate another coarse clock adjustment or for any other reason. During Coarse Clock Adjustment Recovery Mode the Central shall continue broadcasting LMP_CLK_ADJ PDUs (see [Vol 2] Part C, Section 4.1.14.1).

When a Peripheral does not detect any transmissions from its Central it may enter the Synchronization Scan substate (see Section 8.11.1) and scan for synchronization train packets as defined in Section 2.7. $T_{Sync_Scan_Window}$ should be set large enough to enable reception of at least one synchronization train packet. A Peripheral that could have lost its connection to the Central due to coarse clock adjustment should prioritize synchronization scan.

A Peripheral that receives a Synchronization Train packet with the BD_ADDR of its Central shall change the value of peripheral_clock_offset to make CLK correspond to the contents of the packet and then exit the Synchronization Scan substate. If the new clock is in the range $CLK_{OLD}$–LSTO to $CLK_{OLD}$ (where LSTO is the link supervision timeout period), then this shall be treated as a negative change. In this case, the Peripheral shall not transmit any packet and shall ignore all received directed packets until the value of CLK, after being changed, is strictly greater than the value of CLK the last time it transmitted or received (as appropriate) a packet. If the new clock is outside

this range then this shall be treated as a positive change (and thus might involve a clock wrap).

#### 8.6.10.3 Clock Dragging

Clock Dragging means that the Central periodically adjusts its time_base_offset (see Section 2.2.4) until a desired time adjustment has been accomplished.

Each single adjustment shall be performed by a directed packet from the Central to each Peripheral and a response (e.g., Baseband acknowledgment) from each Peripheral. If a Peripheral does not respond, the Central shall suspend its updating of time_base_offset (but should continue to poll that Peripheral) at least until each Peripheral has either responded or has failed to respond for at least *CLK_adj_dragTO*. In case several Peripherals fail to respond, the Central should ensure that each of the failing Peripherals gets at least one *CLK_adj_dragTO* to respond (these may be concurrent for multiple Peripherals). The Central need not allow more than one suspension of *CLK_adj_dragTO* for any given Peripheral during a link supervision timeout for that Peripheral.

A single adjustment shall be less than or equal to 3 µs. Within any period of 125 ms, the total adjustment in a single direction shall be less than or equal to 5 µs.

Note: These requirements apply to the changes to time_base_offset and are applied irrespective of the accuracy of the Central's reference clock (see Section 1.1). This means that Peripherals can observe a change in CLK greater than 20ppm.

If a Central is performing Clock Dragging when it initiates a Coarse Clock Adjustment, a new Clock Dragging, or a sequence containing an instant or timing control flags, or receives a request from a Peripheral to initiate such a sequence, it shall abort the current Clock Dragging before processing the new request or carrying out the sequence. If the Central has suspended clock dragging during *CLK_adj_dragTO*, it shall reject new requests until the timeout period expires or the timeout is cancelled because all of the Peripherals have responded.

### 8.6.11 Slot Availability Mask (SAM)

Slot Availability Mask (SAM) allows two Bluetooth devices to indicate to each other the availability of their time slots for transmission and reception. From the Baseband point of view, SAM provides a map - the SAM slot map - which marks the availability of Bluetooth slots. The availability arises from either external conditions (e.g., MWS coexistence) or internal conditions (e.g., topology management for scatternets). The SAM slot map marks each slot using one of four type codes defined in [Vol 2] Part C, Section 5.2 and repeated for convenience in Table 8.8.

| Slot type code | Meaning |
|---|---|
| 0 | The slot is not available for either transmission or reception |
| 1 | The slot is available for transmission but not reception |
| 2 | The slot is available for reception but not transmission |
| 3 | The slot is available for both transmission and reception |

*Table 8.8: SAM slot types*

A Central may mark Central-to-Peripheral slots available for reception and Peripheral-to-Central slots available for transmission, because such slots can be used in this way for multi-slot packets; similarly for a Peripheral.

A SAM slot map with all slots set to type 3 is equivalent to not using SAM and simply performing normal scheduling.

Figure 8.16 shows an example of a SAM slot map. The Central-to-Peripheral slots are labeled with the letter 'C' and Peripheral-to-Central slots with the letter 'P'.

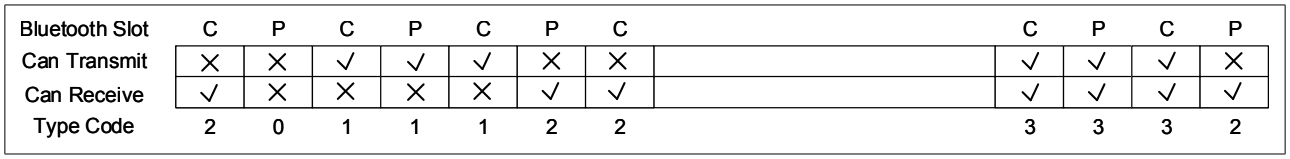



*Figure 8.16: An example of a SAM slot map*

A SAM slot map has a finite length and is repeated indefinitely until changed by the associated LMP procedures. SAM anchor points are spaced regularly with an interval of $T_{SAM}$, in units of slots. Between adjacent SAM anchor points, the slot map indicates when the device is able to transmit and receive. The SAM slot map is effectively repeated every $T_{SAM}$ until it is changed.

The SAM interval $T_{SAM}$ is further divided into $N_{SAM_SM}$ sub-intervals of equal length, $T_{SAM_SM}$, such that $T_{SAM} = T_{SAM_SM} \times N_{SAM_SM}$. $T_{SAM_SM}$ shall be an even integer so as to contain TX/RX slot pairs.

The words "SAM submap" (or simply "submap" when there is no ambiguity from the context) will be used to denote a sub-interval of $T_{SAM_SM}$ slots. Each submap is assigned one of the four usage types defined in [Vol 2] Part C, Section 5.2 and repeated for convenience in Table 8.9.

| Submaptype code | Meaning |
|---|---|
| 0 | Each slot is individually available or unavailable as configured. Slots may have different availabilities for transmission and reception. |
| 1 | All slots are available for transmission and reception. |

| Submaptype code | Meaning |
|---|---|
| 2 | All slots are unavailable for transmission and reception. |
| 3 | Reserved for future use |

*Table 8.9: SAM submap types*

The SAM facilities specify each SAM slot map at the submap granularity. Figure 8.17 shows an example of a SAM slot map in terms of submaps in the local device. Each box represents a sub-interval of $T_{SAM_SM}$ slots, with the label on the box indicating the type of the submap.

At the submap level of granularity, transmit and receive are handled identically; the difference will only appear inside submaps with type 0.

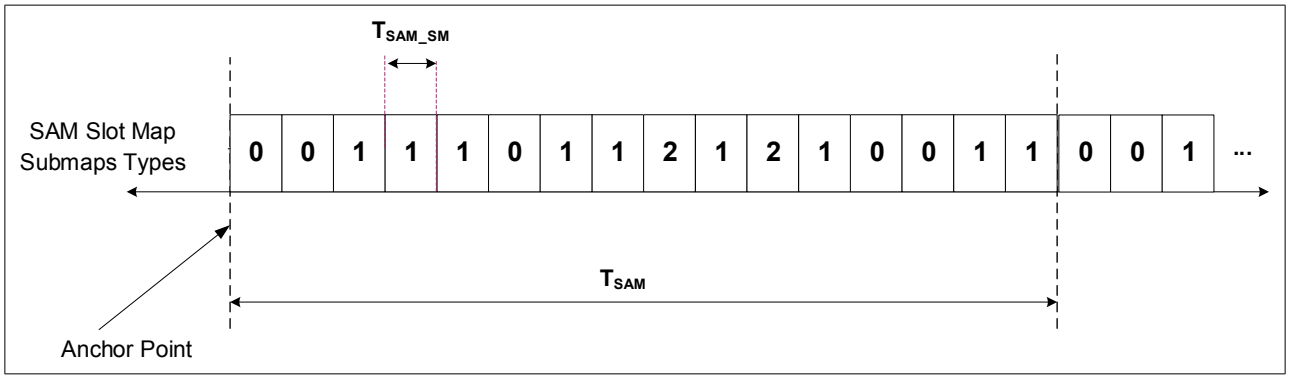



*Figure 8.17: Example of a SAM slot map in terms of submaps in the local device*

In addition to the SAM slot maps, the Controller also maintains a single "SAM type 0 submap" that is used to specify the availability of individual slots for those submaps shown as having type 0. The Controller then combines the SAM slot map at submap granularity with the first $T_{SAM_SM}$ entries of the type 0 submap to generate the effective maps. Figure 8.18 shows a SAM slot map and a SAM type 0 submap together with the resulting effective map.

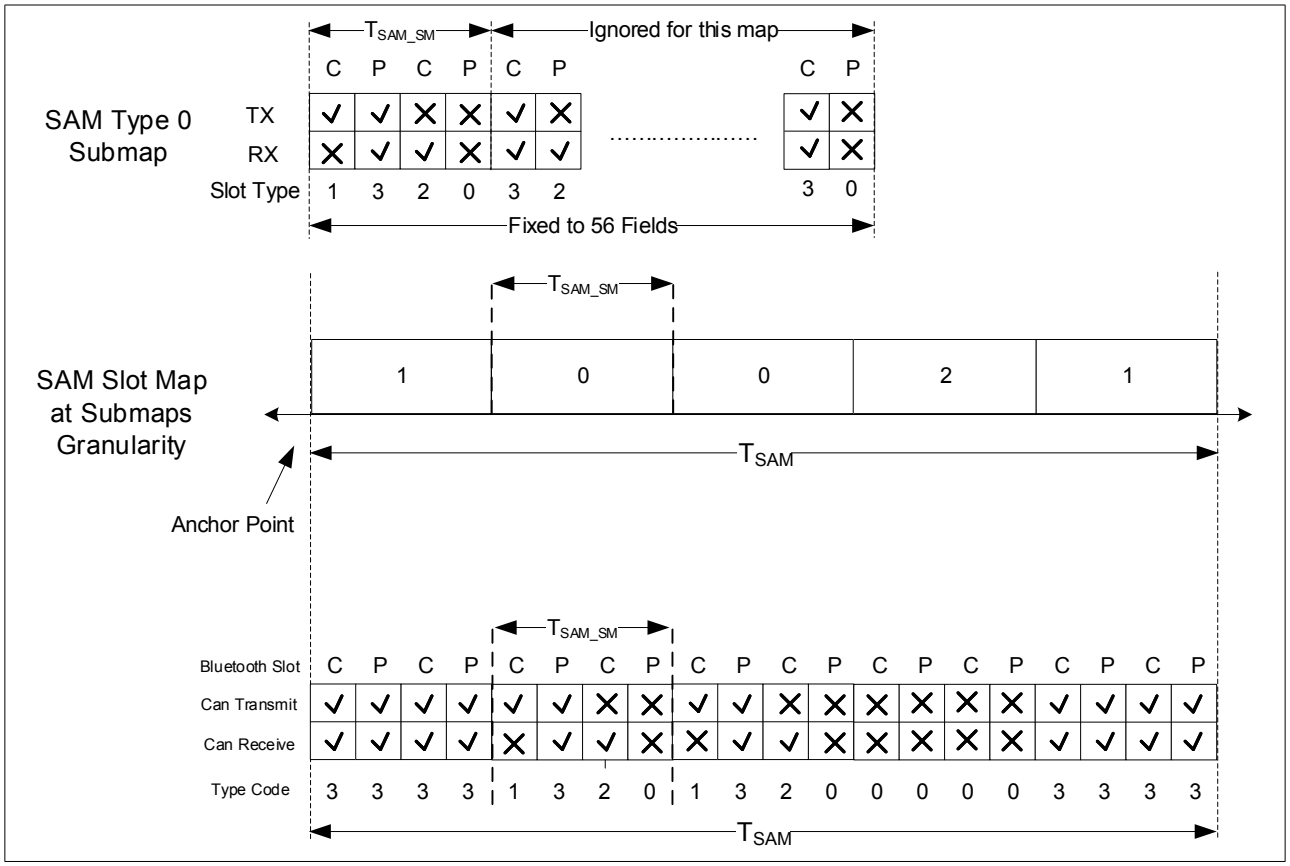



*Figure 8.18: Example of a SAM slot map and SAM type 0 submap being combined*

Figure 8.19 shows an example of a SAM type 0 submap for a collocated MWS.

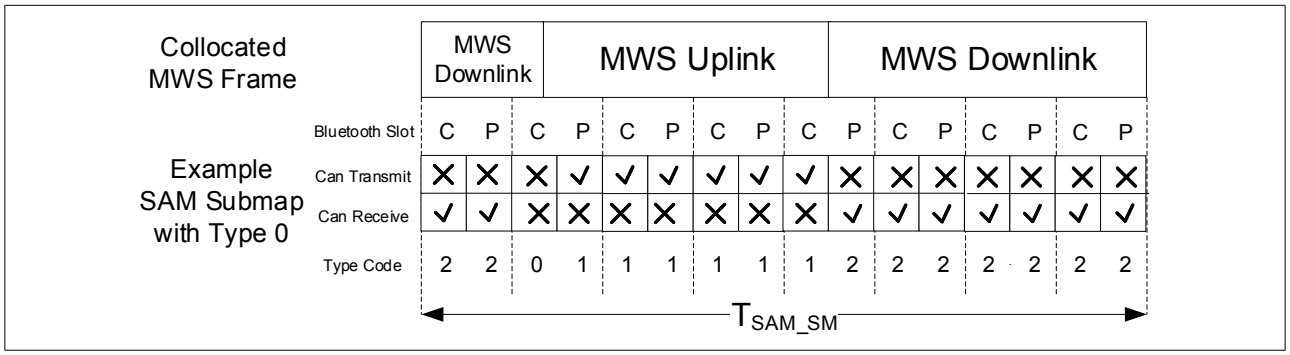



*Figure 8.19: Examples of SAM type 0 submap at a Bluetooth device with a collocated MWS*

The assumption in the example is that, in order to prevent mutual interference between Bluetooth and MWS radios, Bluetooth transmissions are not available during MWS downlink durations and Bluetooth reception is not available during MWS uplink durations. The MWS frame is 10 ms long and the SAM type 0 submap needs to have the same length, so $T_{SAM_SM}$ is 16.

Figure 8.20 shows another example of a collocated Bluetooth device with a low traffic load in the MWS Uplink. In this case, the implementation chooses to mark the MWS Uplink portion available for both TX and RX, because there is a good chance that it can receive Central packets and respond to them without interference from MWS.

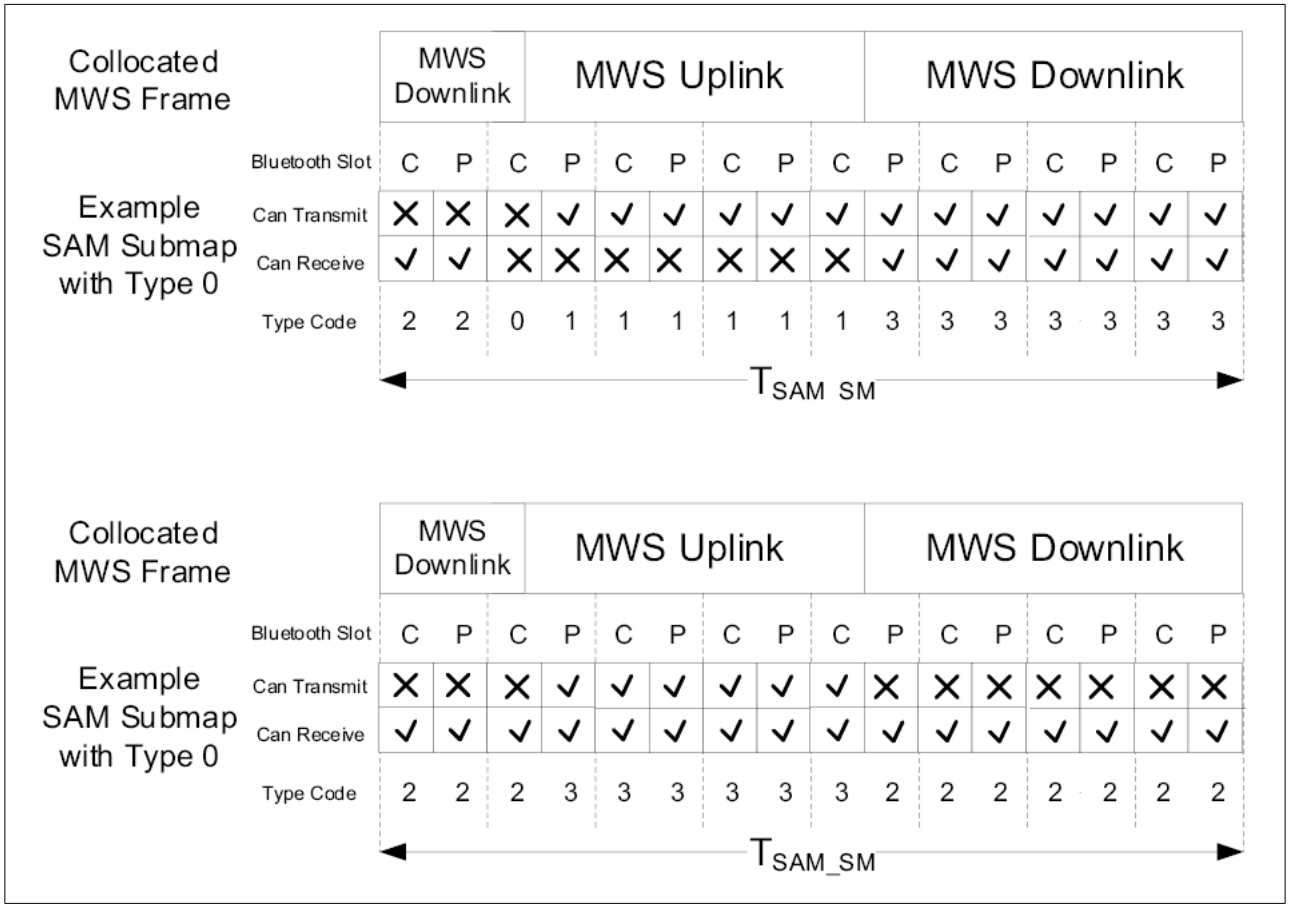



*Figure 8.20: Examples of SAM type 0 submap at a Bluetooth device with a collocated MWS with low uplink traffic load*

SAM allows each device to send its local slot availability to the other, in which case the availability of slots depends on the combined slot maps. The scheduling recommendations for such a SAM slot map are described in Section 8.6.11.2.

A device that supports the SAM feature shall be capable of storing three distinct SAM slot maps from each connected remote device. A remote device can only define one type 0 submap, which can be referenced by any of its defined SAM slot maps. The remote device may redefine its type 0 submap at any time.

### 8.6.11.1 SAM anchor point

SAM slot maps are periodic, defined by an interval $T_{SAM}$ and an offset $D_{SAM}$. Figure 8.21 shows an example of SAM slot maps designed to correspond to periodic MWS active and inactivity cycles.

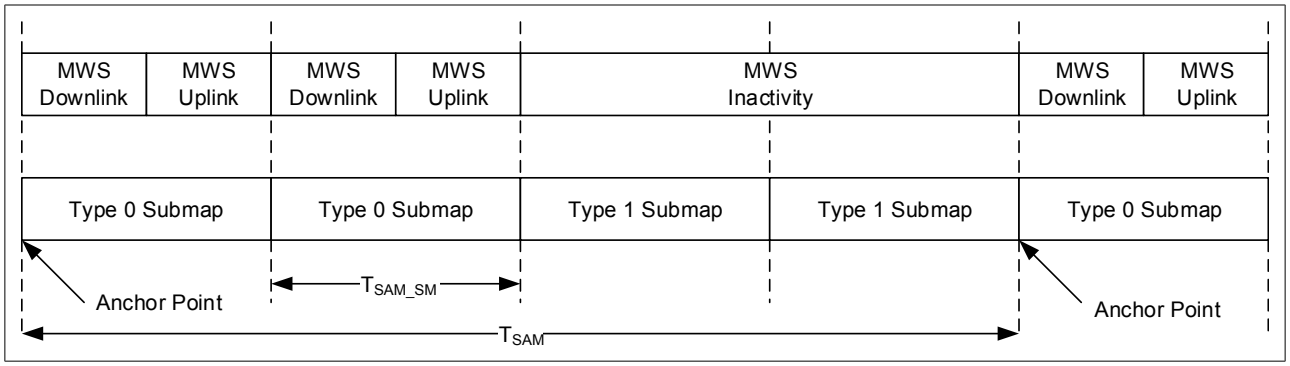



*Figure 8.21: An example of SAM slot maps for a periodic MWS active and inactive cycles*

SAM anchor points are the start points of the first slot in the SAM slot maps. SAM anchor points are computed from $T_{SAM}$ and $D_{SAM}$ using the Central's current clock, which is known to both devices. The first SAM anchor point is indicated by SAM_Instant. In order to prevent problems caused by the clock wrap-around, the LMP message carries an additional timing control flag, which indicates whether initialization procedure 1 or 2 shall be used. The initiating device shall use initialization 1 when the MSB of the Central's current clock ($CLK_{27}$) is 0; it shall use initialization 2 when the MSB of the Central's current clock ($CLK_{27}$) is 1. The responding device shall apply the initialization method indicated by the initialization flag. The SAM anchor points shall be initialized on the slots for which the clock satisfies the applicable equations:

$$CLK_{27-1} \; mod \; T_{SAM} = D_{SAM} \quad \text{for initialization 1}$$

$$(\overline{CLK_{27}}, CLK_{26-1}) \; mod \; T_{SAM} = D_{SAM} \quad \text{for initialization 2}$$

After initialization, the clock value CLK(k+1) for the next SAM anchor point can be found by adding the fixed interval $T_{SAM}$ to the clock value of the current anchor point:

$$CLK_{27-1}(k+1) = CLK_{27-1}(k) + T_{SAM}$$

The SAM anchor point shall be a Central-to-Peripheral slot. As a corollary, the first slot of every SAM submap is also a Central-to-Peripheral slot.

### 8.6.11.2 SAM scheduling

SAM enables each device to send its local slot availability to the other. The scheduler in the Bluetooth Controller should consider the SAM slot maps from both local and remote devices for packet scheduling. The only effect of using the SAM information is to restrict a device's transmission and reception opportunities.

For all ACL packets and for eSCO packets in the retransmission window:

1. The Central should only transmit a packet if the required Central-to-Peripheral slots are marked as available for transmission in the Central's SAM slot map and available for reception in the Peripheral's SAM slot map.
2. The Peripheral should only transmit a packet if the required slots are marked as available for reception in the Central's SAM slot map and available for transmission in the Peripheral's SAM slot map.
3. The Peripheral may choose not to listen to the Central in Central-to-Peripheral slots that are marked either unavailable for transmission in the Central’s SAM slot map or unavailable for reception in the Peripheral’s SAM slot map.

For all ACL packets, the Central should only transmit a packet if the slot immediately following the Central's packet is marked as available for transmission in the Peripheral's SAM slot map and available for reception in the Central's SAM slot map.

For eSCO packets in the retransmission window, the Central should only transmit a packet if either:

1. the slot immediately following the Central's packet is marked as available for transmission in the Peripheral's SAM slot map and available for reception in the Central's SAM slot map;
2. this is the last opportunity for the Central to transmit this packet in this retransmission window; or
3. for all subsequent Central-to-Peripheral slots in the same retransmission window that provide sufficient remaining slots for the Central to transmit the same packet, the rules in this section state that the Central should not transmit the packet.

In SCO and eSCO reserved slots, either the Central or the Peripheral may choose not to transmit a SCO or eSCO packet unless the required slots are marked both as available for transmission in the local SAM slot map and available for reception in the remote SAM slot map.

Figure 8.22 below illustrates possible scheduling results with combined SAM slot maps from Central and Peripheral.

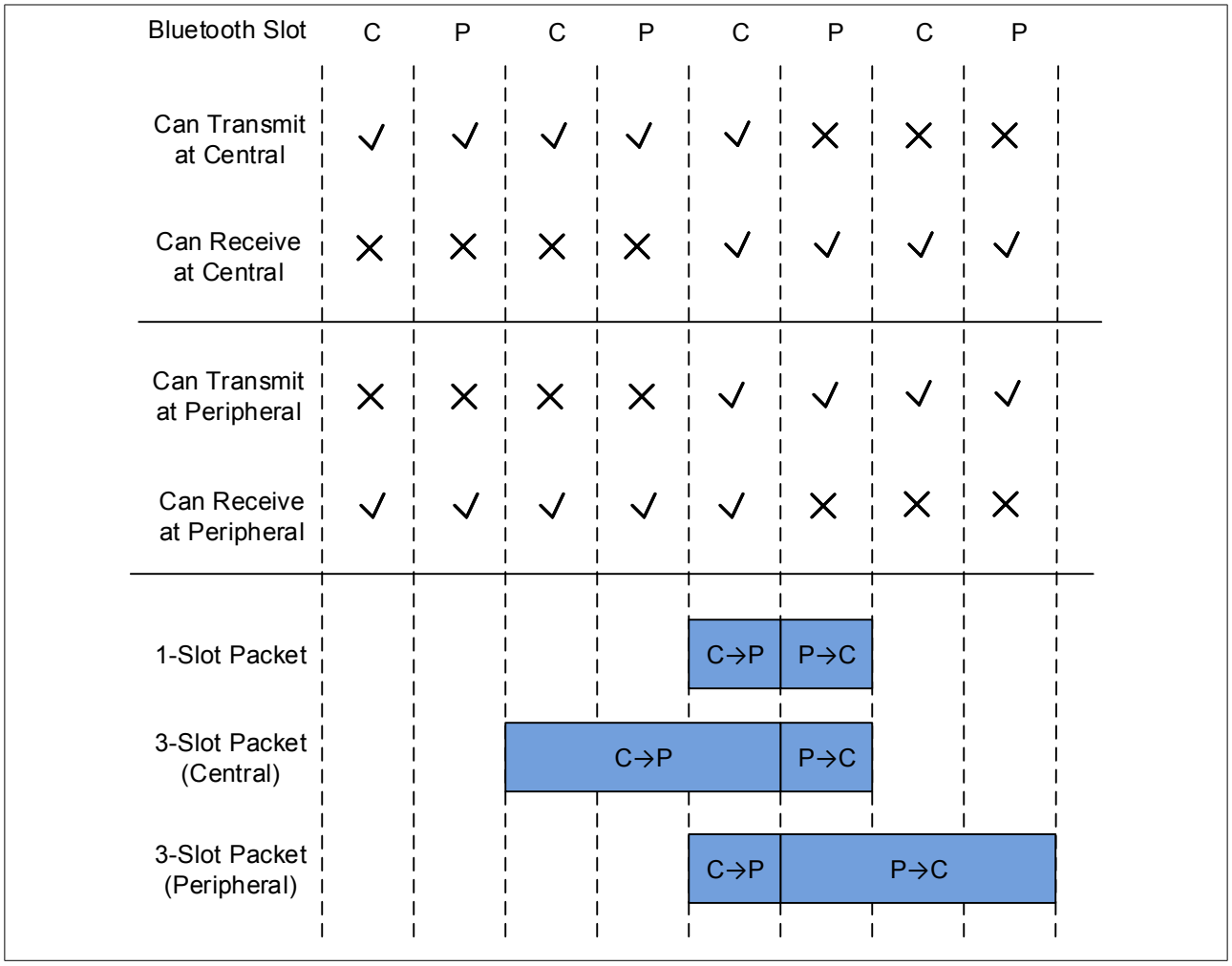



*Figure 8.22: Example of possible scheduling results with combined SAM slot maps from Central and Peripheral*

## 8.7 Sniff mode

In Sniff mode, the duty cycle of the Peripheral's activity in the piconet may be reduced. If a Peripheral is in Active mode on an ACL logical transport, it shall listen in every ACL slot to the Central's traffic, unless that link is being treated as a scatternet link or is absent due to Hold mode. With Sniff mode, the time slots when a Peripheral is listening are reduced, so the Central shall only transmit to a Peripheral in specified time slots. The sniff anchor points are spaced regularly with an interval of $T_{sniff.}$

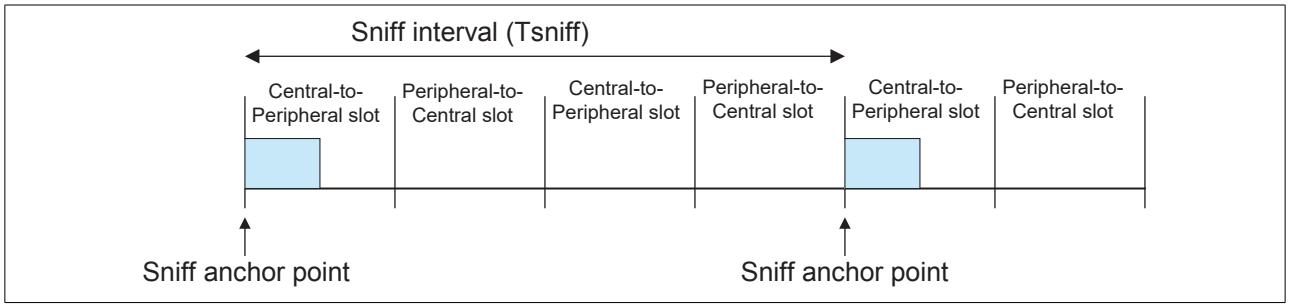



*Figure 8.23: Sniff anchor points*

The Peripheral listens in Central-to-Peripheral transmission slots starting at the sniff anchor point. It shall use the following rules to determine whether to continue listening:

- If fewer than $N_{sniff\ attempt}$ Central-to-Peripheral transmission slots have elapsed since the sniff anchor point then the Peripheral shall continue listening.
- If the Peripheral has received a packet with a matching LT_ADDR that contains ACL data (DM, DH, or DV packets) in the preceding $N_{sniff\ timeout}$ Central-to-Peripheral transmission slots then it shall continue listening.
- If the Peripheral has transmitted a packet containing ACL data (DM, DH, or DV packets) in the preceding $N_{sniff\ timeout}$ Peripheral-to-Central transmission slots then it shall continue listening.
- If the Peripheral has received any packet with a matching LT_ADDR in the preceding $N_{sniff\ timeout}$ Central-to-Peripheral transmission slots then it may continue listening.
- A device may override the rules above and stop listening prior to $N_{sniff\ timeout}$ or the remaining $N_{sniff\ attempt}$ slots if it has activity in another piconet.

It is possible that activity from one sniff timeout may extend to the next sniff anchor point. Any activity from a previous sniff timeout shall not affect activity after the next sniff anchor point. So in the above rules, only the slots since the last sniff anchor point are considered.

Note: $N_{sniff\ attempt}$=1 and $N_{sniff\ timeout}$=0 cause the Peripheral to listen only at the slot beginning at the sniff anchor point, irrespective of packets received from the Central.

$N_{sniff\ attempt}$ =0 shall not be used.

Sniff mode only applies to asynchronous logical transports and their associated LT_ADDR. Sniff mode shall not apply to synchronous logical transports, therefore, both Centrals and Peripherals shall still respect the reserved slots and retransmission windows of synchronous links.

To enter Sniff mode, the Central or Peripheral shall issue a sniff setup message via the LM protocol. This message includes the sniff interval $T_{sniff}$ and an offset $D_{sniff.}$ In

addition, an initialization flag indicates whether initialization procedure 1 or 2 shall be used. The device sending the sniff request shall use initialization 1 when the MSB of the Central's current clock ($CLK_{27}$) is 0; it shall use initialization 2 when the MSB of the Central's current clock ($CLK_{27}$) is 1. The device that receives the sniff request shall apply the initialization method as indicated by the initialization flag irrespective of its own $CLK_{27}$ value. The sniff anchor point determined by the Central and the Peripheral shall be initialized on the slots for which the clock satisfies the applicable equation:

$$\begin{aligned} &\mathrm{CLK}_{27-1} \ mod \ T_{\mathrm{sniff}} = D_{\mathrm{sniff}} && \text{for initialization 1} \\ &\left(\overline{\mathrm{CLK}_{27}}, \mathrm{CLK}_{26-1}\right) \ mod \ T_{\mathrm{sniff}} = D_{\mathrm{sniff}} && \text{for initialization 2} \end{aligned}$$

this implies that $D_{sniff}$ must be even

After initialization, the clock value CLK(k+1) for the next sniff anchor point shall be derived by adding the fixed interval $T_{sniff}$ to the clock value of the current sniff anchor point:

$$\mathrm{CLK}_{27-1}(\mathrm{k}+1) = \mathrm{CLK}_{27-1}(\mathrm{k}) + T_{\mathrm{sniff}}$$

### 8.7.1 Sniff Transition mode

Sniff Transition mode is a special mode which is used during the transition between Sniff mode and Active mode. It is required because during this transition it is unclear which mode (Sniff or Active) the Peripheral is in and it is necessary to ensure that the Peripheral is polled correctly regardless of which mode it is in.

In Sniff Transition mode the Central shall maintain the Active mode poll interval in case the Peripheral is in Active mode. In addition the Central shall poll the Peripheral at least once in the sniff attempt transmit slots starting at each sniff anchor point. This transmission counts for the Active mode polling as well. The Central shall use its high power accurate clock when in Sniff Transition mode.

The precise circumstances under which the Central enters Sniff Transition mode are defined in [Vol 2] Part C, Section 4.5.3.1.

### 8.7.2 Sniff subrating

When sniff subrating is enabled by the Link Manager a device alternates between Sniff mode and Sniff Subrating mode. Sniff Subrating mode allows a device to use a reduced number of sniff anchor points. A device shall transition from Sniff mode to Sniff Subrating mode based on the Sniff mode timeout value (see Section 8.7.2.1). A device shall transition from Sniff Subrating mode to Sniff mode whenever ACL-U, APB-U, ACL-C, or APB-C data is received from the remote device.

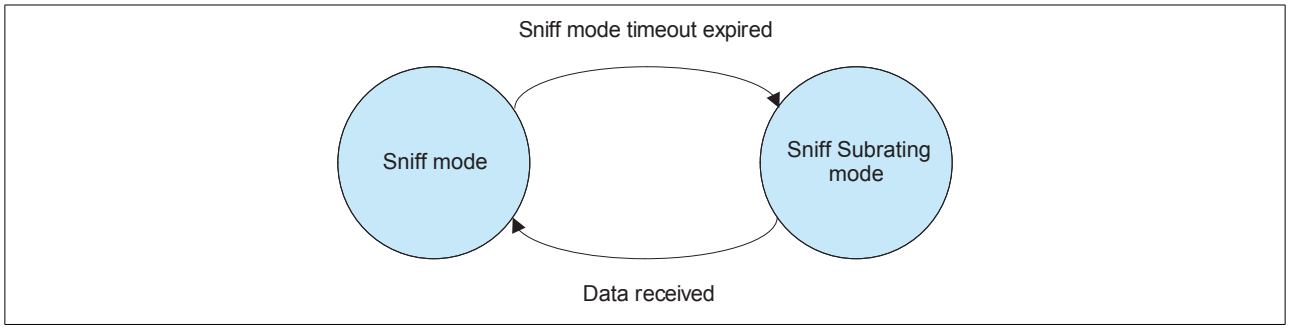



*Figure 8.24: Sniff subrating*

A Peripheral in Sniff Subrating mode shall also temporarily enter Sniff mode after transmitting a packet requiring acknowledgment until the Baseband acknowledgment is received.

Once sniff subrating is enabled the Central and Peripheral may be in Sniff mode or Sniff Subrating mode at different times. The rules defined in Section 8.7.2.2 describe how the two devices maintain synchronization and can reliably exchange data.

### 8.7.2.1 Sniff mode timeout

The Sniff mode timeout value $T_{sniff_mode_timeout}$ used by a device shall be at least the greater of the value specified by the peer over LMP and any value specified by the Host.

Whenever a packet containing ACL-U, APB-U, ACL-C, or APB-C data is received by a device in Sniff Subrating mode it shall transition to Sniff mode, re-start the Sniff mode timeout timer, and shall then use all sniff anchor points until at least $T_{sniff_mode_timeout}$ slots have expired. If ACL-U, APB-U, ACL-C, or APB-C data is received before $T_{sniff_mode_timeout}$ slots have passed since the last ACL-U, APB-U, ACL-C, or APB-C data was received, the Sniff mode timeout timer shall be restarted. NULL and POLL packets do not contain ACL or APB data and shall not restart the Sniff mode timeout timer.

### 8.7.2.2 Sniff Subrating mode

When the Sniff mode timeout has expired a device shall enter sniff subrating mode. In Sniff Subrating mode the mandatory sniff subrating anchor points at which the Central shall transmit to the Peripheral and the Peripheral shall listen for the Central, are defined in Table 8.10 (where M is the max subrate received by the Central, N is the max subrate received by the Peripheral, A is the sniff subrating instant, and k is any positive integer).

$j \leq X$

When sniff subrating is enabled, the rules specified in Section 8.7 for $N_{sniff\ attempt}$ and $N_{sniff\ Timeout}$ shall apply to sniff subrating anchor points.

| | Central | Peripheral |
|---|---|---|
| **M=N** | $CLK_{M(k)} = A + [T_{sniff} \times M \times k]$ | $CLK_{N(k)} = A + [T_{sniff} \times N \times k]$ |
| **M>N** | At least once every j anchor points satisfying $CLK_{N(k)} = A + [T_{sniff} \times N \times k]$, where $j = \lfloor M \div N \rfloor$ | $CLK_{N(k)} = A + [T_{sniff} \times N \times k]$ |
| **M<N** | $CLK_{M(k)} = A + [T_{sniff} \times M \times k]$ | At least once every j anchor points satisfying $CLK_{M(k)} = A + [T_{sniff} \times M \times k]$, where $j = \lfloor N \div M \rfloor$ |

*Table 8.10: Sniff subrating anchor points*

## 8.8 Hold mode

During the Connection state, the ACL logical transport to a Peripheral can be put in a Hold mode. In Hold mode the Peripheral temporarily shall not support ACL packets on the channel. Any synchronous packet during reserved synchronous slots (from SCO and eSCO links) shall be supported. With the Hold mode, capacity can be made free to do other things like scanning, paging, inquiring, or attending another piconet. The device in Hold mode can also enter a low-power sleep mode. During Hold mode, the Peripheral keeps its logical transport address(es) (LT_ADDR).

Prior to entering Hold mode, Central and Peripheral agree on the time duration the Peripheral remains in Hold mode. A timer shall be initialized with the *holdTO* value. When the timer is expired, the Peripheral shall wake up, synchronize to the traffic on the channel and will wait for further Central transmissions.

## 8.9 [This section is no longer used]

## 8.10 Connectionless Peripheral Broadcast mode

In Connectionless Peripheral Broadcast mode, the Transmitter broadcasts packets to zero or more Receivers.

The Transmitter can broadcast messages to multiple Receivers in Connectionless Peripheral Broadcast mode. For this purpose, the Transmitter (Central) shall reserve an LT_ADDR for use only by Connectionless Peripheral Broadcast traffic. In Connectionless Peripheral Broadcast mode, the Transmitter transmits packets at specified intervals.

### 8.10.1 Connectionless Peripheral Broadcast transmit operation

In Connectionless Peripheral Broadcast mode, the Transmitter transmits packets on the CPB logical transport at intervals requested by the Host in Central-to-Peripheral transmission slots.

If the Host has not yet provided the BR/EDR Controller with any data packets to transmit since enabling the broadcast, or if the length of the data is zero, the BR/EDR Controller shall transmit NULL packets. This is different than the case where the Host has provided data since enabling the broadcast but has not provided new data since the previous broadcast packet. In that case, the BR/EDR Controller resends the most recent data.

The Host may provide Connectionless Peripheral Broadcast data through HCI commands. Because HCI commands are limited to 255 bytes, a single command cannot carry the maximum payloads allowed by larger packets, such as DH5. Therefore, HCI commands for Connectionless Peripheral Broadcast allow fragmentation of large payloads at the HCI level. The BR/EDR Controller shall assemble all HCI fragments of a packet before transmission and shall not transmit incomplete packets. Until such assembly is complete, the Controller shall continue to transmit the previous data (if any).

Figure 8.25 shows the timing of Connectionless Peripheral Broadcast packets.

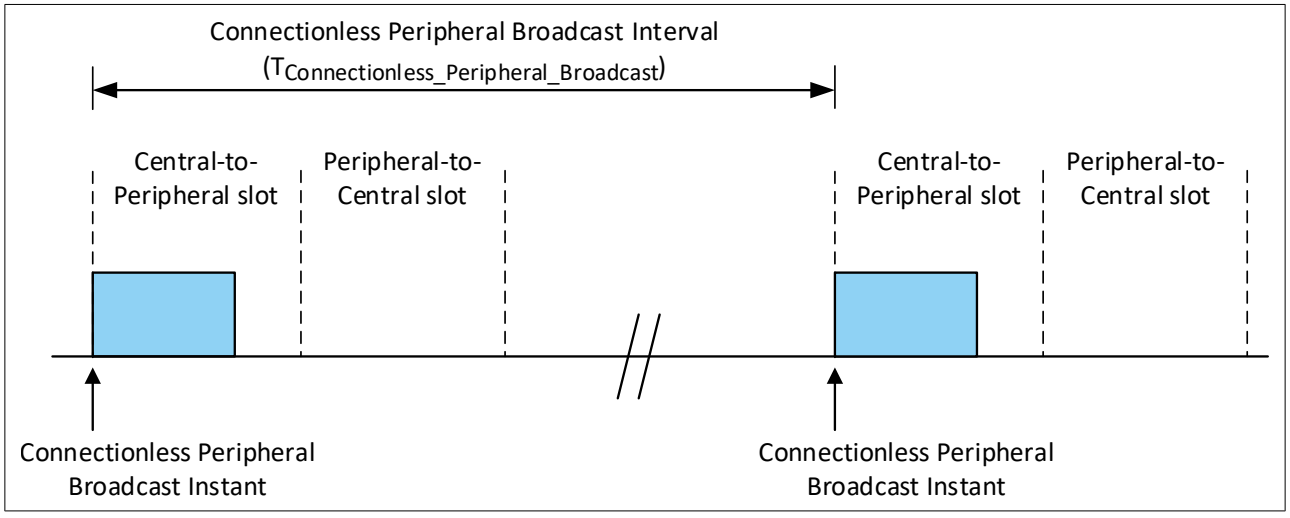



*Figure 8.25: Connectionless Peripheral Broadcast timing*

Connectionless Peripheral Broadcast Instants are separated by the Connectionless Peripheral Broadcast Interval ($T_{Connectionless_Peripheral_Broadcast}$). The Connectionless Peripheral Broadcast interval can be any even value from 0x0002 to 0xFFFE slots and is negotiated between the BR/EDR Controller and the Host during CPB setup. At the start of each Connectionless Peripheral Broadcast Instant, the Connectionless

Peripheral Broadcast packet is transmitted using the adapted piconet physical channel. Connectionless Peripheral Broadcast packets shall not be encrypted.

The Transmitter shall stop transmitting Connectionless Peripheral Broadcast packets after *CPB_supervisionTO* slots have passed without transmission of a Connectionless Peripheral Broadcast packet. *CPB_supervisionTO* may be any even value from 0x0002 to 0xFFFE slots. Connectionless Peripheral Broadcast packets can fail to transmit for a certain continuous period of time due to scheduling conflicts from higher priority traffic.

### 8.10.2 Connectionless Peripheral Broadcast receive operation

In Connectionless Peripheral Broadcast mode, the Receiver is synchronized to a Connectionless Peripheral Broadcast Transmitter and is receiving profile broadcast data on the LT_ADDR reserved by the Transmitter for the CPB logical transport.

When requesting synchronization establishment, the Host provides the Controller with the Transmitter’s clock offset and the next Connectionless Peripheral Broadcast Instant received from the synchronization train. Because of delays in the Host, it is possible that the Connectionless Peripheral Broadcast Instant occurs in the past. The BR/EDR Controller shall support Connectionless Peripheral Broadcast instants at least 1 second in the past. The Receiver should determine whether the Connectionless Peripheral Broadcast Instant is in the past or the future by using the Transmitter’s clock offset and its own clock to estimate the Transmitter's current clock and start listening from the next broadcast instant.

The Skip parameter is set by the Host and specifies the number of consecutive Connectionless Peripheral Broadcast instants which the receiver may skip after successfully receiving a Connectionless Peripheral Broadcast packet. If the Connectionless Peripheral Broadcast packet is received successfully, the payload data shall be forwarded to the Host. If no Connectionless Peripheral Broadcast packet is received by a Receiver during a Connectionless Peripheral Broadcast Instant, the Receiver shall ignore Skip and instead listen at every scheduled Connectionless Peripheral Broadcast Instant until it is able to successfully receive a Connectionless Peripheral Broadcast packet or until *CPB_supervisionTO* number of slots have passed.

The BR/EDR Controller shall stop listening for Connectionless Peripheral Broadcast packets after *CPB_supervisionTO* slots have passed without receiving a Connectionless Peripheral Broadcast packet.

The BR/EDR Controller may transfer Connectionless Peripheral Broadcast data to the Host through HCI events. Because HCI events are limited to 255 bytes a received packet will not necessarily fit into a single HCI event. The BR/EDR Controller shall fragment and transfer such packets using multiple HCI events.

### 8.10.3 AFH in Connectionless Peripheral Broadcast

Connectionless Peripheral Broadcast packets shall be transmitted on the adapted piconet channel and the synchronization train shall always contain the current AFH channel map for the PBD logical link.

## 8.11 Synchronization establishment substates

### 8.11.1 Synchronization Scan substate

The Synchronization Scan substate is used by Peripherals to receive synchronization train packets from the piconet Central. The Synchronization Scan substate can be entered from the Standby or Connection states. In the Synchronization Scan substate, a device uses the Synchronization Scan procedure (see Section 2.7.3). During each synchronization scan window, the device shall listen on a single frequency with its correlator matched to the Central's channel access code (CAC).

If the correlator exceeds the trigger threshold during the Synchronization Scan procedure, the scanning device shall receive the synchronization train packet, whose contents are defined in Table 8.11. Upon reception of the synchronization train packet the device shall exit the Synchronization Scan substate and return to the Standby or Connection state as appropriate. If the packet is not received the device should stay in the Synchronization Scan substate. If the *synchronization_scanTO* expires before reception of a Synchronization Train packet, the device shall return to the Standby state. A device attempting to synchronize to a Connectionless Peripheral Broadcast transport shall ignore any synchronization train packet whose Connectionless Peripheral Broadcast LT_ADDR field in the payload is set to zero.

The synchronization scan may be interrupted by higher priority traffic. In particular, the reserved synchronous slots should have higher priority than the synchronization scan.

### 8.11.2 Synchronization Train substate

The Synchronization Train substate is used by the Central of the piconet to transmit synchronization train packets. The Synchronization Train substate can be entered from the Standby or Connection states. In the Synchronization Train substate, a device uses the Synchronization Train procedure (see Section 2.7.2). During the Synchronization

Train procedure, the Central repeatedly transmits synchronization train packets on the channels specified in Section 2.6.4.8. While in the Synchronization Train substate:

- if Connectionless Peripheral Broadcast mode is enabled, the Central shall continue to transmit Connectionless Peripheral Broadcast packets in addition to synchronization train packets;
- if the Central is in Coarse Clock Adjustment Recovery Mode, it shall continue transmitting and receiving on all active logical transports to all Peripherals that have acknowledged the Coarse Clock Adjustment (see [Vol 2] Part C, Section 4.1.14.1).

Reserved SCO and eSCO slots should have higher priority than the synchronization train. Once the Central enters the Synchronization Train substate, it shall remain in the Synchronization Train substate until *synchronization_trainTO* expires or the Host directs otherwise. The Central shall transition to the Standby or Connection state when exiting the Synchronization Train substate.

The synchronization train packet is a basic rate ACL packet with type DM3 and LT_ADDR of zero. FLOW, ARQN and SEQN shall all be set to zero upon transmission and ignored upon receipt. The HEC LFSR shall be initialized with the Central's UAP. In the payload header, the LLID shall contain the value 0b10 (start of an L2CAP message or no fragmentation). FLOW in the payload header shall be set to zero upon transmission and ignored upon reception. The length of the payload body (LENGTH) shall be 28 bytes. The CRC LFSR shall be initialized with the Central's UAP. Data whitening is not used. Encryption is not used.

There are two possible formats. Format 1 shall be used when the synchronization train is being transmitted by a device which is the Central of a piconet where Connectionless Peripheral Broadcast mode is enabled, and format 2 shall be used otherwise. The format of the payload portion of the DM3 packet used in the synchronization train is shown in Table 8.11.

| Bytes | Field | Length | Format 1 Description | Format 2 Description |
|---|---|---|---|---|
| 0-3 | CLK | 4 Bytes | Current piconet clock (CLK) | Current piconet clock (CLK) |
| 4-7 | Future Connectionless Peripheral Broadcast Instant | 4 Bytes | CLK[27:1] corresponding to the start of a future Connectionless Peripheral Broadcast Instant. | CLK[27:1] + 1600 slots |

| Bytes | Field | Length | Format 1 Description | Format 2 Description |
|---|---|---|---|---|
| 8-17 | AFH Channel Map | 10 Bytes | The current AFH_Channel_-Map used for the PBD logical link. The format is the same as the Link Manager AFH_Channel_Map parameter, see [Vol 2] Part C, Table 5.2. | Unspecified |
| 18-23 | Central BD_ADDR | 6 Bytes | Central's BD_ADDR | Central's BD_ADDR |
| 24-25 | Connectionless Peripheral Broadcast Interval | 2 Bytes | Time duration in slots from the start of a Connectionless Peripheral Broadcast packet to the start of the next Connectionless Peripheral Broadcast packet. Shall be even. | Unspecified |
| 26 | Connectionless Peripheral Broadcast LT_ADDR | 1 Byte | The LT_ADDR reserved for the CPB logical transport. | 0x00 |
| 27 | Service Data | 1 Byte | Defined by the service using the Connectionless Peripheral Broadcast feature. | 0x00 |

*Table 8.11: Synchronization train packet payload body*

The Host provides the Service Data for the synchronization train packet payload body when format 1 is used. The BR/EDR Controller provides the data for all other fields.

All devices in the Synchronization Scan substate shall accept payloads of any length from 28 to 121 bytes and shall ignore bytes 28 (counting from 0) and beyond.

When format 1 is used the Future Connectionless Peripheral Broadcast Instant in the synchronization train packet payload shall correspond to one of the next 4 broadcast instants. Figure 8.26 shows an example of valid and invalid values for this field.

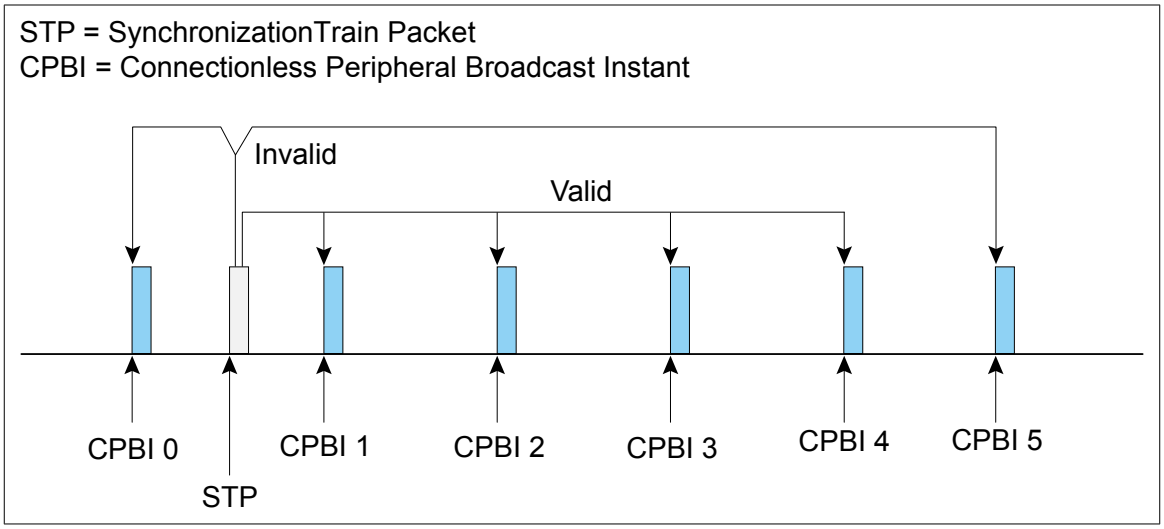



*Figure 8.26: Examples of synchronization train pointing to Connectionless Peripheral Broadcast Instant*

The Central shall not transmit synchronization train packets in a Connectionless Peripheral Broadcast Instant.

# 9 AUDIO

On the air-interface, either a 64 kb/s log PCM (Pulse Code Modulation) format (A-law or μ-law) may be used, or a 64 kb/s CVSD (Continuous Variable Slope Delta Modulation) may be used. The latter format applies an adaptive delta modulation algorithm with syllabic companding.

The voice coding on the line interface is designed to have a quality equal to or better than the quality of 64 kb/s log PCM.

Table 9.1 summarizes the voice coding schemes supported on the air interface. The appropriate voice coding scheme is selected after negotiations between the Link Managers.

| **Voice Codecs** | |
|---|---|
| linear | CVSD |
| 8-bit logarithmic | A-law |
| | μ-law |

*Table 9.1: Voice coding schemes supported on the air interface*

## 9.1 LOG PCM codec

Since the synchronous logical transports on the air-interface can support a 64 kb/s information stream, a 64 kb/s log PCM traffic can be used for transmission. Either A-law or μ-law compression may be applied. In the event that the line interface uses A-law and the air interface uses μ-law or vice versa, a conversion from A-law to μ-law shall be performed. The compression method shall follow ITU-T recommendations G. 711.

## 9.2 CVSD codec

A more robust format for voice over the air interface is delta modulation. This modulation scheme follows the waveform where the output bits indicate whether the prediction value is smaller or larger than the input waveform. To reduce slope overload effects, syllabic companding is applied: the step size is adapted according to the average signal slope. The input to the CVSD encoder shall be 64000 samples per second linear PCM (typically 16 bits, but actual value is implementation specific). Block diagrams of the CVSD encoder and CVSD decoder are shown in Figure 9.1, Figure 9.2 and Figure 9.3. The system shall be clocked at 64 kHz.

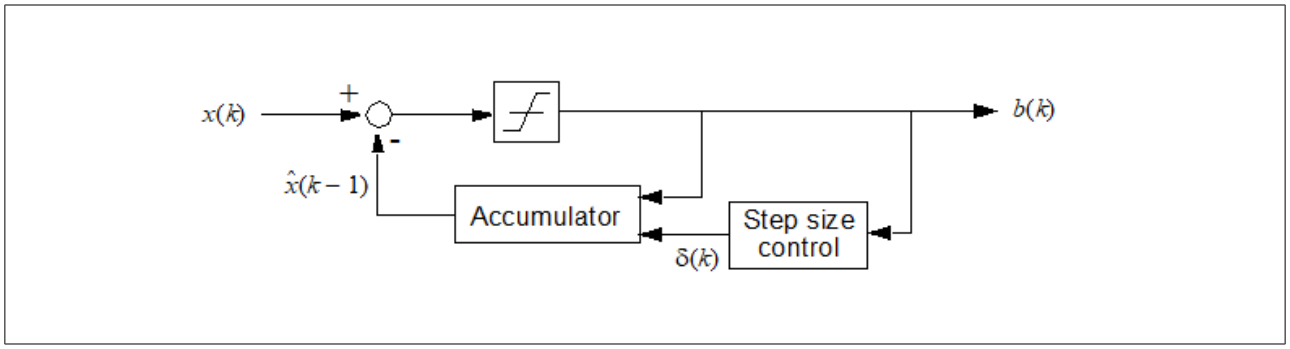



*Figure 9.1: Block diagram of CVSD encoder with syllabic companding*

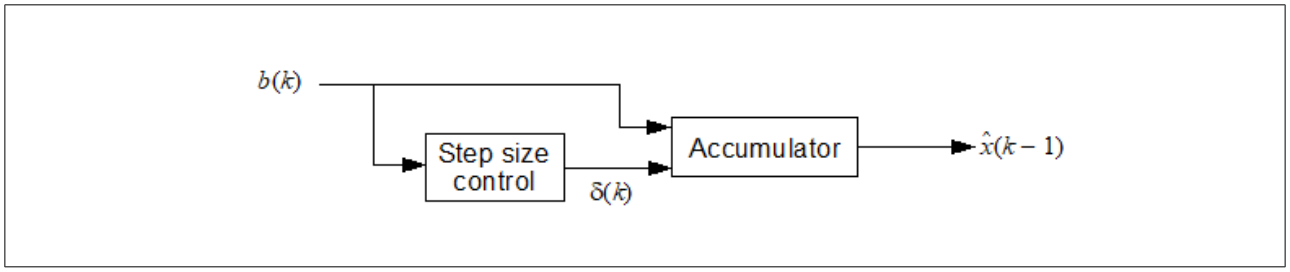



*Figure 9.2: Block diagram of CVSD decoder with syllabic companding*

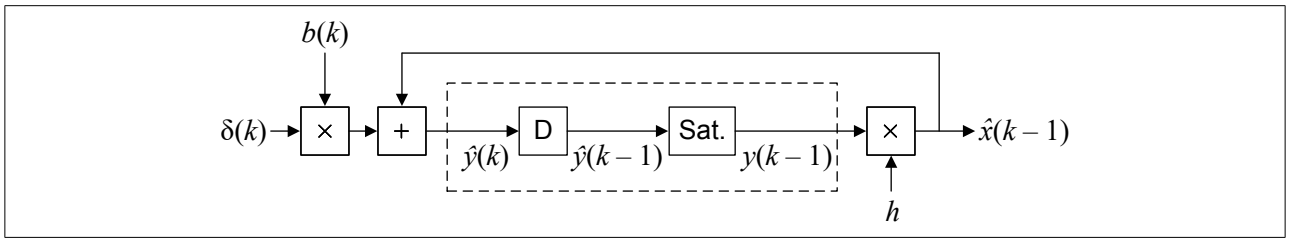



*Figure 9.3: Accumulator procedure*

Let $\mathrm{sgn}(x) = 1$ for $x \geq 0$, otherwise $\mathrm{sgn}(x) = -1$. On air these numbers shall be represented by the sign bit; i.e. negative numbers are mapped to “1” and positive and zero numbers are mapped to “0”.

Denote the CVSD encoder output bit *b*(*k*), the encoder input *x*(*k*), the accumulator contents *y*(*k*), and the step size $\delta(k)$. Furthermore, let *h* be the decay factor for the accumulator, let β denote the decay factor for the step size, and, let α be the syllabic companding parameter. The latter parameter monitors the slope by considering the *K* most recent output bits.

Let

$$\hat{x}(k) = hy(k)\,. \qquad \text{(EQ 14)}$$

Then, the CVSD encoder internal state shall be updated according to the following set of equations:

$$b(k) = \mathrm{sgn}\{x(k) - \hat{x}(k-1)\}, \quad \text{(EQ 15)}$$

$$\alpha = \begin{cases} 1, & \text{if } J \text{ bits in the last } K \text{ output bits are equal,} \\ 0, & \text{otherwise,} \end{cases} \quad \text{(EQ 16)}$$

$$\delta(k) = \begin{cases} \min\{\delta(k-1) + \delta_{min}, \delta_{max}\}, & \alpha = 1, \\ \max\{\beta\delta(k-1), \delta_{min}\}, & \alpha = 0, \end{cases} \quad \text{(EQ 17)}$$

$$y(k) = \begin{cases} \min\{\hat{y}(k), y_{max}\}, & \hat{y}(k) \geq 0\,. \\ \max\{\hat{y}(k), y_{min}\}, & \hat{y}(k) < 0\,. \end{cases} \quad \text{(EQ 18)}$$

where

$$\hat{y}(k) = \hat{x}(k-1) + b(k)\delta(k)\,. \quad \text{(EQ 19)}$$

In these equations, $\delta_{min}$ and $\delta_{max}$ are the minimum and maximum step sizes, and, $y_{min}$ and $y_{max}$ are the accumulator's negative and positive saturation values, respectively. Over air, the bits shall be sent in the same order they are generated by the CVSD encoder.

For a 64 kb/s CVSD, the parameters as shown in Table 9.2 shall be used. The numbers are based on a 16 bit signed number output from the accumulator. These values result in a time constant of 0.5 ms for the accumulator decay, and a time constant of 16 ms for the step size decay

| Parameter | Value |
|---|---|
| $h$ | $1 - \frac{1}{32}$ |
| $\beta$ | $1 - \frac{1}{1024}$ |
| $J$ | 4 |
| $K$ | 4 |
| $\delta_{min}$ | 10 |
| $\delta_{max}$ | 1280 |
| $y_{min}$ | $-2^{15}$ or $-2^{15}+1$ |
| $y_{max}$ | $2^{15}-1$ |

*Table 9.2: CVSD parameter values. The values are based on a 16-bit signed number output from the accumulator.*

## 9.3 Error handling

In the DV, HV3, EV3, EV5, 2-EV3, 3-EV3, 2-EV5 and 3-EV5 packets, the voice is not protected by FEC. The quality of the voice in an error-prone environment then depends on the robustness of the voice coding scheme and, in the case of eSCO, the retransmission scheme. CVSD, in particular, is rather insensitive to random bit errors, which are experienced as white background noise. However, when a packet is rejected because either the channel access code, the HEC test was unsuccessful, or the CRC has failed, measures have to be taken to “fill” in the lost speech segment.

The voice payload in the **HV2** and **EV4** packets are protected by a 2/3 rate FEC. For errors that are detected but cannot be corrected, the receiver should try to minimize the audible effects. For instance, from the 15-bit FEC segment with uncorrected errors, the 10-bit information part as found before the FEC decoder should be used. The **HV1** packet is protected by a 3 bit repetition FEC. For this code, the decoding scheme will always assume zero or one-bit errors. Thus, there exist no detectable but uncorrectable error events for **HV1** packets.

## 9.4 General audio requirements

### 9.4.1 Signal levels

For A-law and μ-law log-PCM encoded signals the requirements on signal levels shall follow the ITU-T recommendation G.711.

Full swing at the 16 bit linear PCM interface to the CVSD encoder is defined to be 3 dBm0.

### 9.4.2 CVSD audio quality

For Bluetooth audio quality the requirements are put on the transmitter side. The 64000 samples per second linear PCM input signal should have negligible spectral power density above 4 kHz. The power spectral density in the 4 kHz to 32 kHz band of the decoded signal at the 64000 samples per second linear PCM output, should be more than 20 dB below the maximum in the 0 kHz to 4 kHz range.

# Appendix A General audio recommendations

The abbreviations in Table A.1 are used only in this appendix.

| | |
|---|---|
| A/D | Analog to digital conversion |
| BTR | Bluetooth Radio |
| D/A | Digital to analog conversion |
| ERP | Ear Reference Point |
| LR | Loudness Rating |
| MRP | Microphone Reference Point |
| PGA | Programmable Gain Amplifier |
| RLR | Receive Loudness Rating |
| SLR | Send Loudness Rating |

*Table A.1: Abbreviations for general audio recommendations*

## A.1 Maximum sound pressure

It is the sole responsibility of each manufacturer to design their audio products in a safe way with regards to injury to the human ear. The Bluetooth Specification doesn't specify maximum sound pressure from an audio device.

## A.2 [This section is no longer used]

## A.3 Audio levels for Bluetooth

Audio levels shall be calculated as SLR and RLR. The calculation methods are specified in ITU-T Recommendation P.79.

The physical test set-up for Handsets and Headsets is described in ITU-T Recommendation P.51 and P.57

The physical test set-up for speakerphones and "Vehicle hands-free systems" is specified in ITU-T Recommendation P.34.

A general equation for computation of LR for telephone sets is given by ITU-T recommendations P.79 and is given by

$$LR = -\frac{10}{m} log_{10}\left(\sum_{i=N_1}^{N_2} 10^{m(s_i - w_i) \div 10}\right), \quad \text{(EQ 20)}$$

where

$m$ is a constant (~ 0.2).
$w_i$ = weighting coefficient (different for the various LRs).
$S_i$ = the sensitivity at frequency $F_i$ , of the electro-acoustic path
$N_1$,$N_2$, consecutive filter bank numbers (Art. Index: 200 Hz to 4000 Hz)

(EQ 20) is used for calculating the SLR according to Figure A.1, and RLR according to Figure A.2. When calculating LRs one must only include those parts of the frequency band where an actual signal transmission can occur in order to ensure that the additive property of LRs is retained. Therefore ITU-T P.79 uses only the frequency band 200 Hz to 4000 Hz in LR computations.

## A.4 Microphone path

**SLR measurement model**

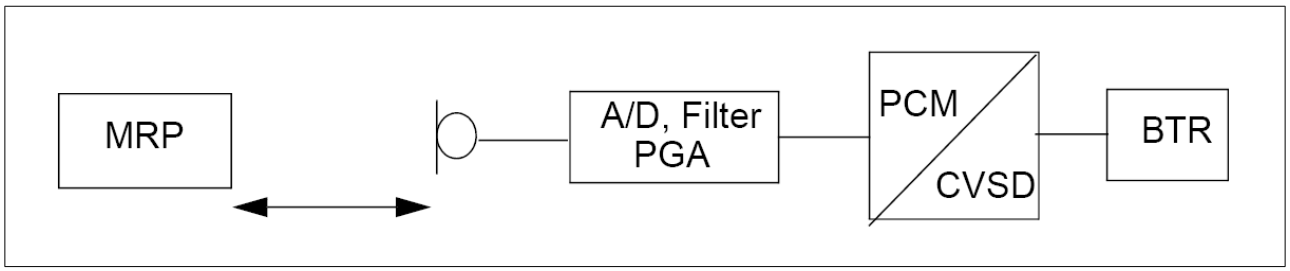



*Figure A.1: SLR measurement set-up*

## A.5 Loudspeaker path

**RLR measurement model**

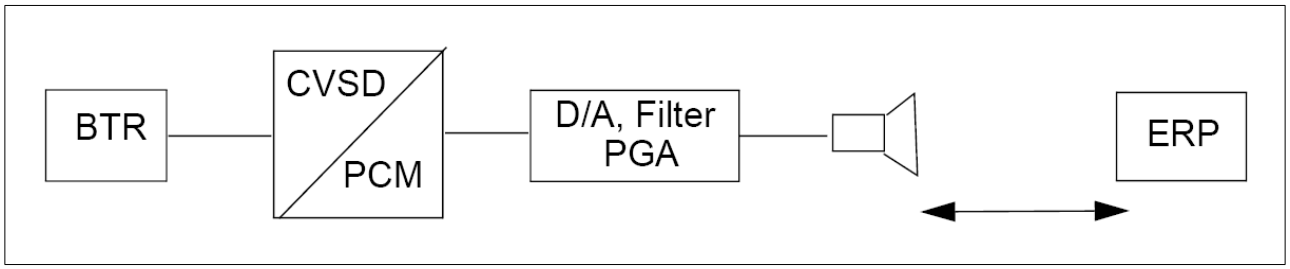



*Figure A.2: RLR measurement set-up*

## A.6 Bluetooth voice interface

The specification for the Bluetooth voice interface should follow in the first place the *ITU-T Recommendations P.79*, which specifies the loudness ratings for telephone sets. These recommendations give general guidelines and specific algorithms used for calculating the loudness ratings of the audio signal with respect to ERP.

For Bluetooth voice interface to the different cellular system terminals, loudness and frequency recommendations based on the cellular standards should be used. For example, GSM 03.50 gives recommendation for both the loudness ratings and frequency mask for a GSM terminal interconnection with Bluetooth.

1- The output of the CVSD decoder are 16-bit linear PCM digital samples, at a sampling frequency of 8000 samples per second. Bluetooth also supports 8-bit log PCM samples of A-law and μ-law type. The sound pressure at the ear reference point for a given 16-bit CVSD sample, should follow the sound pressure level given in the cellular standard specification.

2- A maximum sound pressure which can be represented by a 16-bit linear PCM sample at the output of the CVSD decoder should be specified according to the loudness rating, in ITU P.79 and at PGA value of 0 dB. PGAs are used to control the audio level at the terminals by the user. For conversion between various PCM representations: A-law, μ-law and linear PCM, ITU-T G.711, G.712, G.714 give guidelines and PCM value relationships. Zero-code suppression based on ITU-T G.711 is also recommended to avoid network mismatches.

## A.7 Frequency mask

When interfacing a Bluetooth terminal to a digital cellular mobile terminal, the CVSD decoder signal should conform to the frequency mask given in the GSM cellular standard so that the speech coders operate in the intended manner. A recommendation for a frequency mask is given in Table A.2. Figure A.3 shows a plot of the frequency mask for Bluetooth (solid line). The GSM frequency mask (dotted line) is shown in Figure A.3 for comparison.

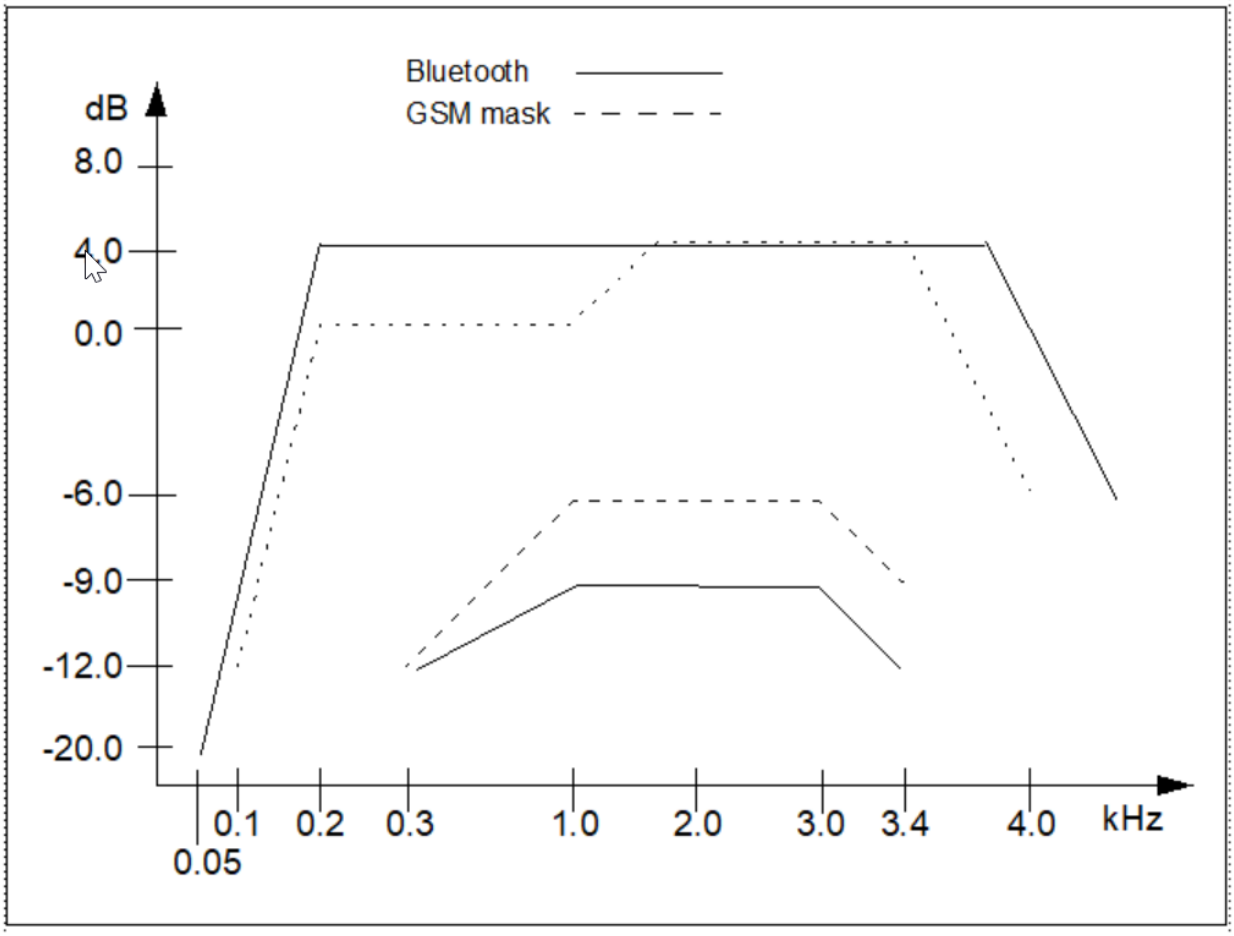



*Figure A.3: Plot of recommended frequency mask for Bluetooth. The GSM send frequency mask is given for comparison (dotted line).*

| Frequency (Hz) | Upper Limit (dB) | Lower Limit (dB) |
|---|---|---|
| 50 | -20 | none |
| 300 | 4 | -12 |
| 1000 | 4 | -9 |
| 2000 | 4 | -9 |
| 3000 | 4 | -9 |
| 3400 | 4 | -12 |
| 4000 | 0 | none |

*Table A.2: Recommended frequency mask for Bluetooth*

# Appendix B Timers

This appendix contains a list of Baseband timers related to inactivity timeout defined in the specification. Definitions and default values of the timers are listed below.

All timer values are given in slots.

## B.1 List of timers

### B.1.1 inquiryTO

The *inquiryTO* defines the number of slots the Inquiry substate shall last. The timer value may be changed by the Host. HCI provides a command to change the timer value.

### B.1.2 pageTO

The *pageTO* defines the number of slots the Page substate can last before a response is received when the *extended_pageTO* is zero. The timer value may be changed by the Host. HCI provides a command to change the timer value.

### B.1.3 extended_pageTO

The *extended_pageTO* defines the number of additional slots the Page substate can last beyond *pageTO* before a response is received. The timer value may be changed by the Host. HCI provides a command to change the timer value.

### B.1.4 pagerespTO

In the Peripheral, *pagerespTO* defines the number of slots the Peripheral shall await the Central's response, FHS packet, after sending the page acknowledgment ID packet. In the Central, *pagerespTO* defines the number of slots the Central should wait for the FHS packet acknowledgment before returning to Page substate. Both Central and Peripheral units should use the same value for this timeout, to allow common page/scan intervals after reaching *pagerespTO*.

The *pagerespTO* value is 8 slots.

### B.1.5 newconnectionTO

Every time a new connection is started through paging, scanning, or role switch, the Central sends a POLL packet as the first packet in the new connection. Transmission and acknowledgment of this POLL packet is used to confirm the new connection. If the POLL packet is not received by the Peripheral or the response packet is not received by

the Central for *newconnectionTO* number of slots, both the Central and the Peripheral should return to the previous substate.

*newconnectionTO* value is 32 slots.

### B.1.6 supervisionTO

The *supervisionTO* is used by both the Central and Peripheral to monitor link loss. If a device does not receive any packets that pass the HEC check and have the proper LT_ADDR for a period of *supervisionTO*, it shall consider the link to be disconnected. The supervision timer keeps running in Hold mode and Sniff mode.

The *supervisionTO* value may be changed by the Host. HCI provides a command to change the timer value. At the Baseband level a default value equivalent to 20 seconds should be used.

### B.1.7 CPB_supervisionTO

The *CPB_supervisionTO* is used by the Peripheral to monitor connection loss and by the Central to monitor scheduling conflicts and shall be between 0x0002 to 0xFFFE with only even values valid. At the Baseband level a default value equivalent to 5.12 seconds should be used. The Host can change the value of *CPB_supervisionTO*.

### B.1.8 synchronization_trainTO

The *synchronization_trainTO* is used by the Central to determine how long to continue broadcasting Synchronization Train packets and shall be between 0x00000002 to 0x07FFFFFE with only even values valid. At the Baseband level a default value equivalent to 120 seconds should be used for Connectionless Peripheral Broadcast and a default value equivalent to 20 seconds should be used for Coarse Clock Adjustment Recovery Mode. The Host may change the value of *synchronization_trainTO* that is used during Connectionless Peripheral Broadcast.

### B.1.9 synchronization_scanTO

The *synchronization_scanTO* is used by the Peripheral to determine when to stop scanning for synchronization train packets. It shall be between 0x0022 to 0xFFFE with only even values valid. At the Baseband level a default value equivalent to 5.12 seconds should be used for Connectionless Peripheral Broadcast and a default value equivalent to 20 seconds should be used during Coarse Clock Adjustment Recovery Mode. The Host may change the value of *synchronization_scanTO* that is used during Connectionless Peripheral Broadcast.

### B.1.10 authenticatedPayloadTO

The *authenticatedPayloadTO* is the maximum amount of time, in seconds, allowed between receiving packets containing a MIC. The Host can change the value of *authenticatedPayloadTO*.

The default value for *authenticatedPayloadTO* is 30 seconds.

### B.1.11 CLK_adj_dragTO

The *CLK_adj_dragTO* is the minimum amount of time, in seconds, the drag shall be suspended when a Peripheral has not responded after the Central has dragged the clock.

The *CLK_adj_dragTO* value shall be 1 second.

# Appendix C Recommendations for AFH operation in Hold, Sniff, and Connectionless Peripheral Broadcast modes

The three possible AFH operation modes for an AFH capable Peripheral in Hold mode and Sniff mode are the same three AFH operation modes used during Connection state:

- *Enabled* (using an AHS with some RF channels unused)
- *Enabled* (using AHS(79))
- *Disabled*

The Central may place an AFH capable Peripheral in any of the three AFH operating modes.

## C.1 Operation at the Central

A Central that has one or more Peripherals in Hold mode or Sniff mode and decides to update them simultaneously shall schedule an *AFH_Instant* for a time that allows it to update all these Peripherals (as well as its active Peripherals) with the new adaptation.

A Central that has multiple Peripherals with non-overlapping “wake” times (e.g. Peripherals in Sniff mode with different timing parameters) may keep them *enabled* on the same adaptation provided that its scheduling of the *AFH_Instant* allows enough time to update them all.

This timing is summarized in Figure C.1. In this example the Central decides that a hop sequence adaptation applying to all its Peripherals at the same time is required. However it cannot schedule an *AFH_Instant* until it has informed its active Peripheral, its Peripheral in Hold mode, and its Peripheral in Sniff mode.

If it decides that only some Peripherals require the new adaptation, it need only take those Peripherals into account when scheduling the *AFH_Instant*.

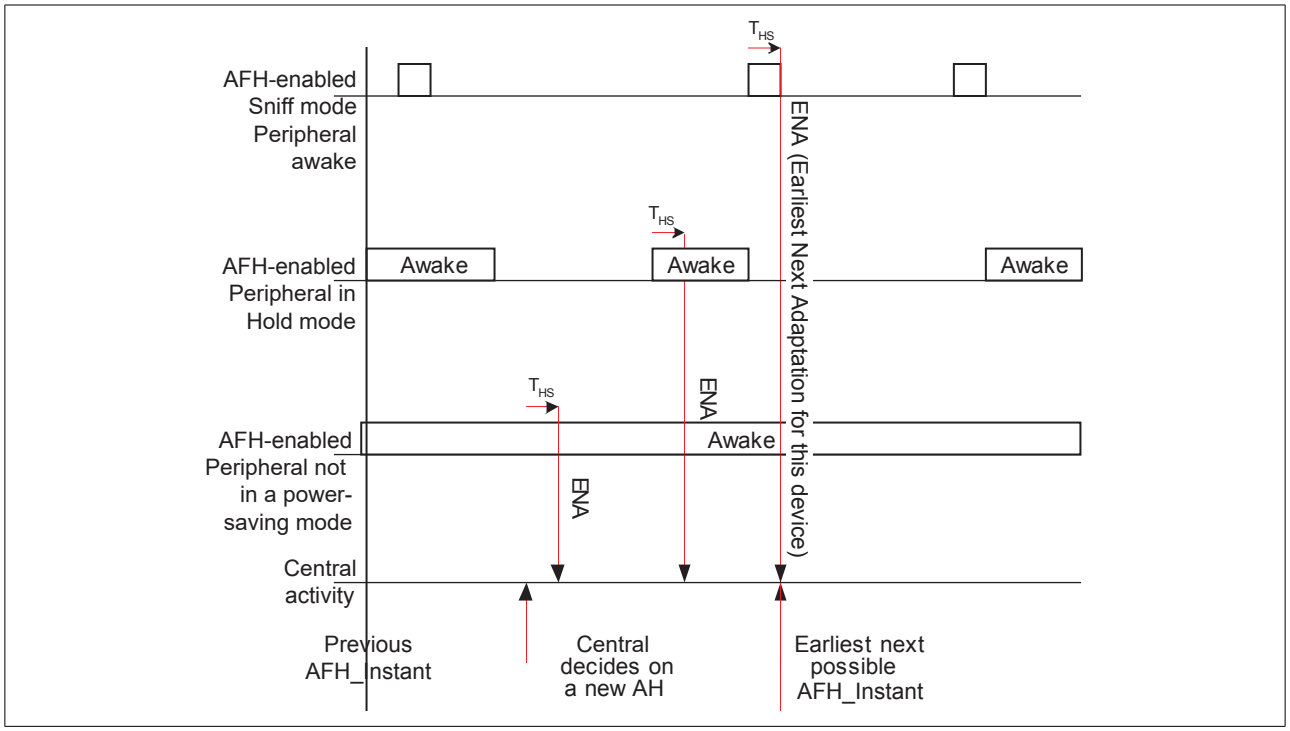



*Figure C.1: Timing constraint on AFH_Instant with Peripherals in Hold and Sniff modes*

## C.2 [This section is no longer used]

## C.3 AFH operation in Sniff mode

Once a Peripheral has been placed in Sniff mode, the Central may periodically change its AHS without taking the Peripheral out of Sniff mode.

## C.4 AFH operation in Hold mode

A Peripheral that is in Hold mode cannot send or receive any LMP messages. Once the Peripheral has left Hold mode the Central may subsequently update the Peripheral's adaptation.

## C.5 AFH operation in Connectionless Peripheral Broadcast

After the Transmitter's BR/EDR Controller notifies the Transmitter's Host that the AFH channel map has changed for the PBD logical link, the Transmitter's Host may restart the synchronization train to allow Receivers to obtain the updated AFH channel map. This is shown in Figure C.2.

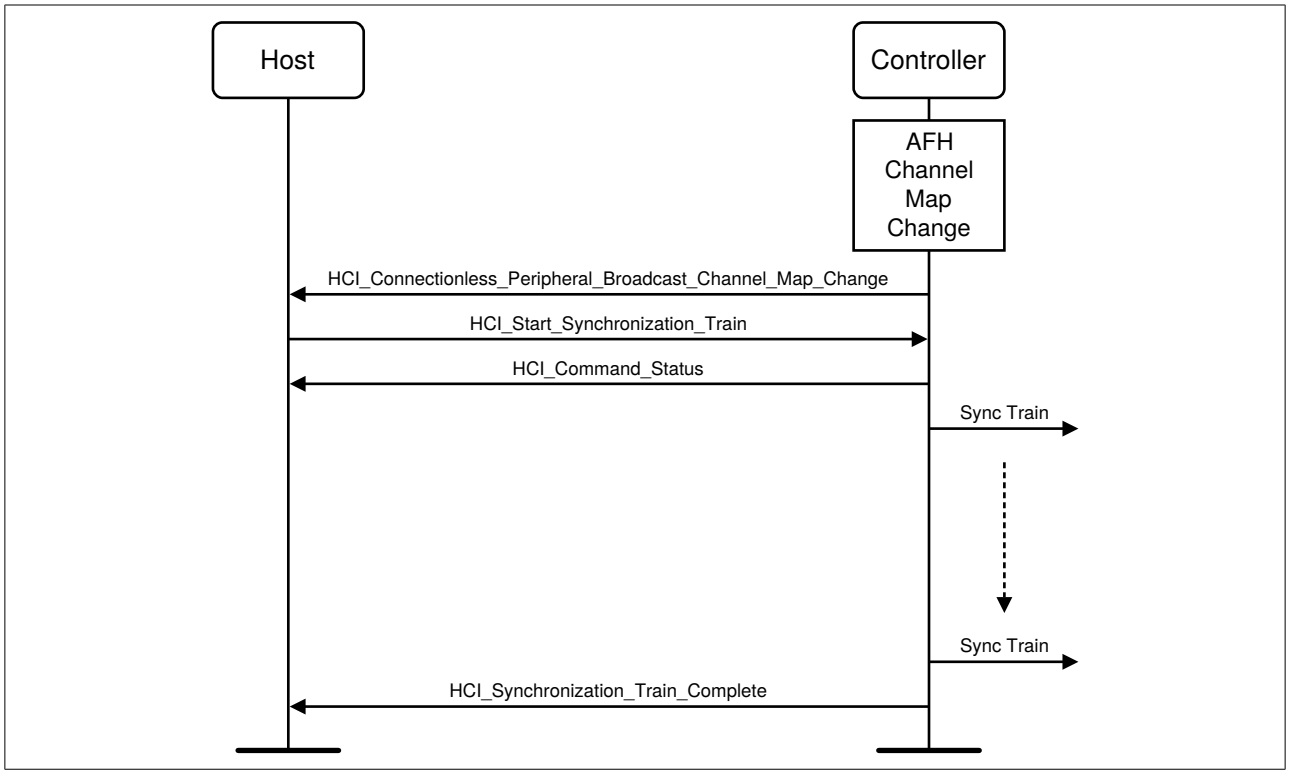



*Figure C.2: AFH map change – Connectionless Peripheral Broadcast Transmitter*

Modification of the AFH channel map will affect Receivers using a different AFH channel map. Depending on the overlap between a Receiver's current AFH channel map and the current AFH channel map of the Transmitter (Central), the Receiver may miss some Connectionless Peripheral Broadcast instants and, in the case of disjoint or nearly disjoint AFH channel maps, the Receiver may lose synchronization. Receivers should monitor the Connectionless Peripheral Broadcast reception rate and obtain the current AFH channel map of the Transmitter (via the synchronization train) if degradation exceeds desired thresholds.

BR/EDR Controller
Part C

# LINK MANAGER PROTOCOL SPECIFICATION

*This Part describes the Link Manager protocol (LMP) which is used for link set-up and control. The signals are interpreted and filtered out by the Link Manager on the receiving side and are not propagated to higher layers.*

# CONTENTS

**1 Introduction ..... 642**

**2 General rules ..... 643**
2.1 Message transport ..... 643
2.2 Synchronization ..... 643
2.3 Packet format ..... 644
2.4 Transactions ..... 645
2.4.1 LMP response timeout ..... 645
2.5 Error handling ..... 646
2.5.1 Transaction collision resolution ..... 647
2.6 Procedure rules ..... 647
2.7 General response messages ..... 648
2.8 LMP message constraints ..... 648

**3 Device features ..... 649**
3.1 General description ..... 649
3.2 Feature definitions ..... 649
3.3 Feature mask definition ..... 654
3.4 Link Manager interoperability policy ..... 657
3.5 Feature requirements ..... 657
3.5.1 [This section is no longer used] ..... 660
3.5.2 [This section is no longer used] ..... 660

**4 Procedure rules ..... 661**
4.1 Connection control ..... 661
4.1.1 Connection establishment ..... 661
4.1.2 Detach ..... 662
4.1.3 Power control ..... 663
4.1.3.1 Enhanced power control ..... 665
4.1.4 Adaptive frequency hopping ..... 666
4.1.4.1 Central enables AFH ..... 667
4.1.4.2 Central disables AFH ..... 668
4.1.4.3 Central updates AFH ..... 668
4.1.4.4 AFH operation in Hold and Sniff modes ..... 669
4.1.5 Channel classification ..... 669
4.1.5.1 Channel classification reporting enabling and disabling ..... 671
4.1.6 Link supervision ..... 671
4.1.7 Channel quality driven data rate change (CQDDR) ..... 672
4.1.8 Quality of service (QoS) ..... 673

4.1.8.1 Central notifies Peripheral of the quality of service ..... 673
4.1.8.2 Device requests new quality of service ..... 674
4.1.9 Paging scheme parameters ..... 674
4.1.9.1 Page mode ..... 675
4.1.9.2 Page Scan mode ..... 675
4.1.10 Control of multi-slot packets ..... 676
4.1.11 Enhanced Data Rate ..... 677
4.1.12 Encapsulated LMP PDUs ..... 678
4.1.12.1 Sending an encapsulated PDU ..... 678
4.1.13 Ping ..... 679
4.1.14 Piconet clock adjustment ..... 680
4.1.14.1 Central coarse adjustment of piconet clock . 681
4.1.14.2 Peripheral request for coarse adjustment of piconet clock ..... 683
4.1.15 Slot Availability Mask ..... 684
4.1.15.1 SAM type 0 submap configuration ..... 686
4.1.15.2 SAM slot map define ..... 687
4.1.15.3 SAM switch sequence ..... 687
4.1.15.4 SAM change during the transmission of a multi-slot packet ..... 688
4.1.15.5 SAM and role switching ..... 688
4.1.15.6 SAM and Sniff mode ..... 688
4.2 Security ..... 688
4.2.1 Authentication ..... 689
4.2.1.1 Claimant has link key (legacy authentication) ..... 689
4.2.1.2 Claimant has no link key (legacy authentication and secure authentication) ... 690
4.2.1.3 Repeated attempts ..... 690
4.2.1.4 Responder has link key (secure authentication) ..... 691
4.2.2 Pairing ..... 692
4.2.2.1 Responder accepts pairing and has a variable PIN ..... 692
4.2.2.2 Responder accepts pairing and has a fixed PIN ..... 693
4.2.2.3 Responder rejects pairing ..... 693
4.2.2.4 Creation of the link key ..... 694
4.2.2.5 Repeated attempts ..... 695
4.2.3 Change link key ..... 695
4.2.4 Change current link key type ..... 696
4.2.4.1 Change to a temporary link key ..... 696

4.2.4.2 Semi-permanent link key becomes current link key ..... 697
4.2.5 Encryption ..... 697
4.2.5.1 Encryption mode ..... 698
4.2.5.2 Encryption key size ..... 699
4.2.5.3 Start encryption ..... 701
4.2.5.4 Stop encryption ..... 701
4.2.5.5 Pause encryption ..... 702
4.2.5.6 Resume encryption ..... 705
4.2.5.7 Change encryption key or random number .. 706
4.2.5.8 Encryption key refresh ..... 706
4.2.6 Request supported encryption key size ..... 707
4.2.7 Secure Simple Pairing ..... 707
4.2.7.1 IO capability exchange ..... 708
4.2.7.2 Public key exchange ..... 710
4.2.7.3 Authentication stage 1 ..... 711
4.2.7.4 Authentication stage 2: DHKey check ..... 719
4.3 Informational requests ..... 721
4.3.1 Timing accuracy ..... 721
4.3.2 Clock offset ..... 722
4.3.3 LMP version ..... 723
4.3.4 Supported features ..... 724
4.3.5 Name request ..... 725
4.4 Role switch ..... 726
4.4.1 Slot offset ..... 726
4.4.2 Role switch ..... 727
4.5 Modes of operation ..... 730
4.5.1 Hold mode ..... 730
4.5.1.1 Central forces Hold mode ..... 730
4.5.1.2 Peripheral forces Hold mode ..... 731
4.5.1.3 Central or Peripheral requests Hold mode ... 731
4.5.2 [This section is no longer used] ..... 732
4.5.3 Sniff mode ..... 732
4.5.3.1 Central or Peripheral requests Sniff mode ... 733
4.5.3.2 Moving a Peripheral from Sniff mode to Active mode ..... 734
4.5.3.3 Sniff subrating ..... 735
4.6 Logical transports ..... 736
4.6.1 SCO logical transport ..... 736
4.6.1.1 Central initiates a SCO link ..... 737
4.6.1.2 Peripheral initiates a SCO link ..... 738
4.6.1.3 Central requests change of SCO parameters ..... 739

4.6.1.4 Peripheral requests change of SCO parameters ..... 739
4.6.1.5 Remove a SCO link ..... 739
4.6.2 eSCO logical transport ..... 739
4.6.2.1 Central initiates an eSCO link ..... 740
4.6.2.2 Peripheral initiates an eSCO link ..... 741
4.6.2.3 Central or Peripheral requests change of eSCO parameters ..... 742
4.6.2.4 Remove an eSCO link ..... 742
4.6.2.5 Rules for the LMP negotiation and renegotiation ..... 743
4.6.2.6 Negotiation state definitions ..... 744
4.7 Test mode ..... 744
4.7.1 Activation and deactivation of Test mode ..... 744
4.7.2 Control of Test mode ..... 745
4.7.3 Summary of Test mode PDUs ..... 746

**5 Summary ..... 749**
5.1 PDU summary ..... 749
5.2 Parameter definitions ..... 757
5.3 LMP encapsulated ..... 768
5.4 Default values ..... 768

**Appendix A Changes to parameter names ..... 770**

# 1 INTRODUCTION

The Link Manager Protocol (LMP) is used to control and negotiate all aspects of the operation of the Bluetooth connection between two devices. This includes the set-up and control of logical transports and logical links, and for control of physical links.

The Link Manager Protocol is used to communicate between the Link Managers (LM) on the two devices. All LMP messages shall apply solely to the physical link and associated logical links and logical transports between the sending and receiving devices.

The protocol is made up of a series of messages which shall be transferred over the ACL-C or APB-C logical link between two devices. LMP messages shall be interpreted and acted-upon by the LM and shall not be directly propagated to higher protocol layers.

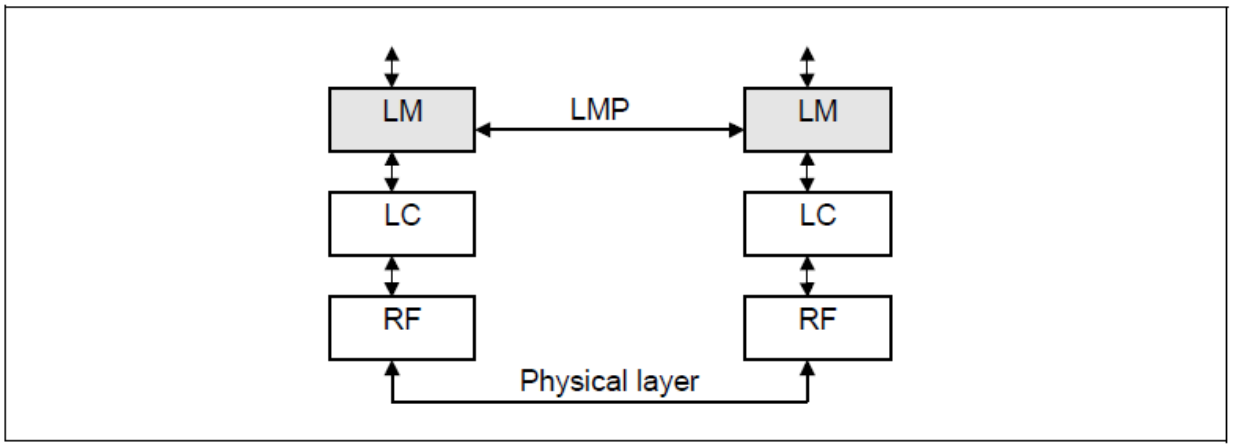



*Figure 1.1: Link Manager Protocol signaling layer*

# 2 GENERAL RULES

## 2.1 Message transport

LMP messages shall be exchanged over the ACL-C logical link that is carried on the default ACL logical transport ([Vol 2] Part B, Section 4.4) or over the APB-C logical link that is carried on the APB logical transport ([Vol 2] Part B, Section 4.6). LMP messages shall be carried on the default ACL logical transport unless specified otherwise in Sections 4 and 5. The ACL-C and APB-C logical links are distinguished from the ACL-U and APB-U logical links (which carry L2CAP and user data) by the Logical Link Identifier (LLID) field carried in the payload header of variable-length packets ([Vol 2] Part B, Section 6.6.2).

The control logical links have a higher priority than other traffic - see [Vol 2] Part B, Section 5.6.

The error detection and correction capabilities of the Baseband ACL logical transport are generally sufficient for the requirements of the LMP. Therefore LMP messages do not contain any additional error detection information beyond what can be realized by means of sanity checks performed on the contents of LMP messages. Any such checks and protections to overcome undetected errors in LMP messages is an implementation matter.

## 2.2 Synchronization

This section explains why many of the LMP message sequences are defined as they are. It does not create any requirements.

LMP messages are carried on the ACL-C and APB-C logical links, which do not guarantee a time to deliver or acknowledge packets. LMP procedures take account of this when synchronizing state changes in the two devices. For example, criteria are defined that specify when a logical transport address (LT_ADDR) may be re-used after it becomes available due to a device leaving the piconet. Other LMP procedures, such as hold or role switch include the Bluetooth clock as a parameter in order to define a fixed synchronization point. The transitions into and out of Sniff mode are protected with a transition mode.

The LC normally attempts to communicate with each Peripheral no less often than every $T_{poll}$ slots (see Section 4.1.8) based on the $T_{poll}$ for that Peripheral.

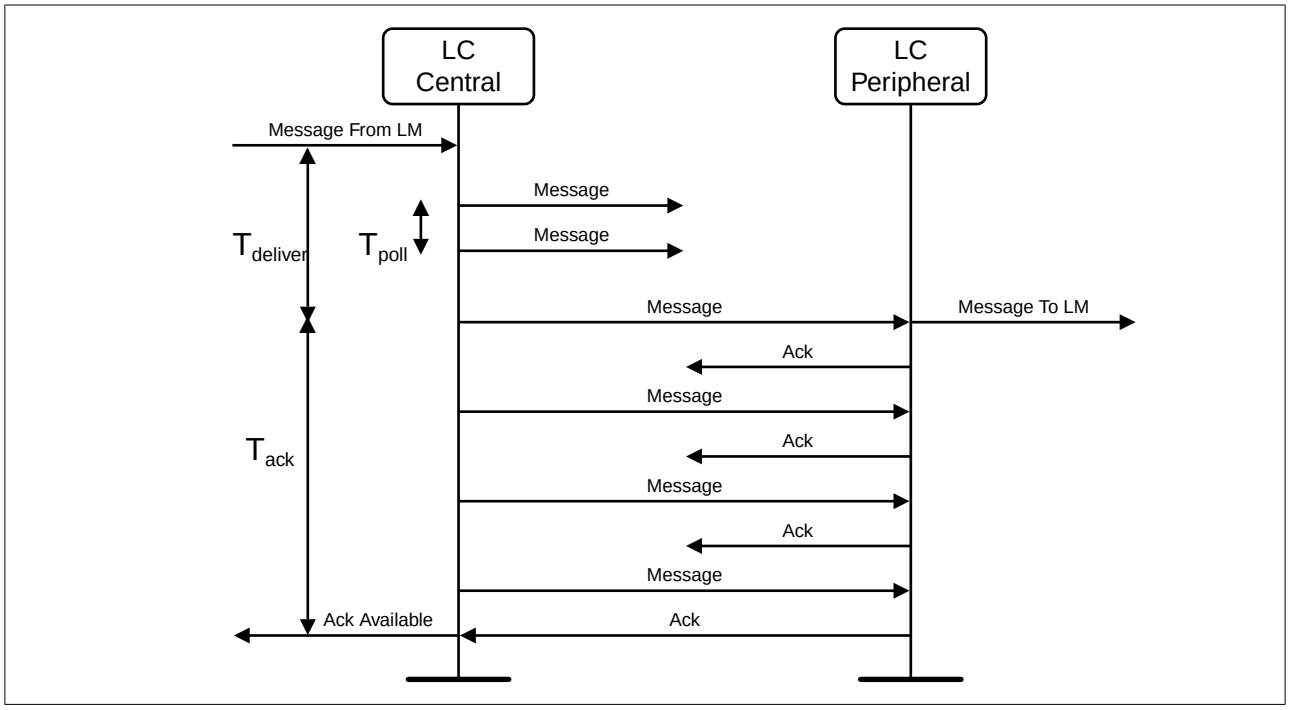



*Figure 2.1: Transmission of a message from Central to Peripheral*[1]

Figure 2.1 illustrates the fundamental problem. It shows the transmission of a packet from the Central to the Peripheral in conditions of heavy interference for illustrative purposes. It is obvious that neither side can determine the value of either $T_{deliver}$ or $T_{ack}$. It is therefore not possible to use simple messages to identify uniquely the instant at which a state change occurs in the other device.

## 2.3 Packet format

Each PDU is assigned either a 7 or a 15 bit opcode used to uniquely identify different types of PDUs, see Table 5.1. The first 7 bits of the opcode and a transaction ID are located in the first byte of the payload body. If the initial 7 bits of the opcode have one of the special escape values 124 to 127 then an additional byte of opcode is located in the second byte of the payload and the combination uniquely identifies the PDU.

The FLOW bit in the packet header is always 1 and shall be ignored on reception.

If the PDU contains one or more parameters these are placed in the payload starting immediately after the opcode, i.e. at byte 2 if the PDU has a 7 bit opcode or byte 3 if the PDU has a 15 bit opcode. The number of bytes used depends on the length of the parameters. The representation of each parameter is determined by its type as specified in [Vol 1] Part E, Section 2.9.

[1]Note the diagram shows the limiting case where the Central transmits the message at intervals of $T_{poll}$. In the case of heavy interference improved performance is gained by transmitting more often.

LMP messages shall be transmitted using DM1 packets, however if an HV1 SCO link is in use and the length of the payload is no greater than 9 bytes then DV packets may be used.

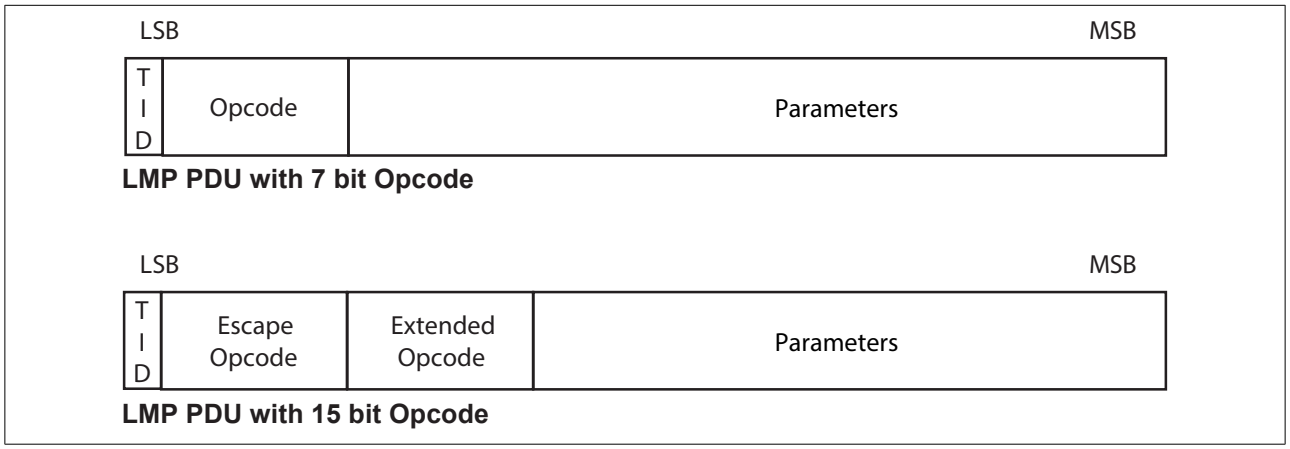



*Figure 2.2: Payload body when LMP PDUs are sent*

## 2.4 Transactions

The LMP operates in terms of transactions. A transaction is a connected set of message exchanges which achieve a particular purpose. All PDUs which form part of the same transaction shall have the same value for the transaction ID which is stored as part of the first byte of the opcode (see Section 2.3).

The transaction ID is in the least significant bit. It shall be 0 if the PDU forms part of a transaction that was initiated by the Central and 1 if the transaction was initiated by the Peripheral.

Each sequence described in Section 4 shall be defined as a transaction. For pairing, see Section 4.2.2, and encryption, see Section 4.2.5, all sequences belonging to each section shall be counted as one transaction and shall use the same transaction ID. For connection establishment, see Section 4.1.1, LMP_HOST_CONNECTION_REQ and the response with LMP_ACCEPTED or LMP_NOT_ACCEPTED shall form one transaction and have the transaction ID of 0. LMP_SETUP_COMPLETE is a stand-alone PDU, which forms a transaction by itself.

For error handling, see Section 2.5, the PDU to be rejected and LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT shall form a single transaction.

### 2.4.1 LMP response timeout

The time between receiving a Baseband packet carrying an LMP PDU and sending a Baseband packet carrying a valid response PDU, according to the procedure rules in

Section 4, shall be less than the LMP Response Timeout. The value of this timeout is 30 seconds. The LMP Response Timeout is applied not only to sequences described in Section 4, but also to the series of the sequences defined as the transactions in Section 4. It shall also be applied to the series of LMP transactions that take place during a period when traffic on the ACL-U logical link is disabled for the duration of the series of LMP transactions, for example during the enabling of encryption.

The LMP Response Timeout shall restart each time an LMP PDU which requires a reply is queued for transmission by the Baseband.

LMP messages sent on the APB-C logical link have special rules and are not subject to the LMP Response Timeout.

## 2.5 Error handling

If the LM receives a PDU with an unknown opcode (e.g. one added in a higher version of the specification), it shall respond with LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Unknown LMP PDU* (0x19). The Opcode parameter shall be the unrecognized opcode.

If the LM receives a PDU which contains valid parameters followed by extra data, it shall either ignore the extra data or shall respond with LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Invalid LMP Parameters* (0x1E).

If the LM receives a PDU which is too short to hold all the parameters, it shall either continue with implementation-specific values for the missing or damaged parameters or shall respond with LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Invalid LMP Parameters* (0x1E).

If the LM receives a PDU with the correct length but with invalid parameters, it shall respond with LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Invalid LMP Parameters* (0x1E).

If the maximum response time (see Section 2.4.1) is exceeded or if a link loss is detected (see [Vol 2] Part B, Section 3.1), the party that waits for the response shall conclude that the procedure has terminated unsuccessfully.

Erroneous LMP messages can be caused by errors on the channel or systematic errors at the transmit side. To detect the latter case, the LM should monitor the number of erroneous messages and disconnect if it exceeds a threshold, which is implementation-dependent.

When the LM receives a PDU that is not allowed, and the PDU normally expects a PDU reply, for example LMP_HOST_CONNECTION_REQ, the LM shall return PDU LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *LMP*

*PDU Not Allowed* (0x24). If the PDU normally doesn't expect a reply, for example LMP_SRES or LMP_TEMP_KEY, the PDU shall be ignored.

If the LM recognizes the PDU as optional but unsupported then it shall reply with LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Unsupported LMP Feature* (0x1A) if the PDU would normally generate a reply; otherwise it shall ignore the PDU and no reply is generated.

### 2.5.1 Transaction collision resolution

Since LMP PDUs are not interpreted in real time, collision situations can occur where both LMs initiate the same procedure and both cannot be completed. In this situation, the Central shall reject the Peripheral-initiated procedure by sending LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *LMP Error Transaction Collision / LL Procedure Collision* (0x23). The Central-initiated procedure shall then be completed.

Collision situations can also occur where both LMs initiate different procedures and both cannot be completed. In this situation, the Central shall reject the Peripheral-initiated procedure by sending LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT with the Error_Code *Different Transaction Collision* (0x2A). The Central-initiated procedure shall then be completed.

## 2.6 Procedure rules

Each procedure is described and depicted with a sequence diagram. The symbols in Figure 2.3 are used in the sequence diagrams:

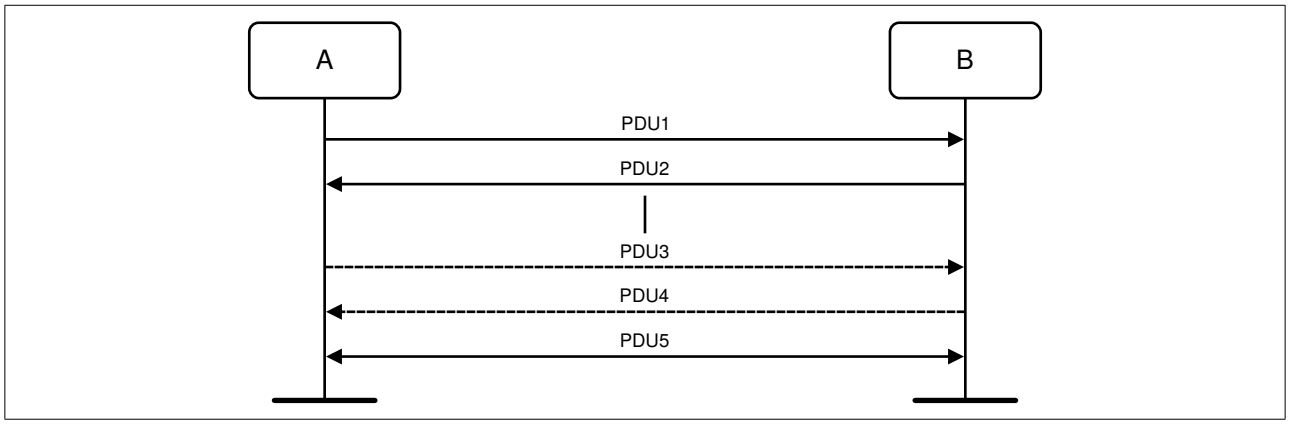



*Figure 2.3: Symbols used in sequence diagrams*

PDU1 is a PDU sent from A to B. PDU2 is a PDU sent from B to A. PDU3 is a PDU that is optionally sent from A to B. PDU4 is a PDU that is optionally sent from B to A. PDU5 is a PDU sent from either A or B. A vertical line indicates that more PDUs can optionally be sent.

## 2.7 General response messages

The PDUs LMP_ACCEPTED, LMP_ACCEPTED_EXT, LMP_NOT_ACCEPTED, and LMP_NOT_ACCEPTED_EXT are used as response messages to other PDUs in a number of different procedures. LMP_ACCEPTED or LMP_ACCEPTED_EXT includes the opcode of the message which is accepted. LMP_NOT_ACCEPTED or LMP_NOT_ACCEPTED_EXT includes the opcode of the message which is not accepted and the error code why it is not accepted.

LMP_ACCEPTED_EXT and LMP_NOT_ACCEPTED_EXT shall be used when the opcode is 15 bits in length. LMP_ACCEPTED and LMP_NOT_ACCEPTED shall be used when the opcode is 7 bits in length.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_ACCEPTED | Opcode |
| M | LMP_NOT_ACCEPTED | Opcode<br>Error_Code |
| O | LMP_ACCEPTED_EXT | Escape_Opcode<br>Extended_Opcode |
| O | LMP_NOT_ACCEPTED_EXT | Escape_Opcode<br>Extended_Opcode<br>Error_Code |

*Table 2.1: General response messages*

## 2.8 LMP message constraints

The following principles are used in the design of LMP.

- No LMP message exceeds the maximum payload length of a single DM1 packet i.e. 17 bytes in length ([Vol 2] Part B, Section 6.5.4.1).
- All LMP messages are of fixed length.
- The LMP version number is not used to indicate the presence or absence of functionality.

# 3 DEVICE FEATURES

## 3.1 General description

Each PDU is either Mandatory or Optional as defined by the M/O field in the tables of Section 4. An M in this field indicates that support for the PDU is mandatory. An O in this field indicates that support for the PDU is optional and it is followed by the number(s) of the feature(s) involved in brackets.

Some features have associated LMP feature bits. Support of these features may be required by Section 3.5 but the LM still considers them to be optional since it must interoperate with devices which do not support them.

The LM does not need to be able to transmit a PDU which is optional. Support of optional PDUs is indicated by a device's features mask. The features mask can be read (see Section 4.3.4). An LM shall not send or be sent any PDU which is incompatible with the features signaled in its or its peer's features mask, as detailed in Section 3.2.

The set of features supported by the LM shall not change while the Controller has a connection to another device. A peer device may cache the device's feature mask or extended feature mask, or the LM may cache a peer's feature mask or extended feature mask, during a connection.

## 3.2 Feature definitions

| Feature | Definition |
|---|---|
| 3-slot Enhanced Data Rate ACL packets | This feature indicates whether the device supports the transmission and reception of three-slot Enhanced Data Rate packets on the ACL-U logical link. |
| 3-slot Enhanced Data Rate eSCO packets | This feature indicates whether the device supports the transmission and reception of 3-slot Enhanced Data Rate packets for the transport of traffic on the eSCO logical transport. |
| 3 slot packets | This feature indicates whether the device supports the transmission and reception of both DM3 and DH3 packets for the transport of traffic on the ACL-U logical link. |
| 5-slot Enhanced Data Rate ACL packets | This feature indicates whether the device supports the transmission and reception of 5-slot Enhanced Data Rate packets for the transport of traffic on the ACL-U logical link. |
| 5 slot packets | This feature indicates whether the device supports the transmission and reception of both DM5 and DH5 packets for the transport of traffic on the ACL-U logical link. |
| AFH capable Central | This indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a Central as defined in [Vol 2] Part B, Section 2.3 using the LMP sequences defined in Section 4.1.4. |

| Feature | Definition |
|---|---|
| AFH capable Peripheral | This feature indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a Peripheral as defined in [Vol 2] Part B, Section 2.3 using the LMP sequences defined in Section 4.1.4. |
| AFH classification Central | This feature indicate whether the device is able to support the AFH classification mechanism as a Central as defined in [Vol 2] Part B, Section 8.6.8 using the LMP sequences defined in Section 4.1.5. |
| AFH classification Peripheral | This feature indicates whether the device is able to support the AFH classification mechanism as a Peripheral as defined in [Vol 2] Part B, Section 8.6.8 using the LMP sequences defined in Section 4.1.5. |
| A-law log synchronous data | This feature indicates whether the device is capable of supporting A-law Log format data as defined in [Vol 2] Part B, Section 9.1 on the SCO and eSCO logical transports. |
| Broadcast encryption | This feature indicates whether the device is capable of supporting broadcast encryption as defined in [Vol 2] Part H, Section 4.2 and also the LMP sequences defined in Section 4.2.6 and Section 4.2.4. |
| Channel Quality Driven Data Rate | This feature indicates whether the LM is capable of recommending a packet type (or types) depending on the channel quality using the LMP sequences defined in Section 4.1.7. |
| Coarse Clock Adjustment | This feature indicates whether the device is able to support coarse clock adjustments using the LMP sequences defined in Section 4.1.14. |
| Connectionless Peripheral Broadcast - Receiver Operation | This feature indicates whether the device supports Connectionless Peripheral Broadcast as Receiver. |
| Connectionless Peripheral Broadcast - Transmitter Operation | This feature indicates whether the device supports Connectionless Peripheral Broadcast as Transmitter. |
| CVSD synchronous data | This feature indicates whether the device is capable of supporting CVSD format data as defined in [Vol 2] Part B, Section 9.2 on the SCO and eSCO logical transports. |
| Encapsulated PDU | This feature indicates whether the device is capable of supporting the Encapsulated PDU mechanism as defined in Section 4.1.12.1. |
| Encryption | This feature indicates whether the device supports the encryption of packet contents using the LMP sequence defined in Section 4.2.5. |
| Enhanced Data Rate ACL 2 Mb/s mode | This feature indicates whether the device supports the transmission and reception of 2-DH1 packets for the transport of traffic on the ACL-U logical link. |
| Enhanced Data Rate ACL 3 Mb/s mode | This feature indicates whether the device supports the transmission and reception of 3-DH1 packets for the transport of traffic on the ACL-U logical link. |

| Feature | Definition |
|---|---|
| Enhanced Data Rate eSCO 2 Mb/s mode | This feature indicates whether the device supports the transmission and reception of 2-EV3 packets for the transport of traffic on the eSCO logical transport. |
| Enhanced Data Rate eSCO 3 Mb/s mode | This feature indicates whether the device supports the transmission and reception of 3-EV3 packets for the transport of traffic on the eSCO logical transport. |
| Enhanced power control | This feature indicates whether the device is able to support enhanced power control using the LMP sequences defined in Section 4.1.3.1. |
| Erroneous Data Reporting | This feature indicates whether the device is able to support the Packet_Status_Flag and the HCI commands HCI_Write_Default_Erroneous_Data_Reporting and HCI_Read_Default_Erroneous_Data_Reporting. |
| EV4 packets | This feature indicates whether the device is capable of supporting the EV4 packet type defined in [Vol 2] Part B, Section 6.5.3.2 on the eSCO logical transport. |
| EV5 packets | This feature indicates whether the device is capable of supporting the EV5 packet type defined in [Vol 2] Part B, Section 6.5.3.3 on the eSCO logical transport. |
| Extended features | This feature indicates whether the device is able to support the extended features mask using the LMP sequences defined in Section 4.3.4. |
| Extended Inquiry Response | This feature indicates whether the device supports extended inquiry response as defined in [Vol 2] Part B, Section 8.4.3. |
| Extended SCO link | This feature indicates whether the device is able to support the eSCO logical transport as defined in [Vol 2] Part B, Section 5.5 and the EV3 packet defined in [Vol 2] Part B, Section 6.5.3.1 using the LMP sequences defined in Section 4.6.2. |
| Flow control lag | This is defined as the total amount of ACL-U data that can be sent following the receipt of a valid payload header with the payload header flow bit set to 0 and is in units of 256 bytes. See further in [Vol 2] Part B, Section 6.6.2. |
| Generalized interlaced scan | This feature indicates whether the device is able to support the generalized interlaced scan mechanism described in [Vol 2] Part B, Section 8.3.1 and Section 8.4.1.<br>The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| Hold mode | This feature indicates whether the device is able to support Hold mode as defined in [Vol 2] Part B, Section 8.8 using the LMP sequences defined in Section 4.5.1. |
| HV2 packets | This feature indicates whether the device is capable of supporting the HV2 packet type as defined in [Vol 2] Part B, Section 6.5.2.2 on the SCO logical transport. |
| HV3 packets | This feature indicates whether the device is capable of supporting the HV3 packet type as defined in [Vol 2] Part B, Section 6.5.2.3 on the SCO logical transport. |
| Inquiry Response Notification event | This feature bit indicates whether the device supports sending the HCI_Inquiry_Response_Notification event to the Host. |

| Feature | Definition |
|---|---|
| Interlaced inquiry scan | This feature indicates whether the device is capable of supporting the interlaced inquiry scan mechanism as defined in [Vol 2] Part B, Section 8.4.1. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| Interlaced page scan | This feature indicates whether the device is capable of supporting the interlaced page scan mechanism as defined in [Vol 2] Part B, Section 8.3.1. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| LE Supported (Controller) | This feature indicates whether the Controller supports LE.<br>The local Host uses this feature bit to determine whether the Controller supports LE.<br>A remote device does not use this feature bit. |
| LE Supported (Host) | This feature indicates that the Host supports LE.<br>The local Host sets this feature bit to indicate to a remote device that the local device is capable of supporting LE.<br>A remote Host uses this feature bit to determine whether an LE connection to the peer device is possible. |
| Link Supervision Timeout Changed event | This feature bit indicates whether the device supports the sending the HCI_Link_Supervision_Timeout_Changed event to the Host. |
| μ-law log synchronous data | This feature indicates whether the device is capable of supporting μ-law Log format data as defined in [Vol 2] Part B, Section 9.1 on the SCO and eSCO logical transports. |
| Non-flushable Packet Boundary Flag | This feature indicates that the device supports the capability to correctly handle HCI ACL Data packets with a Packet_Boundary_Flag value of 00 (Non-Automatically-Flushable packet). |
| Paging parameter negotiation | This feature indicates whether the LM is capable of supporting the signaling of changes in the paging scheme as defined in Section 4.1.9. |
| Pause Encryption | When this feature bit is enabled on both sides, then the encryption pause and resume sequences shall be used. If either side does not support the Pause Encryption feature bit, then the encryption pause and resume sequences shall not be used. |
| Ping | This feature indicates whether the device supports the transmission and reception of ping messages. |
| Power control | This feature indicates whether the device is capable of adjusting its transmission power. This feature indicates the ability to receive the LMP_INCR_POWER and LMP_DECR_POWER PDUs and transmit the LMP_MAX_POWER and LMP_MIN_POWER PDUs, using the LMP sequences defined in Section 4.1.3. These sequences may be used even if the remote device does not support the power control feature, as long as it supports the Power control requests feature. |

| Feature | Definition |
|---|---|
| Power control requests | This feature indicates whether the device is capable of determining if the transmit power level of the other device should be adjusted and will send the LMP_INCR_POWER and LMP_DECR_POWER PDUs to request the adjustments. It indicates the ability to receive the LMP_MAX_POWER and LMP_MIN_POWER PDUs, using the LMP sequences defined in Section 4.1.3. These sequences may be used even if the remote device does not support the RSSI feature, as long as it supports the power control feature. |
| Role switch | This feature indicates whether the device supports the exchange of Central and Peripheral roles as defined by [Vol 2] Part B, Section 8.6.5 using the LMP sequence defined in Section 4.4.2. |
| RSSI with inquiry results | This feature indicates whether the device is capable of reporting the RSSI with inquiry results as defined in [Vol 2] Part B, Section 8.4.2. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| SCO link | This feature shall indicate whether the device is able to support the SCO logical transport as defined in [Vol 2] Part B, Section 4.3, the HV1 packet defined in [Vol 2] Part B, Section 6.5.2.1 and receiving the DV packet defined in [Vol 2] Part B, Section 6.5.2.4 using the LMP sequence defined in Section 4.6.1. |
| Secure Connections (Controller Support) | This feature indicates whether the Controller is able to support AES-CCM, the P-256 Elliptic Curve for Secure Simple Pairing, and enhanced authentication using the LMP sequences defined in Section 4.2.7. |
| Secure Connections (Host Support) | This feature indicates whether the Host is capable of supporting Secure Connections.<br>If HCI is supported, this bit shall be set if the HCI_Write_Secure_Connections_Host_Support command has been issued by the Host. When HCI is not supported, this bit may be set using a vendor specific mechanism. |
| Secure Simple Pairing (Controller Support) | This feature indicates whether the Link Manager is capable of supporting Secure Simple Pairing as defined in Section 4.2.7. |
| Secure Simple Pairing (Host Support) | This feature indicates whether the Host is capable of supporting Secure Simple Pairing as defined in Section 4.2.7.<br>If HCI is supported, this bit shall be set if the HCI_Write_Simple_Pairing_Mode command has been issued to enable the Secure Simple Pairing feature. When HCI is not supported, this bit may be set using a vendor specific mechanism. |
| Simultaneous LE and BR/EDR to Same Device Capable (Controller) | This feature indicates that the Controller supports simultaneous LE and BR/EDR links to the same remote device.<br>The local Host uses this feature bit to determine whether the Controller is capable of supporting simultaneous LE and BR/EDR connections to a remote device.<br>A remote device does not use this feature bit. |
| Slot Availability Mask | This feature indicates whether the device is able to support Slot Availability Mask using the LMP sequences defined in Section 4.1.15. |
| Slot offset | This feature indicates whether the LM supports the transfer of the slot offset using the LMP sequence defined in Section 4.4.1. |

| Feature | Definition |
|---|---|
| Sniff mode | This feature indicates whether the device is able to support Sniff mode as defined in [Vol 2] Part B, Section 8.7 using the LMP sequences defined in Section 4.5.3. |
| Sniff subrating | This feature indicates whether the device is able to support sniff subrating as defined in [Vol 2] Part B, Section 8.7.2 using the LMP sequences defined in Section 4.5.3.3. |
| Synchronization Scan | This feature indicates whether the device supports Synchronization Scan as Peripheral. |
| Synchronization Train | This feature indicates whether the device supports Synchronization Train as Central. |
| Timing accuracy | This feature indicates whether the LM supports requests for timing accuracy using the LMP sequence defined in Section 4.3.1. |
| Train nudging | This feature indicates whether the device is able to support the train nudging mechanism described in [Vol 2] Part B, Section 8.3.2 and Section 8.4.2.<br>The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| Transparent synchronous data | This feature indicates whether the device is capable of supporting transparent synchronous data as defined in [Vol 2] Part B, Section 6.4.3 on the SCO and eSCO logical transports. |
| Variable Inquiry TX Power Level | This feature indicates whether the device is capable of setting the TX power level for Inquiry. |

*Table 3.1: Feature definitions*

## 3.3 Feature mask definition

The features are represented as a bit mask when they are transferred in LMP messages. For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. The single exception is the flow control lag which is coded as a 3 bit field with the least significant bit in byte 2 bit 4 and the most significant bit in byte 2 bit 6. All unassigned feature bits are reserved for future use.

| No. | Supported feature | Byte | Bit |
|---|---|---|---|
| 0 | 3 slot packets | 0 | 0 |
| 1 | 5 slot packets | 0 | 1 |
| 2 | Encryption | 0 | 2 |
| 3 | Slot offset | 0 | 3 |
| 4 | Timing accuracy | 0 | 4 |
| 5 | Role switch | 0 | 5 |
| 6 | Hold mode | 0 | 6 |
| 7 | Sniff mode | 0 | 7 |

| No. | Supported feature | Byte | Bit |
|---|---|---|---|
| 8 | Previously used | 1 | 0 |
| 9 | Power control requests | 1 | 1 |
| 10 | Channel quality driven data rate (CQDDR) | 1 | 2 |
| 11 | SCO link | 1 | 3 |
| 12 | HV2 packets | 1 | 4 |
| 13 | HV3 packets | 1 | 5 |
| 14 | μ-law log synchronous data | 1 | 6 |
| 15 | A-law log synchronous data | 1 | 7 |
| 16 | CVSD synchronous data | 2 | 0 |
| 17 | Paging parameter negotiation | 2 | 1 |
| 18 | Power control | 2 | 2 |
| 19 | Transparent synchronous data | 2 | 3 |
| 20 | Flow control lag (least significant bit) | 2 | 4 |
| 21 | Flow control lag (middle bit) | 2 | 5 |
| 22 | Flow control lag (most significant bit) | 2 | 6 |
| 23 | Broadcast Encryption | 2 | 7 |
| 24 | Reserved for future use | 3 | 0 |
| 25 | Enhanced Data Rate ACL 2 Mb/s mode | 3 | 1 |
| 26 | Enhanced Data Rate ACL 3 Mb/s mode | 3 | 2 |
| 27 | Enhanced inquiry scan Note 1 | 3 | 3 |
| 28 | Interlaced inquiry scan | 3 | 4 |
| 29 | Interlaced page scan | 3 | 5 |
| 30 | RSSI with inquiry results | 3 | 6 |
| 31 | Extended SCO link (EV3 packets) | 3 | 7 |
| 32 | EV4 packets | 4 | 0 |
| 33 | EV5 packets | 4 | 1 |
| 34 | Reserved for future use | 4 | 2 |
| 35 | AFH capable Peripheral | 4 | 3 |
| 36 | AFH classification Peripheral | 4 | 4 |
| 37 | Previously used | 4 | 5 |
| 38 | LE Supported (Controller) | 4 | 6 |
| 39 | 3-slot Enhanced Data Rate ACL packets | 4 | 7 |
| 40 | 5-slot Enhanced Data Rate ACL packets | 5 | 0 |

| No. | Supported feature | Byte | Bit |
|---|---|---|---|
| 41 | Sniff subrating | 5 | 1 |
| 42 | Pause encryption | 5 | 2 |
| 43 | AFH capable Central | 5 | 3 |
| 44 | AFH classification Central | 5 | 4 |
| 45 | Enhanced Data Rate eSCO 2 Mb/s mode | 5 | 5 |
| 46 | Enhanced Data Rate eSCO 3 Mb/s mode | 5 | 6 |
| 47 | 3-slot Enhanced Data Rate eSCO packets | 5 | 7 |
| 48 | Extended Inquiry Response | 6 | 0 |
| 49 | Simultaneous LE and BR/EDR to Same Device Capable (Controller) | 6 | 1 |
| 50 | Reserved for future use | 6 | 2 |
| 51 | Secure Simple Pairing (Controller Support) | 6 | 3 |
| 52 | Encapsulated PDU | 6 | 4 |
| 53 | Erroneous Data Reporting | 6 | 5 |
| 54 | Non-flushable Packet Boundary Flag | 6 | 6 |
| 55 | Reserved for future use | 6 | 7 |
| 56 | HCI_Link_Supervision_Timeout_Changed event | 7 | 0 |
| 57 | Variable Inquiry TX Power Level | 7 | 1 |
| 58 | Enhanced Power Control | 7 | 2 |
| 59 | Reserved for future use | 7 | 3 |
| 60 | Reserved for future use | 7 | 4 |
| 61 | Reserved for future use | 7 | 5 |
| 62 | Reserved for future use | 7 | 6 |
| 63 | Extended features | 7 | 7 |

*Table 3.2: Feature mask page 0 definitions*

Note [1] Feature bit 27 (“Enhanced Inquiry Scan”) is no longer used in the specification. Devices may set this bit but are not required to.

| No. | Supported Feature | Byte | Bit |
|---|---|---|---|
| 64 | Secure Simple Pairing (Host Support) | 0 | 0 |
| 65 | LE Supported (Host) | 0 | 1 |
| 66 | Previously used | 0 | 2 |
| 67 | Secure Connections (Host Support) | 0 | 3 |

*Table 3.3: Feature mask page 1 definitions*

| No. | Supported Feature | Byte | Bit |
|---|---|---|---|
| 128 | Connectionless Peripheral Broadcast – Transmitter Operation | 0 | 0 |
| 129 | Connectionless Peripheral Broadcast – Receiver Operation | 0 | 1 |
| 130 | Synchronization Train | 0 | 2 |
| 131 | Synchronization Scan | 0 | 3 |
| 132 | HCI_Inquiry_Response_Notification event | 0 | 4 |
| 133 | Generalized interlaced scan | 0 | 5 |
| 134 | Coarse Clock Adjustment | 0 | 6 |
| 135 | Reserved for future use | 0 | 7 |
| 136 | Secure Connections (Controller Support) | 1 | 0 |
| 137 | Ping | 1 | 1 |
| 138 | Slot Availability Mask | 1 | 2 |
| 139 | Train nudging | 1 | 3 |

*Table 3.4: Feature mask page 2 definitions*

## 3.4 Link Manager interoperability policy

Link managers of any version shall interpret using the lowest common subset of functionality by reading the LMP features mask (defined in Table 3.2).

An optional LMP PDU shall only be sent to a device if the corresponding feature bit is set in its feature mask. The exception to this are certain PDUs (see Section 4.1.1) which can be sent before the features mask is requested.

Note: A higher version device with a restricted feature set is indistinguishable from a lower version device with the same features.

## 3.5 Feature requirements

Note: In the tables in this section, numbers in parentheses before feature names are the corresponding feature bit numbers (see Section 3.3).

Any device supporting any feature with a feature bit number greater than or equal to 64 shall support Extended features and shall set feature bit 63.

The features listed in Table 3.5 are mandatory in this version of the specification (see Section 3.1) and these feature bits shall be set.

| Feature |
|---|
| (2) Encryption |
| (42) Pause Encryption |
| (51) Secure Simple Pairing (Controller Support) |
| (52) Encapsulated PDU |

*Table 3.5: Mandatory features*

For each row of Table 3.6, either every feature named in that row shall be supported or none of the features named in that row shall be supported.

| Features | |
|---|---|
| (7) Sniff mode | (41) Sniff subrating |

*Table 3.6: Mutually supporting features*

For each row of Table 3.7, not more than one feature in that row shall be supported.

| Features | |
|---|---|
| (23) Broadcast encryption | (134) Coarse clock adjustment |

*Table 3.7: Mutually exclusive features*

For each row of Table 3.8, if the feature named in the first column is supported then the feature named in the second column shall be supported.

| Feature | Required feature |
|---|---|
| (5) Role switch | (3) Slot offset |
| (12) HV2 packets | (11) SCO link |
| (13) HV3 packets | (11) SCO link |
| (14) μ-law log synchronous data | (11) SCO link *or*<br>(31) Extended SCO link (EV3 packets) |
| (15) A-law log synchronous data | (11) SCO link *or*<br>(31) Extended SCO link (EV3 packets) |
| (16) CVSD log synchronous data | (11) SCO link *or*<br>(31) Extended SCO link (EV3 packets) |
| (19) Transparent synchronous data | (11) SCO link *or*<br>(31) Extended SCO link (EV3 packets) |
| (26) Enhanced Data Rate ACL 3 Mb/s mode | (25) Enhanced Data Rate ACL 2 Mb/s mode |
| (32) EV4 packets | (31) Extended SCO link (EV3 packets) |
| (33) EV5 packets | (31) Extended SCO link (EV3 packets) |

| Feature | Required feature |
|---|---|
| (36) AFH classification Peripheral | (35) AFH capable Peripheral |
| (39) 3-slot Enhanced Data Rate ACL packets | (25) Enhanced Data Rate ACL 2 Mb/s mode |
| (40) 5-slot Enhanced Data Rate ACL packets | (25) Enhanced Data Rate ACL 2 Mb/s mode |
| (44) AFH classification Central | (43) AFH capable Central |
| (45) Enhanced Data Rate eSCO 2 Mb/s mode | (31) Extended SCO link (EV3 packets) |
| (46) Enhanced Data Rate eSCO 3 Mb/s mode | (45) Enhanced Data Rate eSCO 2 Mb/s mode |
| (47) 3-slot Enhanced Data Rate eSCO packets | (45) Enhanced Data Rate eSCO 2 Mb/s mode |
| (48) Extended inquiry response | (30) RSSI with inquiry results |
| (49) Simultaneous LE and BR/EDR to Same Device Capable (Controller) | (38) LE Supported (Controller) |
| (53) Erroneous data reporting | (11) SCO link *or*<br>(31) Extended SCO link (EV3 packets) |
| (58) Enhanced power control | (9) Power control requests *and*<br>(18) Power control |
| (65) LE Supported (Host) | (38) LE Supported (Controller) |
| (67) Secure Connections (Host Support) | (64) Secure Simple Pairing (Host Support) *and*<br>(136) Secure Connections (Controller Support) |
| (128) Connectionless Peripheral Broadcast – Transmitter Operation | (130) Synchronization Train |
| (129) Connectionless Peripheral Broadcast – Receiver Operation | (131) Synchronization Scan |
| (133) Generalized interlaced scan | (28) Interlaced inquiry scan *or*<br>(29) Interlaced page scan |
| (134) Coarse clock adjustment | (35) AFH capable Peripheral *and*<br>(43) AFH capable Central *and*<br>(130) Synchronization Train *and*<br>(131) Synchronization Scan |
| (136) Secure Connections (Controller Support) | (137) Ping |

*Table 3.8: Features that require other features*

For each row of Table 3.9, the feature in the first column shall be supported if and only if the implementation supports HCI and the HCI command or event named in the second column.

| Feature | HCI command or event |
|---|---|
| (53) Erroneous Data Reporting | HCI_Write_Default_Erroneous_Data_Reporting |
| (56) Link Supervision Timeout Changed event | HCI_Link_Supervision_Timeout_Changed |
| (132) Inquiry Response Notification event | HCI_Inquiry_Response_Notification |

*Table 3.9: Features relating to HCI commands and events*

### 3.5.1 [This section is no longer used]

### 3.5.2 [This section is no longer used]

# 4 PROCEDURE RULES

## 4.1 Connection control

### 4.1.1 Connection establishment

After the paging procedure, LMP procedures for clock offset request, LMP version, supported features, name request and detach may then be initiated.

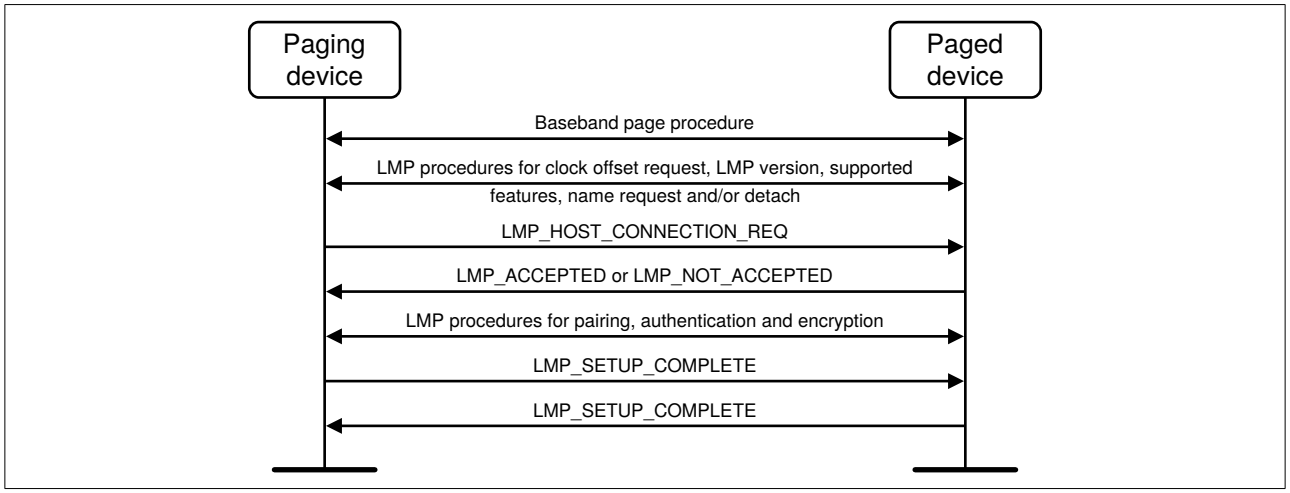



*Figure 4.1: Connection establishment*

When the paging device needs to create a connection involving layers above LM, it sends an LMP_HOST_CONNECTION_REQ PDU. When the other side receives this message, the Host is informed about the incoming connection. The remote device can accept or reject the connection request by sending an LMP_ACCEPTED PDU or an LMP_NOT_ACCEPTED PDU. Alternatively, if the Peripheral needs a role switch, see Section 4.4.2, it sends an LMP_SLOT_OFFSET PDU and LMP_SWITCH_REQ PDU after it has received an LMP_HOST_CONNECTION_REQ PDU. If the role switch fails the LM shall continue with the creation of the connection unless this cannot be supported due to limited resources in which case the connection shall be terminated with an LMP_DETACH PDU with Error_Code *Remote Device Terminated Connection due to Low Resources* (0x14). When the role switch has been successfully completed, the old Peripheral will reply with an LMP_ACCEPTED PDU or an LMP_NOT_ACCEPTED PDU to the LMP_HOST_CONNECTION_REQ PDU (with the transaction ID set to 0).

If the paging device receives an LMP_NOT_ACCEPTED PDU in response to LMP_HOST_CONNECTION_REQ it shall immediately disconnect the link using the mechanism described in Section 4.1.2.

If the LMP_HOST_CONNECTION_REQ PDU is accepted, LMP security procedures (pairing, authentication and encryption) may be invoked. When a device is not going to initiate any more security procedures during connection establishment it sends an LMP_SETUP_COMPLETE PDU. When both devices have sent LMP_SETUP_COMPLETE PDUs the traffic can be transferred on the BR/EDR ACL logical transport.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_HOST_CONNECTION_REQ | *none* |
| M | LMP_SETUP_COMPLETE | *none* |

*Table 4.1: Connection establishment PDU*

### 4.1.2 Detach

The connection between two Bluetooth devices may be detached anytime by the Central or the Peripheral. An Error_Code parameter is included in the message to inform the other party of why the connection is detached.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_DETACH | Error_Code |

*Table 4.2: Detach PDU*

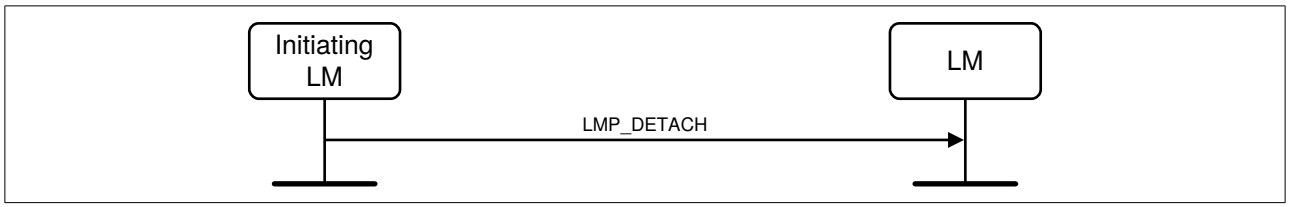
The initiating LM shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). The initiating LM then queues the LMP_DETACH for transmission and it shall start a timer for $6 \times T_{poll}$ slots where $T_{poll}$ is the poll interval for the connection. If the initiating LM receives the Baseband acknowledgment before the timer expires it starts a timer for $3 \times T_{poll}$ slots. When this timer expires, and if the initiating LM is the Central, the LT_ADDR(s) may be re-used immediately. If the initial timer expires then the initiating LM drops the link and starts a timer for $T_{linksupervisiontimeout}$ slots after which the LT_ADDR(s) may be re-used if the initiating LM is the Central.

When the receiving LM receives the LMP_DETACH, it shall start a timer for $6 \times T_{poll}$ slots if it is the Central and $3 \times T_{poll}$ if it is the Peripheral. On timer expiration, the link shall be detached and, if the receiving LM is the Central, the LT_ADDR(s) may be re-used immediately. If the receiver never receives the LMP_DETACH then a link supervision timeout will occur, the link will be detached, and the LT_ADDR may be re-used immediately.

If at any time during this or any other LMP sequence the Link supervision timeout expires then the link shall be terminated immediately and the LT_ADDR(S) may be re-used immediately.

If the connection is in Hold mode, the initiating LM shall wait for Hold mode to end before initiating the procedure defined above. If the connection is in Sniff mode,

the initiating LM shall perform the procedure to exit Sniff mode before initiating the procedure defined above. If the procedure to exit Sniff mode does not complete within the LMP response timeout (30 seconds) the procedure defined above shall be initiated anyway.




*Sequence 1: Connection closed by sending LMP_DETACH*

If there are any SCO or eSCO connections open on the same physical link, sending or receiving the LMP_DETACH PDU implicitly removes them and the initiating LM should not send separate PDUs for this purpose when detaching.

### 4.1.3 Power control

If the received signal characteristics differ too much from the preferred value of a Bluetooth device, it may request an increase or a decrease of the other device's transmit power level. A device's transmit power level is a property of the physical link, and affects all logical transports carried over the physical link. Power control requests carried over the default ACL-C logical link shall only affect the physical link associated with the requesting device's default ACL-C logical link; they shall not affect the power level used on the physical links to other Peripherals.

Two power control mechanisms are specified: legacy power control (see Table 4.3) and enhanced power control (see Table 4.4 and Section 4.1.3.1).

| M/O | PDU | Contents |
|---|---|---|
| O(9) | LMP_INCR_POWER_REQ | Reserved |
| O(9) | LMP_DECR_POWER_REQ | Reserved |
| O(18) | LMP_MAX_POWER | *none* |
| O(18) | LMP_MIN_POWER | *none* |

*Table 4.3: Legacy power control PDU*

| M/O | PDU | Contents |
|---|---|---|
| O(58) | LMP_POWER_CONTROL_REQ | Power_Adj_Req |
| O(58) | LMP_POWER_CONTROL_RES | Power_Adj_Rsp |

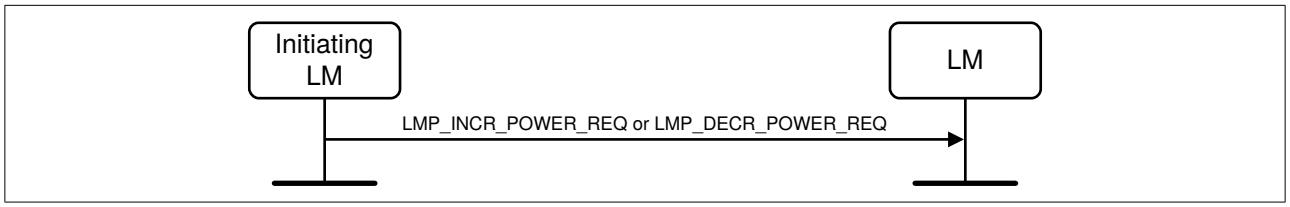
*Table 4.4: Enhanced power control PDU*

The power adjustment requests may be made at any time using the legacy power control mechanism following a successful Baseband Paging procedure and before

the link manager supported features responses have been processed. After the Link Manager supported features responses have been processed, if both devices support enhanced power control (see Section 4.1.3.1) then only enhanced power control shall be used. Otherwise, if either device supports only the legacy power control mechanism then only legacy power control shall be used.

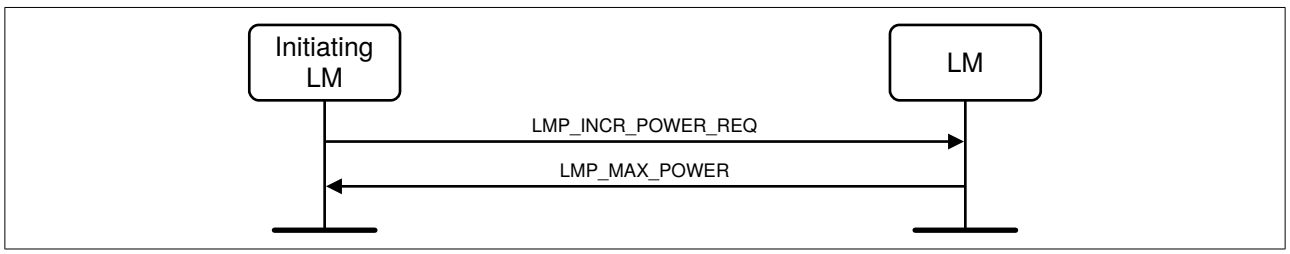
If a device does not support power control requests this is indicated in the supported features list and thus no power control requests shall be sent after the supported features response has been processed. Prior to this time, a power control adjustment might be sent, and if the recipient does not support power control, it shall send LMP_MAX_POWER in response to LMP_INCR_POWER_REQ and LMP_MIN_POWER in response to LMP_DECR_POWER_REQ or shall send LMP_NOT_ACCEPTED with the Error_Code *Unsupported LMP Feature* (0x1A) in response to either PDU.

Upon receipt of an LMP_INCR_POWER_REQ PDU or LMP_DECR_POWER_REQ PDU the output power shall be increased or decreased one step. See [Vol 2] Part A, Section 5.2 for the definition of the step size.

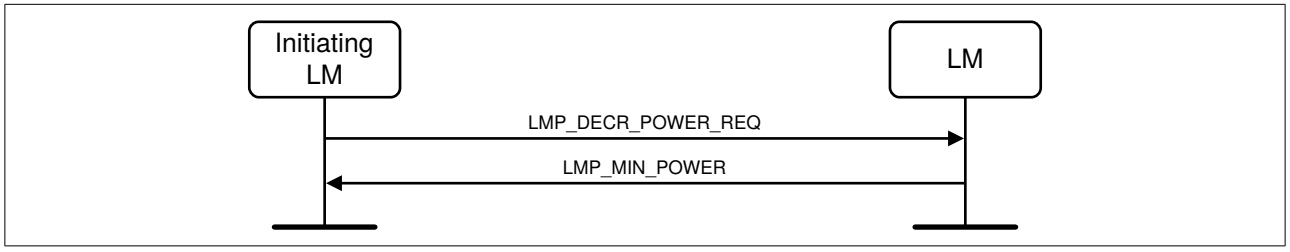



*Sequence 2: A device requests a change of the other device's TX power*

If the receiver of LMP_INCR_POWER_REQ is at maximum power LMP_MAX_POWER shall be returned. The device shall only request an increase again after having requested a decrease at least once. If the receiver of LMP_DECR_POWER_REQ is at minimum power then LMP_MIN_POWER shall be returned and the device shall only request a decrease after having requested an increase at least once.




*Sequence 3: The TX power cannot be increased*




*Sequence 4: The TX power cannot be decreased*

One byte in LMP_INCR/DECR_POWER_REQ is reserved for future use.

#### 4.1.3.1 Enhanced power control

Enhanced power control shall only be used when both devices support the enhanced power control LMP feature. Legacy power control shall not be used when both devices support the enhanced power control LMP feature.

##### 4.1.3.1.1 Sending enhanced power control requests

To adjust the remote device's output power, a device shall send the LMP_POWER_CONTROL_REQ PDU with the Power_Adj_Req parameter.

The Power_Adj_Req parameter may be set to: one step up, one step down, or all the way to the max power level. The remote device shall respond with an LMP_POWER_CONTROL_RES PDU. The responder shall transmit the LMP_POWER_CONTROL_RES PDU at the new power level. Upon reception of the LMP_POWER_CONTROL_RES PDU, the initiating device should restart any processes used to determine whether additional power level changes are required.

If the receiver of the LMP_POWER_CONTROL_REQ PDU has indicated that the output power levels for all of the supported modulation modes are at maximum the requesting device shall only request an increase again after having requested a decrease at least once. If the receiver of the LMP_POWER_CONTROL_REQ PDU has indicated that the output power levels for all of the supported modulation modes are at minimum the requesting device shall only request a decrease after having requested an increase at least once.

A new LMP_POWER_CONTROL_REQ PDU shall not be sent until an LMP_POWER_CONTROL_RES PDU has been received.

##### 4.1.3.1.2 Responding to enhanced power control requests

When a device receives an LMP_POWER_CONTROL_REQ PDU with the Power_Adj_Req parameter set to "increment one step," all supported modulations that are not at the maximum level shall be increased one step.

When a device receives an LMP_POWER_CONTROL_REQ PDU with the Power_Adj_Req parameter set to "decrement one step", all supported modulations that are not at the minimum level shall be decreased one step.

Implementations shall not violate the relative power level between modulations (see [Vol 2] Part A, Section 5.2).

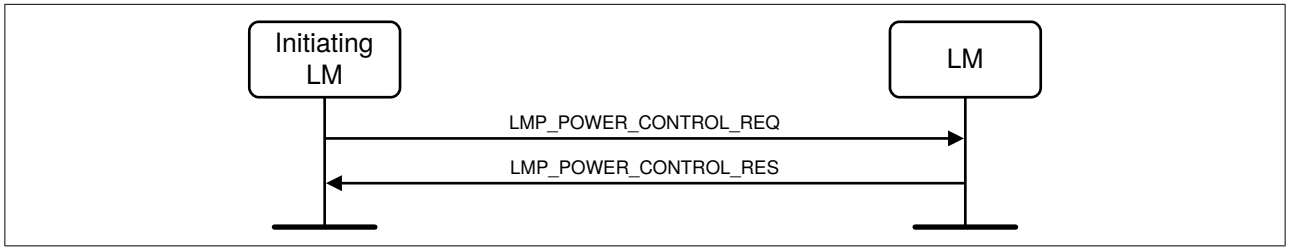
When a device receives an LMP_POWER_CONTROL_REQ PDU with the Power_Adj_Req parameter set to "increment power to maximum value", each supported modulation mode shall be set to its maximum power level.

Note: See [Vol 2] Part A, Section 5.2 for requirements on the relative power levels of different modulation modes.

The responding LM shall send the LMP_POWER_CONTROL_RES PDU to indicate the status for every modulation. The Power_Adj_Rsp parameter has three 2-bit fields indicating the status for each modulation mode: GFSK, π/4-DQPSK and 8DPSK. Each 2-bit field shall be set to one of the following values: *not supported*, *changed one step, max power*, or *min power*. The changed one step value shall only be used when the power level for that modulation mode has not reached the minimum or maximum level.

The *not supported* value shall be used for each unsupported modulation type.

The responder shall transmit the LMP_POWER_CONTROL_RES PDU at the new transmit power level and shall not change its power level until requested by the remote device by a subsequent LMP_POWER_CONTROL_REQ PDU.




*Sequence 5: A device responds to a power control request*

### 4.1.4 Adaptive frequency hopping

AFH is used to improve the performance of physical links in the presence of interference as well as reducing the interference caused by physical links on other devices in the ISM band. AFH shall only be used during Connection state and for Connectionless Peripheral Broadcast.

| M/O | PDU | Contents |
|---|---|---|
| O(35) Rx<br>O(43) Tx | LMP_SET_AFH | AFH_Instant,<br>AFH_Mode,<br>AFH_Channel_Map |

*Table 4.5: AFH PDU*

The LMP_SET_AFH PDU contains three parameters: AFH_Instant, AFH_Mode, and AFH_Channel_Map. The parameter, AFH_Instant, specifies the instant at which the hopset switch shall become effective. This is specified as a Bluetooth Clock value of the Central's clock, that is available to both devices. The AFH instant is chosen by the Central and shall be an even value at least $6 \times T_{poll}$ or 96 slots (whichever is greater) in the future. The AFH_Instant shall be within 12 hours of the current clock value. The parameter AFH_Mode, specifies whether AFH shall be enabled or disabled. The parameter AFH_Channel_Map, specifies the set of channels that shall be used if AFH is enabled.

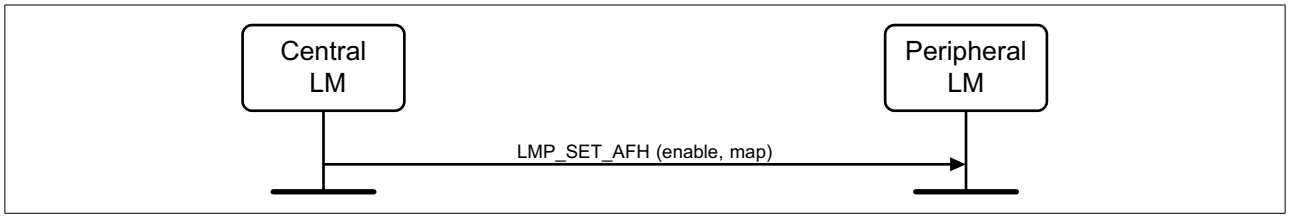
When the LMP_SET_AFH PDU is received the AFH instant shall be compared with the current Bluetooth clock value. If it is in the past then the AFH_Instant has passed and the Peripheral shall immediately configure the hop selection kernel (see [Vol 2] Part B, Section 2.6.3) with the new AFH_Mode and AFH_Channel_Map specified in the LMP_SET_AFH PDU. If it is in the future then a timer shall be started to expire at the AFH instant. When this timer expires it shall configure the hop selection kernel with the new AFH_Mode and AFH_Channel_Map.

The Central shall not send a new LMP_SET_AFH PDU to a Peripheral until it has received the Baseband acknowledgment for any previous LMP_SET_AFH addressed to that Peripheral and the instant has passed.

Role switch while AFH is enabled shall follow the procedures define by [Vol 2] Part B, Section 8.6.5.

#### 4.1.4.1 Central enables AFH

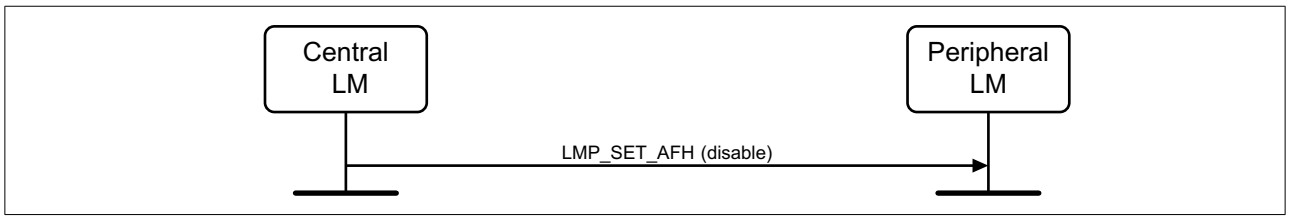
Prior to enabling AFH the Central LM shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). The Central shall then enable AFH on a physical link by sending the LMP_SET_AFH PDU with AFH_Mode set to enabled, the AFH_Channel_Map parameter containing the set of used and unused channels, and an AFH_instant. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The Central considers the physical link to have entered AFH-enabled operation once the Baseband acknowledgment has been received and the AFH_Instant has passed. Once the Baseband acknowledgment has been received the Central shall restart transmission on the ACL-U logical link.




*Sequence 6: Central enables AFH*

#### 4.1.4.2 Central disables AFH

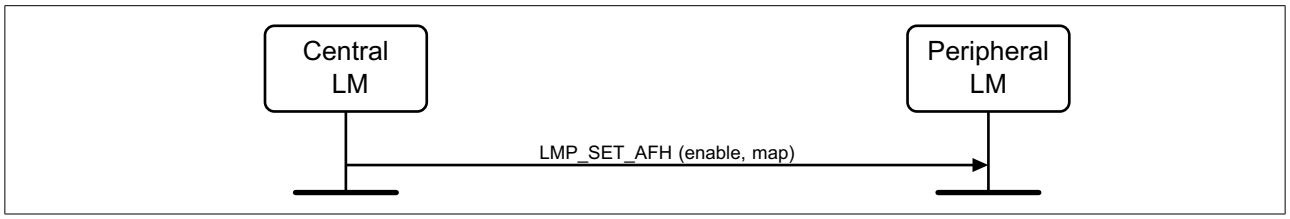
Prior to disabling AFH the Central LM shall pause traffic on the ACL-U logical link ([Vol 2] Part B, Section 5.3.1). The Central shall then disable AFH operation on a physical link by sending the LMP_SET_AFH PDU with AFH_mode set to disabled and an AFH_Instant. The AFH_Channel_Map parameter is not valid when AFH_Mode is disabled. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The Central considers the physical link to have entered AFH-disabled operation once the Baseband acknowledgment has been received and the AFH_Instant has passed. Once the Baseband acknowledgment has been received the Central shall restart transmission on the ACL-U logical link.




*Sequence 7: Central disables AFH*

#### 4.1.4.3 Central updates AFH

A Central shall update the AFH parameters on a physical link by sending the LMP_SET_AFH PDU with AFH_Mode set to enabled, an AFH_Instant and a new AFH_Channel_Map. The Central shall consider the Peripheral to have the updated AFH parameters once the Baseband acknowledgment has been received and the AFH_Instant has passed.




*Sequence 8: Central updates AFH*

#### 4.1.4.4 AFH operation in Hold and Sniff modes

A Peripheral in Hold mode or Sniff mode shall retain the AFH_Mode and AFH_Channel_Map it was using prior to entering those modes. A Central may change the AFH_Mode while a Peripheral is in Sniff mode.

A Central that receives a request from an AFH-enabled Peripheral to enter Hold mode or Sniff mode and decides to operate the Peripheral using a different hop sequence shall respond with an LMP_SET_AFH PDU specifying the new hop sequence.

The Central continues with the LMP signaling, for Hold or Sniff initiation, once the Baseband acknowledgment for the LMP_SET_AFH PDU has been received. The Central may delay the continuation of this LMP signaling until after the instant. An AFH capable Peripheral shall support both of these cases.

A Central that receives a request from an AFH-enabled Peripheral to enter Hold mode or Sniff mode and decides not to change the Peripheral's hop sequence shall respond exactly as it would do without AFH. In this case, AFH operation has no effect on the LMP signaling.

### 4.1.5 Channel classification

A Central may request channel classification information from a Peripheral that supports AFH.

A Peripheral that supports the AFH_classification_Peripheral feature shall perform channel classification and reporting according to its AFH_reporting_mode. The Central shall control the AFH_reporting_mode using the LMP_CHANNEL_CLASSIFICATION_REQ PDU. The Peripheral shall report its channel classification using the LMP_CHANNEL_CLASSIFICATION PDU.

The Peripheral shall report pairs of channels as *good*, *bad* or *unknown*. See Table 5.2 for the detailed format of the AFH_Channel_Classification parameter. When one channel in the $n^{th}$ channel pair is good and the other channel is unknown the $n^{th}$ channel pair shall be reported as good. When one channel in the $n^{th}$ channel pair is bad and the other is unknown the $n^{th}$ channel pair shall be reported as bad. It is implementation dependent what to report when one channel in a channel pair is good and the other is bad.

| M/O | PDU | Contents |
|---|---|---|
| O(36) Rx<br>O(44) Tx | LMP_CHANNEL_CLASSIFICATION_REQ | AFH_Reporting_Mode,<br>AFH_Min_Interval,<br>AFH_Max_Interval |
| O(36) Tx<br>O(44) Rx | LMP_CHANNEL_CLASSIFICATION | AFH_Channel_-Classification |

*Table 4.6: Channel classification PDU*

The LMP_CHANNEL_CLASSIFICATION_REQ PDU contains three parameters: AFH_Reporting_Mode, AFH_Min_Interval, and AFH_Max_Interval. In the disabled state, the Peripheral shall not generate any channel classification reports. The parameter AFH_min_interval, defines the minimum amount of time from the last LMP_CHANNEL_CLASSIFICATION command that was sent before the next LMP_CHANNEL_CLASSIFICATION PDU may be sent. The parameter AFH_max_interval, defines the maximum amount of time between the change in the channel classification being detected by a Peripheral and its generation of an LMP_CHANNEL_CLASSIFICATION PDU. The AFH_max_interval shall be equal to or larger than AFH_min_interval.

The AFH_reporting_mode parameter shall determine if the Peripheral is in the enabled or disabled state. The default state, prior to receipt of any LMP_CHANNEL_CLASSIFICATION_REQ PDUs, shall be disabled.

AFH_reporting_mode is implicitly set to the disabled state when any of the following occur:

- Establishment of a connection at the Baseband level
- Role switch
- Entry to Hold mode operation

and may be implicitly set to the disabled state when any of the following occur:

- The Peripheral disables AFH after receipt of an LMP_SET_AFH PDU (see Section 4.1.4)
- AFH_reporting_mode is set to enabled when the Peripheral has AFH disabled

AFH_reporting_mode is implicitly restored to its former value when any of the following occur:

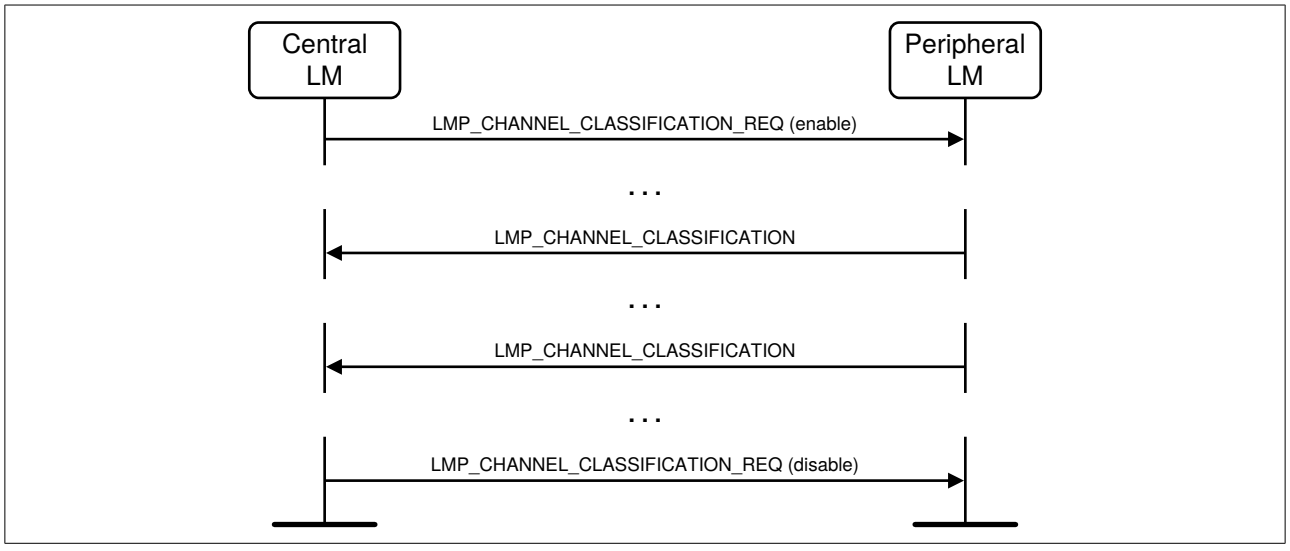
- Exit from Hold mode
- Failure of role switch
- The Peripheral enables AFH after receipt of an LMP_SET_AFH PDU

#### 4.1.5.1 Channel classification reporting enabling and disabling

A Central enables Peripheral channel classification reporting by sending the LMP_CHANNEL_CLASSIFICATION_REQ PDU with the AFH_reporting_mode parameter set to enabled.

When a Peripheral has had classification reporting enabled by the Central it shall send the LMP_CHANNEL_CLASSIFICATION PDU according to the information in the latest LMP_CHANNEL_CLASSIFICATION_REQ PDU. The LMP_CHANNEL_CLASSIFICATION PDU shall not be sent if there has been no change in the Peripheral's channel classification.

A Central disables Peripheral channel classification reporting by sending the LMP_CHANNEL_CLASSIFICATION_REQ PDU with the AFH_reporting_mode parameter set to disabled.



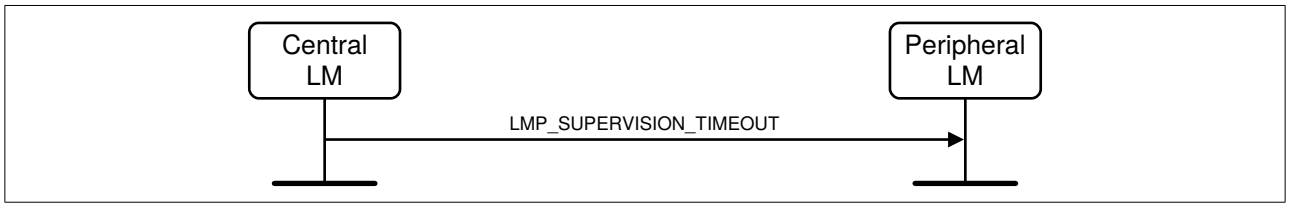

*Sequence 9: Channel classification reporting*

### 4.1.6 Link supervision

Each physical link has a timer that is used for link supervision. This timer is used to detect physical link loss caused by devices moving out of range, or being blocked by interference, a device's power-down, or other similar failure cases. Link supervision is specified in [Vol 2] Part B, Section 3.1.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_SUPERVISION_TIMEOUT | Supervision_Timeout |

*Table 4.7: Set supervision timeout PDU*




*Sequence 10: Setting the link supervision timeout*

### 4.1.7 Channel quality driven data rate change (CQDDR)

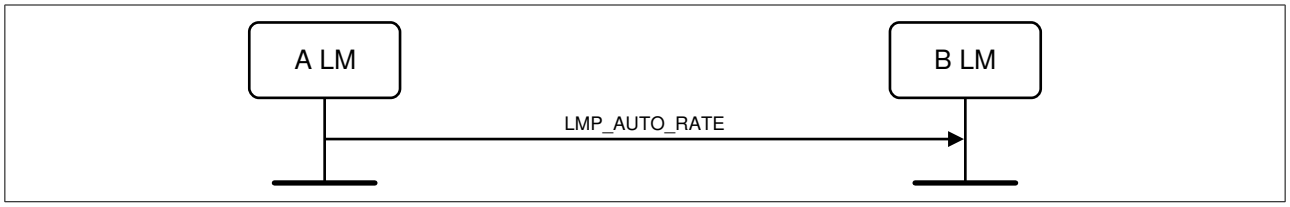
The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. Device A sends the LMP_AUTO_RATE PDU once to notify device B to enable this feature. Once enabled, device B may request packet type(s) that A should transmit by sending the LMP_PREFERRED_RATE PDU. This PDU has a parameter which determines the preferred error coding (with or without 2/3 FEC) and optionally the preferred size in slots of the packets. Device A is not required to change to the packet type specified by this parameter. Device A shall not send a packet that is larger than max slots (see Section 4.1.10) even if the preferred size is greater than this value.

The Data_Rate parameter includes the preferred rate for Basic Rate and Enhanced Data Rate modes. When operating in Basic Rate mode, the device shall use bits 0 to 2 to determine the preferred data rate. When operating in Enhanced Data Rate mode, the device shall use bits 3 to 6 to determine the preferred data rate. For devices that support Enhanced Data Rate, the preferred rates for both Basic Rate and Enhanced Data Rate modes shall be valid at all times.

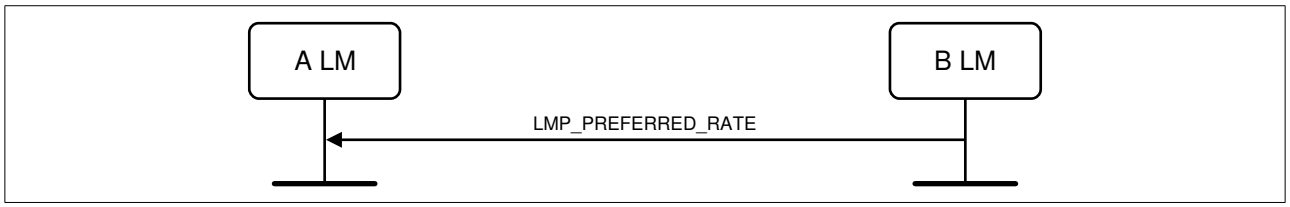
These PDUs may be sent at any time after connection setup is completed.

| M/O | PDU | Contents |
|---|---|---|
| O(10) | LMP_AUTO_RATE | *none* |
| O(10) | LMP_PREFERRED_RATE | Data_Rate |

*Table 4.8: Quality-driven change of the data rate PDU*




*Sequence 11: A notifies B to enable CQDDR*




*Sequence 12: B sends A a preferred packet type*

### 4.1.8 Quality of service (QoS)

The LM provides QoS capabilities. A poll interval, $T_{poll}$ , that is defined as the maximum time between transmissions from the Central to a particular Peripheral on the ACL logical transport, is used to support bandwidth allocation and latency control - see [Vol 2] Part B, Section 8.6.1 for details. The poll interval shall be met in the Active and Sniff modes except when there are collisions with page, page scan, inquiry and inquiry scan, during time critical LMP sequences in the current piconet and any other piconets in which the Bluetooth device is a member, and during critical Baseband sequences (such as the page response, initial Connection state until the first POLL, and role switch). These PDUs may be sent at any time after connection setup is completed.

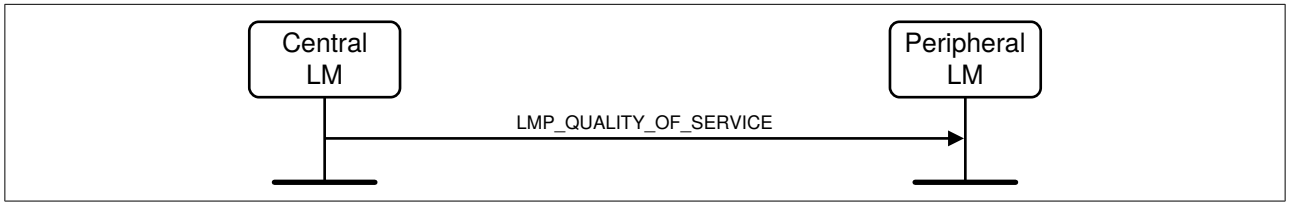
Central and Peripheral negotiate the number of repetitions for broadcast packets ($N_{BC}$ ), see [Vol 2] Part B, Section 7.6.5.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_QUALITY_OF_SERVICE | Poll_Interval<br>$N_{BC}$ |
| M | LMP_QUALITY_OF_SERVICE_REQ | Poll_Interval<br>$N_{BC}$ |

*Table 4.9: Quality of service PDU*

#### 4.1.8.1 Central notifies Peripheral of the quality of service

The Central notifies the Peripheral of the new poll interval and $N_{BC}$ by sending the LMP_QUALITY_OF_SERVICE PDU. The Peripheral cannot reject the notification.

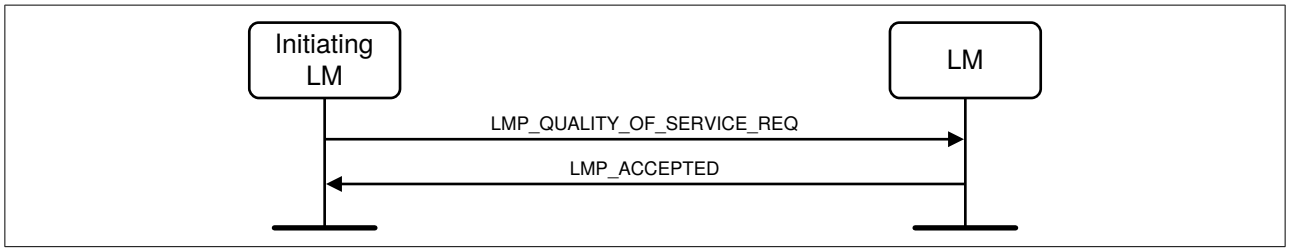



*Sequence 13: Central notifies Peripheral of quality of service*

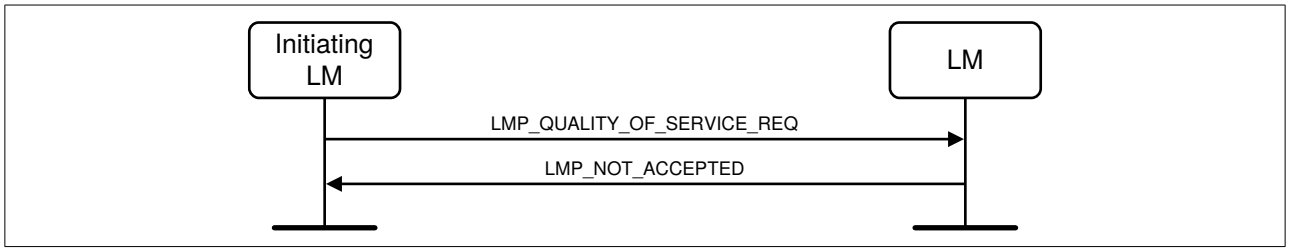
#### 4.1.8.2 Device requests new quality of service

Either the Central or the Peripheral may request a new poll interval and $N_{BC}$ by sending an LMP_QUALITY_OF_SERVICE_REQ PDU. The parameter $N_{BC}$ is meaningful only when it is sent by a Central to a Peripheral. For transmission of LMP_QUALITY_OF_SERVICE_REQ PDUs from a Peripheral, this parameter shall be ignored by the Central. The request can be accepted or rejected. This allows the Central and Peripheral to dynamically negotiate the quality of service as needed.

The selected poll interval by the Peripheral shall be less than or equal to the specified Access Latency for the outgoing traffic of the ACL link (see [Vol 3] Part A, Section 5.3).




*Sequence 14: Device accepts new quality of service*




*Sequence 15: Device rejects new quality of service*

### 4.1.9 Paging scheme parameters

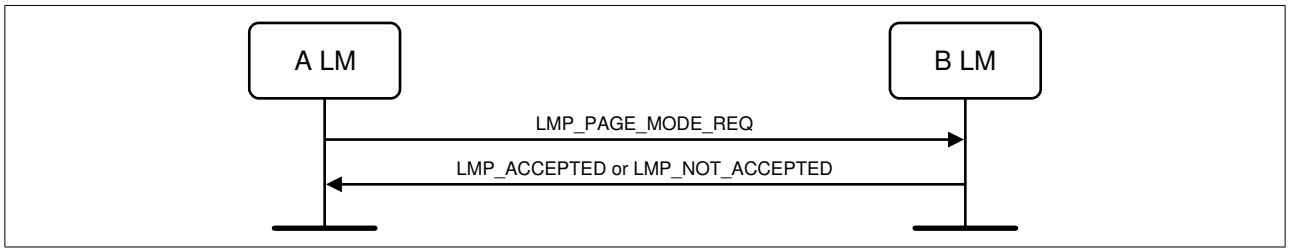
LMP provides a means to negotiate the paging scheme parameters that are used the next time a device is paged.

| M/O | PDU | Contents |
|---|---|---|
| O(17) | LMP_PAGE_MODE_REQ | Paging_Scheme<br>Paging_Scheme_Settings |
| O(17) | LMP_PAGE_SCAN_MODE_REQ | Paging_Scheme<br>Paging_Scheme_Settings |

*Table 4.10: Paging scheme request PDU*

#### 4.1.9.1 Page mode

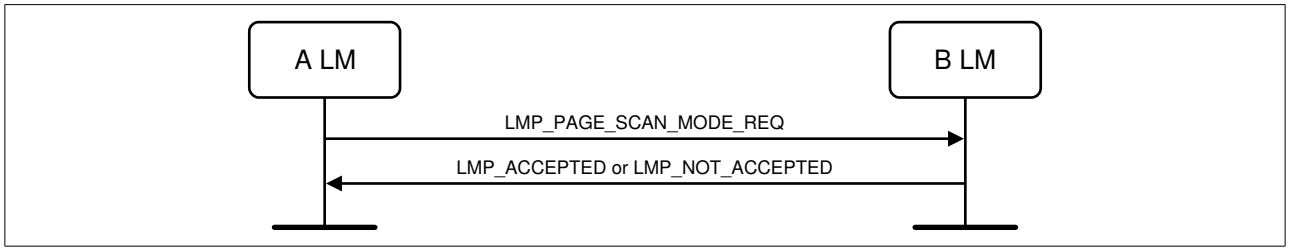
This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting shall not be changed. A request to switch to a reserved paging scheme shall be rejected.




*Sequence 16: Negotiation for Page mode*

#### 4.1.9.2 Page Scan mode

This procedure is initiated from device A and negotiates the paging scheme and paging scheme settings used when device B pages device A. Device A proposes a paging scheme and paging scheme settings and device B may accept or reject. On reject the old setting is not changed. A request specifying the mandatory scheme shall be accepted. A request to switch to a reserved scheme shall be rejected. This procedure should be used when device A changes its paging scheme settings. A Peripheral should also send this message to the Central after connection establishment, to inform the Central of the Peripheral's current paging scheme and paging scheme settings.



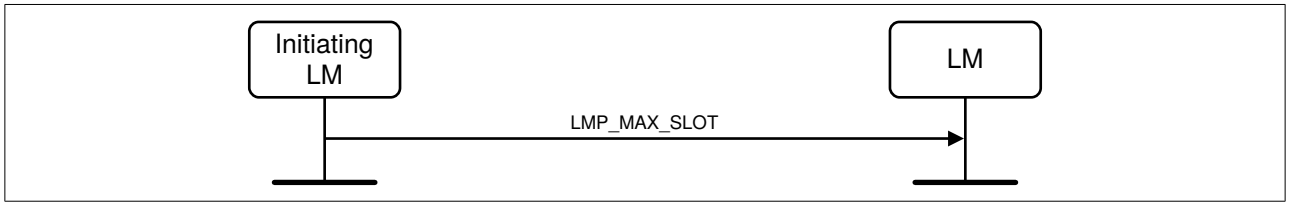

*Sequence 17: Negotiation for Page Scan mode*

### 4.1.10 Control of multi-slot packets

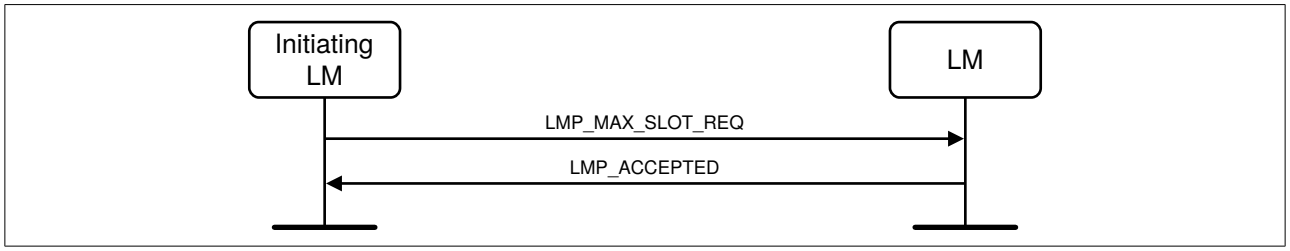
The number of consecutive slots used by a device on an ACL-U logical link can be limited. It does not affect traffic on the eSCO links where the packet sizes are defined as part of link setup. A device allows the remote device to use a maximum number of slots by sending the PDU LMP_MAX_SLOT providing a Max_Slots parameter. Each device can request to use a maximal number of slots by sending the PDU LMP_MAX_SLOT_REQ providing a Max_Slots parameter. After a new connection (as a result of page or page scan), or after a role switch, the value shall be 1 slot. These PDUs can be sent at any time after connection setup is completed.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_MAX_SLOT | Max_Slots |
| M | LMP_MAX_SLOT_REQ | Max_Slots |

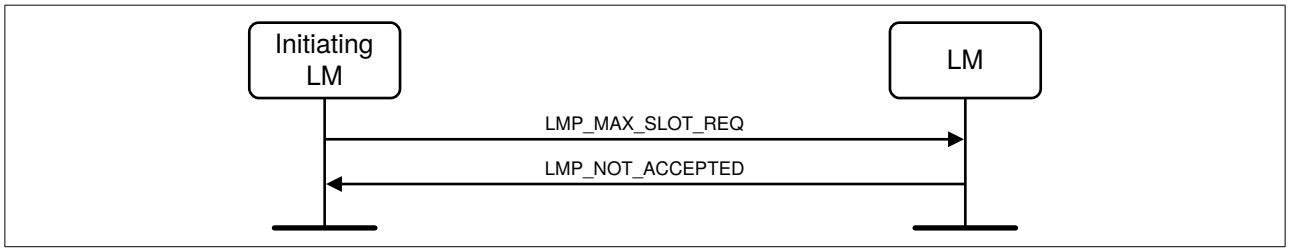
*Table 4.11: Multi-slot packet control PDU*




*Sequence 18: Device allows Remote Device to use a maximum number of slots*




*Sequence 19: Device requests a maximum number of slots. Remote Device accepts.*




*Sequence 20: Device requests a maximum number of slots. Remote Device rejects*

### 4.1.11 Enhanced Data Rate

A device may change the packet type table, ptt, to select which if any of the packets and modulation modes are to be used on an ACL logical transport.

Either the Central or the Peripheral may request a new packet type table and therefore the packets and modulation mode to be used on this ACL link. After a new Baseband connection, as a result of page or page scan, the default value for ptt shall be 0.

The change of the modulation mode for an ACL logical transport shall not affect the packet and packet types used for an associated SCO logical transport on the same LT_ADDR.

Note: Enhanced Data Rate eSCO links are negotiated using the LMP eSCO link_req as described in Section 4.6.2.

Before changing the packet type table, the initiator shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions. It shall then send the LMP_PACKET_TYPE_TABLE_REQ PDU.

If the receiver rejects the change, then it shall respond with an LMP_NOT_ACCEPTED_EXT PDU.

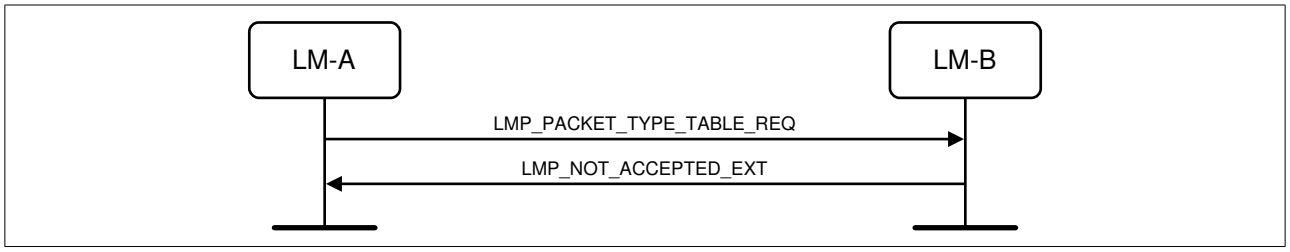
If the receiver accepts the change, then it shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions, it shall change to the new packet type table and shall respond with an LMP_ACCEPTED_EXT PDU. When it receives the Baseband acknowledgment for the LMP_ACCEPTED_EXT PDU it shall restart ACL-U transmissions.

When the initiator receives an LMP_NOT_ACCEPTED_EXT PDU the initiator shall restart ACL-U transmissions.

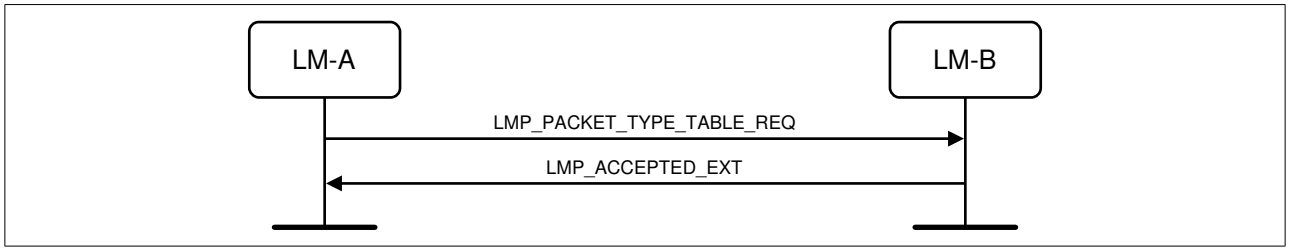
When the initiator receives an LMP_ACCEPTED_EXT PDU it shall change the packet type table and restart ACL-U transmissions.

| M/O | PDU | Contents |
|---|---|---|
| O(25) | LMP_PACKET_TYPE_TABLE_REQ | Packet_Type_Table |

*Table 4.12: Enhanced Data Rate PDUs*




*Sequence 21: Packet type table change is rejected*




*Sequence 22: Packet type table change is accepted*

### 4.1.12 Encapsulated LMP PDUs

Some transactions require sending LMP payload data that is longer than 16 octets. To enable a link manager to send a large PDU, an encapsulated LMP PDU is defined. An encapsulated LMP PDU is composed of a minimum of two LMP messages, a header PDU and one or more payload PDUs.

| M/O | PDU | Contents |
|---|---|---|
| O(52) | LMP_ENCAPSULATED_HEADER | Encap_Major_Type<br>Encap_Minor_Type<br>Encap_Payload_Length |
| O(52) | LMP_ENCAPSULATED_PAYLOAD | Encap_Data |

*Table 4.13: Encapsulated LMP PDUs*

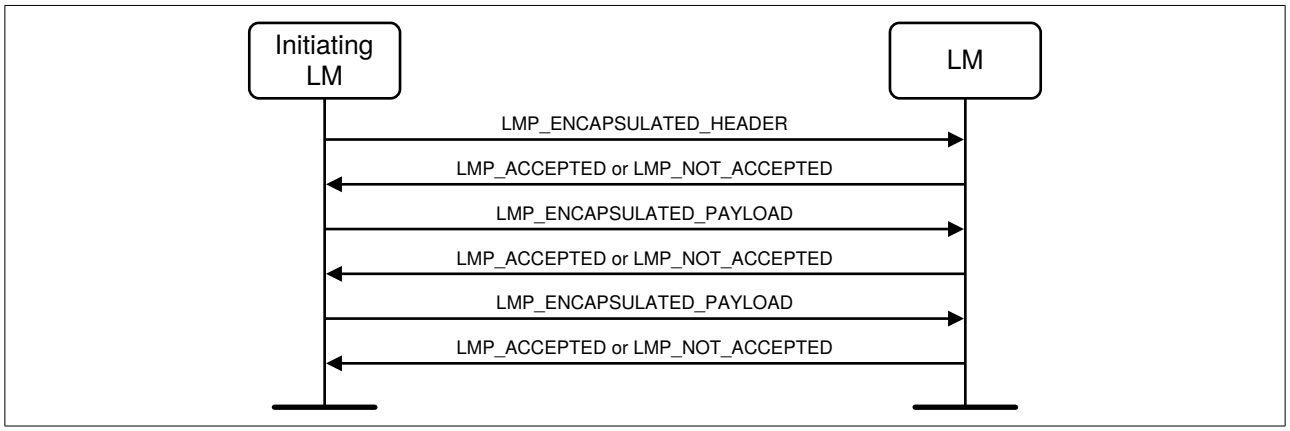
#### 4.1.12.1 Sending an encapsulated PDU

The LMP_ENCAPSULATED_HEADER PDU shall be sent by the initiating device when it needs to send an encapsulated PDU. This PDU shall be either accepted or rejected using the LMP_ACCEPTED or LMP_NOT_ACCEPTED PDUs. If the major and minor types are not supported the PDU shall be rejected with Error_Code *Unsupported LMP Parameter Value* (0x20). If the LMP_ENCAPSULATED_HEADER PDU is accepted, then one or more LMP_ENCAPSULATED_PAYLOAD PDUs will be sent with the encapsulated data sent in sequence, 16 octets at a time, or if this is last packet, the correct number of octets and then zero padded.

Each LMP_ENCAPSULATED_PAYLOAD PDU shall be accepted or rejected. If the LMP_ENCAPSULATED_HEADER PDU is rejected, then the

opcode in the LMP_NOT_ACCEPTED PDU shall be the opcode for the LMP_ENCAPSULATED_HEADER and not the Encap_Major_Type or Encap_Minor_Type. If the LMP_ENCAPSULATED_PAYLOAD PDU is rejected, then the opcode in the LMP_NOT_ACCEPTED PDU shall be the opcode for the LMP_ENCAPSULATED_PAYLOAD.

A responding device may reject the final LMP_ENCAPSULATED_PAYLOAD PDU after accepting the LMP_ENCAPSULATED_HEADER PDU. This is so that the link manager can still reject an encapsulated message after all the data has been received.




*Sequence 23: Sending an encapsulated PDU*

Between sending an LMP_ENCAPSULATED_HEADER PDU and an LMP_ENCAPSULATED_PAYLOAD PDU, or between each of the LMP_ENCAPSULATED_PAYLOAD PDUs, either device shall be able to send the following LMP PDUs without causing a "different transaction collision" error.

- LMP_CHANNEL_CLASSIFICATION
- LMP_DECR_POWER_REQ
- LMP_DETACH
- LMP_INCR_POWER_REQ
- LMP_MAX_POWER
- LMP_MAX_SLOT
- LMP_MIN_POWER
- LMP_PREFERRED_RATE
- LMP_SET_AFH

### 4.1.13 Ping

When both devices support the Ping feature, a Link Manager may verify the presence of the remote Link Manager by using the LMP ping mechanism. The LMP ping mechanism

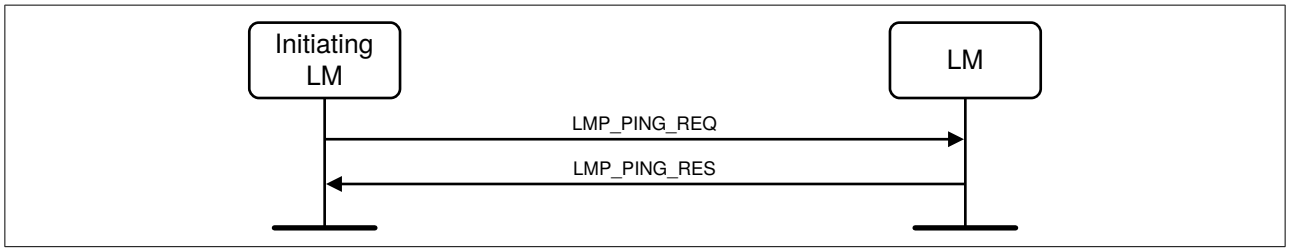
may also be used to verify message integrity on the ACL logical transport by forcing the remote device to send an ACL packet that contains a MIC.

Note: Since the LMP_PING_RES PDU contains a MIC and will update the packet counter, the sender of the LMP_PING_REQ PDU can verify that packets with earlier packet counter values have not been deliberately suppressed by an attacker.

| M/O | PDU | Contents |
|---|---|---|
| O(137) | LMP_PING_REQ | *none* |
| O(137) | LMP_PING_RES | *none* |

*Table 4.14: Ping PDUs*

The initiating link manager sends the LMP_PING_REQ PDU. The responding link manager responds with the LMP_PING_RES PDU. The initiating link manager may be a Central or Peripheral.




*Sequence 24: Ping PDUs*

The Link Manager shall send an LMP_PING_REQ PDU when the remote device has not sent a packet containing a payload protected by a MIC within an authenticated payload timeout set by the Host (see [Vol 4] Part E, Section 7.3.94). The Link Manager should send an LMP_PING_REQ PDU in advance enough of the expiration of the authenticated payload timeout to allow the remote device reasonable time to respond with an LMP_PING_RES PDU before the timeout expires.

### 4.1.14 Piconet clock adjustment

The Central may adjust the piconet clock (e.g. to align with an external time base) by initiating the Coarse Clock Adjustment or Clock Dragging procedures. A Peripheral may request that the Central adjust the piconet clock.

| M/O | PDU | Contents |
|---|---|---|
| O(134) | LMP_CLK_ADJ | Clk_Adj_ID<br>Clk_Adj_Instant<br>Clk_Adj_Offset<br>Clk_Adj_Slots<br>Clk_Adj_Mode<br>Clk_Adj_Clk |
| O(134) | LMP_CLK_ADJ_ACK | Clk_Adj_ID |
| O(134) | LMP_CLK_ADJ_REQ | Clk_Adj_Offset<br>Clk_Adj_Slots<br>Clk_Adj_Period |

*Table 4.15: Piconet clock adjustment PDUs*

Piconet clock adjustment requests may be made at any time after the Link Manager supported features responses have been processed.

The Central is free to reject any request for an adjustment; an implementation may, for instance, accept coarse clock adjustment requests from one Peripheral but reject all requests from any other Peripheral.

#### 4.1.14.1 Central coarse adjustment of piconet clock

The coarse clock adjustment procedure shall only be used if all connected Peripherals declare support for the Coarse Clock Adjustment feature.

For all devices that may use Coarse Clock Adjustment Recovery Mode (see [Vol 2] Part B, Section 8.6.10.2), the RF channel indices used for the Synchronization Train (see [Vol 2] Part B, Section 2.6.4.8) shall be marked as unused in the AFH_Channel_Map for all logical links.

The Central may use the Clk_Adj_Period parameter from one or more Peripherals, in addition to other scheduling considerations, to select an optimal value of Clk_Adj_Slots.

The Central selects the parameters of the coarse clock adjustment. Some parameters shall remain the same for every LMP_CLK_ADJ PDU associated with a given adjustment. These are the number of slots (Clk_Adj_Slots) between 0 and 255, an offset within a slot (Clk_Adj_Offset) between –624 µs and 624 µs, a coarse clock adjustment instant (Clk_Adj_Instant), which is specified in the old time domain before the instant, and an identifier for the instant (Clk_Adj_ID). The Clk_Adj_Instant shall be at least 12 slots in the future and no more than 12 hours away. The Clk_Adj_ID shall be different for any two adjacent adjustments and for any two adjustments whose instants are closer together in time than the longest Link Supervision Timeout period

for any Peripheral. Each LMP_CLK_ADJ PDU also contains the value of CLK when it is transmitted (Clk_Adj_Clk) and a flag Clk_Adj_Mode, which shall be set to Before Instant if the LMP_CLK_ADJ PDU is sent before the instant or to After Instant if the LMP_CLK_ADJ PDU is sent on or after the instant.

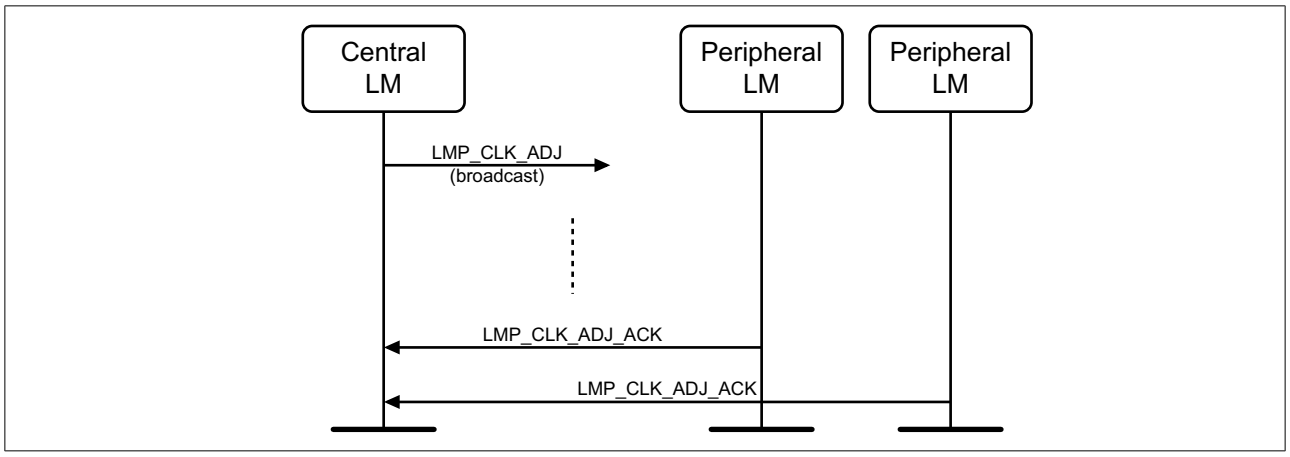
To initiate the coarse adjustment, the Central starts broadcasting the LMP_CLK_ADJ PDU using the APB-C logical link and, unless all Peripherals have acknowledged, it should broadcast the LMP_CLK_ADJ PDU at least 6 times before the adjustment instant. The Central shall continue to broadcast the LMP_CLK_ADJ PDU until all Peripherals have acknowledged the adjustment with an LMP_CLK_ADJ_ACK PDU or the Central has left Coarse Clock Adjustment Recovery Mode.

The Central shall not initiate a coarse clock adjustment if there are any transactions outstanding that involve LMP PDUs containing an instant or timing control flags including, but not limited to: role switch, adaptive frequency hopping, SCO, eSCO, sniff, sniff subrating, hold, and encryption pause and resume.

The Central shall not initiate another sequence containing an instant or timing control flags before an outstanding clock adjustment instant has been reached, while waiting for *CLK_adj_dragTO* to expire (see [Vol 2] Part B, Appendix B), or while in Coarse Clock Adjustment Recovery Mode (see [Vol 2] Part B, Section 8.6.10.2); however the latter may be terminated by the Central in order to initiate the request. If a request to initiate such a sequence is received before an outstanding clock adjustment instant is reached or *CLK_adj_dragTO* has been cancelled or has expired, the request shall be rejected with the Error_Code *Different Transaction Collision* (0x2A).

A Peripheral shall discard and not acknowledge an LMP_CLK_ADJ PDU with a value of Clk_Adj_ID that is the same as that in the most recently received LMP_CLK_ADJ PDU, if any, and any LMP_CLK_ADJ PDU with any parameters outside the valid range (see Section 5.2).

On receipt of an LMP_CLK_ADJ PDU that passes these checks, the Peripheral shall reply with an LMP_CLK_ADJ_ACK PDU (on the ACL-C logical link) containing the same value of Clk_Adj_ID as in the LMP_CLK_ADJ PDU and shall change the value of peripheral_clock_offset, if not already performed, as described in [Vol 2] Part B, Section 8.6.10.1, either at the adjustment instant or, if that has passed, immediately.




*Sequence 25: Central initiates a coarse clock adjustment*

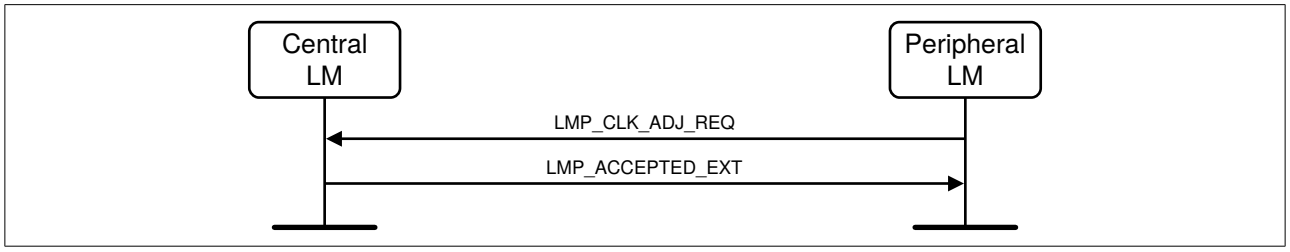
#### 4.1.14.2 Peripheral request for coarse adjustment of piconet clock

The Peripheral may request a coarse adjustment of the piconet clock by sending the LMP_CLK_ADJ_REQ PDU to the Central. The LMP_CLK_ADJ_REQ PDU contains three parameters: Clk_Adj_Offset, Clk_Adj_Slots, and Clk_Adj_Period. The parameters Clk_Adj_Offset and Clk_Adj_Slots have the same meaning as in the LMP_CLK_ADJ PDU and Clk_Adj_Slots or Clk_Adj_Offset shall be non-zero. If non-zero, the parameter Clk_Adj_Period is an indication to the Central that the request is for any of a set of possible adjustments, all of which are equally acceptable to the Peripheral. These adjustments are those where the number of slots is equal to Clk_Adj_Slots + N × Clk_Adj_Period, where N is zero or a positive integer, and the offset within a slot equals Clk_Adj_Offset. For example, if Clk_Adj_Slots is 18 and Clk_Adj_Period is 48, then the acceptable adjustments are 18, 66, 114, 162, and 210 slots. A value of zero for Clk_Adj_Period indicates that an adjustment of any number of slots is equally acceptable to the Peripheral (that is, the Central may ignore the value of Clk_Adj_Slots). A request for a coarse adjustment shall also update the Central's local record of Clk_Adj_Period for the Peripheral.

The Peripheral may indicate its preferred Clk_Adj_Period to the Central without requesting a clock adjustment. In this case it shall set Clk_Adj_Slots and Clk_Adj_Offset to zero. Upon receiving an LMP_CLK_ADJ_REQ PDU with Clk_Adj_Slots and Clk_Adj_Offset set to zero, the Central shall respond with an LMP_ACCEPTED_EXT PDU and update its local record of Clk_Adj_Period for the Peripheral, but shall not initialize any clock adjustment. The Central is not required to honor a Peripheral's preferred Clk_Adj_Period when making coarse clock adjustments.

The Central may accept the coarse clock adjustment request from the Peripheral by sending an LMP_ACCEPTED_EXT PDU. This indicates that the Central intends to carry out a coarse clock adjustment with the requested value of Clk_Adj_Offset;

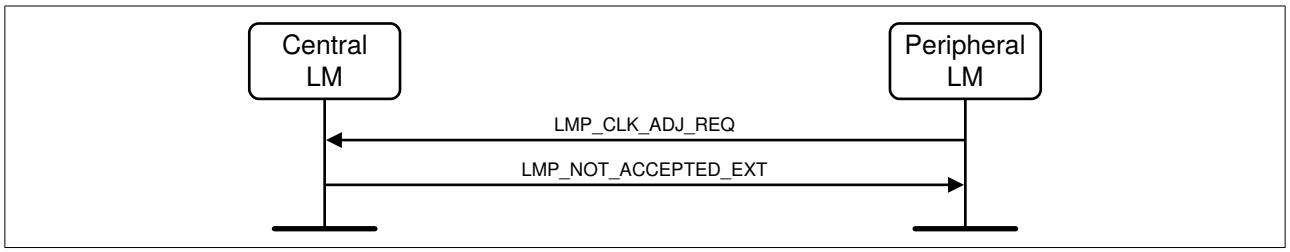
however, the value of Clk_Adj_Slots may be different to that requested and might not follow the Clk_Adj_Period parameter.




*Sequence 26: Central accepts Peripheral request for a coarse clock adjustment*

The Central may reject the coarse clock adjustment request from the Peripheral by sending an LMP_NOT_ACCEPTED_EXT PDU with the Error_Code *Command Disallowed* (0x0C) (see [Vol 1] Part F, Section 2.12). The Central's decision not to follow the Clk_Adj_Slots or Clk_Adj_Period parameters does not constitute a rejection.

The Central may implement the coarse clock adjustment request from the Peripheral by carrying out clock dragging (see [Vol 2] Part B, Section 8.6.10.3) to achieve the requested clock adjustment. If so, it shall send an LMP_NOT_ACCEPTED_EXT PDU with the Error_Code *Coarse Clock Adjustment Rejected but Will Try to Adjust Using Clock Dragging* (0x40) (see [Vol 1] Part F, Section 2.61).




*Sequence 27: Central rejects a Peripheral request for coarse clock adjustment*

### 4.1.15 Slot Availability Mask

SAM LMP sequences may be initiated at any time after Link Manager supported features responses have been processed.

Either the Central or the Peripheral may initiate the LMP_SAM_SET_TYPE0 PDU to configure the SAM type 0 submap of the local device. The initiation may be triggered by the HCI command HCI_Set_External_Frame_Configuration for MWS coexistence.

Either the Central or the Peripheral may initiate the LMP_SAM_DEFINE_MAP PDU to define a new or modify an existing local SAM slot map. The initiation may be triggered by the HCI command HCI_Set_MWS_PATTERN_Configuration for MWS coexistence. If a type 0 submap is used with the LMP_SAM_DEFINE_MAP PDU, the Controller shall

first execute the SAM set type 0 LMP sequence to configure the SAM type 0 submap, then the SAM define map LMP sequence itself.

Either the Central or the Peripheral may initiate the LMP_SAM_SWITCH PDU to switch to a local SAM slot map that has been defined by previous SAM set type 0 and SAM define map LMP sequences. This may be triggered by the real-time signals (e.g. MWS_PATTERN_Index, FRAME_SYNC, etc.) from the Coexistence Logical Interface (see [Vol 7] Part A), which contain information for the SAM_Index to be activated and the SAM anchor point for MWS coexistence. Piconet Clock Adjustment may be performed to create more available slot pairs per MWS frame before initiating the SAM switch LMP sequence.

These SAM LMP sequences will be described in detail in the following subsections.

| M/O | PDU | Contents |
|---|---|---|
| O(138) | LMP_SAM_SET_TYPE0 | Update_Mode<br>SAM_Type0_Submap |
| O(138) | LMP_SAM_DEFINE_MAP | SAM_Index<br>$T_{SAM_SM}$<br>$N_{SAM_SM}$<br>SAM_Submaps |
| O(138) | LMP_SAM_SWITCH | SAM_Index<br>Timing_Control_Flags<br>$D_{SAM}$<br>SAM_Instant |

*Table 4.16: PDUs used for SAM negotiation*

The LMP_SAM_SET_TYPE0 PDU contains the following SAM parameters from the sender:

1. Update_Mode: When Update_Mode is set to 0, the existing slot maps containing the type 0 submap are invalidated and the corresponding SAM_Index shall not be used until the map with that index has been redefined in a subsequent SAM define map LMP sequence. When Update_Mode is set to 1, the new type 0 submap shall take effect immediately. When Update_Mode is set to 2, the new type 0 submap shall take effect at the start of the next sub-interval.
2. SAM_Type0_Submap: This parameter specifies the types of each slot in the type 0 submap. The length of this parameter corresponds to 56 slots but, when it is used in a map, only the first $T_{SAM_SM}$ entries are significant.

   Different maps can have different values of $T_{SAM_SM}$.

The LMP_SAM_DEFINE_MAP PDU contains the following parameters from the sender:

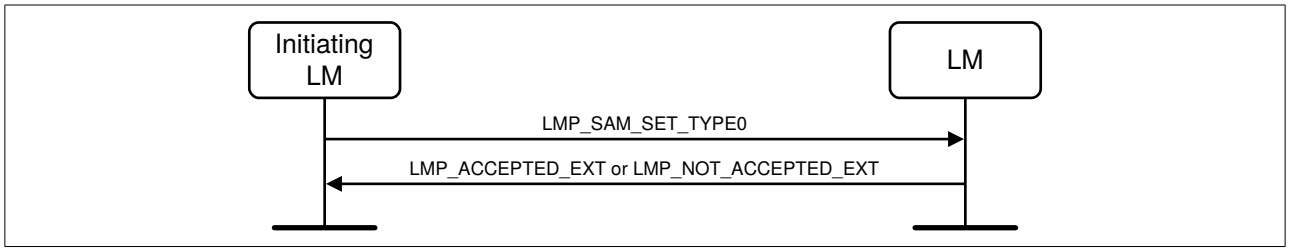
1. SAM_Index: Index of the SAM slot map this PDU applies to. If the same SAM_Index has been previously defined, then the previous map parameters associated with the map shall be discarded and replaced by the parameters contained in this PDU.
2. $T_{SAM_SM}$: Length of each SAM submap in slots. It shall be an even integer in the range 2 to 56.
3. $N_{SAM_SM}$: The number of SAM submaps in the SAM slot map.
4. SAM_Submaps: This parameter specifies the types of the SAM submaps, using 2 bits for each submap. The length of this parameter corresponds to 48 submaps, but only the first $N_{SAM_SM}$ entries are significant.

The LMP_SAM_SWITCH PDU contains the following parameters from the sender:

1. SAM_Index: Index of the SAM slot map. It shall either be the index of a valid map, in which case SAM should be enabled and that map selected, or shall be 0xFF, which indicates that SAM should be disabled (meaning that, for SAM purposes, all slots are available for both transmission and reception). Disabling SAM does not invalidate a SAM slot map or the type 0 submap.
2. Timing_Control_Flags: This parameter is used to avoid clock wrap-around during the sequence, using one of the two initialization rules.
3. $D_{SAM}$: SAM offset that is used to compute the SAM anchor point. Only even values are valid.
4. SAM_Instant: Bits 27:1 of the Central's clock value when the SAM slot map is to be activated. This is used to indicate the first anchor point of the slot map. The new SAM slot map may be activated at SAM_Instant even if the Baseband acknowledgment is not received.

#### 4.1.15.1 SAM type 0 submap configuration

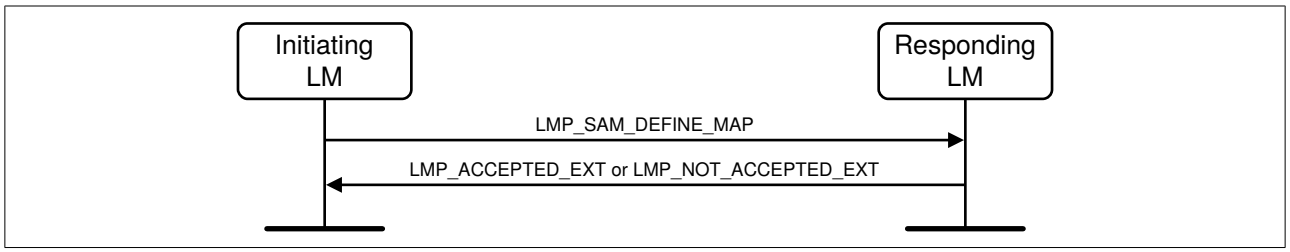
Either the Central or the Peripheral may initiate the SAM type 0 submap configuration sequence by sending an LMP_SAM_SET_TYPE0 PDU to the responder. Both devices shall save the SAM parameters contained in this PDU. Update_Mode shall not be set to 0 if the active SAM slot map contains a type 0 submap. If the responder can accept the parameters, it shall send an LMP_ACCEPTED_EXT PDU. If the Update_Mode parameter is set to 0 and the active SAM slot map contains a type 0 submap, it shall send an LMP_NOT_ACCEPTED_EXT PDU with the Error_Code *Invalid LMP Parameters* (0x1E).




*Sequence 28: SAM type 0 submap configuration sequence*

#### 4.1.15.2 SAM slot map define

Either the Central or the Peripheral may initiate the SAM slot map define sequence by sending an LMP_SAM_DEFINE_MAP PDU. If $N_{SAM_SM}$ is set to zero, the map shall be deleted. The SAM_Index parameter shall not be 0xFF or that of the currently selected map. The slot map may only contain a type 0 submap if the LMP_SAM_SET_TYPE0 sequence has already been used to define the type 0 submap. If the responder can accept the parameters, it shall send an LMP_ACCEPTED_EXT PDU. If a SAM type 0 submap is used in an LMP_SAM_DEFINE_MAP PDU without successfully completing the LMP_SAM_SET_TYPE0 sequence in advance, it shall send an LMP_NOT_ACCEPTED_EXT PDU with the Error_Code *Type0 Submap Not Defined* (0x41).



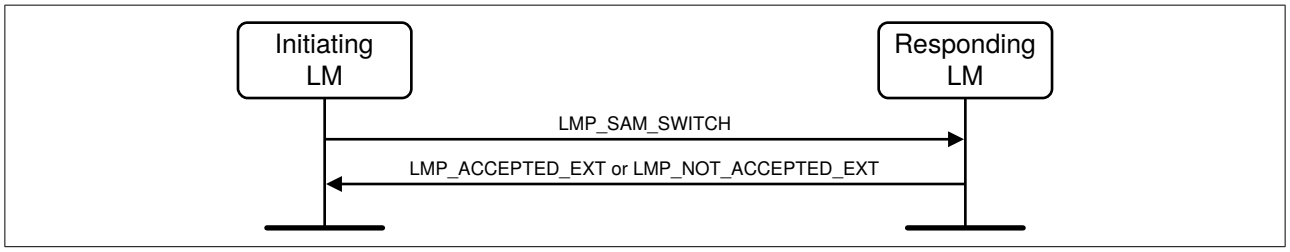

*Sequence 29: SAM slot map define sequence*

#### 4.1.15.3 SAM switch sequence

Either the Central or the Peripheral may initiate the SAM switch sequence by sending an LMP_SAM_SWITCH PDU to the responder. There are two initialization procedures to prevent problems caused by clock wrap-around; see [Vol 2] Part B, Section 8.6.11.1 for details. The SAM_Index shall refer to a SAM slot map that is currently valid (the special value 0xFF is always valid). If the responder can accept the parameters, it shall send an LMP_ACCEPTED_EXT PDU. If the SAM_Index is invalid (e.g., the associated SAM_Index configuration has not yet been defined with LMP_SAM_DEFINE_MAP_SEQUENCE), it shall send an LMP_NOT_ACCEPTED_EXT PDU with the Error_Code *Invalid LMP Parameters* (0x1E).

Each Controller shall notify its Host of a successful SAM switch sequence specifying a different SAM_Index to that currently selected (it may, but need not, notify the Host if the

sequence specifies the current map). The currently selected map remains in effect if the change is not accepted.




*Sequence 30: SAM switch sequence*

#### 4.1.15.4 SAM change during the transmission of a multi-slot packet

SAM slot map changes do not affect the transmission of a multi-slot packet that spans the change instant. In the case of a Central-to-Peripheral packet, the new map can mean that the Peripheral may choose not to reply to the packet.

#### 4.1.15.5 SAM and role switching

The Central and Peripheral shall disable SAM at the role switch instant. If the role switch fails, both devices shall resume using the SAM parameters prior to the role switch instant

SAM mode may be reconfigured and re-enabled after the successful completion of a role switch using the new piconet clock.

#### 4.1.15.6 SAM and Sniff mode

If sniff negotiation is requested when SAM is already enabled, the device should align sniff with SAM by choosing the sniff anchor point to be an available Central-to-Peripheral slot and the sniff period to be an integer multiple of $T_{SAM}$. If such alignment is not possible, the sniff configuration shall take precedence, even if this requires a device to transmit or receive on a slot marked as unavailable by SAM. The SAM slot maps shall be reinstated when devices exit Sniff mode.

Note: Even though Sniff mode overrides the SAM slot maps, a device still might not be able to transmit or receive because of scatternet commitments, a coexistence clash, or some other reason.

## 4.2 Security

Each random number mentioned in this section shall be created according to [Vol 2] Part H, Section 2.

### 4.2.1 Authentication

Two authentication procedures are defined: legacy and secure authentication. Legacy authentication shall be performed when at least one device does not support both the Secure Connections (Controller Support) and Secure Connections (Host Support) features and the local device allows legacy authentication to be used. Secure authentication shall be performed when both devices support the Secure Connections (Controller Support) and Secure Connections (Host Support) features.

The legacy authentication procedure is based on a challenge-response scheme as described in [Vol 2] Part H, Section 3.2.2. The verifier sends an LMP_AU_RAND PDU that contains a random number (the challenge) to the claimant. The claimant calculates a response, that is a function of this challenge, the claimant's BD_ADDR and a secret key. The response is sent back to the verifier, which checks if the response was correct or not. The response shall be calculated as described in [Vol 2] Part H, Section 6.1. A successful calculation of the authentication response requires that two devices share a secret key. This key is created as described in Section 4.2.2. Both the Central and the Peripheral can be verifiers. The sequences for legacy authentication are described in Section 4.2.1.1 and Section 4.2.1.2.

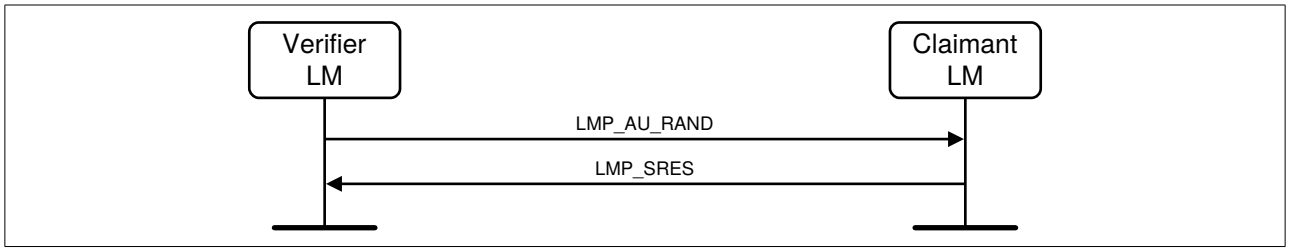
The secure authentication procedure is a challenge-response scheme as described in [Vol 2] Part H, Section 5. The verifier sends an LMP_AU_RAND PDU that contains a random number (the challenge) to the claimant. The claimant responds with another LMP_AU_RAND PDU also containing a random number. Both Link Managers calculate the Device Authentication Key using the h4 function (see [Vol 2] Part H, Section 7.7.7) and then calculate the SRES_C, SRES_P and ACO values using the h5 function (see [Vol 2] Part H, Section 7.7.8). The Peripheral (regardless of whether it was the verifier or claimant) sends its response (SRES_P) to the Central. The Central sends its response (SRES_C) to the Peripheral. A successful calculation of the authentication response requires that two devices share a secret key. The sequences for secure authentication are described in Section 4.2.1.2 and Section 4.2.1.4.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_AU_RAND | Random_Number |
| M | LMP_SRES | Authentication_Rsp |

*Table 4.17: Authentication PDUs*

#### 4.2.1.1 Claimant has link key (legacy authentication)

If the claimant has a link key associated with the verifier, it shall calculate the response and sends it to the verifier with LMP_SRES. The verifier checks the response. If the response is not correct, the verifier may end the connection by sending an LMP_DETACH PDU with the Error_Code *Authentication Failure* (0x05), see Section 4.1.2.




*Sequence 31: Authentication. Claimant has link key.*

Upon reception of an LMP_AU_RAND, an LM shall reply with LMP_SRES before initiating its own authentication.

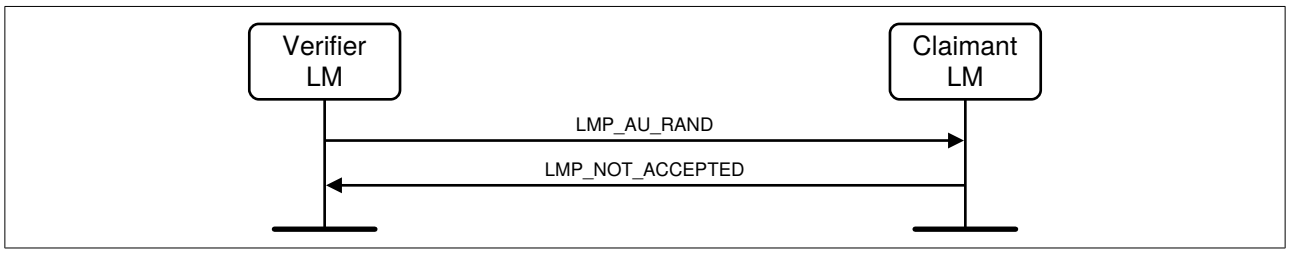
Note: There can be conflicting actions from the Central and the Peripheral which could lead to the two devices having different Authenticated Ciphering Offsets (ACOs, see [Vol 2] Part H, Section 6.1) when they calculate the encryption key. The following procedures assure that this cannot occur:

- procedure in Section 2.5.1 (Transaction Collision Resolution) when the Central and Peripheral simultaneously initiate an authentication.
- procedure in Section 4.2.5.1 when encryption parameters are being negotiated.

#### 4.2.1.2 Claimant has no link key (legacy authentication and secure authentication)

If the claimant does not have a link key associated with the verifier it shall send an LMP_NOT_ACCEPTED PDU with the Error_Code *PIN or Key Missing* (0x06) after receiving an LMP_AU_RAND PDU ([Vol 1] Part F, Section 2.6).

Note: This sequence is identical for legacy authentication and secure authentication.




*Sequence 32: Authentication fails. Claimant has no link key.*

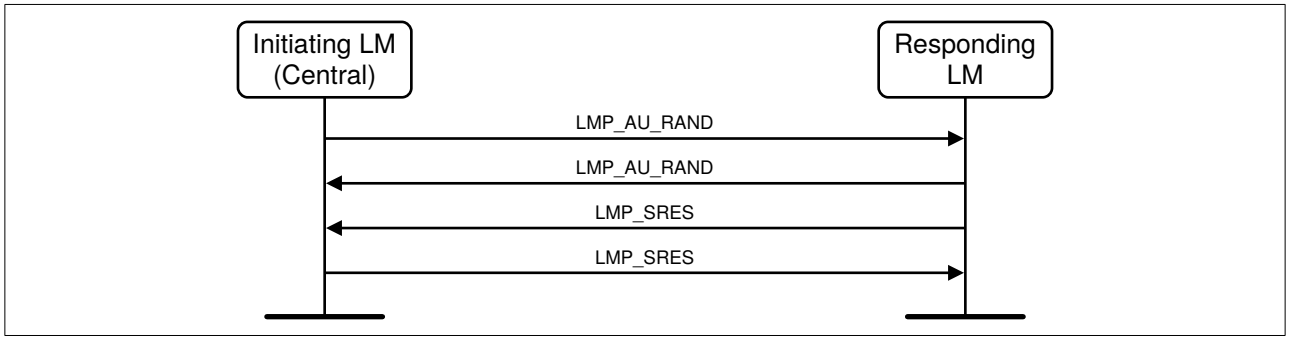
#### 4.2.1.3 Repeated attempts

The scheme described in [Vol 2] Part H, Section 5.1 shall be applied when an authentication fails.

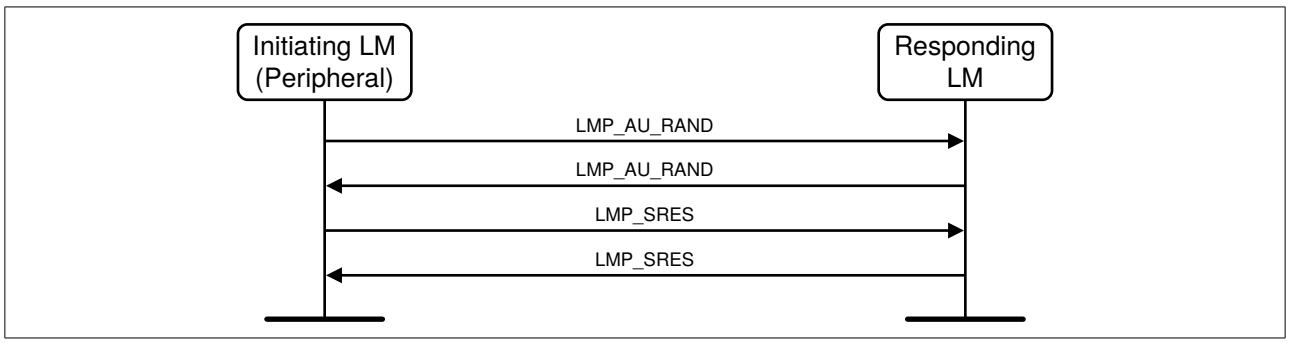
#### 4.2.1.4 Responder has link key (secure authentication)

Both the Central and Peripheral LM act as verifier and claimant. The initiator is the device that sends the LMP_AU_RAND PDU first.

The initiator sends an LMP_AU_RAND PDU to the responder. If the responder has a link key associated with the initiator, it shall respond with an LMP_AU_RAND PDU. The initiator and responder shall calculate the response. The Peripheral responds first with an LMP_SRES PDU containing SRES_P. The Central shall then respond with an LMP_SRES PDU containing SRES_C. The Central shall verify that the SRES_P sent by the Peripheral matches the SRES_P calculated by the Central. The Peripheral shall verify that the SRES_C sent by the Central matches the SRES_C calculated by the Peripheral. If the response is not correct, then the device receiving the wrong SRES value may end the connection by sending an LMP_DETACH PDU with the Error_Code *Authentication Failure* (0x05) (see Section 4.1.2).




*Sequence 33: Secure authentication. Responder has link key. Initiator is Central.*




*Sequence 34: Secure authentication. Responder has link key. Initiator is Peripheral.*

In the case where the Central and Peripheral both initiate the secure authentication sequence, the Central shall reject the Peripheral's LMP_AU_RAND PDU with an LMP_NOT_ACCEPTED PDU with the Error_Code *LMP Error Transaction Collision / LL Procedure Collision* (0x23). The Peripheral shall send another LMP_AU_RAND PDU with transaction ID set to 0 (Central).

Note: Secure Authentication is a mutual authentication.

### 4.2.2 Pairing

When two devices do not have a common link key an initialization key ($K_{init}$) shall be created using either the pairing or Secure Simple Pairing procedures. When pairing is used, $K_{init}$ shall be created based on a PIN, and a random number and a BD_ADDR. $K_{init}$ shall be created as specified in [Vol 2] Part H, Section 6.3. When both devices have calculated $K_{init}$ the link key shall be created, and a mutual authentication is performed. The pairing procedure starts with a device sending an LMP_IN_RAND PDU; this device is referred to as the "initiating LM" or "initiator" in Section 4.2.2.1 - Section 4.2.2.5. The other device is referred to as the "responding LM" or "responder". The PDUs used in the pairing procedure are:

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_IN_RAND | Random_Number |
| M | LMP_AU_RAND | Random_Number |
| M | LMP_SRES | Authentication_Rsp |
| M | LMP_COMB_KEY | Random_Number |
| M | LMP_UNIT_KEY | Key |

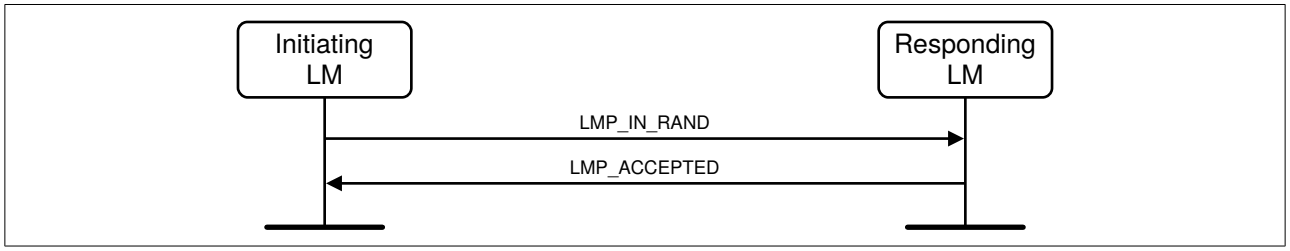
*Table 4.18: Pairing PDUs*

The Link Manager shall not send an LMP_UNIT_KEY PDU.

All sequences described in Section 4.2.2, including the mutual authentication after the link key has been created, shall form a single transaction. The transaction ID from the first LMP_IN_RAND shall be used for all subsequent sequences.

#### 4.2.2.1 Responder accepts pairing and has a variable PIN

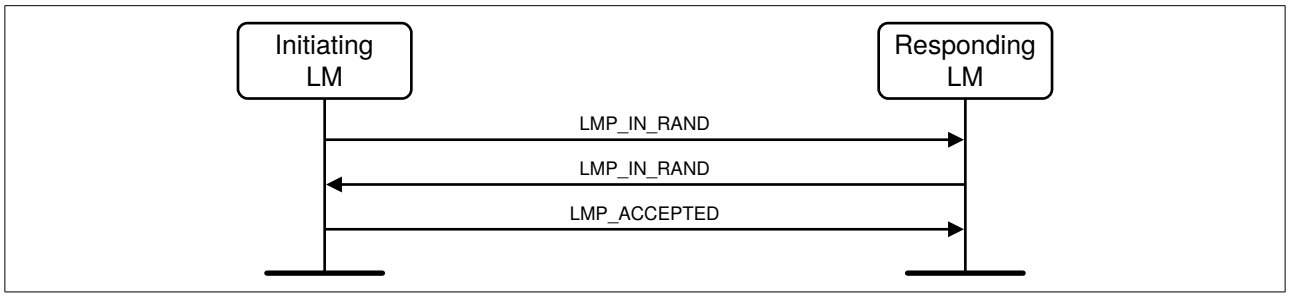
If the responder accepts the pairing and has a variable PIN, then it shall reply with an LMP_ACCEPTED PDU. Both devices shall then calculate $K_{init}$ based on the BD_ADDR of the responder and the procedure continues with creation of the link key; see Section 4.2.2.4.




*Sequence 35: Pairing accepted. Responder has a variable PIN. Initiator has a variable or fixed PIN.*

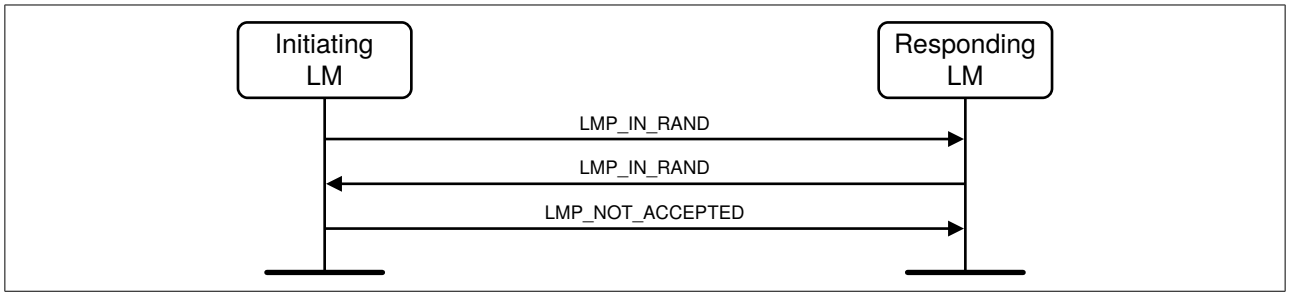
#### 4.2.2.2 Responder accepts pairing and has a fixed PIN

If the responder accepts the pairing and has a fixed PIN, then it shall generate a new random number and send it back in an LMP_IN_RAND PDU. If the initiator has a variable PIN, then it shall accept the LMP_IN_RAND PDU and shall respond with an LMP_ACCEPTED PDU. Both sides shall then calculate $K_{init}$ based on the last IN_RAND and the BD_ADDR of the initiator. The procedure continues with creation of the link key; see Section 4.2.2.4.




*Sequence 36: Responder has a fixed PIN and initiator has a variable PIN*

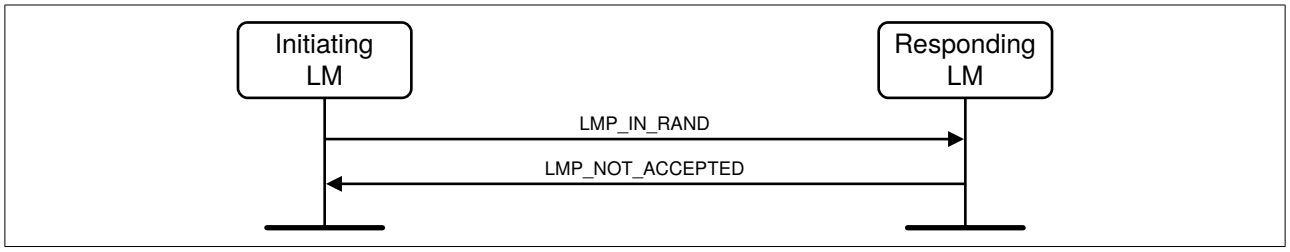
If the responder has a fixed PIN and the initiator also has a fixed PIN, then the second LMP_IN_RAND shall be rejected by the initiator sending an LMP_NOT_ACCEPTED PDU with the Error_Code *Pairing Not Allowed* (0x18).




*Sequence 37: Both devices have a fixed PIN*

#### 4.2.2.3 Responder rejects pairing

If the responder rejects pairing (e.g., because the user has disabled pairing on the device) it shall send an LMP_NOT_ACCEPTED PDU with the Error_Code *Pairing Not Allowed* (0x18) after receiving an LMP_IN_RAND PDU.




*Sequence 38: Responder rejects pairing*

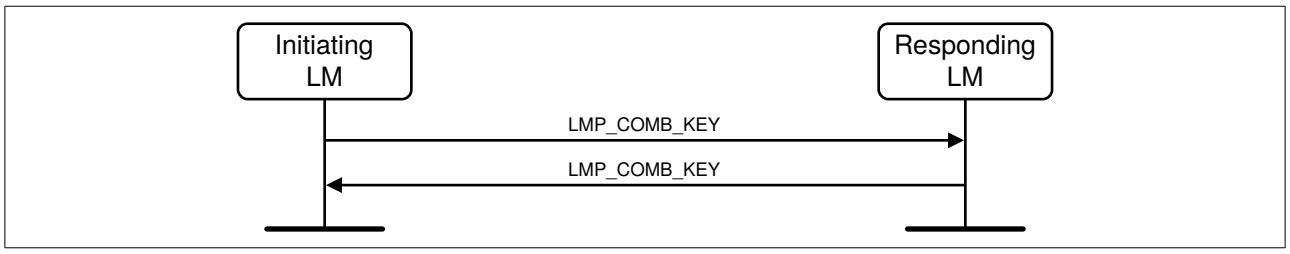
#### 4.2.2.4 Creation of the link key

When $K_{init}$ is calculated in both devices the link key shall be created. This link key will be used in the authentication between the two devices for all subsequent connections until it is changed; see Section 4.2.3 and Section 4.2.4. The link key created in the pairing procedure will be a combination key; both devices send an LMP_COMB_KEY PDU and the link key shall be calculated as described in [Vol 2] Part H, Section 3.2.

The content of the LMP_COMB_KEY PDU is LK_RAND bitwise XORed with $K_{init}$. Any device configured to use a combination key shall store the link key.

When the new link key has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. After mutual authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key.

Note: This will cause a new encryption key to be generated from the ACO created during the mutual authentication process and, when E0 encryption is used, the random number sent in the LMP_START_ENCRYPTION_REQ PDU which occurs in response to the resumption of encryption.




*Sequence 39: Creation of the link key*

If, in sequence 39, either device sends an LMP_UNIT_KEY PDU, the other device shall respond with an LMP_NOT_ACCEPTED PDU with the Error_Code set to *Pairing with Unit Key Not Supported* (0x29).

#### 4.2.2.5 Repeated attempts

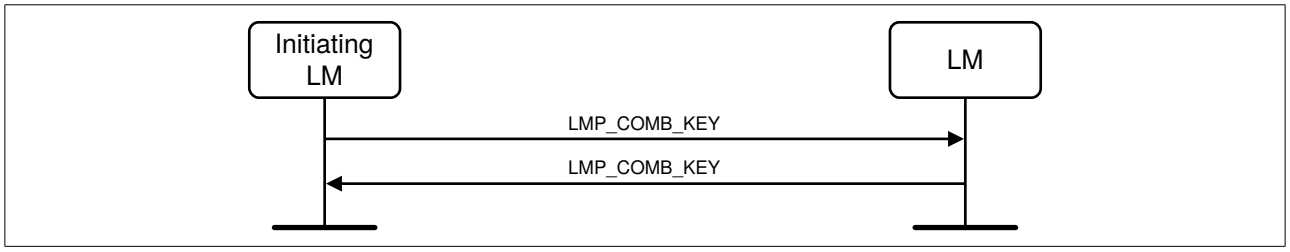
When the authentication after creation of the link key fails because of an incorrect Authentication_Rsp, the scheme described in [Vol 2] Part H, Section 5.1 shall be applied.

### 4.2.3 Change link key

The link key can be changed. The contents of the LMP_COMB_KEY PDU is protected by a bitwise XOR with the current link key.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_COMB_KEY | Random_Number |

*Table 4.19: Change link key PDU*




*Sequence 40: Successful change of the link key*

The new link key shall be stored and the old link key shall be discarded. The new link key shall be used as link key for all the following connections between the two devices until the link key is changed again. The new link key also becomes the current link key. It will remain the current link key until the link key is changed again, or until a temporary link key is created, see Section 4.2.4.

When the new link key has been created, mutual or secure authentication shall be performed to confirm that the same link key has been created in both devices.

If both devices support the Secure Connections (Controller Support) and Secure Connections (Host Support) features, the device that initiated the change link key shall initiate the Secure Authentication procedure. Otherwise, the first authentication in the mutual authentication is performed with the device that initiated the change link key as verifier. When finalized an authentication in the reverse direction is performed.

After mutual or secure authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key.

Note: This will cause a new encryption key to be generated from the ACO created during the mutual authentication process, and when E0 encryption is used, the random number sent in the LMP_START_ENCRYPTION_REQ PDU which occurs in response to the resumption of encryption.

### 4.2.4 Change current link key type

The current link key can be a semi-permanent link key or a temporary link key. It may be changed temporarily, but the change shall only be valid for the current connection, see [Vol 2] Part H, Section 3.1. Changing to a temporary link key is necessary if the piconet is to support encrypted broadcast. The current link key shall not be changed before the connection establishment procedure has completed.

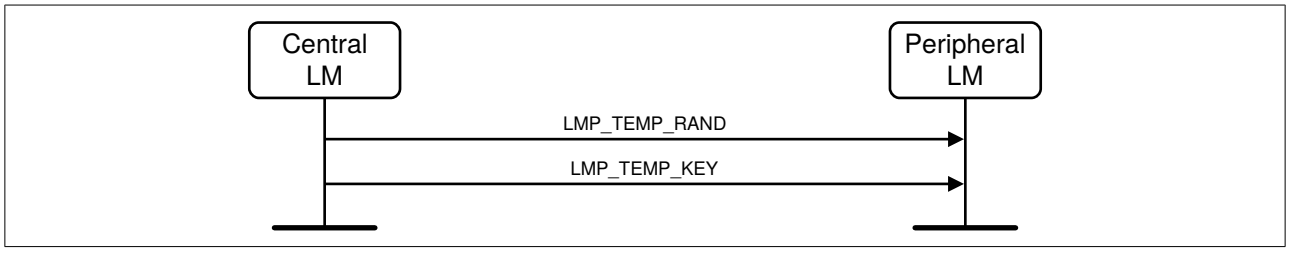
| M/O | PDU | Contents |
|---|---|---|
| O(23) | LMP_TEMP_RAND | Random_Number |
| O(23) | LMP_TEMP_KEY | Key |
| O(23) | LMP_USE_SEMI_PERMANENT_KEY | *none* |

*Table 4.20: Change current link key PDU*

#### 4.2.4.1 Change to a temporary link key

The Central starts by creating the temporary link key K_temp as specified in [Vol 2] Part H (EQ 4). Then the Central shall generate a random number, RAND, and shall send it to the Peripheral in an LMP_TEMP_RAND PDU. Both sides then calculate an overlay denoted OVL as OVL= $E_{22}$ (current link key, RAND, 16). The Central shall then send K_temp protected by XORing with OVL to the Peripheral in an LMP_TEMP_KEY PDU. The Peripheral calculates K_temp, based on OVL, that becomes the current link key. It shall be the current link key until the devices fall back to the semi-permanent link key, see Section 4.2.4.2.

Note: The terminology in this section is the same as used in [Vol 2] Part H, Section 3.2.8.




*Sequence 41: Change to a temporary link key*

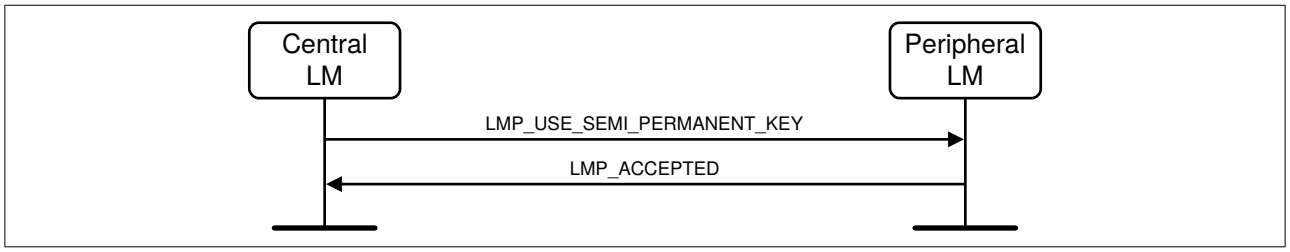
All sequences described in Section 4.2.4.1, including the mutual authentication after K_temp has been created, shall form a single transaction. The transaction ID shall be set to 0.

When the devices have changed to the temporary key, a mutual authentication shall be made to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication shall be performed with the Central as verifier. When finalized an authentication in the reverse direction is performed.

If the mutual authentication fails at either side, then the LM of the verifier shall start the detach procedure.

#### 4.2.4.2 Semi-permanent link key becomes current link key

After the current link key has been changed to K_temp, this change can be undone and the semi-permanent link key becomes the current link key again. If encryption is used on the link, the procedure to go back to the semi-permanent link key shall be immediately followed by the Central stopping encryption using the procedure described in Section 4.2.5.4. Encryption may be restarted by the Central according to the procedures in Section 4.2.5.3. This is to assure that encryption with encryption parameters known by other devices in the piconet is not used when the semi-permanent link key is the current link key.




*Sequence 42: Link key changed to the semi-permanent link key*

### 4.2.5 Encryption

If at least one authentication has been performed encryption may be used. Two encryption mechanisms are defined: E0 encryption (legacy) and AES-CCM encryption. If both devices support the Secure Connections (Controller Support) and Secure Connections (Host Support) features, then AES-CCM shall be used when encryption is enabled. If at least one device does not support both the Secure Connections (Controller Support) and Secure Connections (Host Support) features, then E0 shall be used when encryption is enabled.

In order for the Central to use the same encryption parameters for all Peripherals in the piconet where E0 encryption would be used it shall issue a temporary key, K_temp. The Central shall make this key the current link key for all Peripherals in the piconet where E0 encryption would be used before encryption is started, see Section 4.2.4. This is required if broadcast packets are to be encrypted.

Note: Packets encrypted with broadcast encryption can not be received by Peripherals that have AES-CCM encryption enabled. When the Central and all the Peripherals in the piconet support Secure Connections, broadcast packets will not be encrypted and can be received by Peripherals that support Secure Connections.

| M/O | PDU | Contents |
|---|---|---|
| O (2) | LMP_ENCRYPTION_MODE_REQ | Encryption_Mode |
| O (2) | LMP_ENCRYPTION_KEY_SIZE_REQ | Key_Size |
| O (2) | LMP_START_ENCRYPTION_REQ | Random_Number |
| O (2) | LMP_STOP_ENCRYPTION_REQ | *none* |
| O (42) | LMP_PAUSE_ENCRYPTION_REQ | *none* |
| O (42) | LMP_RESUME_ENCRYPTION_REQ | *none* |
| O (136) | LMP_PAUSE_ENCRYPTION_AES_REQ | Random_Number |

*Table 4.21: Encryption handling PDU*

All sequences described in Section 4.2.5 shall form a single transaction. The transaction ID from the LMP_ENCRYPTION_MODE_REQ PDU shall be used for all start encryption and stop encryption sequences.

Where the specification requires a device to resume encryption as part of one procedure and then pause it as part of a following procedure, the device may omit both the resume and the subsequent pause.

#### 4.2.5.1 Encryption mode

The Central and the Peripheral must agree upon whether to use encryption (Encryption_Mode=1 in LMP_ENCRYPTION_MODE_REQ) or not (Encryption_Mode=0). If the semi-permanent key is used, encryption shall only apply to point-to-point packets. If the temporary link key is used, encryption shall apply to both point-to-point packets and broadcast packets. If Central and Peripheral agree on the encryption mode, the Central continues to give more detailed information about the encryption.

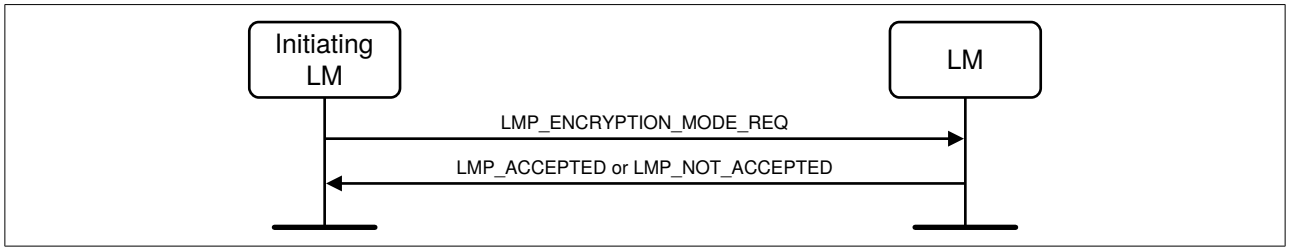
If a device receives an LMP_ENCRYPTION_MODE_REQ with an Encryption_Mode value of 2 then the device should treat it as if the Encryption_Mode value was 1.

If both devices support both the Secure Connections (Controller Support) and Secure Connections (Host Support) features, setting the Encryption_Mode to 0 shall not be allowed. If a device receives an LMP_ENCRYPTION_MODE_REQ PDU with Encryption_Mode set to 0, it shall respond with an LMP_NOT_ACCEPTED PDU with Error_Code *Encryption Mode Not Allowed* (0x25).

The initiating LM shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). The initiating device shall then send the

LMP_ENCRYPTION_MODE_REQ PDU. If the responding device accepts the change in encryption mode then it shall complete the transmission of the current packet on the ACL logical transport and shall then suspend transmission on the ACL-U logical link. The responding device shall then send the LMP_ACCEPTED PDU.

ACL-U logical link traffic shall only be resumed after the attempt to encrypt or decrypt the logical transport is completed, i.e. at the end of Sequence 43 (on failure), 45, 46, or 47.




*Sequence 43: Negotiation for encryption mode*

After a device has sent an LMP_ENCRYPTION_MODE_REQ PDU it shall not send an LMP_AU_RAND PDU before encryption is started. After a device has received an LMP_ENCRYPTION_MODE_REQ PDU and sent an LMP_ACCEPTED PDU it shall not send an LMP_AU_RAND PDU before encryption is started. If an LMP_AU_RAND PDU is sent violating these rules, the claimant shall respond with an LMP_NOT_ACCEPTED PDU with the Error_Code *LMP PDU Not Allowed* (0x24). This assures that devices will not have different ACOs when they calculate the encryption key. If the encryption mode is not accepted or the encryption key size negotiation results in disagreement the devices may send an LMP_AU_RAND PDU again.

#### 4.2.5.2 Encryption key size

Note: This section uses the same terms as in [Vol 2] Part H, Section 4.1.

The Central sends an LMP_ENCRYPTION_KEY_SIZE_REQ PDU including the suggested key size L_sug_c, that shall initially be equal to L_max_c. If L_min_p ≤ L_sug_c ≤ L_max_p, the Peripheral shall respond with an LMP_ACCEPTED PDU and L_sug_c shall be used as the key size.

If L_sug_c > L_max_p, the Peripheral shall send back an LMP_ENCRYPTION_KEY_SIZE_REQ PDU including the Peripheral's suggested key size L_sug_p set to L_max_p. If L_sug_c < L_min_p, the Peripheral shall send back an LMP_NOT_ACCEPTED PDU with the Error_Code *Unsupported LMP Parameter Value* (0x20) and the devices shall not communicate using encryption.

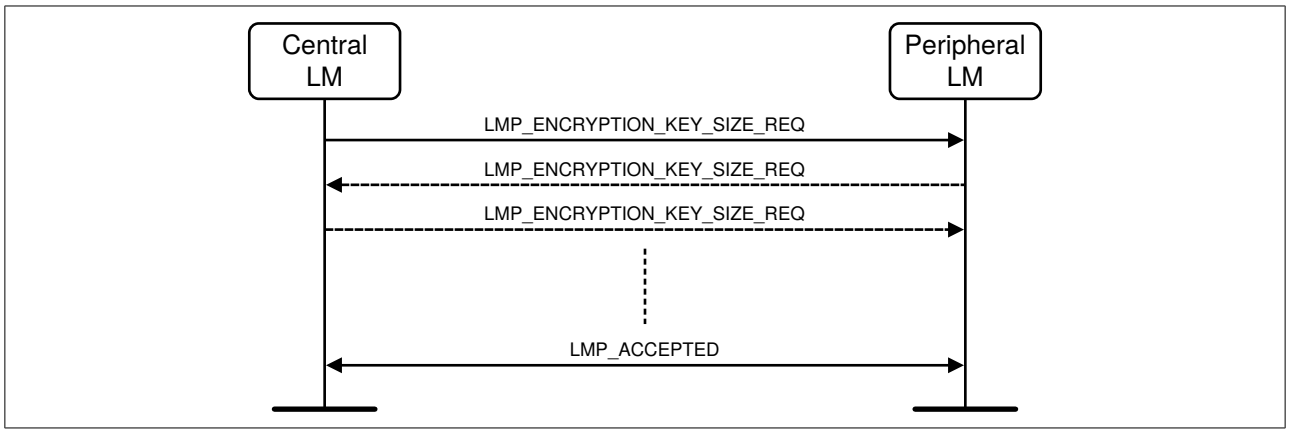
If the Peripheral sends back an LMP_ENCRYPTION_KEY_SIZE_REQ PDU, then the Central performs the corresponding test on the Peripheral's suggestion. This

procedure is repeated until a key size agreement is reached or it becomes clear that no such agreement can be reached. If an agreement is reached a device sends an LMP_ACCEPTED PDU and the key size in the last LMP_ENCRYPTION_KEY_SIZE_REQ PDU shall be used.

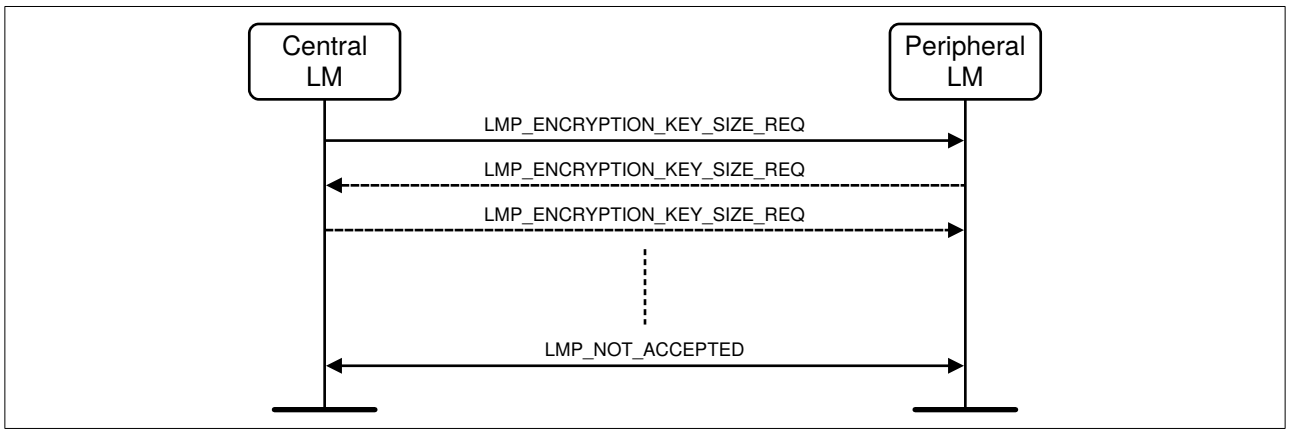
If a key size is agreed, encryption is then started; see Section 4.2.5.3. If an agreement is not reached a device sends an LMP_NOT_ACCEPTED PDU with the Error_Code *Unsupported LMP Parameter Value* (0x20) and the devices shall not communicate using encryption.

L_max_c, L_min_c, L_max_p, and L_min_p shall be set to at least 7 octets. The values of L_max_c, L_min_c, L_max_p, and L_min_p shall not change during an ACL connection between the Central and the Peripheral.

Note: If the Host of either the Central or the Peripheral uses services that require security mode 4 (see [Vol 3] Part C, Section 5.2.2.8), a key size longer than the key size negotiated by the two Link Managers can be enforced.




*Sequence 44: Encryption key size negotiation successful*




*Sequence 45: Encryption key size negotiation failed*

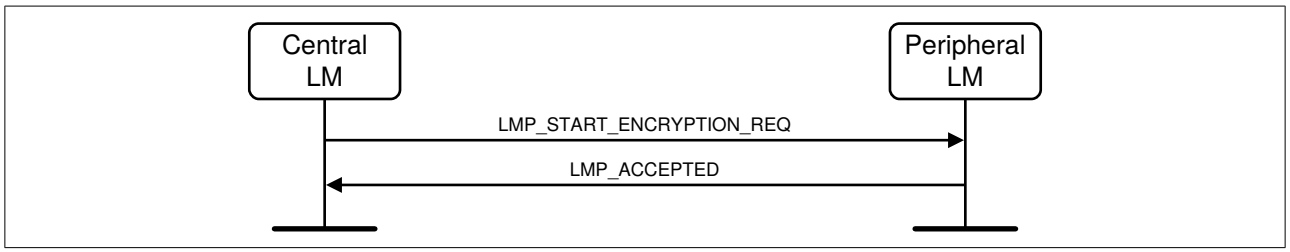
#### 4.2.5.3 Start encryption

To start encryption, the Central issues the random number EN_RAND and calculates the encryption key. See [Vol 2] Part H, Section 3.2.5. The random number shall be the same for all Peripherals in the piconet when broadcast encryption is used. The Central then sends an LMP_START_ENCRYPTION_REQ PDU, that includes EN_RAND.

The Peripheral shall calculate the encryption key when this message is received and shall acknowledge with an LMP_ACCEPTED PDU. For E0, the encryption key shall be calculated using the E3 algorithm (see [Vol 2] Part H, Section 6.4). For AES-CCM, the encryption key shall be calculated using the h3 algorithm (see [Vol 2] Part H, Section 7.7.6).

Note: For AES-CCM, the EN_RAND is not used when creating the encryption key.

If encryption has been paused, then this sequence shall not be used.




*Sequence 46: Start encryption*

Starting encryption shall be performed in three steps:

1. Central is configured to transmit unencrypted packets and to receive encrypted packets.
2. Peripheral is configured to transmit and receive encrypted packets.
3. Central is configured to transmit and receive encrypted packets.

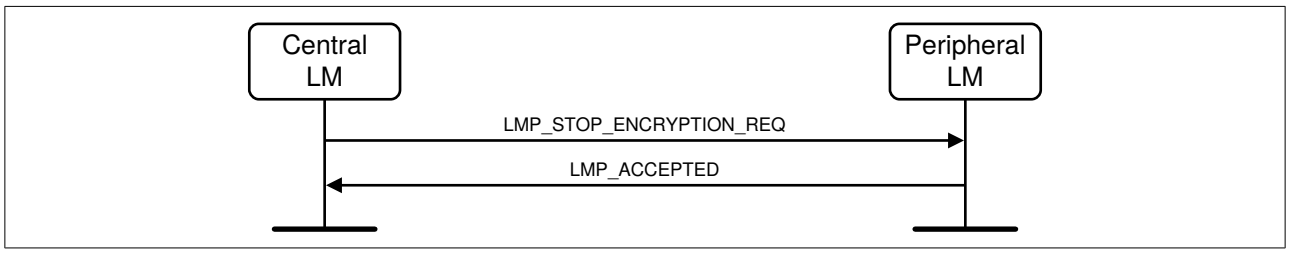
Between step 1 and step 2, Central-to-Peripheral transmission is possible. This is when an LMP_START_ENCRYPTION_REQ PDU is transmitted. Step 2 is triggered when the Peripheral receives this message. Between step 2 and step 3, Peripheral-to-Central transmission is possible. This is when an LMP_ACCEPTED PDU is transmitted. Step 3 is triggered when the Central receives this message.

#### 4.2.5.4 Stop encryption

To stop encryption a device shall send an LMP_ENCRYPTION_MODE_REQ PDU with the parameter Encryption_Mode equal to 0 (no encryption). The other device responds with an LMP_ACCEPTED PDU or an LMP_NOT_ACCEPTED PDU (the procedure is described in Sequence 43 in Section 4.2.5.1). If accepted, encryption shall be stopped

by the Central sending an LMP_STOP_ENCRYPTION_REQ PDU and the Peripheral shall respond with an LMP_ACCEPTED PDU according to Sequence 47.

If encryption has been paused, then this sequence shall not be used.




*Sequence 47: Stop encryption*

Stopping encryption shall be performed in three steps, similar to the procedure for starting encryption.

1. Central is configured to transmit encrypted packets and to receive unencrypted packets.
2. Peripheral is configured to transmit and receive unencrypted packets.
3. Central is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 Central to Peripheral transmission is possible. This is when an LMP_STOP_ENCRYPTION_REQ PDU is transmitted. Step 2 is triggered when the Peripheral receives this message. Between step 2 and step 3 Peripheral to Central transmission is possible. This is when an LMP_ACCEPTED PDU is transmitted. Step 3 is triggered when the Central receives this message.

#### 4.2.5.5 Pause encryption

For E0 encryption:

- To pause encryption without disabling encryption, a device shall finalize the transmission of the current ACL-U data packet and then send an LMP_PAUSE_ENCRYPTION_REQ PDU (with the transaction ID set to the role of the device at the time the LMP_PAUSE_ENCRYPTION_REQ PDU is sent).

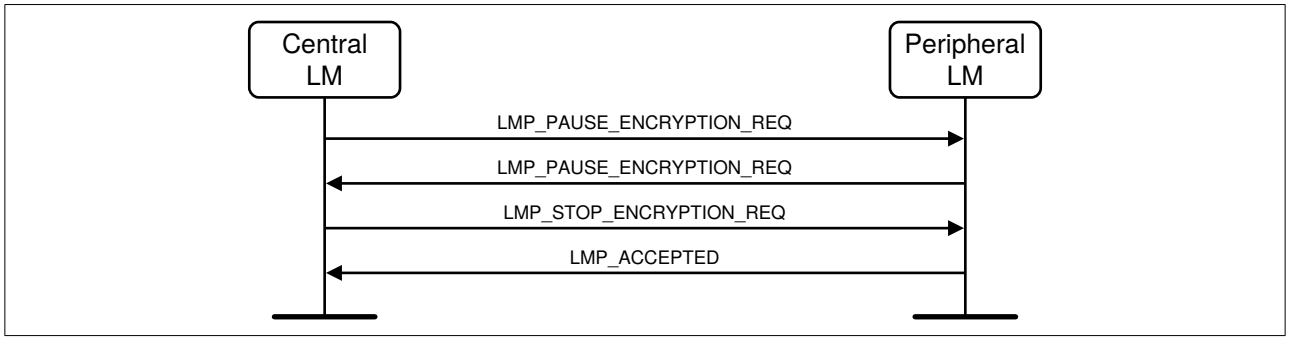
For AES-CCM encryption:

- To pause encryption without disabling encryption, a device shall finalize the transmission of the current ACL-U data packet and then send an LMP_PAUSE_ENCRYPTION_AES_REQ PDU (with the transaction ID set to the role of the device at the time the LMP_PAUSE_ENCRYPTION_AES_REQ PDU is sent). The LMP_PAUSE_ENCRYPTION_AES_REQ PDU includes a random number EN_RAND.

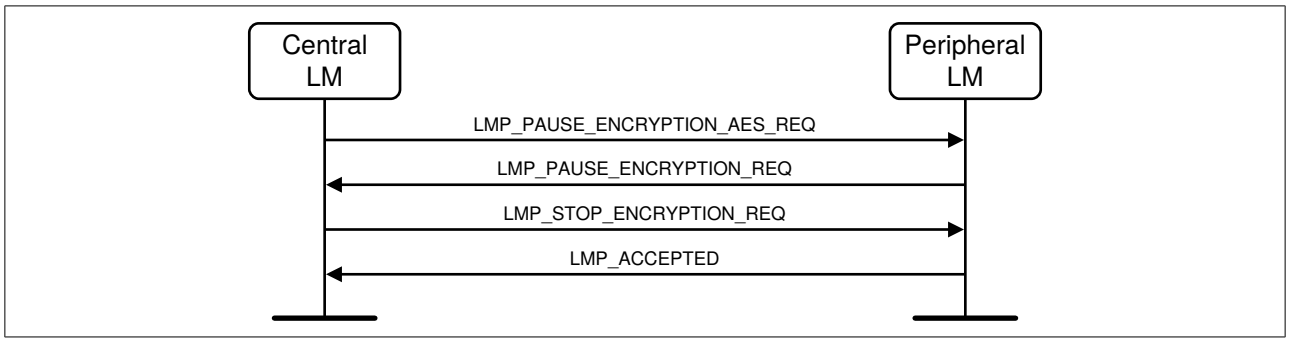
If the responding device is a Central, then the Central shall finalize the transmission of the current ACL-U data packet and then respond with an LMP_STOP_ENCRYPTION_REQ PDU to the Peripheral.

If the responding device is a Peripheral, then the Peripheral shall finalize the transmission of the current ACL-U data packet and then respond with an LMP_PAUSE_ENCRYPTION_REQ PDU. The Central shall respond to the LMP_PAUSE_ENCRYPTION_REQ PDU with an LMP_STOP_ENCRYPTION_REQ PDU to the Peripheral.

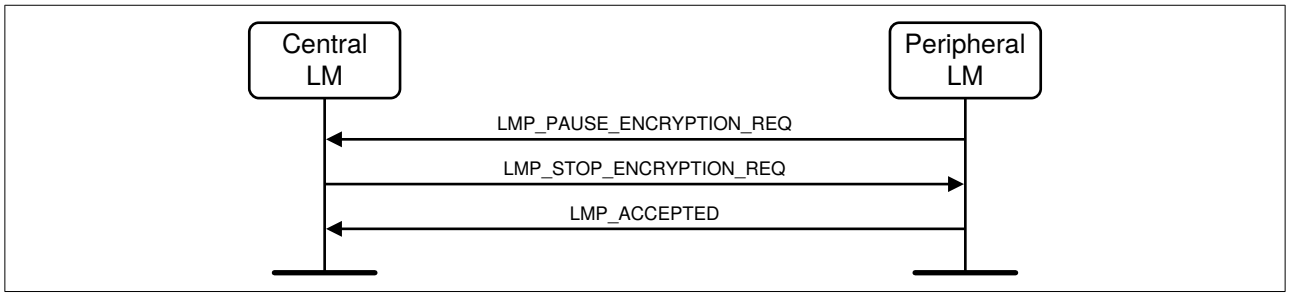
When the Peripheral receives the LMP_STOP_ENCRYPTION_REQ PDU it shall respond with an LMP_ACCEPTED PDU.



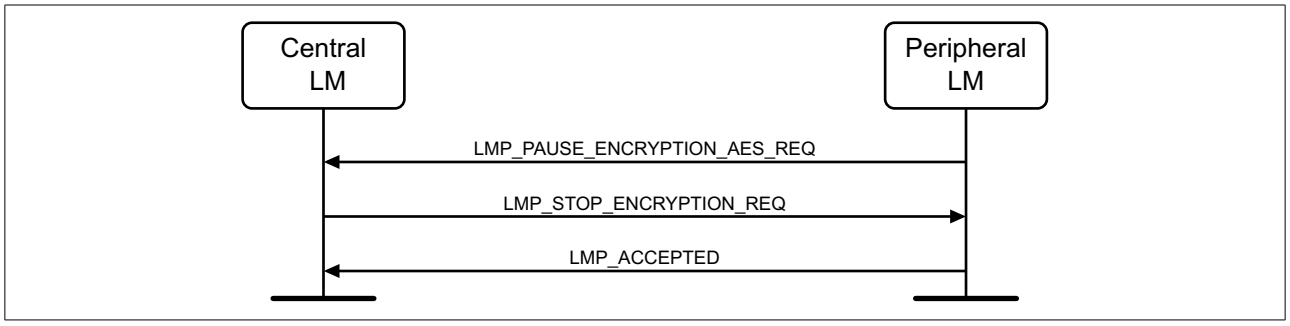

*Sequence 48: Central-initiated pausing of encryption (E0)*




*Sequence 49: Central-initiated pausing of encryption (AES-CCM)*




*Sequence 50: Peripheral-initiated pausing of encryption (E0)*




*Sequence 51: Peripheral-initiated pausing of encryption (AES-CCM)*

For E0 encryption:

- The LMP_PAUSE_ENCRYPTION_REQ PDU and LMP_STOP_ENCRYPTION_REQ PDU shall only be rejected when a transaction collision needs to be resolved.

For AES-CCM encryption:

- The LMP_PAUSE_ENCRYPTION_AES_REQ PDU and LMP_STOP_ENCRYPTION_REQ PDU shall only be rejected when a transaction collision needs to be resolved.

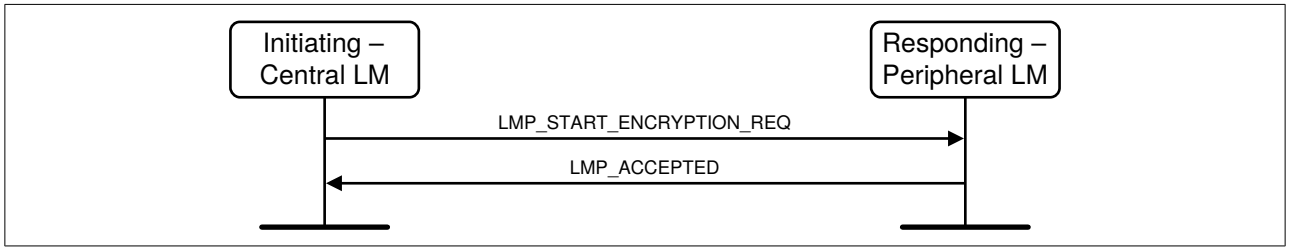
Pausing encryption shall be performed in three steps, similar to the procedure for stopping encryption.

1. Central is configured to transmit encrypted packets and to receive unencrypted packets.
2. Peripheral is configured to transmit and receive unencrypted packets.
3. Central is configured to transmit and receive unencrypted packets.

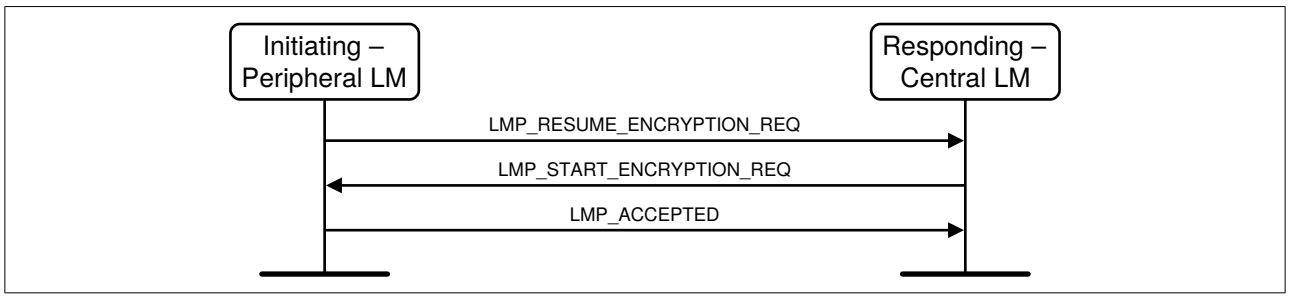
Between step 1 and step 2 Central-to-Peripheral transmission is possible. This is when the LMP_STOP_ENCRYPTION_REQ PDU is transmitted from the Central to the Peripheral. Step 2 is triggered when the Peripheral receives this message. Between step 2 and step 3 Peripheral-to-Central transmission is possible. This is when an LMP_ACCEPTED PDU is transmitted. Step 3 is triggered when the Central receives this message.

Note: A device can only restart ACL-U data traffic by resuming encryption using the procedures in Section 4.2.5.6.

#### 4.2.5.6 Resume encryption




*Sequence 52: Initiating Central LM resumes encryption*




*Sequence 53: Initiating Peripheral LM resumes encryption*

For E0 encryption:

- If the responding device is a Peripheral, then the Peripheral shall calculate the encryption key and respond with an LMP_ACCEPTED PDU.
- If the responding device is a Central, then the Central shall respond with an LMP_START_ENCRYPTION_REQ PDU. The Peripheral, upon receiving the LMP_START_ENCRYPTION_REQ PDU from the Central, shall calculate the encryption key and respond with an LMP_ACCEPTED PDU.

For AES-CCM encryption:

- If the resume encryption was the result of a Change Connection Link Key procedure, the LMs shall calculate a new encryption key using the h3 algorithm (see [Vol 2] Part H, Section 7.7.6).
- If the responding device is a Peripheral, then the Peripheral shall respond with an LMP_ACCEPTED PDU.
- If the responding device is a Central, then the Central shall respond with an LMP_START_ENCRYPTION_REQ PDU. The Peripheral, upon receiving the LMP_START_ENCRYPTION_REQ PDU from the Central, shall respond with an LMP_ACCEPTED PDU.

Note: When AES-CCM is used, the EN_RAND value in the LMP_START_ENCRYPTION_REQ PDU is not used.

The LMP_RESUME_ENCRYPTION_REQ PDU and the LMP_START_ENCRYPTION_REQ PDU shall not be rejected.

Resuming encryption shall be performed in three steps, similar to the procedure for starting encryption:

1. Central is configured to transmit unencrypted packets and to receive encrypted packets.
2. Peripheral is configured to transmit and receive encrypted packets.
3. Central is configured to transmit and receive encrypted packets.

Between step 1 and step 2, Central-to-Peripheral transmission is possible. This is when the LMP_START_ENCRYPTION_REQ PDU is transmitted from the Central. Step 2 is triggered when the Peripheral receives this message. Between step 2 and step 3, Peripheral-to-Central transmission is possible. This is when an LMP_ACCEPTED PDU is transmitted. Step 3 is triggered when the Central receives this message.

Note: For a Peripheral-initiated resumption of encryption, step 1 is not started when the Central has received the LMP_RESUME_ENCRYPTION_REQ PDU from the Peripheral, but when the Central sends the LMP_START_ENCRYPTION_REQ PDU.

Once encryption has been resumed, the device shall restart ACL-U traffic.

#### 4.2.5.7 Change encryption key or random number

If the encryption key or encryption random number need to be changed, or if the current link key needs to be changed according to the procedures in Section 4.2.3 and Section 4.2.4, encryption shall be paused and resumed after completion, using the procedures in Section 4.2.5.5 and Section 4.2.5.6, for the new parameters to be valid. If the Pause Encryption feature is not supported by both devices, encryption shall be stopped and re-started after completion, using the procedures in Section 4.2.5.3 and Section 4.2.5.4, for the new parameters to be valid.

#### 4.2.5.8 Encryption key refresh

When E0 encryption is used, the Link Manager shall refresh the encryption key within $2^{28}$ ticks of the Bluetooth clock from the previous start or resume of encryption. When AES-CCM encryption is used, the Link Manager shall refresh the encryption key before either the PayloadCounter or dayCounter roll over. To refresh the encryption key, the Link Manager shall Pause encryption using the procedure in Section 4.2.5.5 and immediately Resume encryption using the procedure in Section 4.2.5.6.

Note: The roll over will occur at least $2^{38}$ ticks of the Bluetooth clock after the previous start of encryption.

If the encryption key has not been refreshed before the rollover event, the link shall be terminated immediately.

### 4.2.6 Request supported encryption key size

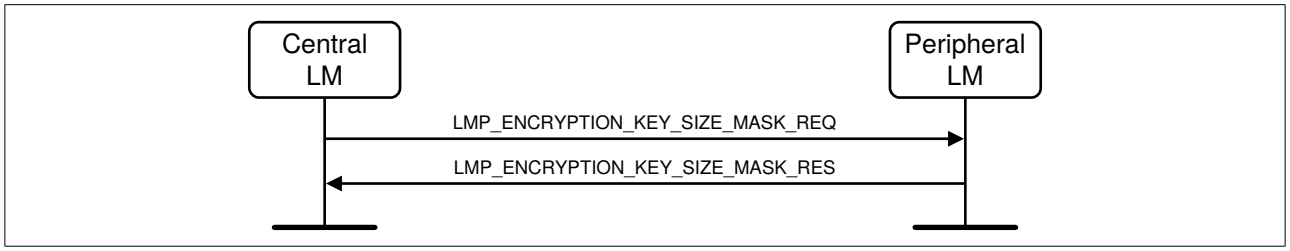
It is possible for the Central to request a Peripheral's supported encryption key sizes.

| M/O | PDU | Contents |
|---|---|---|
| O(23) | LMP_ENCRYPTION_KEY_SIZE_MASK_REQ | *none* |
| O(23) | LMP_ENCRYPTION_KEY_SIZE_MASK_RES | Key_Size_Mask |

*Table 4.22: Encryption key size request PDU*

The Central shall send an LMP_ENCRYPTION_KEY_MASK_REQ PDU to the Peripheral to obtain the Peripheral’s supported encryption key sizes.

The Peripheral shall return a bit mask indicating all broadcast encryption key sizes supported. The least significant bit shall indicate support for a key size of 1, the next most significant bit shall indicate support for a key size of 2 and so on up to a key size of 16. In all cases a bit set to 1 shall indicate support for a key size; a bit set to 0 shall indicate that the key size is not supported. Key sizes less than 7 shall not be supported.




*Sequence 54: Request for supported encryption key sizes*

### 4.2.7 Secure Simple Pairing

There are four stages defined in the Secure Simple Pairing LM process:

- IO capabilities exchange
- Public key exchange
- Authentication stage 1
- Authentication stage 2

The devices shall first exchange the IO capabilities to determine the proper algorithm to be used. Three algorithms have been specified: Numeric comparison, Passkey entry, Out of band.

In following sections, the device requesting the IO capabilities is referred to as the "Initiating LM" or "Initiator". The other device is referred to as the "LM" or "Responder". This designation remains throughout the entire Secure Simple Pairing procedure.

| M/O | PDU | Contents |
|---|---|---|
| O(51) | LMP_IO_CAPABILITY_REQ | IO_Capabilities,<br>OOB_Auth_Data,<br>Authentication_Requirements |
| O(51) | LMP_IO_CAPABILITY_RES | IO_Capabilities,<br>OOB_Auth_Data,<br>Authentication_Requirements |
| O(51) | LMP_SIMPLE_PAIRING_CONFIRM | Commitment_Value |
| O(51) | LMP_SIMPLE_PAIRING_NUMBER | Nonce_Value |
| O(51) | LMP_DHKEY_CHECK | Confirmation_Value |
| O(51) | LMP_NUMERIC_COMPARISON_FAILED | *none* |
| O(51) | LMP_OOB_FAILED | *none* |
| O(51) | LMP_KEYPRESS_NOTIFICATION | Notification_Type |
| O(51) | LMP_PASSKEY_ENTRY_FAILED | *none* |

*Table 4.23: Secure simple pairing PDUs*

When Secure Simple Pairing mode is disabled, the Link Manager shall reject all Secure Simple Pairing LMP procedures in this section with the error code *Secure Simple Pairing not Supported by Host* (0x37).

#### 4.2.7.1 IO capability exchange

The Link Managers shall use the local and remote IO capabilities to determine which association model shall be used.

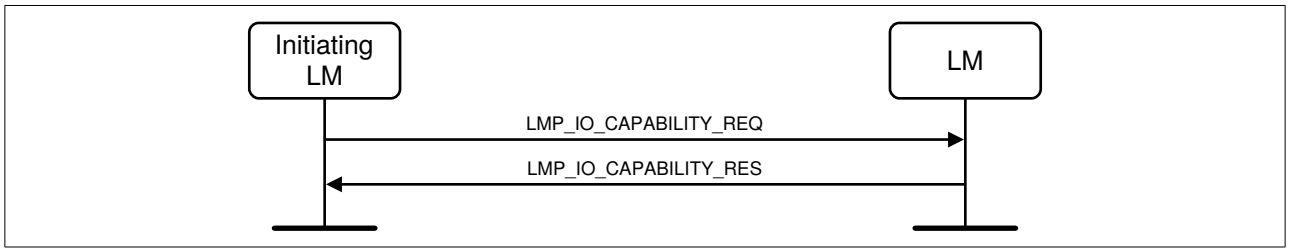
If Secure Simple Pairing mode is enabled, then the Initiator shall request the IO capabilities from the Host. The initiator shall send an LMP_IO_CAPABILITY_REQ PDU to the Responder. If Secure Simple Pairing mode is enabled on the responding device, then it shall reply with an LMP_IO_CAPABILITY_RES PDU containing its IO capabilities description.

The OOB_Auth_Data parameter shall be set as shown in Table 4.24.

| | | | Local Device | |
|---|---|---|---|---|
| | | | **Does not support Secure Connections** | **Supports Secure Connections** |
| **Remote Device** | **Does not have OOB Data** | | No OOB Data Received | No OOB Data Received |
| | **Does not support Secure Connections. P-192 OOB Data Available** | | OOB Data Received | OOB Data Received |
| | **Supports Secure Connections** | Only P-192 OOB Data Available | OOB Data Received | No OOB Data Received |
| | | P-192 OOB Data and P-256 OOB Data Available | OOB Data Received [1] | OOB Data Received |
| | | Only P-256 OOB Data Available | No OOB Data Received [1] | OOB Data Received |

*Table 4.24: Setting the OOB_Auth_Data parameter*

[1] A device that does not support Secure Connections cannot generate P-256 OOB data.

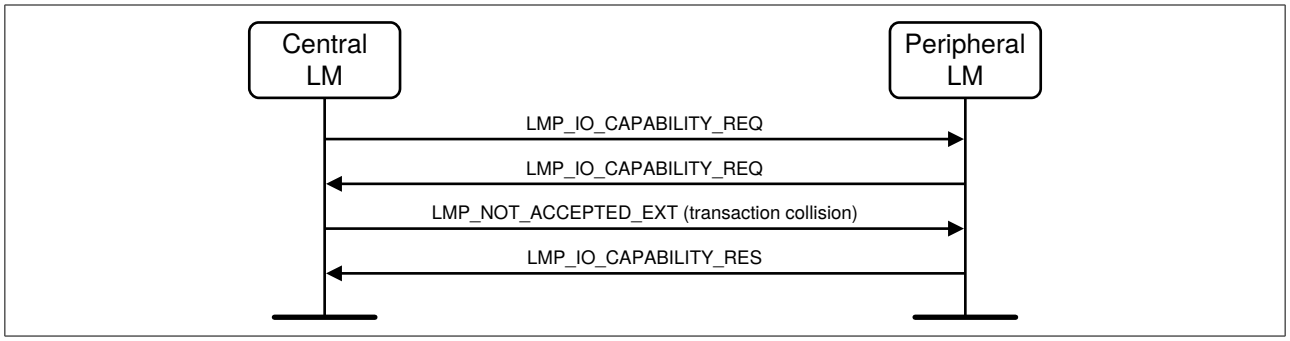



*Sequence 55: IO capability exchange*

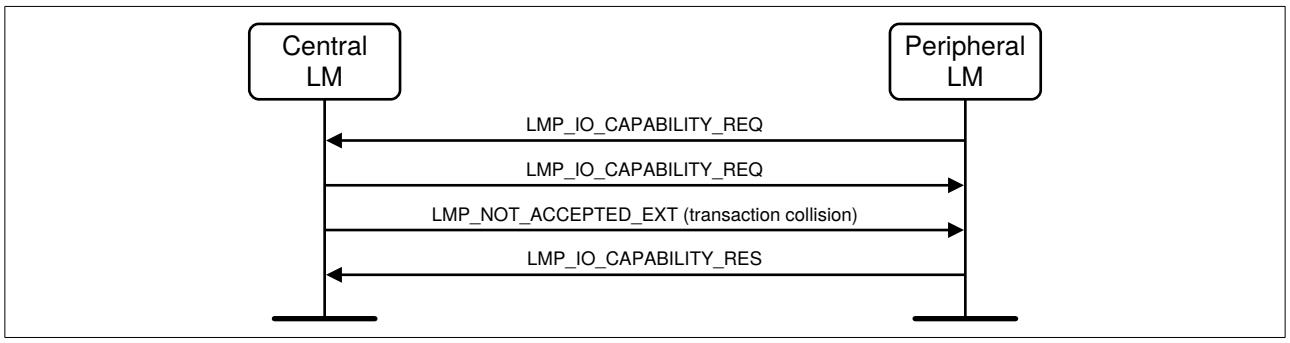
#### 4.2.7.1.1 IO capability exchange transaction collision and resolution

If both Link Managers attempt to become the initiator of the IO capability exchange, the Central's Link Manager shall send an LMP_NOT_ACCEPTED_EXT PDU with Error_Code *LMP Error Transaction Collision / LL Procedure Collision* (0x23). The Peripheral's Link Manager shall respond with the LMP_IO_CAPABILITY_RES PDU.

The Central's LM shall remain the initiating LM for the remainder of the Secure Simple Pairing sequences.




*Sequence 56: IO capability exchange with transaction collision (Central transmitting LMP_IO_CAPABILITY_REQ first) and resolution*




*Sequence 57: IO capability exchange with transaction collision (Peripheral transmitting LMP_IO_CAPABILITY_REQ first) and resolution*

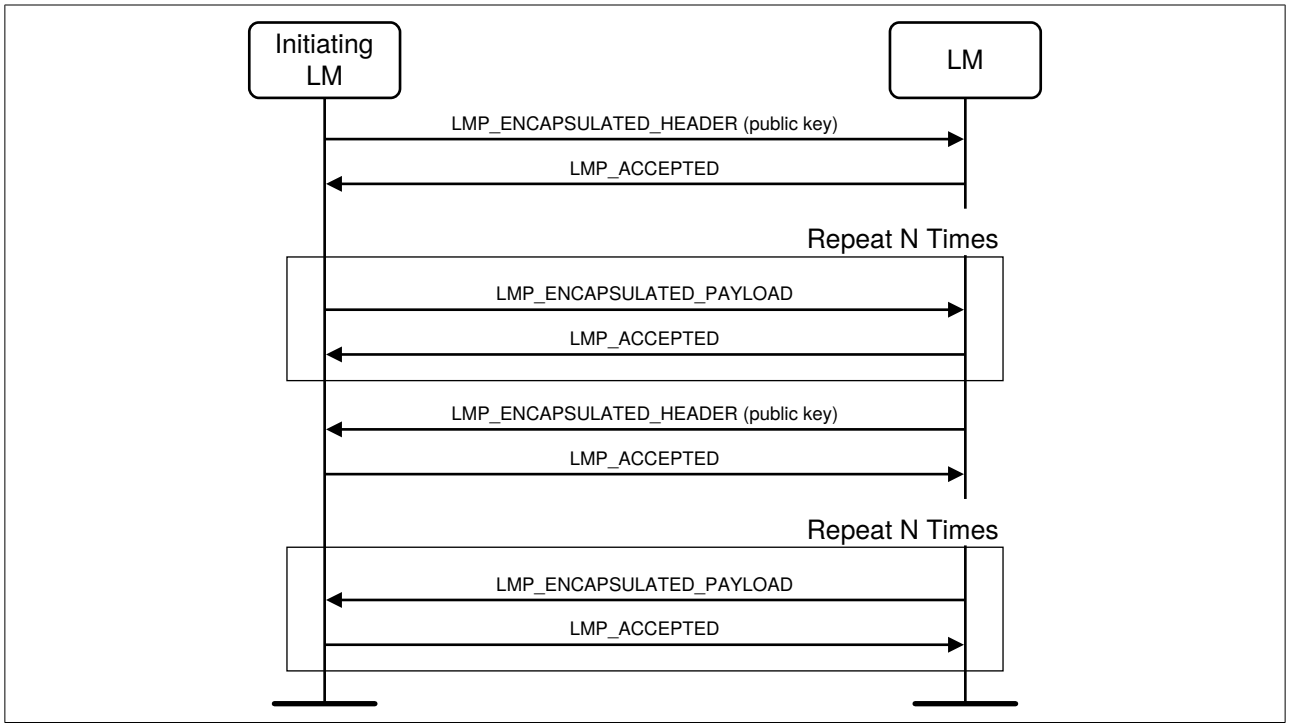
#### 4.2.7.2 Public key exchange

Once the IO capabilities are exchanged, public keys shall be exchanged between the two devices.

Since the public key size is longer than the payload body length of a DM1 packet, the exchange shall be done using the LMP_ENCAPSULATED_HEADER and LMP_ENCAPSULATED_PAYLOAD PDUs as defined in Section 4.1.12.

When at least one device does not support Secure Connections, the P-192 curve shall be used for calculating the public and private keys. When both devices support Secure Connections, the P-256 curve shall be used for calculating the public and private keys.

The Initiator shall first send its public key, and the Responder shall reply with its public key.




*Sequence 58: Public key exchange*

A public key shall be considered as received when the last LMP_ENCAPSULATED_PAYLOAD has been received and the associated LMP_ACCEPTED PDU has been sent.

The device can then start computing its Diffie Hellman Key.

#### 4.2.7.3 Authentication stage 1

One of the following procedures shall be used in Authentication stage 1:

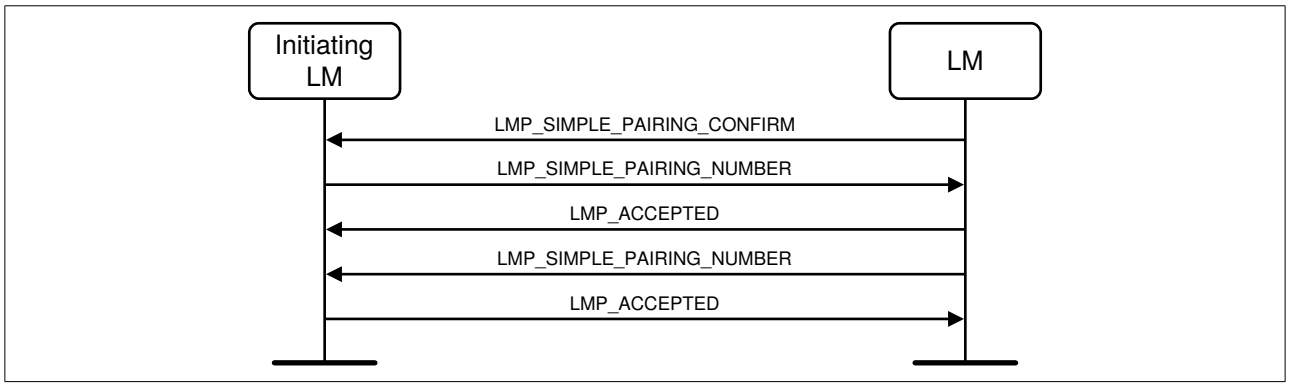
- If one or both devices have the OOB_Auth_Data parameter set to Received, the Out-of-Band procedure shall be used.
- If both devices have the Authentication_Requirements parameter set to one of the man-in-the middle (MITM) Protection Not Required options, the Numeric Comparison procedure shall be used.
- If one or both devices have the Authentication_Requirements parameter set to one of the MITM Protection Required options, the Passkey Entry procedure shall be used if the either the local or remote IO Capability is set to KeyboardOnly and the other IO capability is not set to NoInputNoOutput. Otherwise, the Numeric Comparison authentication procedure shall be used.

#### 4.2.7.3.1 Authentication stage 1: Numeric Comparison

Once public keys have been exchanged, both devices shall generate a random number.

The Responder shall compute its commitment as defined in [Vol 2] Part H, Section 7.7.1, and shall send this value to the Initiator by using LMP_SIMPLE_PAIRING_CONFIRM PDU. The Initiator shall then send an LMP_SIMPLE_PAIRING_NUMBER PDU with its generated random number. The Responder shall acknowledge by sending an LMP_ACCEPTED PDU.

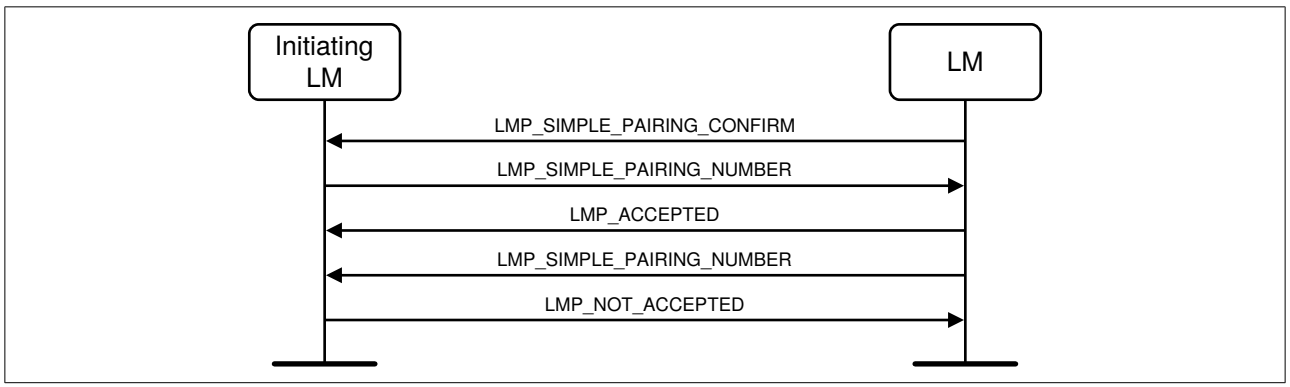
The Responder shall then send an LMP_SIMPLE_PAIRING_NUMBER containing its own generated random number. Upon reception, the Initiator shall calculate the commitment as defined in [Vol 2] Part H, Section 7.7.1 and compare it to the one received previously with the LMP_SIMPLE_PAIRING_CONFIRM PDU. If both values are equal, the Initiator shall respond with an LMP_ACCEPTED PDU.




*Sequence 59: Numeric Comparison Authentication: Commitment check success*

##### *4.2.7.3.1.1 Commitment check failure*

If the calculated commitment by the Initiator is not equal to the received commitment, the Initiator shall abort the Secure Simple Pairing process by sending an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure."

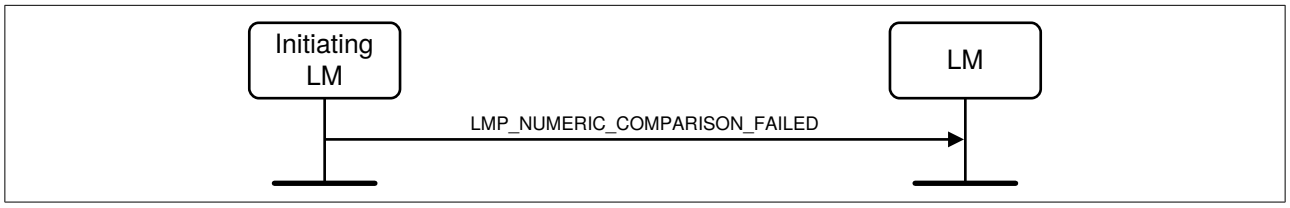



*Sequence 60: Numeric Comparison authentication: Commitment check failure*

#### 4.2.7.3.1.2 Numeric Comparison failure on Initiator side

If the user on the initiating side indicates that the confirm values do not match (e.g., as indicated by the HCI_User_Confirmation_Request_Negative_Reply command) the initiating LM shall send an LMP_NUMERIC_COMPARISON_FAILURE PDU.

The Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

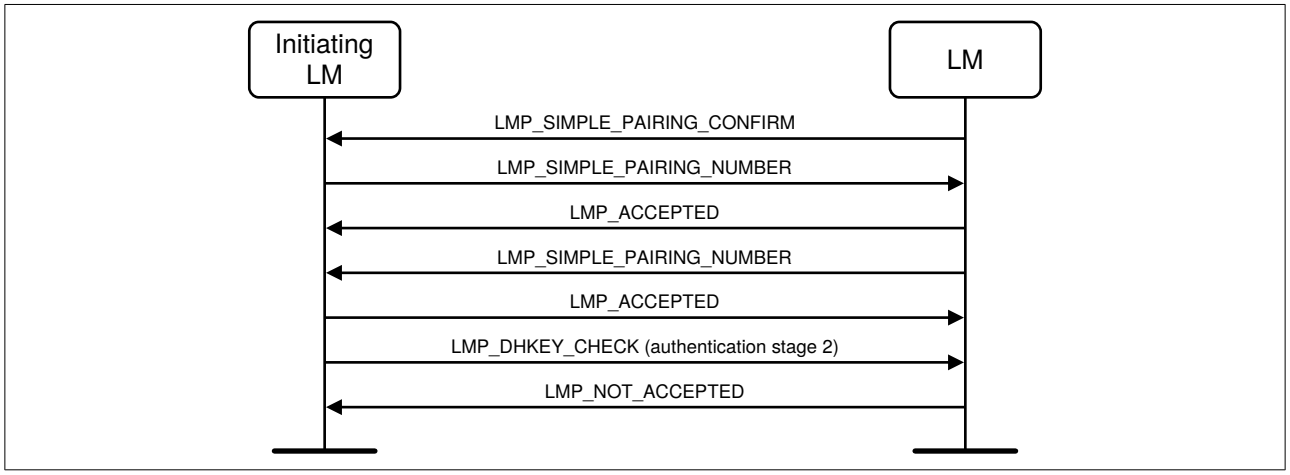



*Sequence 61: Authentication stage 1: Numeric Comparison failure on Initiator side*

#### 4.2.7.3.1.3 Numeric Comparison failure on Responder side

If the user on the responding side indicates that the confirm values do not match (e.g., as indicated by the HCI_User_Confirmation_Request_Negative_Reply command) the responding LM shall send an LMP_NOT_ACCEPTED PDU in response to the LMP_DHKEY_CHECK PDU sent in authentication stage 2 (see Section 4.2.7.4).

The Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

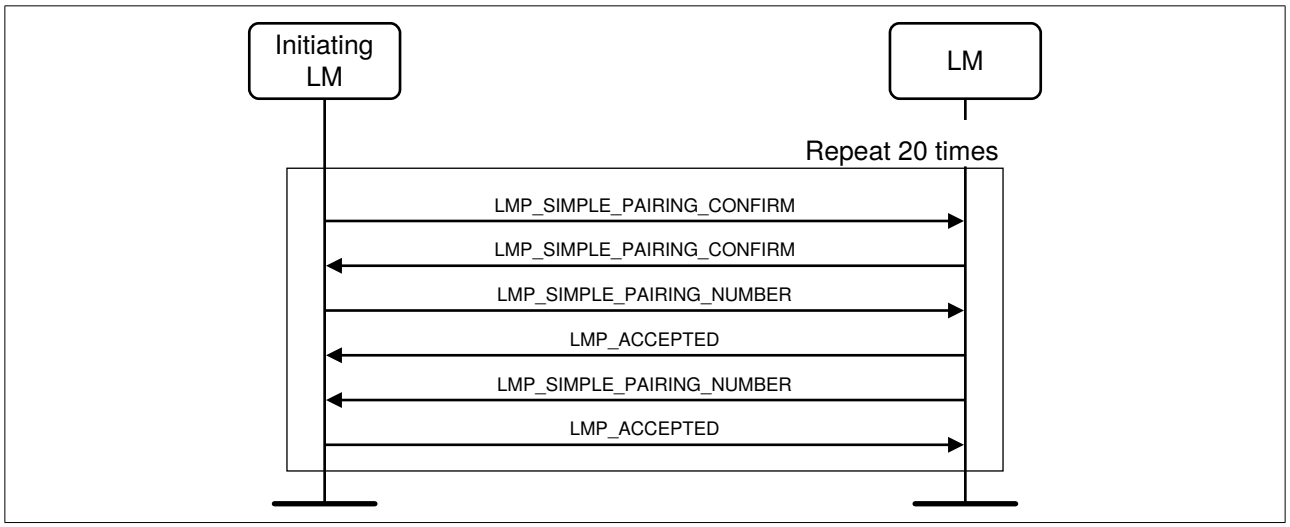



*Sequence 62: Authentication stage 1: Numeric Comparison failure on Responder side*

#### 4.2.7.3.2 Authentication stage 1: Passkey Entry Authentication

The initiating LM shall start the following procedure after receiving the Passkey Reply from the Host. This procedure shall be repeated 20 times:

1. The Initiator and the Responder shall generate a new random number.
2. The Initiator and the Responder shall calculate the local commitment value using the exchanged public keys, the local random number, and the passkey from the local Host, according to [Vol 2] Part H, Section 7.7.1.
3. The Initiator shall send an LMP_SIMPLE_PAIRING_CONFIRM PDU with the commitment it calculated in step 2.
4. The Responder shall respond with an LMP_SIMPLE_PAIRING_CONFIRM PDU with the commitment it calculated in step 2.
5. The Initiator shall then send an LMP_SIMPLE_PAIRING_NUMBER PDU with the random number it generated in step 1.
6. The Responder shall then calculate commitment from the exchanged public keys, the random number it received and the passkey from the local Host, according to [Vol 2] Part H, Section 7.7.1. If the calculated commitment and the received commitment are equal, the Responder shall reply with an LMP_ACCEPTED PDU.
7. The Responder shall then send an LMP_SIMPLE_PAIRING_NUMBER PDU with the random value it generated in step 1.
8. The Initiator shall calculate the commitment using the exchanged public keys, the random number it received, and the passkey from the local Host, according to [Vol 2] Part H, Section 7.7.1. If the calculated commitment is equal to the received commitment, the Initiator shall reply with an LMP_ACCEPTED PDU.



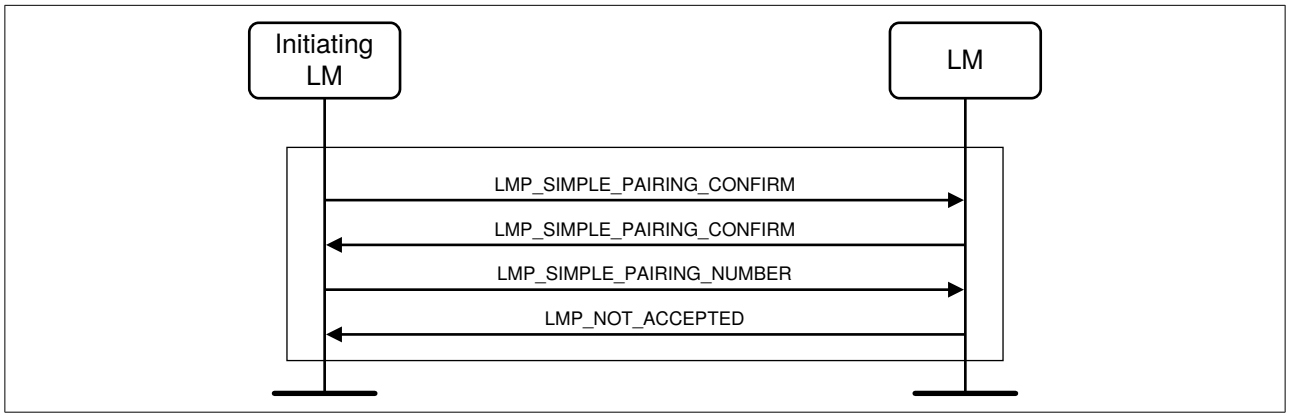

*Sequence 63: Authentication passkey entry*

When this procedure has successfully passed 20 times, the Secure Simple Pairing process continues with the Authentication step 2 as described in Section 4.2.7.4.

#### 4.2.7.3.2.1 Commitment check failure on the Responder side

If during one of the 20 repetitions, the commitment calculated by the Responder is not equal to the one received from the Initiator (step 6), the Responder shall abort the Secure Simple Pairing process by sending an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure."

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.

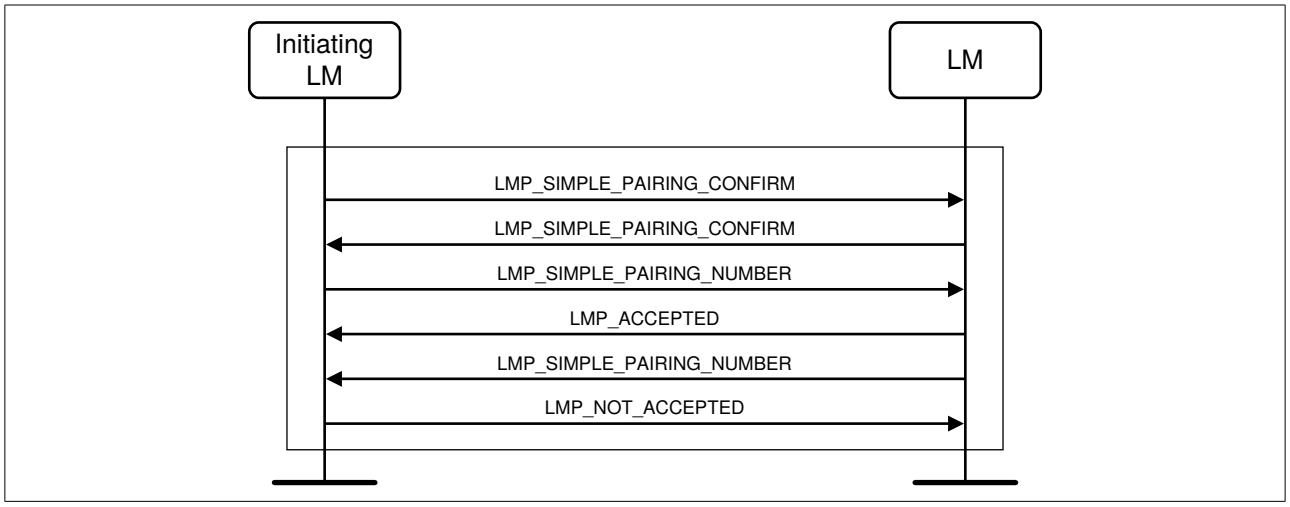



*Sequence 64: Authentication passkey entry: Commitment check failure on the Responder side*

#### 4.2.7.3.2.2 Commitment check failure on the Initiator side

If during one of the 20 repetitions, the commitment calculated by the Initiator is not equal to the one received from the Responder (step 8), the Initiator shall abort the Secure Simple Pairing process by sending an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure".

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.

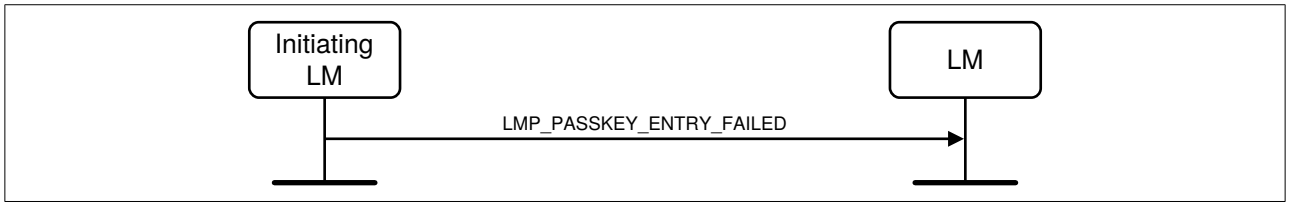



*Sequence 65: Authentication passkey entry: Commitment check failure on the Initiator side*

#### *4.2.7.3.2.3 Passkey Entry failure on Initiator side*

If the initiating side indicates that the passkey was not entered or canceled (e.g., as indicated by the HCI_User_Passkey_Request_Negative_Reply command) the initiating LM shall send an LMP_PASSKEY_ENTRY_FAILED PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

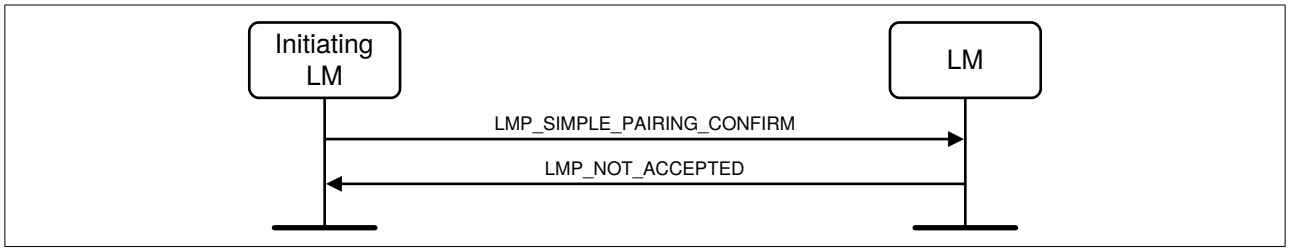



*Sequence 66: Authentication stage 1: Passkey Entry failure on Initiator side*

### 4.2.7.3.3 Passkey Entry failure on Responding side

If the responding side indicates that the passkey was not entered or canceled (e.g., as indicated by the HCI_User_Passkey_Request_Negative_Reply command), the responding LM shall send an LMP_NOT_ACCEPTED PDU in response to the LMP_SIMPLE_PAIRING_CONFIRM PDU.

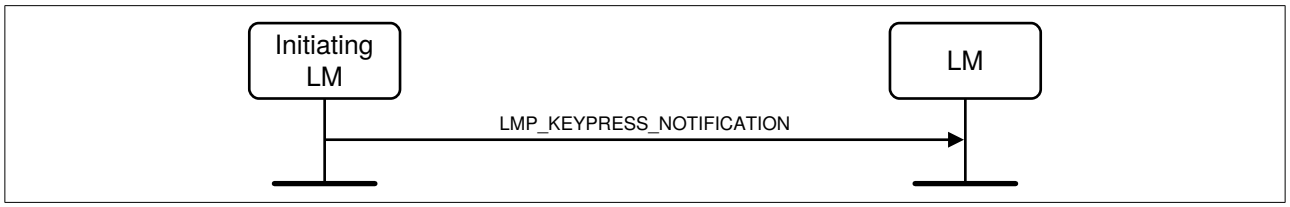
The Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 67: Authentication stage 1: Passkey Entry failure on Responding side*

#### 4.2.7.3.4 Keypress notifications

A Controller that allows the Host to change its IO capabilities shall send notifications on key presses to the remote side using the LMP_KEYPRESS_NOTIFICATION PDU when the Host sets the IO capabilities to KeyboardOnly IO and when Secure Simple Pairing is supported on the Host and Controller.




*Sequence 68: Keypress notifications*

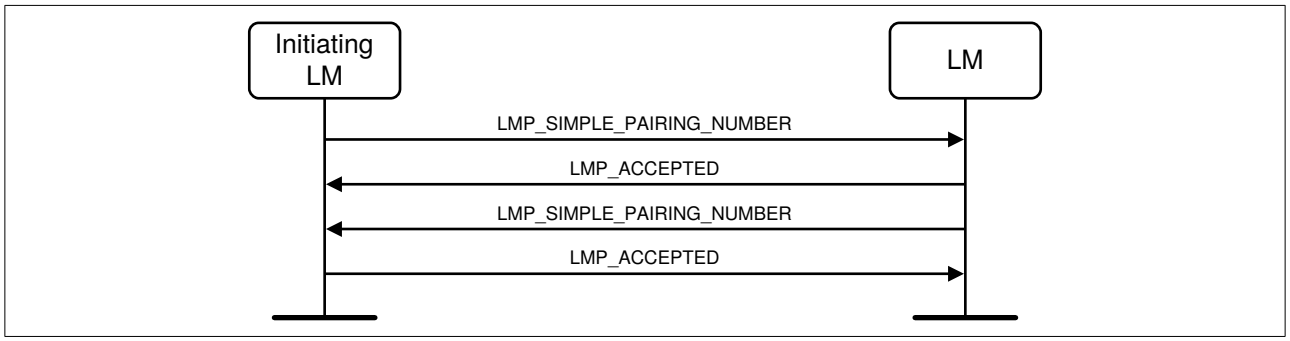
#### 4.2.7.3.5 Authentication stage 1: OOB

Upon reception of the OOB information (as defined in [Vol 2] Part H, Section 7.2.2) from the Host, the devices shall compare the received commitment from its Host, with the one calculated using the secret number received from the Host and the public key received from the remote device. If the local Host has not set the OOB Authentication Data Present, the LM shall set the remote secret number to zero and base the subsequent calculations on this value.

If the commitment check on the initiator is valid, the Initiator shall then generate a random number (nonce) and send it to the Responder using an LMP_SIMPLE_PAIRING_NUMBER PDU. If the commitment succeeds, the Responder shall acknowledge by sending an LMP_ACCEPTED PDU otherwise it shall send an LMP_NOT_ACCEPTED PDU. The Responder shall then generate a random number (nonce) and send it to the Initiator using an LMP_SIMPLE_PAIRING_NUMBER PDU. If the commitment succeeds, the Initiator shall acknowledge by sending an LMP_ACCEPTED PDU otherwise it shall send an LMP_NOT_ACCEPTED PDU.

If the commitment values don't match in the Initiator, the procedure in Section 4.2.7.3.5.2 shall apply.

If the commitment values don't match in the Responder, the procedure in Section 4.2.7.3.5.1 shall apply.



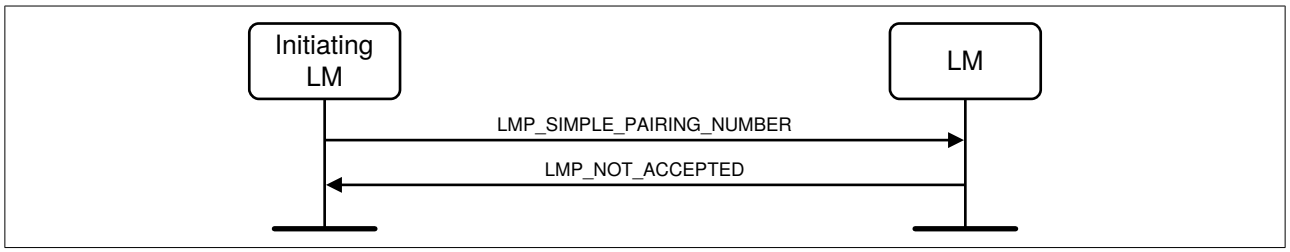

*Sequence 69: Authentication OOB: Only one device is OOB-r capable*

When these operations have been passed successfully, Secure Simple Pairing procedures continue with Authentication step 2 as described in Section 4.2.7.4.

#### *4.2.7.3.5.1 Commitment check failure on the Responder side*

If the commitment received OOB from the Host is not equal to the calculated commitment, the Responder shall send an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure" in response to the LMP_SIMPLE_PAIRING_NUMBER PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

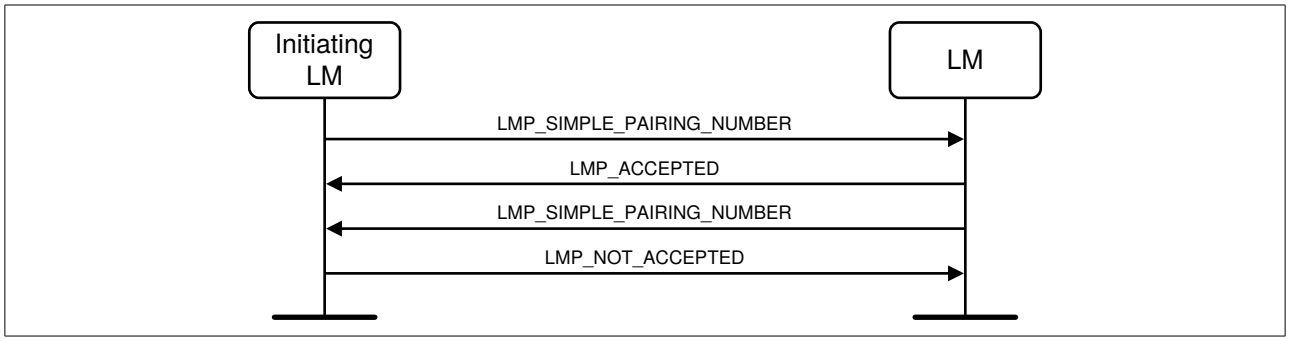



*Sequence 70: Authentication stage 1 OOB: Commitment check failure on the Responder side*

#### *4.2.7.3.5.2 Commitment check failure on the Initiator side*

If the commitment received OOB from the Host is not equal to the calculated commitment, the Initiator shall send an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure" in response to the LMP_SIMPLE_PAIRING_NUMBER PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

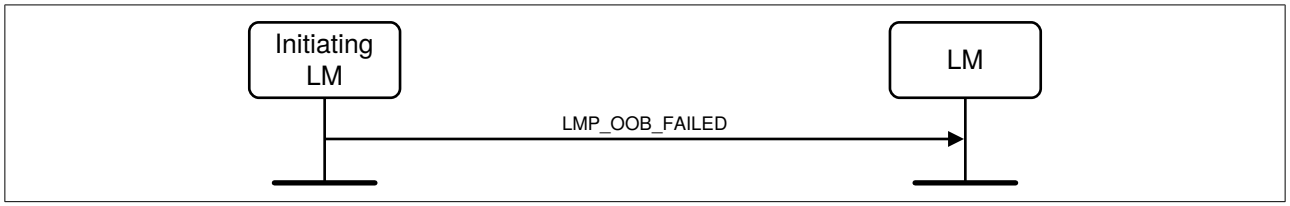



*Sequence 71: Authentication stage 1 OOB: Commitment check failure on the Initiator side*

#### 4.2.7.3.6 Out-of-band information not available on the Initiator side

If the Host on the initiating side does not have out-of-band information the Initiator shall send an LMP_OOB_FAILED PDU.

The Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 72: Authentication stage 1 OOB: OOB information not available on the Initiator side*

### 4.2.7.4 Authentication stage 2: DHKey check

At this stage, both devices compute new confirmation values based on Diffie-Hellman key and previously exchanged information according to [Vol 2] Part H, Section 7.7.4.

The Initiator shall send an LMP_DHKEY_CHECK PDU to the Responder. If the Initiator has determined that the received public key is invalid (see [Vol 2] Part H, Section 7.6), the PDU shall include a confirmation value that is different from the computed confirmation value (for example, substituting a randomly generated number). Otherwise, the PDU shall include the computed confirmation value.

Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4, or if the received public key is invalid (see [Vol 2] Part H, Section 7.6), then the Responder shall follow the procedure in Section 4.2.7.4.1. Otherwise it shall reply with an LMP_ACCEPTED PDU.

The Responder shall then send an LMP_DHKEY_CHECK PDU, including the confirmation value it has computed, to the Initiator. Upon reception, if the received value is not equal to the one calculated according to [Vol 2] Part H, Section 7.7.4,

or if the received public key is invalid (see [Vol 2] Part H, Section 7.6), then the Initiator shall follow the procedure in Section 4.2.7.4.1. Otherwise it shall reply with an LMP_ACCEPTED PDU.

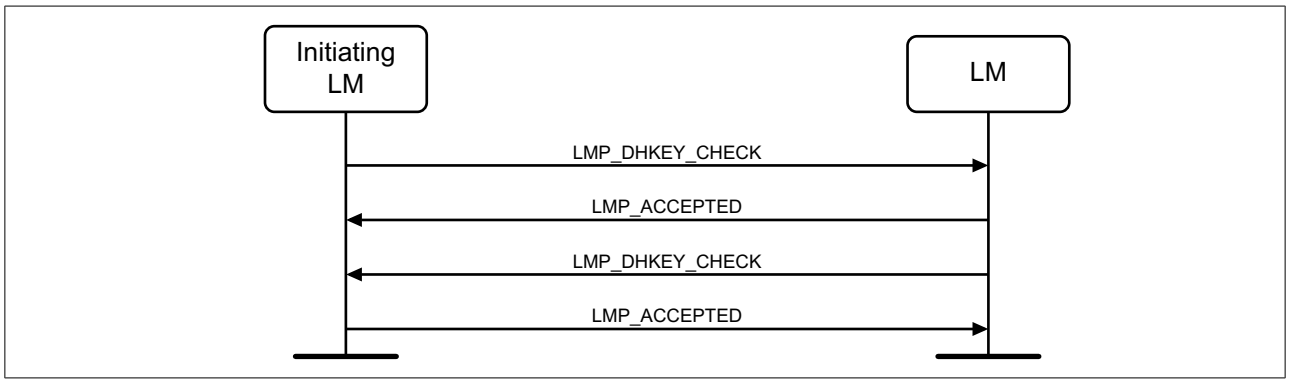
At this point, both devices shall compute the link key according to [Vol 2] Part H, Section 7.7.3.

If at least one device does not support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start standard mutual authentication as described in Section 4.2.1.1

If both devices support both the Secure Connections (Controller Support) and the Secure Connections (Host Support) features, the Initiator shall then start secure authentication as described in Section 4.2.1.4.

After secure authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key.

Note: This will cause a new encryption key to be generated using the h3 function including the ACO created during the secure authentication process.



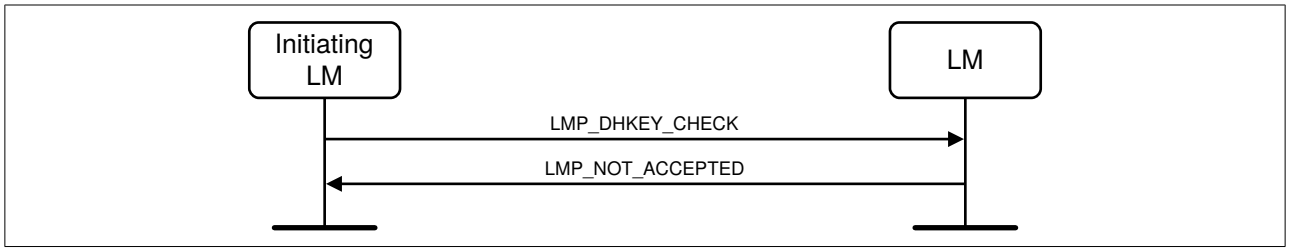

*Sequence 73: DHKey check*

A device that detects an invalid public key (see [Vol 2] Part H, Section 7.6) from the peer at any point during the Secure Simple Pairing process shall fail the pairing process and therefore not create a link key.

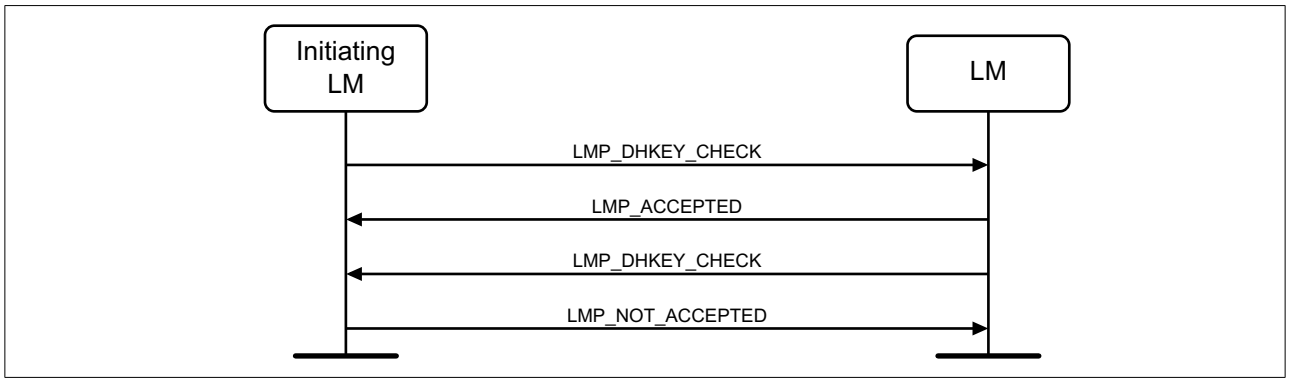
#### 4.2.7.4.1 Check failure

If either Link Manager receives a public key that is invalid (see [Vol 2] Part H, Section 7.6), it shall send an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure". If either Link Manager receives a confirmation value via LMP that is not equal to the one it has calculated according to [Vol 2] Part H, Section 7.7.4, it shall send an LMP_NOT_ACCEPTED PDU with reason "Authentication Failure".

The Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 74: DHKey check: Check failure on the Responder side*




*Sequence 75: DHKey check: Check failure on the Initiator side*

*4.2.7.4.1.1 [This section is no longer used]*

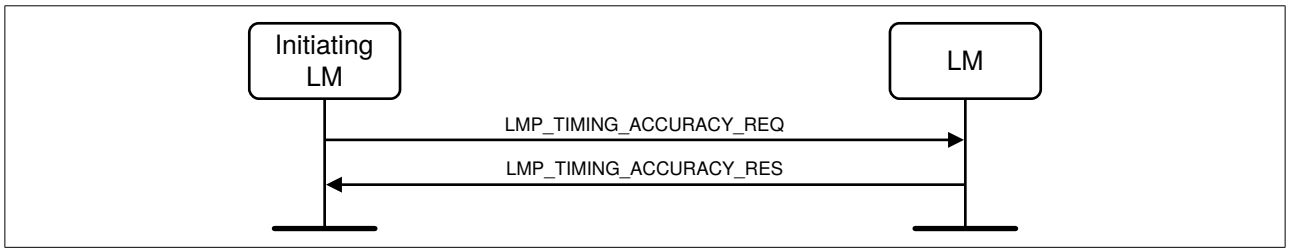
## 4.3 Informational requests

### 4.3.1 Timing accuracy

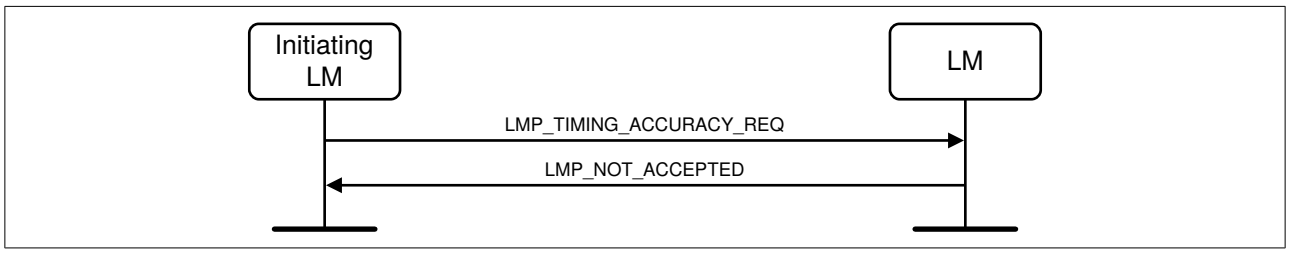
LMP supports requests for the timing accuracy. This information can be used to minimize the scan window during piconet physical channel re-synchronization (see [Vol 2] Part B, Section 2.2.5.2). The timing accuracy parameters returned are the long term drift measured in ppm and the long term jitter measured in µs of the worst case clock used. These parameters are fixed for a certain device and shall be identical when requested several times. Reported time accuracy shall not include changes caused by performing Piconet Clock Adjustment. If timing accuracy information has not been received from the remote device, the worst-case values (drift = 250 ppm and jitter = 10 µs) shall be used.

| M/O | PDU | Contents |
|---|---|---|
| O(4) | LMP_TIMING_ACCURACY_REQ | *none* |
| O(4) | LMP_TIMING_ACCURACY_RES | Drift<br>Jitter |

*Table 4.25: Request limited timing PDU*




*Sequence 76: The requested device supports timing accuracy information*




*Sequence 77: The requested device does not support timing accuracy information*

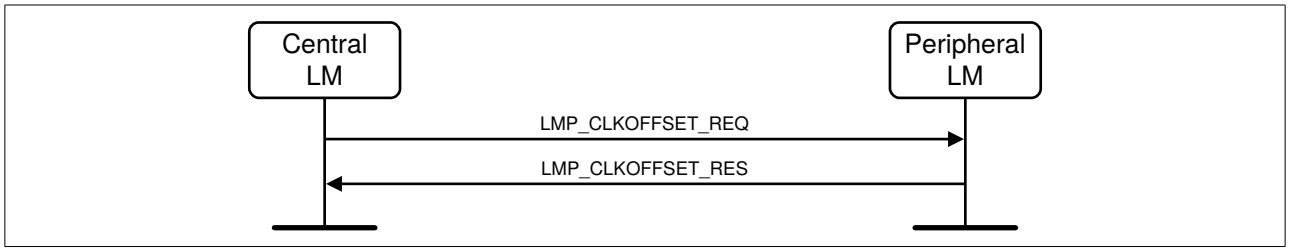
### 4.3.2 Clock offset

The clock offset can be used to speed up the paging time the next time the same device is paged. The Central can request the clock offset at anytime following a successful Baseband Paging procedure (i.e., before, during or after connection setup). The clock offset shall be defined by the following equation:

$(CLKN_{16\text{-}2\ Peripheral} - CLKN_{16\text{-}2\ Central})$ *mod* $2^{15}$.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_CLKOFFSET_REQ | *none* |
| M | LMP_CLKOFFSET_RES | Clock_Offset |

*Table 4.26: PDUs used for clock offset request*




*Sequence 78: Clock offset requested*

### 4.3.3 LMP version

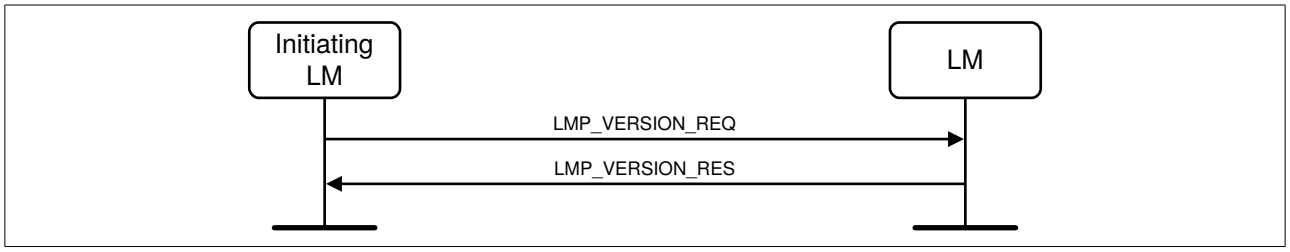
LMP supports requests for the version of the LM protocol. The LMP_VERSION_REQ and LMP_VERSION_RES PDUs contain three parameters: Version, Company_Identifier and Subversion. Version specifies the version of the Bluetooth LMP specification that the device supports. All companies that create a unique implementation of the LM shall have their own Company_Identifier. The same company is also responsible for the administration and maintenance of the Subversion. It is recommended that each company has a unique Subversion for each RF/BB/LM implementation. For a given Version and Company_Identifier, the values of the Subversion shall increase each time a new implementation is released. For both Company_Identifier and Subversion the value 0xFFFF means that no valid number applies. There is no ability to negotiate the version of the LMP. The sequence below is only used to exchange the parameters. LMP version can be requested at anytime following a successful Baseband Paging procedure.

Note: A given value for Version does not indicate that the device supports all the features in the corresponding version of the specification; the relevant feature bits (see Section 4.3.4) should be checked instead.

Note: A larger value for Version does not necessarily indicate a higher version of the specification.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_VERSION_REQ | Version<br>Company_Identifier<br>Subversion |
| M | LMP_VERSION_RES | Version<br>Company_Identifier<br>Subversion |

*Table 4.27: PDUs used for LMP version request*




*Sequence 79: Request for LMP version*

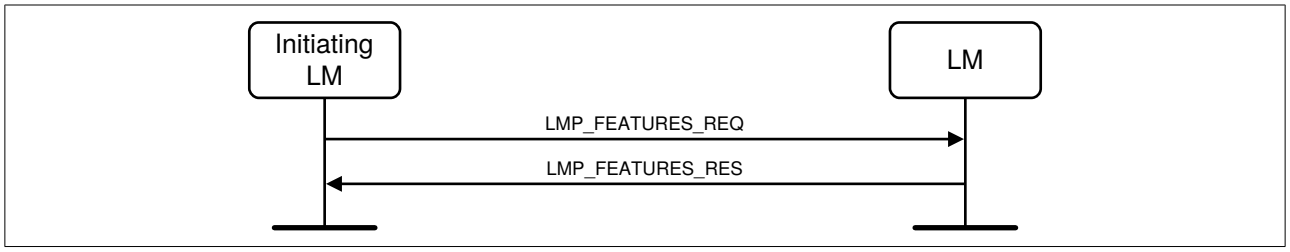
### 4.3.4 Supported features

The supported features may be requested at anytime following a successful Baseband Paging procedure by sending the LMP_FEATURES_REQ PDU. Upon reception of an LMP_FEATURES_REQ PDU, the receiving device shall return an LMP_FEATURES_RES PDU.

The extended features mask provides support for more than 64 features. Support for the extended features mask is indicated by the presence of the appropriate bit in the LMP features mask. The LMP_FEATURES_REQ_EXT and LMP_FEATURES_RES_EXT PDUs operate in precisely the same way as the LMP_FEATURES_REQ and LMP_FEATURES_RES PDUs except that they allow the various pages of the extended features mask to be requested. The LMP_FEATURES_REQ_EXT may be sent at any time following the exchange of the LMP_FEATURES_REQ and LMP_FEATURES_RSP PDUs.

The LMP_FEATURES_REQ_EXT PDU contains a Features_Page parameter that specifies which page is requested and the contents of that page for the requesting device. Pages are numbered from 0 to 255 with page 0 corresponding to the normal features mask. Each page consists of 64 bits. If a device does not support any page number it shall return an Extended_Features parameter with every bit set to 0. It also contains the maximum features page number containing any non-zero bit for this device. The recipient of an LMP_FEATURES_REQ_EXT PDU shall respond with an LMP_FEATURES_RES_EXT PDU containing the same page number and the appropriate features page along with its own maximum features page number.

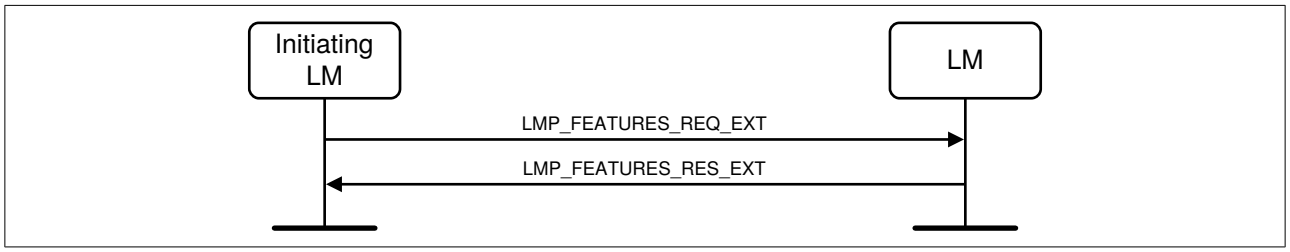
If the extended features request is not supported then all bits in all extended features pages for that device shall be assumed to be zero.




*Sequence 80: Request for supported features*

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_FEATURES_REQ | Features |
| M | LMP_FEATURES_RES | Features |
| O(63) | LMP_FEATURES_REQ_EXT | Features_Page<br>Max_Supported_Page<br>Extended_Features |
| O(63) | LMP_FEATURES_RES_EXT | Features_Page<br>Max_Supported_Page<br>Extended_Features |

*Table 4.28: PDUs used for features request*




*Sequence 81: Request for extended features*

### 4.3.5 Name request

LMP supports name request to another device. The name is a user-friendly name associated with the device and consists of a maximum of 248 bytes coded according to the UTF-8 standard (more specifically, the type utf8s{248z} defined in [Vol 1] Part E, Section 2.9.3). The name is fragmented over one or more DM1 packets. When an LMP_NAME_REQ PDU is sent, a Name_Offset indicates which fragment is expected. The corresponding LMP_NAME_RES PDU carries the same Name_Offset,

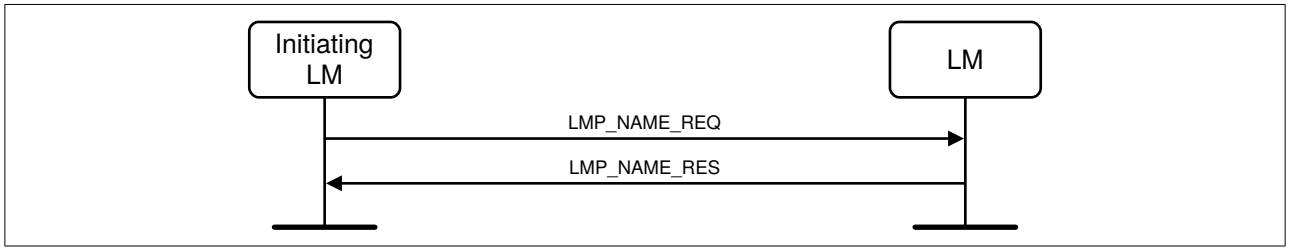
the Name_Length indicating the total number of bytes in the name of the device and the Name_Fragment, where:

- Name_Fragment(N) = name(N + Name_Offset), if (N + Name_Offset) < Name_Length
- Name_Fragment(N) = 0,otherwise.

Here 0 ≤ N ≤ 13. In the first sent LMP_NAME_REQ PDU, Name_Offset=0. Sequence 82 is then repeated until the initiator has collected all fragments of the name. The name request may be made at any time following a successful Baseband Paging procedure.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_NAME_REQ | Name_Offset |
| M | LMP_NAME_RES | Name_Offset<br>Name_Length<br>Name_Fragment |

*Table 4.29: Name request PDUs*




*Sequence 82: Request for device name*

## 4.4 Role switch

### 4.4.1 Slot offset

With LMP_SLOT_OFFSET the information about the difference between the slot boundaries in different piconets is transmitted. The LMP_SLOT_OFFSET PDU may be sent anytime after the Baseband Paging procedure has completed if the ACL logical transport is in Active mode and a synchronous logical transport is not being negotiated by the LM. This PDU carries the parameters Slot_Offset and BD_ADDR. The Slot_Offset shall be the time, in microseconds, from the start of a Central transmission in the current piconet to the start of the next following Central transmission in the piconet where the BD_ADDR device (normally the Peripheral) is Central at the time that the request is interpreted by the BD_ADDR device.

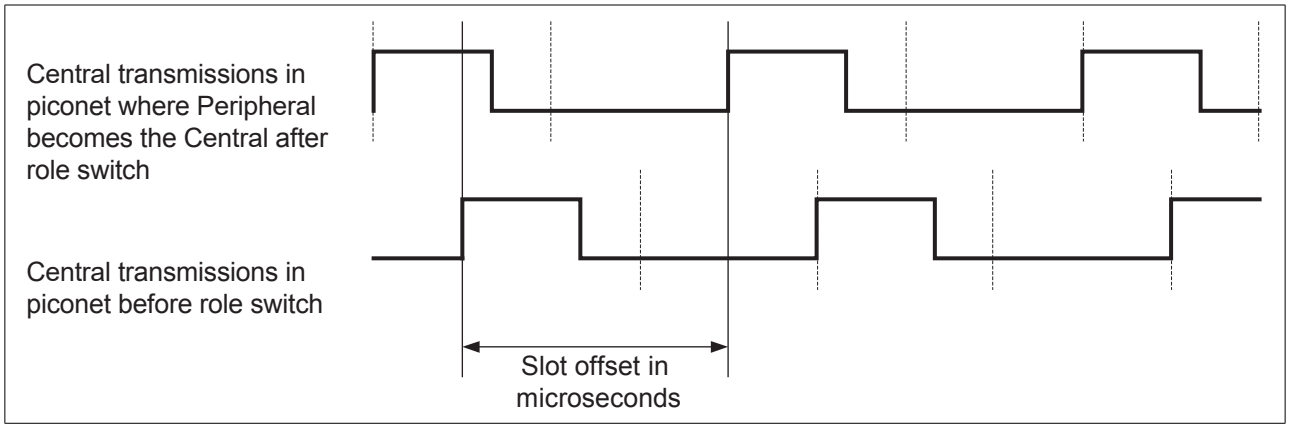
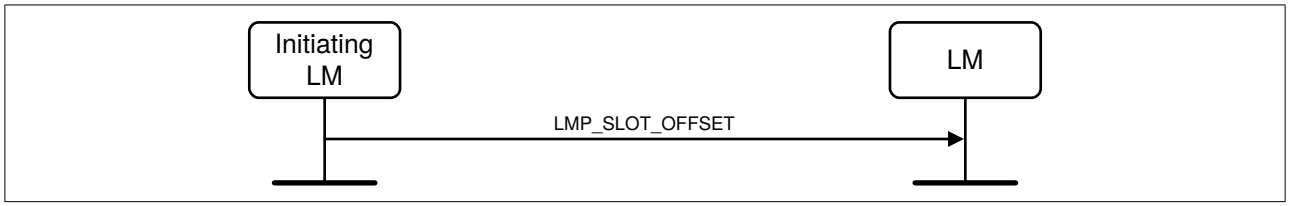



*Figure 4.2: Slot offset for role switch*

See Section 4.4.2 for the use of LMP_SLOT_OFFSET in the context of the role switch. In the case of role switch the BD_ADDR is that of the Peripheral.

| M/O | PDU | Contents |
|---|---|---|
| O(3) | LMP_SLOT_OFFSET | Slot_Offset<br>BD_ADDR |

*Table 4.30: Slot offset PDU*




*Sequence 83: Slot offset information is sent*

### 4.4.2 Role switch

Since the paging device always becomes the Central of the piconet, a role switch is sometimes needed; see [Vol 2] Part B, Section 8.6.5. A role switch may be performed as part of connection establishment (see Section 4.1.1) or any time after connection establishment has completed.

The LMP_SWITCH_REQ shall be sent only if the ACL logical transport is in Active mode, encryption is stopped or paused, and all synchronous logical transports on the same physical link are disabled. LMP_SWITCH_REQ shall not be initiated or accepted while a synchronous logical transport is being negotiated by the LM.

| M/O | PDU | Contents |
|---|---|---|
| O(5) | LMP_SWITCH_REQ | Switch_Instant |
| O(5) | LMP_SLOT_OFFSET | Slot_Offset<br>BD_ADDR |

*Table 4.31: Role switch PDUs*

Role switch is performed by an initiating device and a responding device.

First, the initiating LM shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). Next, if the Encryption_Mode is set to "encryption" then it shall initiate either the pause encryption sequence (see Section 4.2.5.5.) if both devices support pausing encryption or the stop encryption sequence (see Section 4.2.5.4) otherwise. If it is the Peripheral, it shall next send an LMP_SLOT_OFFSET PDU. Finally, it shall send an LMP_SWITCH_REQ PDU.

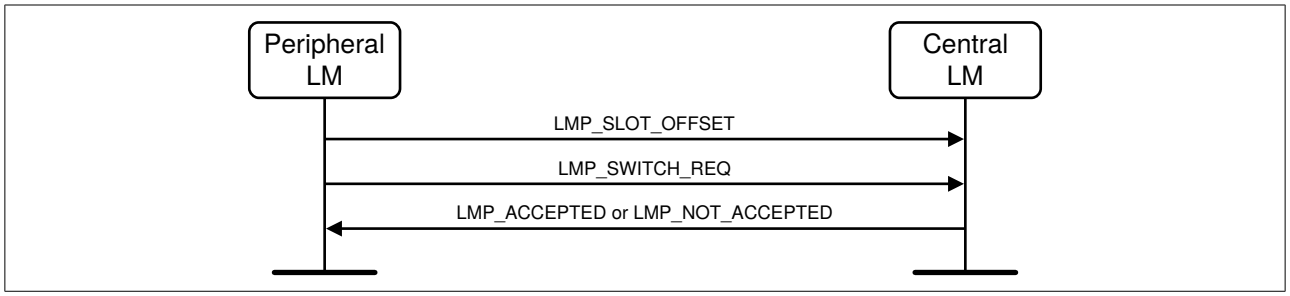
If the responding device accepts the role switch, then first it shall pause traffic on the ACL-U logical link if it is not already paused. If it is the Peripheral, it shall next send an LMP_SLOT_OFFSET PDU. Finally, it shall send an LMP_ACCEPTED PDU. The two devices shall then perform the Baseband Role Switch procedure. If it rejects the role switch, it shall send an LMP_NOT_ACCEPTED PDU.

When the role switch has been completed at the Baseband level (successfully or not), or has been rejected, then:

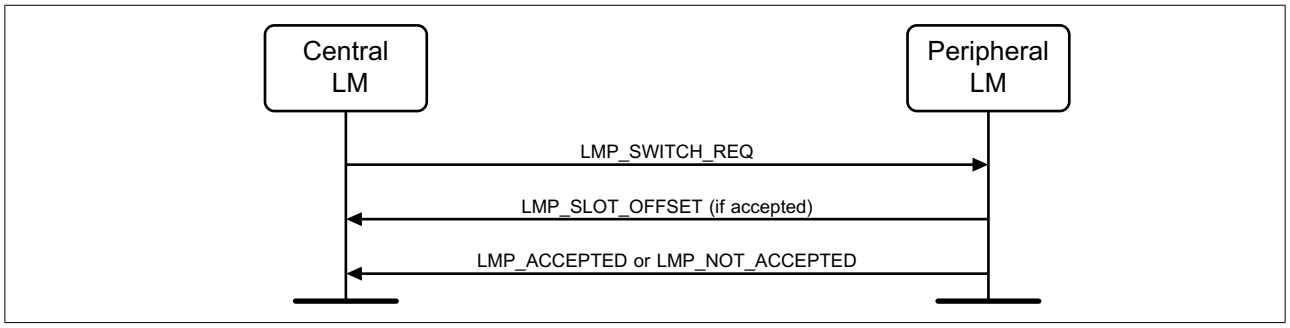
- If the initiating device had paused encryption, it shall initiate the resume encryption sequence (see Section 4.2.5.6). When AES-CCM encryption is used, the LMs shall calculate a new encryption key using the h3 algorithm (see [Vol 2] Part H, Section 7.7.6) with the BD_ADDRs for the new Central and Peripheral prior to resuming encryption.
- If the initiating device had stopped encryption, it shall initiate the start encryption sequence (see Section 4.2.5.3).

Both devices shall then re-enable transmission on the ACL-U logical link.

The transaction ID for the role switch PDUs in the sequence (LMP_SLOT_OFFSET and LMP_SWITCH_REQ along with the associated LMP_ACCEPTED or LMP_NOT_ACCEPTED) shall be set to 0 when the initiating device is the Central and 1 when it is the Peripheral.




*Sequence 84: Role switch (Peripheral initiated)*




*Sequence 85: Role switch (Central initiated)*

The LMP_SWITCH_REQ PDU contains a parameter, Switch_Instant, which specifies the instant at which the TDD switch is performed. This is specified as a Bluetooth clock value of the Central's clock, that is available to both devices. This instant is chosen by the sender of the message and shall be at least $2 \times T_{poll}$ or 32 (whichever is greater) slots in the future. The switch instant shall be within 12 hours of the current clock value to avoid clock wrap.

The sender of the LMP_SWITCH_REQ PDU selects the switch instant and queues the LMP_SWITCH_REQ PDU to LC for transmission and starts a timer to expire at the switch instant. When the timer expires it initiates the mode switch. In the case of a Central initiated switch if the LMP_SLOT_OFFSET PDU has not been received by the switch instant the role switch is carried out without an estimate of the Peripheral's slot offset. If an LMP_NOT_ACCEPTED PDU is received before the timer expires then the timer is stopped and the role switch shall not be initiated.

When the LMP_SWITCH_REQ is received the switch instant is compared with the Central's current clock value. If it is in the past then the instant has been passed and an LMP_NOT_ACCEPTED PDU with the Error_Code *Instant Passed* (0x28) shall be returned. If it is in the future then an LMP_ACCEPTED PDU shall be returned, assuming the role switch is accepted, and a timer is started to expire at the switch instant. When this timer expires the role switch shall be initiated.

After a successful role switch the supervision timeout and poll interval ($T_{poll}$) shall be set to their default values. The authentication state and the ACO shall remain unchanged.

Adaptive Frequency Hopping shall follow the procedures described in [Vol 2] Part B, Section 8.6.5. The default value for Max_Slots shall be used.

A role switch, whether successful or failed, does not affect the state of the Link Manager and of the ACL-C and ACL-U logical links except where explicitly stated.

## 4.5 Modes of operation

### 4.5.1 Hold mode

The ACL logical transport of a connection between two Bluetooth devices can be placed in Hold mode for a specified hold time. See [Vol 2] Part B, Section 8.8 for details.

| M/O | PDU | Contents |
|---|---|---|
| O(6) | LMP_HOLD | Hold_Time, Hold_Instant |
| O(6) | LMP_HOLD_REQ | Hold_Time, Hold_Instant |

*Table 4.32: Hold mode PDUs*

The LMP_HOLD and LMP_HOLD_REQ PDUs both contain a parameter, Hold_Instant, that specifies the instant at which the hold becomes effective. This is specified as a Bluetooth clock value of the Central's clock, that is available to both devices. The hold instant is chosen by the sender of the message and should be at least $6 \times T_{poll}$ slots in the future. The hold instant shall be within 12 hours of the current clock value to avoid clock wrap.

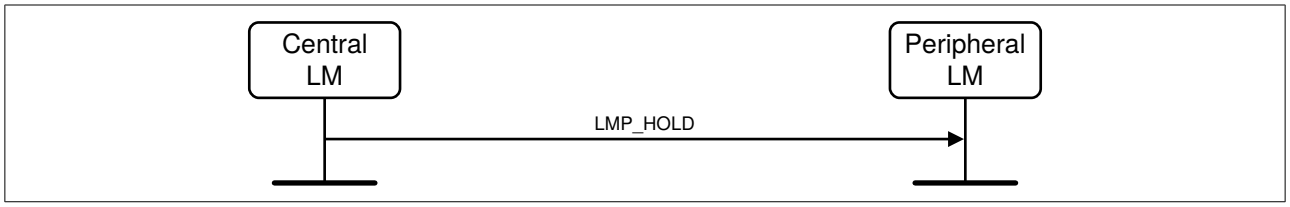
#### 4.5.1.1 Central forces Hold mode

The Central may force Hold mode if there has previously been a request for Hold mode that has been accepted by the Peripheral. The hold time included in the PDU when the Central forces Hold mode shall not be longer than any hold time the Peripheral has previously accepted when there was a request for Hold mode.

The Central LM shall first pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). It shall select the hold instant and queue the LMP_HOLD PDU to its LC for transmission. It shall then start a timer to wait until the hold instant occurs. When this timer expires then the connection shall enter Hold mode. If the Baseband acknowledgment for the LMP_HOLD PDU is not received then the Central may enter Hold mode, but it shall not use its low accuracy clock during the hold.

When the Peripheral LM receives an LMP_HOLD PDU it compares the hold instant with the Central's current clock value. If it is in the future then it starts a timer to expire at this instant and enters Hold mode when it expires.

When the Central LM exits from Hold mode it re-enables transmission on the ACL-U logical link.




*Sequence 86: Central forces Peripheral into Hold mode*

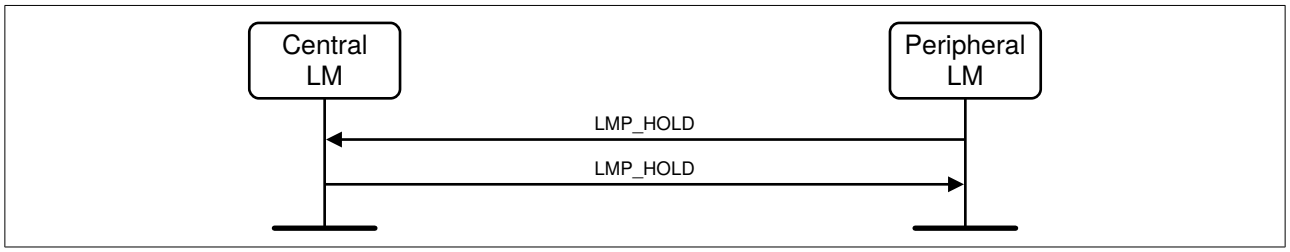
#### 4.5.1.2 Peripheral forces Hold mode

The Peripheral may force Hold mode if there has previously been a request for Hold mode that has been accepted by the Central. The hold time included in the PDU when the Peripheral forces Hold mode shall not be longer than any hold time the Central has previously accepted when there was a request for Hold mode.

The Peripheral LM shall first complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link. It shall select the hold instant and queue the LMP_HOLD PDU to its LC for transmission. It shall then wait for an LMP_HOLD PDU from the Central acting according to the procedure described in Section 4.5.1.1.

When the Central LM receives an LMP_HOLD PDU it shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). It shall then inspect the hold instant. If this is less than $6 \times T_{poll}$ slots in the future it shall modify the instant so that it is at least $6 \times T_{poll}$ slots in the future. It shall then send an LMP_HOLD PDU using the mechanism described in Section 4.5.1.1.

When the Central and Peripheral LMs exit from Hold mode they shall re-enable transmission on the ACL-U logical link.




*Sequence 87: Peripheral forces Central into Hold mode*

#### 4.5.1.3 Central or Peripheral requests Hold mode

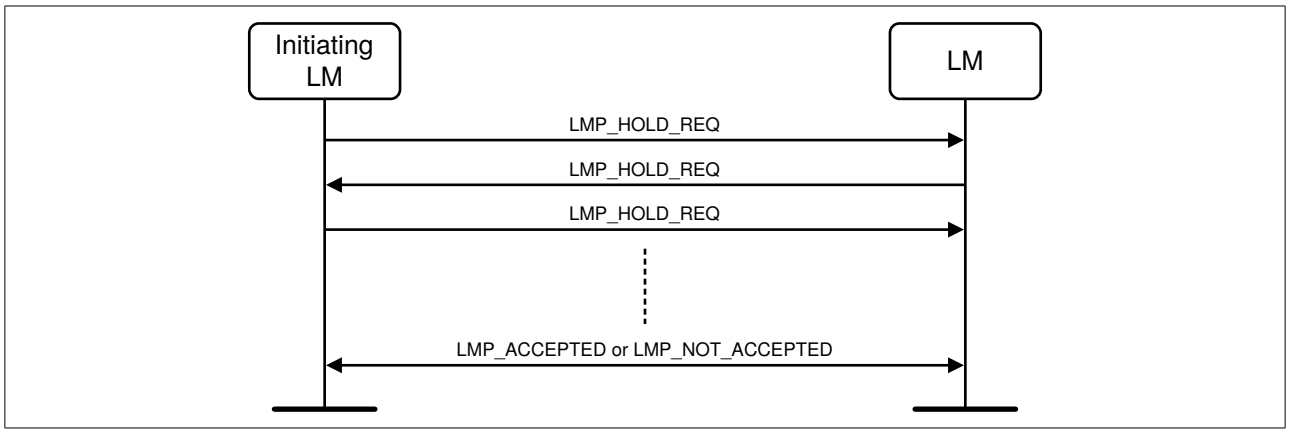
The Central or the Peripheral may request to enter Hold mode. Upon receipt of the request, the same request with modified parameters may be returned or the negotiation may be terminated. If an agreement is seen an LMP_ACCEPTED PDU terminates the negotiation and the ACL link is placed in Hold mode. If no agreement is seen, an

LMP_NOT_ACCEPTED PDU with the Error_Code *Unsupported LMP Parameter Value* (0x20) terminates the negotiation and Hold mode is not entered.

The initiating LM shall pause traffic on the ACL-U logical link (see [Vol 2] Part B, Section 5.3.1). On receiving an LMP_HOLD_REQ PDU the receiving LM shall complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link.

The LM sending the LMP_HOLD_REQ PDU selects the hold instant, that shall be at least $9 \times T_{poll}$ slots in the future. If this is a response to a previous LMP_HOLD_REQ PDU and the contained hold instant is at least $9 \times T_{poll}$ slots in the future then this shall be used. The LMP_HOLD_REQ PDU shall then be queued to its LC for transmission and a timer shall be started to expire at this instant and the connection enters Hold mode when it expires unless an LMP_NOT_ACCEPTED or LMP_HOLD_REQ PDU is received by its LM before that point. If the LM receiving LMP_HOLD_REQ PDU agrees to enter Hold mode it shall return an LMP_ACCEPTED PDU and shall start a timer to expire at the hold instant. When this timer expires it enters Hold mode.

When each LM exits from Hold mode it shall re-enable transmission on the ACL-U logical link.




*Sequence 88: Negotiation for Hold mode*

### 4.5.2 [This section is no longer used]

### 4.5.3 Sniff mode

To enter Sniff mode, Central and Peripheral negotiate a sniff interval $T_{Sniff}$ and a sniff offset, $D_{Sniff}$, that specifies the timing of the sniff slots. The offset determines the time of the first sniff slot; after that the sniff slots follow periodically with the sniff interval $T_{Sniff}$. To avoid clock wrap-around during the initialization, one of two options is chosen for the calculation of the first sniff slot. The Timing_Control_Flags parameter in the sniff request message indicates this.

When the ACL logical transport is in Sniff mode the Central shall only start a transmission in the sniff slots. Two parameters control the listening activity in the Peripheral: the Sniff_Attempt and the Sniff_Timeout. The Sniff_Attempt parameter determines for how many slots the Peripheral shall listen when the Peripheral is not treating this as a scatternet link, beginning at the sniff slot, even if it does not receive a packet with its own LT_ADDR. The Sniff_Timeout parameter determines for how many additional slots the Peripheral shall listen when the Peripheral is not treating this as a scatternet link if it continues to receive only packets with its own LT_ADDR. It is not possible to modify the sniff parameters while the device is in Sniff mode.

#### 4.5.3.1 Central or Peripheral requests Sniff mode

Either the Central or the Peripheral may request entry to Sniff mode. The process is initiated by sending an LMP_SNIFF_REQ PDU containing a set of parameters. The receiving LM shall then decide whether to reject the attempt by sending an LMP_NOT_ACCEPTED PDU, to suggest different parameters by replying with an LMP_SNIFF_REQ PDU or to accept the request.

| M/O | PDU | Contents |
|---|---|---|
| O(7) | LMP_SNIFF_REQ | Timing_Control_Flags<br>$D_{Sniff}$<br>$T_{Sniff}$<br>Sniff_Attempt<br>Sniff_Timeout |
| O(7) | LMP_UNSNIFF_REQ | - |
| O(41) | LMP_SNIFF_SUBRATING_REQ | Max_Sniff_Subrate<br>Min_Sniff_Mode_Timeout<br>Sniff_Subrating_Instant |
| O(41) | LMP_SNIFF_SUBRATING_RES | Max_Sniff_Subrate<br>Min_Sniff_Mode_Timeout<br>Sniff_Subrating_Instant |

*Table 4.33: Sniff mode PDUs*

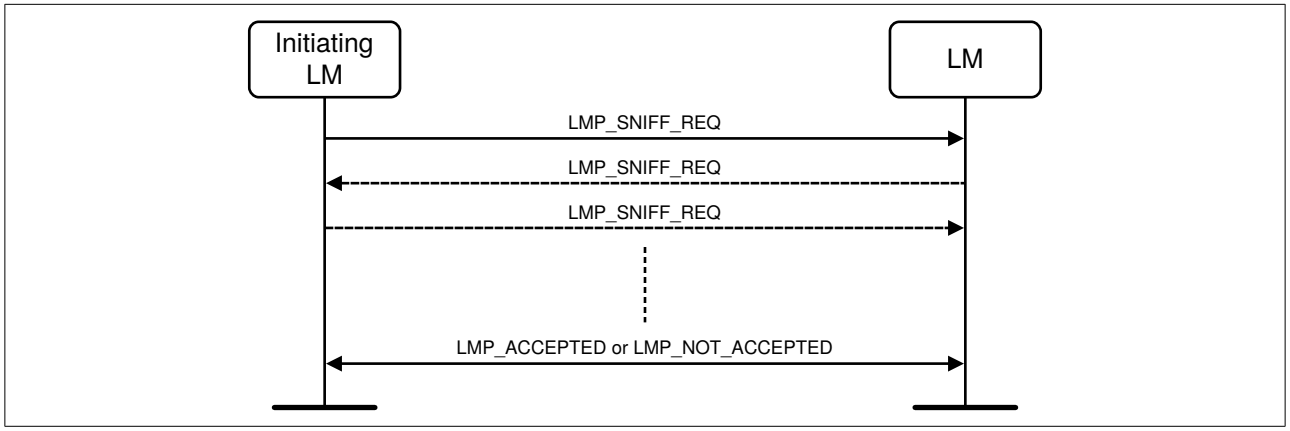
Before the first time that the Central sends LMP_SNIFF_REQ in a transaction it shall enter Sniff Transition mode. If the Central receives or sends an LMP_NOT_ACCEPTED PDU it shall exit from Sniff Transition mode.

If the Central receives an LMP_SNIFF_REQ PDU it shall enter Sniff Transition mode (if not already in it) and then determine whether to accept the request. The Central may reply with an LMP_NOT_ACCEPTED PDU or, if it can enter Sniff mode but requires a different set of parameters, it shall respond with an LMP_SNIFF_REQ PDU containing the new parameters. If the Central decides that the parameters are acceptable then it

shall send an LMP_ACCEPTED PDU; when it receives the Baseband acknowledgment for this PDU it shall exit Sniff Transition mode and enter Sniff mode.

If the Central receives an LMP_ACCEPTED PDU the Central shall exit from Sniff Transition mode and enter Sniff mode.

If the Peripheral receives an LMP_SNIFF_REQ PDU it shall determine whether to accept the request. The Peripheral may reply with an LMP_NOT_ACCEPTED PDU to reject the request or, if it can enter Sniff mode but requires a different set of parameters, it shall respond with an LMP_SNIFF_REQ PDU containing the new parameters. If the Peripheral decides that the parameters are acceptable then it shall send an LMP_ACCEPTED PDU and enter Sniff mode. If the Peripheral receives an LMP_NOT_ACCEPTED PDU it shall terminate the attempt to enter Sniff mode.



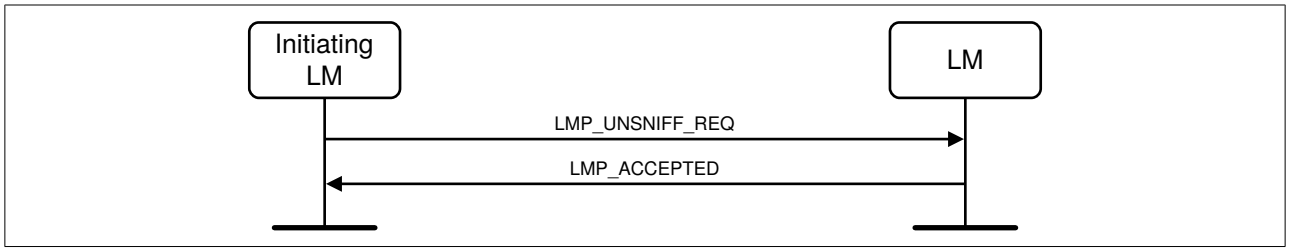

*Sequence 89: Negotiation for Sniff mode*

#### 4.5.3.2 Moving a Peripheral from Sniff mode to Active mode

Sniff mode may be exited by either the Central or the Peripheral sending an LMP_UNSNIFF_REQ PDU. The requested device shall reply with an LMP_ACCEPTED PDU.

If the Central requests an exit from Sniff mode it shall enter Sniff Transition mode and then send an LMP_UNSNIFF_REQ PDU. When the Peripheral receives the LMP_UNSNIFF_REQ it shall exit from Sniff mode and reply with an LMP_ACCEPTED PDU. When the Central receives the LMP_ACCEPTED PDU it shall exit from Sniff Transition mode and enter Active mode.

If the Peripheral requests an exit from Sniff mode it shall send an LMP_UNSNIFF_REQ PDU. When the Central receives the LMP_UNSNIFF_REQ PDU it shall enter Sniff Transition mode and then send an LMP_ACCEPTED PDU. When the Peripheral receives the LMP_ACCEPTED PDU it shall exit from Sniff mode and enter Active mode. When the Central receives the Baseband acknowledgment for the LMP_ACCEPTED PDU it shall leave Sniff Transition mode and enter Active mode.




*Sequence 90: Peripheral moved from Sniff mode to Active mode*

#### 4.5.3.3 Sniff subrating

Once Sniff mode has been started, sniff subrating may be initiated by either Link Manager.

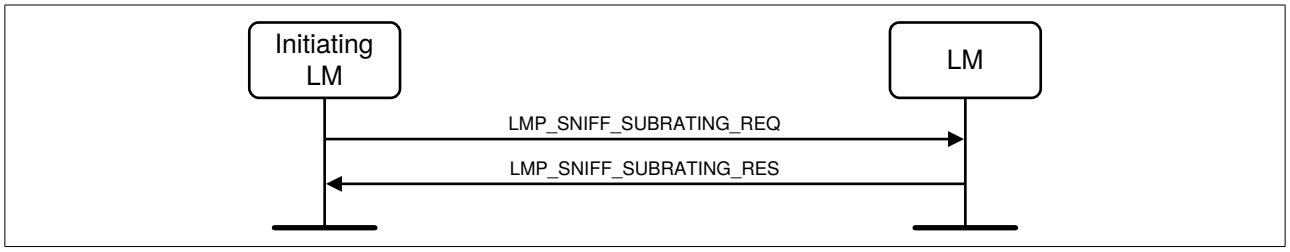
The LMP_SNIFF_SUBRATING_REQ and LMP_SNIFF_SUBRATING_RES PDUs specify parameters that the peer and initiating device shall use respectively for sniff subrating.

The Sniff_Subrating_Instant value shall be used to calculate the first sniff subrating anchor point. The Sniff_Subrating_Instant value shall be a maximum of $2^{16}$ time slots (40.9 seconds) from the Central's current clock and shall be a sniff anchor point. The Sniff_Subrating_Instant value should indicate a clock value in the future with respect to the clock value when the LMP message is first sent.

If the LMP_SNIFF_SUBRATING_REQ PDU is sent by the Central, the Sniff_Subrating_Instant value shall be used. The Peripheral shall reply with an LMP_SNIFF_SUBRATING_RES PDU using the same Sniff_Subrating_Instant value given by the Central.

When the LMP_SNIFF_SUBRATING_REQ PDU is sent by the Peripheral, the Sniff_Subrating_Instant value shall be ignored. The Central shall reply with an LMP_SNIFF_SUBRATING_RES PDU with the Sniff_Subrating_Instant value that shall be used for sniff subrating.

The initiating device shall not transition to the new sniff subrating parameters until the sniff subrating instant has passed and the LMP_SNIFF_SUBRATING_RES PDU has been received. The non-initiating device shall remain in Sniff mode and shall not transition to the new sniff subrating parameters until after the sniff subrating instant has passed and the Baseband acknowledgment of the LMP_SNIFF_SUBRATING_RES PDU has been received.




*Sequence 91: LM accepts sniff subrating request*

A device shall not send a new LMP_SNIFF_SUBRATING_REQ PDU until the previous sniff subrating transaction has completed and the sniff subrating instant has passed.

The maximum clock interval between two sniff subrating anchor points shall be less than the link supervision timeout. If the link supervision timeout needs to be updated to a shorter value than the clock interval between two sniff subrating anchor points, the Central shall disable sniff subrating, shall send the LMP_SUPERVISION_TIMEOUT PDU with the new supervision timeout value, and shall start using the new supervision timeout value after receiving a Baseband acknowledgment for the LMP_SUPERVISION_TIMEOUT PDU. Upon reception of the LMP_SUPERVISION_TIMEOUT PDU the Peripheral shall disable sniff subrating and shall start using the new supervision timeout value.

The Central shall initiate sniff subrating with the Max_Sniff_Subrate parameter less than the new supervision timeout. The Peripheral shall respond with the LMP_SNIFF_SUBRATING_RES PDU with the Max_Sniff_Subrate parameter less than the new supervision timeout.

## 4.6 Logical transports

When a connection is first established between two devices the connection consists of the ACL logical transport (carrying the ACL-C logical link for LMP messages and the ACL-U logical link for L2CAP data) and the APB logical transport carrying the APB-C logical links for LMP messages and the APB-U logical link for L2CAP data. One or more synchronous logical transports (SCO or eSCO) may then be added. A new logical transport shall not be created if it would cause all slots to be allocated to reserved slots on secondary LT_ADDRs.

### 4.6.1 SCO logical transport

The SCO logical transport reserves slots separated by the SCO interval, $T_{sco}$. The first slot reserved for the SCO logical transport is defined by $T_{sco}$ and the SCO offset, $D_{sco}$. See [Vol 2] Part B, Section 8.6.2 for details. A device shall initiate a request for HV2 or HV3 packet type only if the other device supports that type. A device shall initiate CVSD, μ-law or A-law coding or uncoded (transparent) data only if the other device supports the corresponding feature. To avoid problems with a wrap-around of

the clock during initialization of the SCO logical transport, the Timing_Control_Flags parameter is used to indicate how the first SCO slot shall be calculated. The SCO link is distinguished from all other SCO links by a SCO handle. The SCO handle zero shall not be used.

The Link Manager shall not initiate a SCO connection, and shall reject any request for a SCO connection, while AES-CCM encryption is in use.

Note: The Peripheral interprets the initialization flag in the Timing_Control_Flags in terms of the Central's Bluetooth clock. See [Vol 2] Part B, Section 8.6.2.

| M/O | PDU | Contents |
|---|---|---|
| O(11) | LMP_SCO_LINK_REQ | SCO_Handle<br>Timing_Control_Flags<br>$D_{sco}$<br>$T_{sco}$<br>SCO_Packet<br>Air_Mode |
| O(11) | LMP_REMOVE_SCO_LINK_REQ | SCO_Handle<br>Error_Code |

*Table 4.34: SCO link management PDUs*

#### 4.6.1.1 Central initiates a SCO link

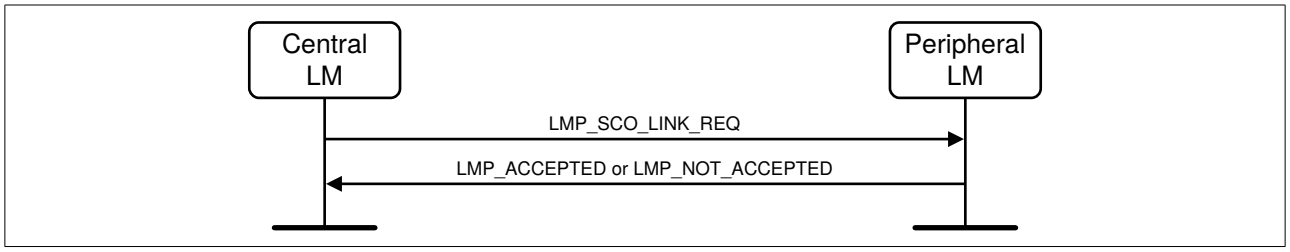
When establishing a SCO link the Central sends a request, a LMP_SCO_LINK_REQ PDU, with parameters that specify the timing, packet type and coding that will be used on the SCO link. Each of the SCO packet types supports three different voice coding formats on the air-interface: μ-law log PCM, A-law log PCM and CVSD. The air coding by log PCM or CVSD may be deactivated to achieve a transparent synchronous data link at 64 kb/s.

The slots used for the SCO links are determined by three parameters controlled by the Central: $T_{sco}$, $D_{sco}$ and a flag indicating how the first SCO slot is calculated. After the first slot, the SCO slots follow periodically at an interval of $T_{sco}$.

If the Peripheral does not accept the SCO link, but can operate with a different set of SCO parameters, it can indicate what it does not accept in the Error_Code parameter of an LMP_NOT_ACCEPTED PDU. The Central may then issue a new request with modified parameters.

The SCO handle in the message shall be different from existing SCO link(s).

If the SCO packet type is HV1 the LMP_ACCEPTED shall be sent using the DM1 packet.

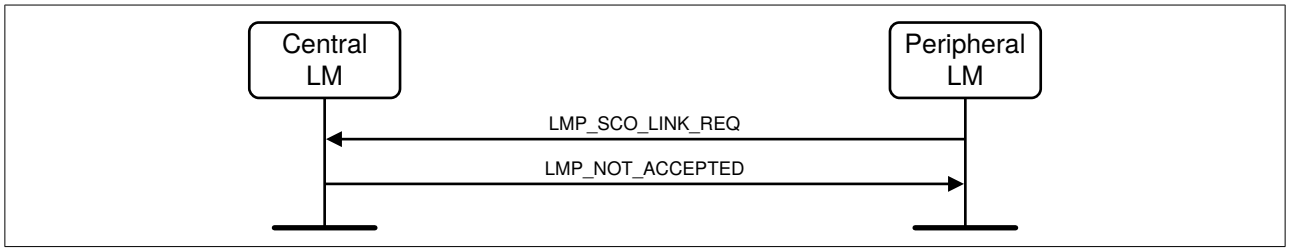



*Sequence 92: Central requests a SCO link*

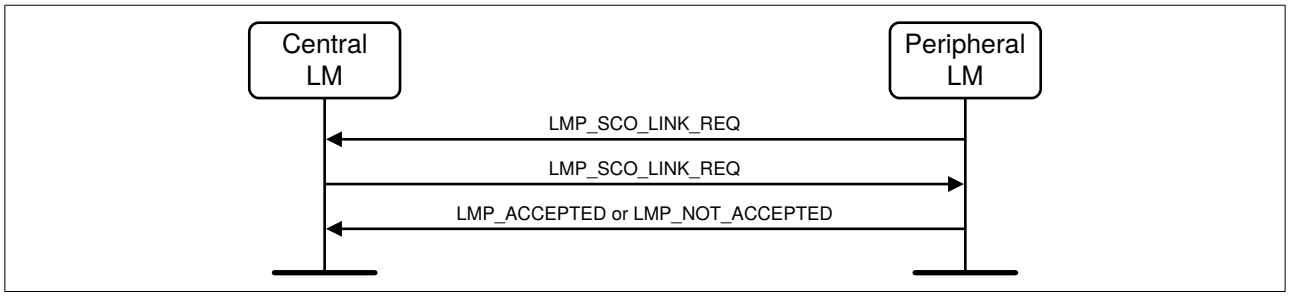
#### 4.6.1.2 Peripheral initiates a SCO link

The Peripheral may initiate the establishment of a SCO link. The Peripheral sends an LMP_SCO_LINK_REQ PDU with the SCO handle set to zero; the Timing_Control_Flags parameter shall be ignored by the Central, while the $D_{SCO}$ parameter should be set to 0. If the Central is not capable of establishing a SCO link, it replies with an LMP_NOT_ACCEPTED PDU. Otherwise it sends back an LMP_SCO_LINK_REQ PDU. This message includes the assigned SCO handle, $D_{sco}$ and the timing control flags. The Central should try to use the same parameters as in the Peripheral request; if the Central cannot meet that request, it is allowed to use other values. The Peripheral shall then reply with LMP_ACCEPTED or LMP_NOT_ACCEPTED PDU.

If the SCO packet type is HV1 the LMP_ACCEPTED shall be sent using the DM1 packet.




*Sequence 93: Central rejects Peripheral's request for a SCO link*




*Sequence 94: Central accepts Peripheral's request for a SCO link*

#### 4.6.1.3 Central requests change of SCO parameters

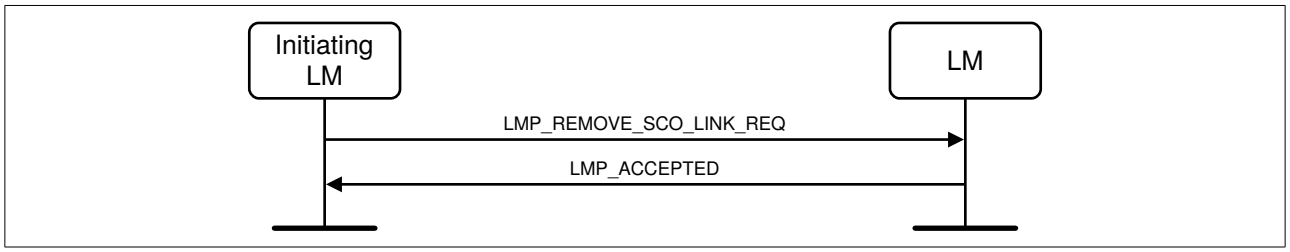
The Central sends an LMP_SCO_LINK_REQ PDU, where the SCO_Handle is the handle of the SCO link the Central is attempting to change parameters for. If the Peripheral accepts the new parameters, it replies with an LMP_ACCEPTED PDU and the SCO link will change to the new parameters. If the Peripheral does not accept the new parameters, it shall reply with an LMP_NOT_ACCEPTED PDU and the SCO link is left unchanged. When the Peripheral replies with an LMP_NOT_ACCEPTED PDU it shall indicate in the Error_Code parameter what it does not accept. The Central may then try to change the SCO link again with modified parameters. The sequence is the same as in Section 4.6.1.1.

#### 4.6.1.4 Peripheral requests change of SCO parameters

The Peripheral sends an LMP_SCO_LINK_REQ PDU, where the SCO_Handle is the handle of the SCO link to be changed. The parameters Timing_Control_Flags and $D_{sco}$ in this PDU shall be ignored by the Central. If the Central does not accept the new parameters it shall reply with an LMP_NOT_ACCEPTED PDU and the SCO link is left unchanged. If the Central accepts the new parameters it shall reply with an LMP_SCO_LINK_REQ PDU containing the same parameters as in the Peripheral request. When receiving this message the Peripheral replies with an LMP_NOT_ACCEPTED PDU if it does not accept the new parameters. The SCO link is then left unchanged. If the Peripheral accepts the new parameters it replies with an LMP_ACCEPTED PDU and the SCO link will change to the new parameters. The sequence is the same as in Section 4.6.1.2.

#### 4.6.1.5 Remove a SCO link

The Central or Peripheral may remove the SCO link by sending a request including the SCO handle of the SCO link to be removed and an Error_Code indicating why the SCO link is removed. The receiving side shall respond with an LMP_ACCEPTED PDU.




*Sequence 95: SCO link removed*

### 4.6.2 eSCO logical transport

After an ACL link has been established, one or more extended SCO (eSCO) links can be set up to the remote device. The eSCO links are similar to SCO links using

timing control flags, an interval $T_{eSCO}$ and an offset $D_{eSCO}$. As opposed to SCO links, eSCO links have a configurable data rate that may be asymmetric, and can be set up to provide limited retransmissions of lost or damaged packets inside a retransmission window of size $W_{eSCO}$. The $D_{eSCO}$ shall be based on CLK.

Note: The Peripheral interprets the initialization flag in the Timing_Control_Flags in terms of the Central's Bluetooth clock. See [Vol 2] Part B, Section 8.6.2.

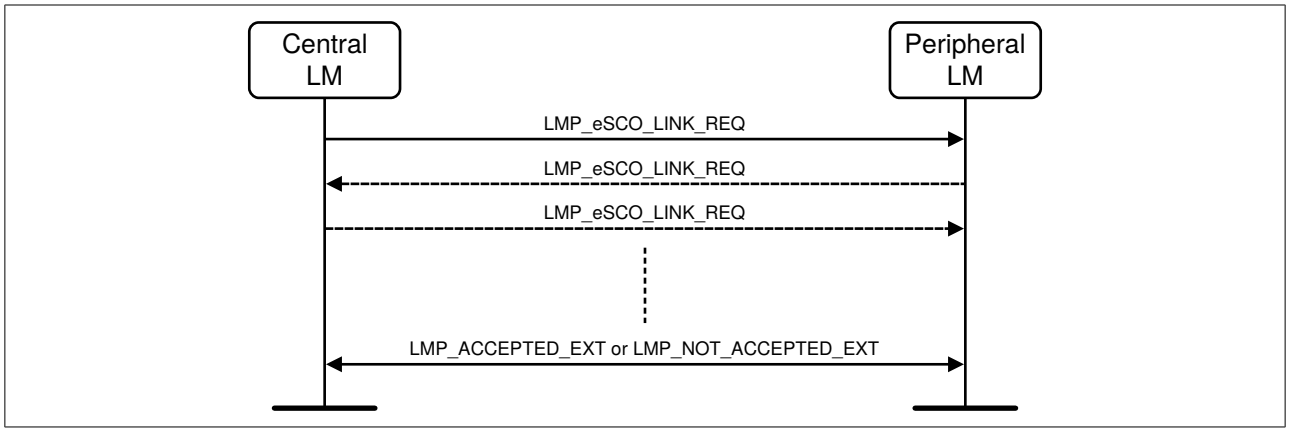
The parameters $D_{eSCO}$, $T_{eSCO}$, $W_{eSCO}$, eSCO_Packet_Type C→P, eSCO_Packet_Type P→C, Packet_Length C→P, Packet_Length P→C are henceforth referred to as the negotiable parameters.

| M/O | PDU | Contents |
|---|---|---|
| O(31) | LMP_eSCO_LINK_REQ | eSCO_Handle<br>eSCO_LT_ADDR<br>Timing_Control_Flags<br>$D_{eSCO}$<br>$T_{eSCO}$<br>$W_{eSCO}$<br>eSCO_Packet_Type C→P<br>eSCO_Packet_Type P→C<br>Packet_Length C→P<br>Packet_Length P→C<br>Air_Mode<br>Negotiation_State |
| O(31) | LMP_REMOVE_eSCO_LINK_REQ | eSCO_Handle<br>Error_Code |

*Table 4.35: PDUs used for managing the eSCO links*

#### 4.6.2.1 Central initiates an eSCO link

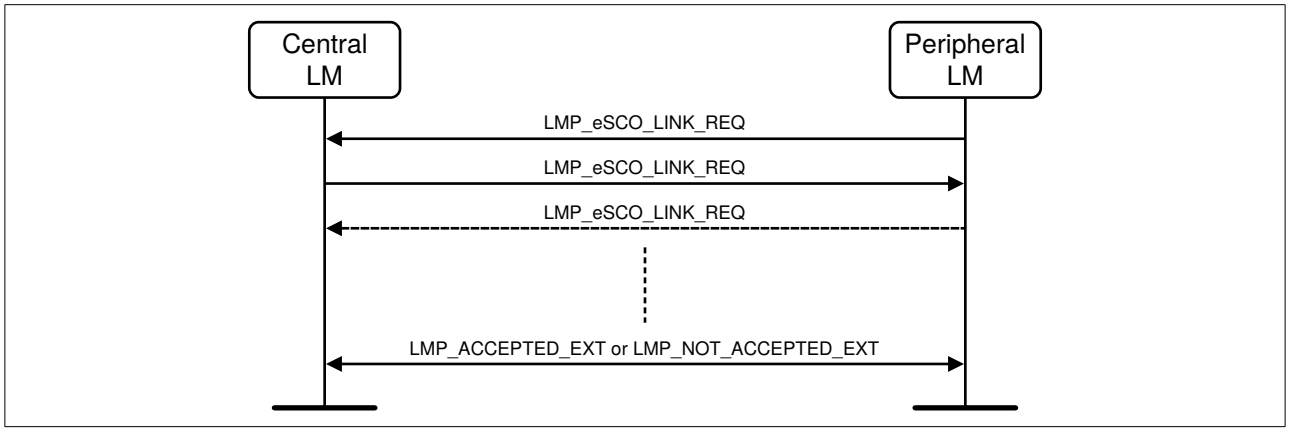
When establishing an eSCO link the Central sends an LMP_eSCO_LINK_REQ PDU specifying all parameters. The Peripheral may accept this with an LMP_ACCEPTED_EXT PDU, reject it with an LMP_NOT_ACCEPTED_EXT PDU, or respond with its own LMP_eSCO_LINK_REQ specifying alternatives for some or all parameters. The Peripheral shall not negotiate the eSCO_Handle or eSCO_LT_ADDR parameters. The negotiation of parameters continues until the Central or Peripheral either accepts the latest parameters with an LMP_ACCEPTED_EXT PDU, or terminates the negotiation with an LMP_NOT_ACCEPTED_EXT PDU. The negotiation shall use the procedures defined in Section 4.6.2.5.




*Sequence 96: Central requests an eSCO link*

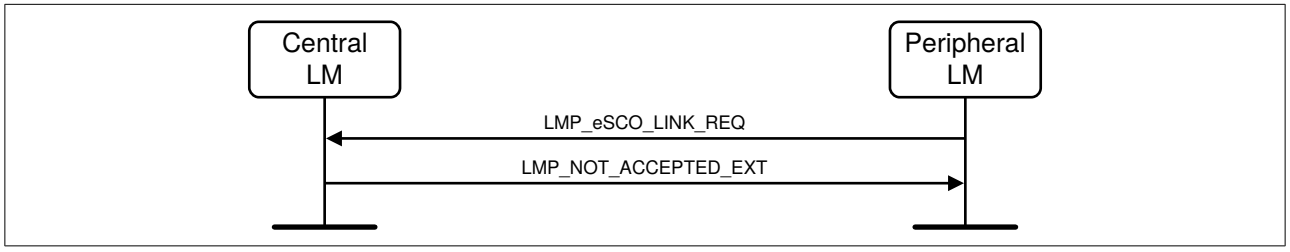
#### 4.6.2.2 Peripheral initiates an eSCO link

When attempting to establish an eSCO link the Peripheral shall send an LMP_eSCO_LINK_REQ PDU specifying all parameters except eSCO_LT_ADDR, which is reserved for future use, and eSCO_Handle, which shall be set to zero. The Central may respond to this with an LMP_eSCO_LINK_REQ PDU, filling in these missing parameters, and potentially changing the other requested parameters. The Peripheral can accept this with an LMP_ACCEPTED_EXT PDU, or respond with a further LMP_eSCO_LINK_REQ PDU specifying alternatives for some or all of the parameters. The negotiation of parameters continues until the Central or Peripheral either accepts the latest parameters with an LMP_ACCEPTED_EXT PDU, or terminates the negotiation with an LMP_NOT_ACCEPTED_EXT PDU.




*Sequence 97: Peripheral requests an eSCO link*

The Central may reject the request immediately with an LMP_NOT_ACCEPTED_EXT PDU. The negotiation shall use the procedures defined in Section 4.6.2.5.




*Sequence 98: Central rejects Peripheral's request for an eSCO link*

#### 4.6.2.3 Central or Peripheral requests change of eSCO parameters

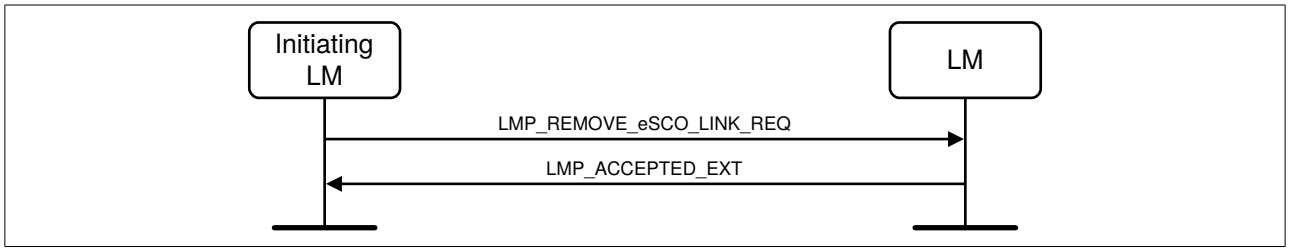
The Central or Peripheral may request a renegotiation of the eSCO parameters. The Central or Peripheral shall send an LMP_eSCO_LINK_REQ PDU with the eSCO handle of the eSCO link the device is attempting to renegotiate. The remote device may accept the changed parameters immediately with LMP_ACCEPTED_EXT PDU, or the negotiation may be continued with further LMP_eSCO_LINK_REQ PDUs until the Central or Peripheral accepts the latest parameters with an LMP_ACCEPTED_EXT PDU or terminates the negotiation with an LMP_NOT_ACCEPTED_EXT PDU. In the case of termination with an LMP_NOT_ACCEPTED_EXT PDU, the eSCO link continues on the previously negotiated parameters.

The sequence is the same as in Section 4.6.2.2.

During re-negotiation, the eSCO LT_ADDR and eSCO handle shall not be re-negotiated and shall be set to the originally negotiated values. The negotiation shall use the procedures defined in Section 4.6.2.5.

#### 4.6.2.4 Remove an eSCO link

Either the Central or Peripheral may remove the eSCO link by sending an LMP_REMOVE_eSCO_LINK_REQ PDU including the eSCO handle of the eSCO link to be removed and an Error_Code indicating why the eSCO link is removed. The receiving side shall respond with an LMP_ACCEPTED_EXT PDU. The Peripheral shall shut down its eSCO logical transport before sending its PDU (LMP_REMOVE_eSCO_LINK_REQ for Peripheral initiated removal, LMP_ACCEPTED_EXT for Central initiated). The Central shall not reassign the eSCO LT_ADDR until the end of the removal procedure. If, for some reason, such as the Peripheral being out of range, the procedure cannot be completed, the Central shall not reassign the eSCO LT_ADDR until the primary LT_ADDR is available for reuse (supervision timeout).




*Sequence 99: eSCO link removed*

#### 4.6.2.5 Rules for the LMP negotiation and renegotiation

**Rule 1:** the Negotiation_State shall be set to 0 by the initiating LM. After the initial LMP_eSCO_LINK_REQ is sent the Negotiation_State shall not be set to 0.

**Rule 2:** if the bandwidth (defined as 1600 times the packet length in bytes divided by $T_{eSCO}$ in slots) for either RX or TX or the Air_Mode cannot be accepted the device shall send LMP_NOT_ACCEPTED_EXT with the appropriate Error_Code.

**Rule 3:** Bandwidth and Air_Mode are not negotiable and shall not be changed for the duration of the negotiation. Once one side has rejected the negotiation (with LMP_NOT_ACCEPTED_EXT) a new negotiation may be started with different bandwidth and Air_Mode parameters.

**Rule 4:** if the parameters will cause a latency violation ($T_{eSCO}$ + $W_{eSCO}$ + reserved synchronous slots > allowed local latency) the device should propose new parameters that shall not cause a reserved slot violation or latency violation for the device that is sending the parameters. In this case the Negotiation_State shall be set to 3. Otherwise the device shall send LMP_NOT_ACCEPTED_EXT.

**Rule 5:** once a device has received an LMP_eSCO_LINK_REQ with the Negotiation_State set to 3 (latency violation), the device shall not propose any combination of packet type, $T_{eSCO}$, and $W_{eSCO}$ that will give an equal or larger latency than the combination that caused the latency violation for the other device.

**Rule 6:** if the parameters cause both a reserved slot violation and a latency violation or if the parameters are not supported and a latency violation occurs, then the device shall set the Negotiation_State to 3 (latency violation).

**Rule 7:** if the parameters will cause a reserved slot violation the device should propose new parameters that shall not cause a reserved slot violation. In this case the Negotiation_State shall be set to 2. Otherwise the device shall send LMP_NOT_ACCEPTED_EXT.

**Rule 8:** If the requested parameters are not supported the device should propose a setting that is supported, and set the Negotiation_State to 4. If it is not possible to find such a parameter set, the device shall send LMP_NOT_ACCEPTED_EXT.

**Rule 9:** when proposing new parameters for reasons other than a latency violation, reserved slot violation, or configuration not supported, the Negotiation_State shall be set to 1.

#### 4.6.2.6 Negotiation state definitions

Reserved Slot Violation: a reserved slot violation is when the receiving LM cannot setup the requested eSCO logical transport because the eSCO reserved slots would overlap with other regularly scheduled slots (e.g. other synchronous reserved slots or sniff anchor points).

Latency Violation: a latency violation is when the receiving LM cannot setup the requested eSCO logical transport because the latency ($W_{eSCO}$+$T_{eSCO}$ + reserved synchronous slots) is greater than the maximum allowed latency.

Configuration not supported: The combination of parameters requested is not inside the supported range for the device.

## 4.7 Test mode

LMP has PDUs to support testing of the Bluetooth radio and Baseband. See [Vol 3] Part D, Section 1 for a detailed description of these test modes.

### 4.7.1 Activation and deactivation of Test mode

The activation may be carried out locally (via a HW or SW interface), or using the air interface.

- For activation over the air interface, entering the test mode shall be locally enabled for security and type approval reasons. The implementation of this local enabling is not subject to standardization.

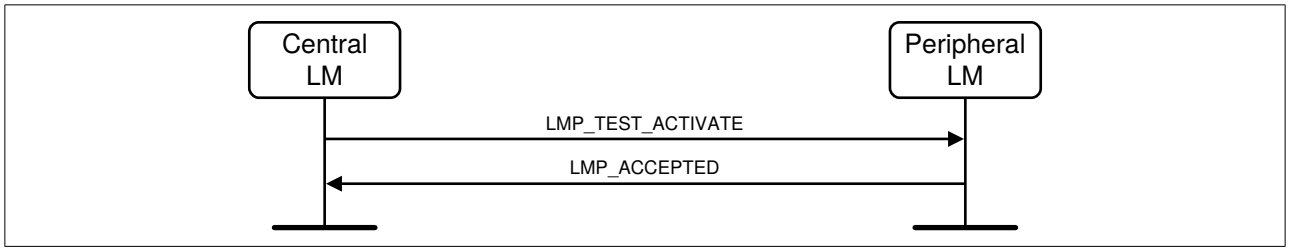
  The Lower Tester sends an LMP command that shall force the IUT to enter test mode. The IUT shall terminate all normal operation before entering the test mode.

  The IUT shall return an LMP_ACCEPTED PDU on reception of an activation command. An LMP_NOT_ACCEPTED PDU shall be returned if the IUT is not locally enabled.
- If the activation is performed locally using a HW or SW interface, the IUT shall terminate all normal operation before entering the test mode.

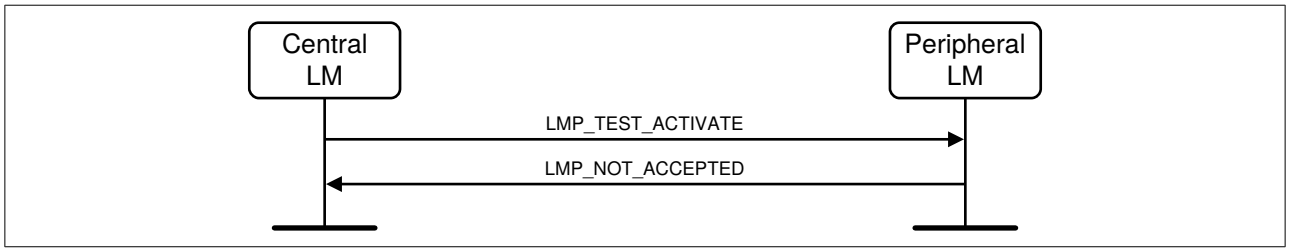
  Until a connection to the Lower Tester exists, the device shall perform page scan and inquiry scan. Extended scan activity is recommended.

The test mode is activated by sending an LMP_TEST_ACTIVATE PDU to the implementation under test (IUT). The IUT is always the Peripheral. The LM shall be able to receive this message at any time. If entering test mode is locally enabled in

the IUT, then it shall respond with an LMP_ACCEPTED PDU and test mode is entered. Otherwise the IUT responds with an LMP_NOT_ACCEPTED PDU and the IUT remains in normal operation. The Error_Code in the LMP_NOT_ACCEPTED PDU shall be *LMP PDU Not Allowed* (0x24).




*Sequence 100: Activation of test mode successful*




*Sequence 101: Activation of Test mode fails. Peripheral is not allowed to enter Test mode.*

The test mode can be deactivated in two ways. Sending an LMP_TEST_CONTROL PDU with Test_Scenario set to "exit test mode" exits the test mode and the Peripheral returns to normal operation still connected to the Central. Sending an LMP_DETACH PDU to the IUT ends the test mode and the connection.

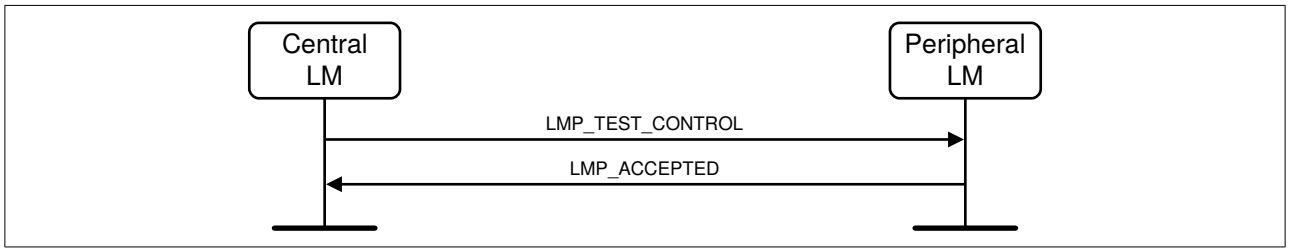
### 4.7.2 Control of Test mode

Control and configuration is performed using special LMP commands (see Section 4.7.3). These commands shall be rejected if the Bluetooth device is not in test mode. In this case, an LMP_NOT_ACCEPTED shall be returned. The IUT shall return an LMP_ACCEPTED on reception of a control command when in test mode.

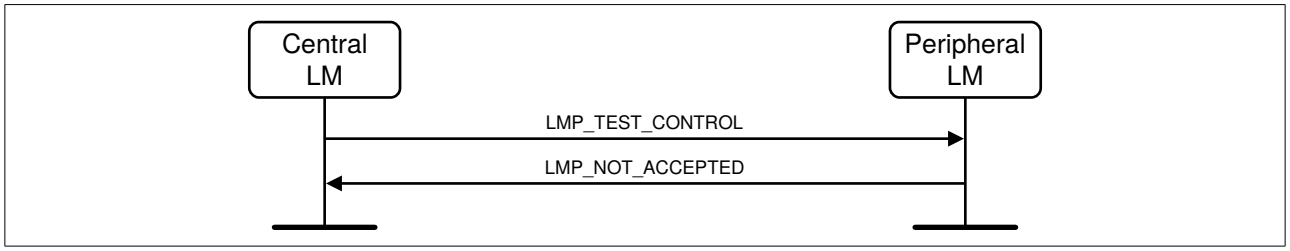
A Bluetooth device in test mode shall ignore all LMP commands not related to control of the test mode. LMP commands dealing with power control and the request for LMP features (LMP_FEATURES_REQ), and adaptive frequency hopping (LMP_SET_AFH, LMP_CHANNEL_CLASSIFICATION_REQ and LMP_CHANNEL_CLASSIFICATION) are allowed in test mode; the normal procedures are also used to test the adaptive power control.

The IUT shall leave the test mode when an LMP_DETACH command is received or an LMP_TEST_CONTROL command is received with Test_Scenario set to 'exit test mode'.

When the IUT has entered test mode, the LMP_TEST_CONTROL PDU can be sent to the IUT to start a specific test. This PDU is acknowledged with an LMP_ACCEPTED PDU. If a device that is not in test mode receives an LMP_TEST_CONTROL PDU, then it responds with an LMP_NOT_ACCEPTED PDU, where the Error_Code shall be *LMP PDU Not Allowed* (0x24).




*Sequence 102: Control of Test mode successful*




*Sequence 103: Control of Test mode rejected since Peripheral is not in Test mode*

### 4.7.3 Summary of Test mode PDUs

Table 4.36 lists all LMP messages used for test mode. To ensure that the contents of the LMP_TEST_CONTROL PDU are suitably whitened (important when sent in transmitter mode), all parameters listed in Table 4.37 are XORed with 0x55 before being sent.

| LMP PDU | PDU number | Possible Direction | Contents | Position in Payload |
|---|---|---|---|---|
| LMP_TEST_ACTIVATE | 56 | C → P | *none* | |
| LMP_TEST_CONTROL | 57 | C → P | Test_Scenario | 2 |
| | | | Hopping_Mode | 3 |
| | | | Tx_Frequency | 4 |
| | | | Rx_Frequency | 5 |
| | | | Power_Mode | 6 |
| | | | Poll_Period | 7 |
| | | | Packet_Type | 8 |
| | | | Test_Data_Length | 9-10 |

| LMP PDU | PDU number | Possible Direction | Contents | Position in Payload |
|---|---|---|---|---|
| LMP_DETACH | 7 | C → P | *none* | |
| LMP_ACCEPTED | 3 | C ← P | *none* | |
| LMP_NOT_ACCEPTED | 4 | C ← P | *none* | |

*Table 4.36: LMP messages used for Test mode*

| Name | Length (bytes) | Type | Detailed |
|---|---|---|---|
| Test_Scenario | 1 | uint8 | 0 Pause Test Mode<br>1 Transmitter test – 0 pattern<br>2 Transmitter test – 1 pattern<br>3 Transmitter test – 1010 pattern<br>4 Pseudorandom bit sequence<br>5 Closed Loop Back – ACL packets<br>6 Closed Loop Back – Synchronous packets<br>7 ACL packets without whitening<br>8 Synchronous packets without whitening<br>9 Transmitter test – 1111 0000 pattern<br>10–254 Reserved for future use<br>255 Exit Test Mode<br>The value is XORed with 0x55. |
| Hopping_Mode | 1 | uint8 | 0 RX/TX on single frequency<br>1 Normal hopping<br>2–255 Reserved for future use<br>The value is XORed with 0x55. |
| Tx_Frequency | 1 | uint8 | f = [2402 + k] MHz<br>The value is XORed with 0x55. |
| Rx_Frequency | 1 | uint8 | f = [2402 + k] MHz<br>The value is XORed with 0x55. |
| Power_Mode | 1 | uint8 | 0 fixed TX output power<br>1 adaptive power control<br>The value is XORed with 0x55. |
| Poll_Period | 1 | uint8 | Unit: 1.25 ms<br>The value is XORed with 0x55. |

| Name | Length (bytes) | Type | Detailed |
|---|---|---|---|
| Packet_Type | 1 | uint8 | Bits 3-0<br>numbering as in packet header, see [Vol 2] Part B, Section 6.4.2<br>Bits 7-4<br>0: ACL/SCO<br>1: eSCO<br>2: Enhanced Data Rate ACL<br>3: Enhanced Data Rate eSCO<br>4-15: Reserved for future use<br>The value is XORed with 0x55. |
| Test_Data_Length | 2 | uint16 | This is equal to the length of user data in [Vol 2] Part B, Section 6.5, in bytes, excluding the payload header and CRC if present in the relevant type of packet.<br>The value is XORed with 0x5555. |

*Table 4.37: Parameters used in LMP_TEST_CONTROL PDU*

The control PDU is used for both transmitter and loop back tests. The following restrictions apply for the parameter settings:

| Parameter | Restrictions Transmitter Test | Restrictions Loopback Test |
|---|---|---|
| Tx_Frequency | 0 ≤ k ≤ 78 | 0 ≤ k ≤ 78 |
| Rx_Frequency | same as Tx_Frequency | 0 ≤ k ≤ 78 |
| Poll_Period | *none* | not applicable (set to 0) |
| Test_Data_Length | Depends on packet type. See [Vol 2] Part B, Table 6.8 and [Vol 2] Part B, Table 6.9 | For ACL and SCO packets:<br>not applicable (set to 0)<br>For eSCO packets:<br>see [Vol 2] Part B, Table 6.8 |

*Table 4.38: Restrictions on the parameters in the LMP_TEST_CONTROL PDU*

# 5 SUMMARY

## 5.1 PDU summary

Where the "Possible direction" column in Table 5.1 shows "B", the PDU is sent on the APB-C logical link. All other PDUs are sent on the ACL-C logical link and only in the direction(s) indicated.

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_ACCEPTED | 2 | 3 | DM1/DV | C ↔ P | Opcode | 2 |
| LMP_ACCEPTED_-EXT | 4 | 127/01 | DM1 | C ↔ P | Escape_Opcode | 3 |
| | | | | | Extended_-Opcode | 4 |
| LMP_AU_RAND | 17 | 11 | DM1 | C ↔ P | Random_Number | 2–17 |
| LMP_AUTO_RATE | 1 | 35 | DM1/DV | C ↔ P | *none* | |
| LMP_CHANNEL_-CLASSIFICATION | 12 | 127/17 | DM1 | C ← P | AFH_Channel_-Classification | 3–12 |
| LMP_CHANNEL_-CLASSIFICATION_-REQ | 7 | 127/16 | DM1 | C → P | AFH_Reporting_-Mode | 3 |
| | | | | | AFH_Min_Interval | 4–5 |
| | | | | | AFH_Max_-Interval | 6–7 |
| LMP_CLK_ADJ | 15 | 127/5 | DM1 | B | Clk_Adj_ID | 3 |
| | | | | | Clk_Adj_Instant | 4–7 |
| | | | | | Clk_Adj_Offset | 8–9 |
| | | | | | Clk_Adj_Slots | 10 |
| | | | | | Clk_Adj_Mode | 11 |
| | | | | | Clk_Adj_Clk | 12–15 |
| LMP_CLK_ADJ_-ACK | 3 | 127/6 | DM1 | C ← P | Clk_Adj_ID | 3 |
| LMP_CLK_ADJ_-REQ | 6 | 127/7 | DM1 | C ← P | Clk_Adj_Offset | 3–4 |
| | | | | | Clk_Adj_Slots | 5 |
| | | | | | Clk_Adj_Period | 6 |
| LMP_CLKOFFSET_-REQ | 1 | 5 | DM1/DV | C → P | *none* | |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_CLKOFFSET_-RES | 3 | 6 | DM1/DV | C ← P | Clock_Offset | 2–3 |
| LMP_COMB_KEY | 17 | 9 | DM1 | C ↔ P | Random_Number | 2–17 |
| LMP_DECR_-POWER_REQ | 2 | 32 | DM1/DV | C ↔ P | Reserved | 2 |
| LMP_DETACH | 2 | 7 | DM1/DV | C ↔ P | Error_Code | 2 |
| LMP_DHKEY_-CHECK | 17 | 65 | DM1 | C ↔ P | Confirmation_-Value | 2–17 |
| LMP_-ENCAPSULATED_-HEADER | 4 | 61 | DM1 | C ↔ P | Encap_Major_-Type | 2 |
| | | | | | Encap_Minor_-Type | 3 |
| | | | | | Encap_Payload_-Length | 4 |
| LMP_-ENCAPSULATED_-PAYLOAD | 17 | 62 | DM1 | C ↔ P | Encap_Data | 2–17 |
| LMP_-ENCRYPTION_-KEY_SIZE_MASK_-REQ | 1 | 58 | DM1 | C → P | *none* | |
| LMP_-ENCRYPTION_-KEY_SIZE_MASK_-RES | 3 | 59 | DM1 | C ← P | Key_Size_Mask | 2–3 |
| LMP_-ENCRYPTION_-KEY_SIZE_REQ | 2 | 16 | DM1/DV | C ↔ P | Key_Size | 2 |
| LMP_-ENCRYPTION_-MODE_REQ | 2 | 15 | DM1/DV | C ↔ P | Encryption_Mode | 2 |
| LMP_eSCO_LINK_-REQ (see Note 1) | 16 | 127/12 | DM1 | C ↔ P | eSCO_Handle | 3 |
| | | | | | eSCO_LT_ADDR | 4 |
| | | | | | Timing_Control_-Flags | 5 |
| | | | | | $D_{eSCO}$ | 6 |
| | | | | | $T_{eSCO}$ | 7 |
| | | | | | $W_{eSCO}$ | 8 |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| | | | | | eSCO_Packet_-Type C→P | 9 |
| | | | | | eSCO_Packet_-Type P→C | 10 |
| | | | | | Packet_Length C→P | 11–12 |
| | | | | | Packet_Length P→C | 13–14 |
| | | | | | Air_Mode | 15 |
| | | | | | Negotiation_State | 16 |
| LMP_FEATURES_-REQ | 9 | 39 | DM1/DV | C ↔ P | Features | 2–9 |
| LMP_FEATURES_-REQ_EXT | 12 | 127/03 | DM1 | C ↔ P | Features_Page | 3 |
| | | | | | Max_Supported_-Page | 4 |
| | | | | | Extended_-Features | 5–12 |
| LMP_FEATURES_-RES | 9 | 40 | DM1/DV | C ↔ P | Features | 2–9 |
| LMP_FEATURES_-RES_EXT | 12 | 127/04 | DM1 | C ↔ P | Features_Page | 3 |
| | | | | | Max_Supported_-Page | 4 |
| | | | | | Extended_-Features | 5–12 |
| LMP_HOLD | 7 | 20 | DM1/DV | C ↔ P | Hold_Time | 2–3 |
| | | | | | Hold_Instant | 4–7 |
| LMP_HOLD_REQ | 7 | 21 | DM1/DV | C ↔ P | Hold_Time | 2–3 |
| | | | | | Hold_Instant | 4–7 |
| LMP_HOST_-CONNECTION_REQ | 1 | 51 | DM1/DV | C ↔ P | *none* | |
| LMP_IN_RAND | 17 | 8 | DM1 | C ↔ P | Random_Number | 2–17 |
| LMP_INCR_-POWER_REQ | 2 | 31 | DM1/DV | C ↔ P | Reserved | 2 |
| LMP_IO_-CAPABILITY_REQ | 5 | 127/25 | DM1 | C ↔ P | IO_Capabilities | 3 |
| | | | | | OOB_Auth_Data | 4 |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| | | | | | Authentication_-Requirements | 5 |
| LMP_IO_-CAPABILITY_RES | 5 | 127/26 | DM1 | C ↔ P | IO_Capabilities | 3 |
| | | | | | OOB_Auth_Data | 4 |
| | | | | | Authentication_-Requirements | 5 |
| LMP_KEYPRESS_-NOTIFICATION | 3 | 127/30 | DM1 | C ↔ P | Notification_Type | 2 |
| LMP_MAX_POWER | 1 | 33 | DM1/DV | C ↔ P | *none* | |
| LMP_MAX_SLOT | 2 | 45 | DM1/DV | C ↔ P | Max_Slots | 2 |
| LMP_MAX_SLOT_-REQ | 2 | 46 | DM1/DV | C ↔ P | Max_Slots | 2 |
| LMP_MIN_POWER | 1 | 34 | DM1/DV | C ↔ P | *none* | |
| LMP_NAME_REQ | 2 | 1 | DM1/DV | C ↔ P | Name_Offset | 2 |
| LMP_NAME_RES | 17 | 2 | DM1 | C ↔ P | Name_Offset | 2 |
| | | | | | Name_Length | 3 |
| | | | | | Name_Fragment | 4–17 |
| LMP_NOT_-ACCEPTED | 3 | 4 | DM1/DV | C ↔ P | Opcode | 2 |
| | | | | | Error_Code | 3 |
| LMP_NOT_-ACCEPTED_EXT | 5 | 127/02 | DM1 | C ↔ P | Escape_Opcode | 3 |
| | | | | | Extended_-Opcode | 4 |
| | | | | | Error_Code | 5 |
| LMP_NUMERIC_-COMPARISON_-FAILED | 2 | 127/27 | DM1 | C ↔ P | *none* | |
| LMP_OOB_FAILED | 2 | 127/29 | DM1 | C ↔ P | *none* | |
| LMP_PACKET_-TYPE_TABLE_REQ | 3 | 127/11 | DM1 | C ↔ P | Packet_Type_-Table | 3 |
| LMP_PAGE_-MODE_REQ | 3 | 53 | DM1/DV | C ↔ P | Paging_Scheme | 2 |
| | | | | | Paging_-Scheme_Settings | 3 |
| LMP_PAGE_-SCAN_MODE_REQ | 3 | 54 | DM1/DV | C ↔ P | Paging_Scheme | 2 |
| | | | | | Paging_-Scheme_Settings | 3 |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_PASSKEY_-FAILED | 2 | 127/28 | DM1 | C ↔ P | *none* | |
| LMP_PAUSE_-ENCRYPTION_-AES_REQ | 17 | 66 | DM1 | C ↔ P | Random_Number | 2–17 |
| LMP_PAUSE_-ENCRYPTION_REQ | 2 | 127/23 | DM1 | C ↔ P | *none* | |
| LMP_PING_REQ | 2 | 127/33 | DM1 | C ↔ P | *none* | |
| LMP_PING_RES | 2 | 127/34 | DM1 | C ↔ P | *none* | |
| LMP_POWER_-CONTROL_REQ | 3 | 127/31 | DM1/DV | C ↔ P | Power_Adj_Req | 3 |
| LMP_POWER_-CONTROL_RES | 3 | 127/32 | DM1/DV | C ↔ P | Power_Adj_Rsp | 3 |
| LMP_PREFER-RED_RATE | 2 | 36 | DM1/DV | C ↔ P | Data_Rate | 2 |
| LMP_QUALITY_-OF_SERVICE | 4 | 41 | DM1/DV | C → P | Poll_Interval | 2–3 |
| | | | | | $N_{BC}$ | 4 |
| LMP_QUALITY_-OF_SERVICE_REQ | 4 | 42 | DM1/DV | C ↔ P | Poll_Interval | 2–3 |
| | | | | | $N_{BC}$ | 4 |
| LMP_REMOVE_-eSCO_LINK_REQ (see Note 1) | 4 | 127/13 | DM1 | C ↔ P | eSCO_Handle | 3 |
| | | | | | Error_Code | 4 |
| LMP_REMOVE_-SCO_LINK_REQ | 3 | 44 | DM1/DV | C ↔ P | SCO_Handle | 2 |
| | | | | | Error_Code | 3 |
| LMP_RESUME_-ENCRYPTION_REQ | 2 | 127/24 | DM1 | C ← P | *none* | |
| LMP_SAM_-DEFINE_MAP | 17 | 127/36 | DM1 | C ↔ P | SAM_Index | 3 |
| | | | | | $T_{SAM_SM}$ | 4 |
| | | | | | $N_{SAM_SM}$ | 5 |
| | | | | | SAM_Submaps | 6–17 |
| LMP_SAM_SET_-TYPE0 | 17 | 127/35 | DM1 | C ↔ P | Update_Mode | 3 |
| | | | | | SAM_Type0_-Submap | 4–17 |
| LMP_SAM_SWITCH | 9 | 127/37 | DM1 | C ↔ P | SAM_Index | 3 |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| | | | | | Timing_Control_-Flags | 4 |
| | | | | | $D_{SAM}$ | 5 |
| | | | | | SAM_Instant | 6–9 |
| LMP_SCO_LINK_-REQ | 7 | 43 | DM1/DV | C ↔ P | SCO_Handle | 2 |
| | | | | | Timing_Control_-Flags | 3 |
| | | | | | $D_{sco}$ | 4 |
| | | | | | $T_{sco}$ | 5 |
| | | | | | SCO_Packet | 6 |
| | | | | | Air_Mode | 7 |
| LMP_SET_AFH | 16 | 60 | DM1 | C → P | AFH_Instant | 2–5 |
| | | | | | AFH_Mode | 6 |
| | | | | | AFH_Channel_-Map | 7–16 |
| LMP_SETUP_-COMPLETE | 1 | 49 | DM1 | C ↔ P | *none* | |
| LMP_SIMPLE_-PAIRING_CONFIRM | 17 | 63 | DM1 | C ↔ P | Commitment_-Value | 2–17 |
| LMP_SIMPLE_-PAIRING_NUMBER | 17 | 64 | DM1 | C ↔ P | Nonce_Value | 2–17 |
| LMP_SLOT_-OFFSET | 9 | 52 | DM1/DV | C ↔ P | Slot_Offset | 2–3 |
| | | | | | BD_ADDR | 4–9 |
| LMP_SNIFF_REQ | 10 | 23 | DM1 | C ↔ P | Timing_Control_-Flags | 2 |
| | | | | | $D_{Sniff}$ | 3–4 |
| | | | | | $T_{Sniff}$ | 5–6 |
| | | | | | Sniff_Attempt | 7–8 |
| | | | | | Sniff_Timeout | 9–10 |
| LMP_SNIFF_-SUBRATING_REQ | 9 | 127/21 | DM1 | C ↔ P | Max_Sniff_-Subrate | 3 |
| | | | | | Min_Sniff_Mode_-Timeout | 4–5 |
| | | | | | Sniff_Subrating_-Instant | 6–9 |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_SNIFF_-SUBRATING_RES | 9 | 127/22 | DM1 | C ↔ P | Max_Sniff_-Subrate | 3 |
| | | | | | Min_Sniff_Mode_-Timeout | 4–5 |
| | | | | | Sniff_Subrating_-Instant | 6–9 |
| LMP_SRES | 5 | 12 | DM1/DV | C ↔ P | Authentication_-Rsp | 2–5 |
| LMP_START_-ENCRYPTION_REQ | 17 | 17 | DM1 | C → P | Random_Number | 2–17 |
| LMP_STOP_-ENCRYPTION_REQ | 1 | 18 | DM1/DV | C → P | *none* | |
| LMP_SUPERVI-SION_TIMEOUT | 3 | 55 | DM1/DV | C → P | Supervision_-Timeout | 2–3 |
| LMP_SWITCH_REQ | 5 | 19 | DM1 | C ↔ P | Switch_Instant | 2–5 |
| LMP_TEMP_KEY | 17 | 14 | DM1 | C → P | Key | 2–17 |
| LMP_TEMP_RAND | 17 | 13 | DM1 | C → P | Random_Number | 2–17 |
| LMP_TEST_-ACTIVATE | 1 | 56 | DM1/DV | C → P | *none* | |
| LMP_TEST_-CONTROL | 10 | 57 | DM1 | C → P | Test_Scenario | 2 |
| | | | | | Hopping_Mode | 3 |
| | | | | | Tx_Frequency | 4 |
| | | | | | Rx_Frequency | 5 |
| | | | | | Power_Mode | 6 |
| | | | | | Poll_Period | 7 |
| | | | | | Packet_Type | 8 |
| | | | | | Test_Data_Length | 9–10 |
| LMP_TIMING_-ACCURACY_REQ | 1 | 47 | DM1/DV | C ↔ P | *none* | |
| LMP_TIMING_-ACCURACY_RES | 3 | 48 | DM1/DV | C ↔ P | Drift | 2 |
| | | | | | Jitter | 3 |
| LMP_UNIT_KEY | 17 | 10 | DM1 | C ↔ P | Key | 2–17 |
| LMP_UNSNIFF_-REQ | 1 | 24 | DM1/DV | C ↔ P | *none* | |

| LMP PDU | Length (bytes) | Opcode | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_USE_SEMI_-PERMANENT_KEY | 1 | 50 | DM1/DV | C → P | *none* | |
| LMP_VERSION_-REQ | 6 | 37 | DM1/DV | C ↔ P | Version | 2 |
| | | | | | Company_-Identifier | 3–4 |
| | | | | | Subversion | 5–6 |
| LMP_VERSION_-RES | 6 | 38 | DM1/DV | C ↔ P | Version | 2 |
| | | | | | Company_-Identifier | 3–4 |
| | | | | | Subversion | 5–6 |

*Table 5.1: Coding of the different LM PDUs*

**Note 1.** Parameters coincide with their namesakes in LMP_SCO_LINK_REQ and LMP_REMOVE_SCO_LINK_REQ apart from the following:

1. eSCO_LT_ADDR - the eSCO connection will be active on an additional LT_ADDR that needs to be defined. The Central is not allowed to re-assign an active eSCO link to a different LT_ADDR.
2. $D_{eSCO}$, $T_{eSCO}$ - as per LMP_SCO_LINK_REQ but with a greater flexibility in values (e.g. no longer fixed with respect to HV1, HV2, and HV3 packet choice).
3. $W_{eSCO}$ - the eSCO retransmission window size (in slots)
4. packet type and packet length may be prescribed differently in Central-to-Peripheral or Peripheral-to-Central directions for asynchronous eSCO links
5. packet length (in bytes) - eSCO packet types no longer have fixed length
6. negotiation state – this is used to better enable the negotiation of the negotiable parameters: $D_{eSCO}$, $T_{eSCO}$, $W_{eSCO}$, eSCO_Packet_Type C→P, eSCO_Packet_Type P→C, Packet_Length C→P, Packet_Length P→C. When responding to an eSCO link request with a new suggestion for these parameters, this flag may be set to 1 to indicate that the last received negotiable parameters are possible, but the new parameters specified in the response eSCO link request would be preferable, to 2 to indicate that the last received negotiable parameters are not possible as they cause a reserved slot violation or to 3 to indicate that the last received negotiable parameters would cause a latency violation. The flag is set to zero in the initiating LMP_eSCO_LINK_REQ.

Note: The following opcodes were previously used: 22, 25 to 30. All other opcodes not listed in Table 5.1 are reserved for future use.

## 5.2 Parameter definitions

Where no mandatory range is given, the mandatory range consists of all valid values that are not reserved for future use.

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Access_Scheme | 1 | uint4 | | 0: polling technique<br>1-15: reserved for future use |
| AFH_Channel_Classification | 10 | uint2 [40] | | The $n^{th}$ (numbering from 0) element defines the classification of channels *2n* and *2n+1*, other than the $39^{th}$ element which just contains the classification of channel 78.<br>The value of each element indicates:<br>0 = unknown<br>1 = good<br>2 = reserved for future use<br>3 = bad |
| AFH_Channel_Map | 10 | uint1 [80] | | If *AFH_Mode* is not 1 (enabled), the contents of this parameter are reserved for future use. Otherwise:<br>The $n^{th}$ (numbering from 0) element (in the range 0 to 78) contains the value for channel *n*.<br>Element 79 is reserved for future use.<br>The value of each element indicates:<br>0: channel *n* is unused<br>1: channel *n* is used |
| AFH_Instant | 4 | uint32 | slots | Bits 27:1 of the Central's clock value at the time of switching hop sequences. Only even values are valid. |
| AFH_Max_Interval | 2 | uint16 | slots | Range is 0x0640 to 0xBB80 slots (1 to 30 s)<br>Only even values are valid |
| AFH_Min_Interval | 2 | uint16 | slots | Range is 0x0640 to 0xBB80 slots (1 to 30 s)<br>Only even values are valid |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| AFH_Mode | 1 | uint8 | | 0: disabled<br>1: enabled<br>2-255: reserved for future use |
| AFH_Reporting_Mode | 1 | uint8 | | 0: disabled<br>1: enabled<br>2-255: reserved for future use |
| Air_Mode | 1 | uint8 | | 0: μ-law log<br>1: A-law log<br>2: CVSD<br>3: transparent data<br>4-255: reserved for future use |
| Authentication_Rsp | 4 | multiple bytes | | |
| Authentication_-Requirements | 1 | uint8 | | 0x00: MITM Protection Not Required – No Bonding<br>0x01: MITM Protection Required – No Bonding<br>0x02: MITM Protection Not Required – Dedicated Bonding<br>0x03: MITM Protection Required – Dedicated Bonding<br>0x04: MITM Protection Not Required – General Bonding<br>0x05: MITM Protection Required – General Bonding<br>0x06 to 0xFF: reserved for future use |
| BD_ADDR | 6 | multiple bytes | | BD_ADDR of the sending device |
| Clk_Adj_Clk | 4 | uint32 | slot pairs | CLK[27:2] for the moment that the PDU is transmitted. |
| Clk_Adj_ID | 1 | uint8 | | Clk_Adj_ID is chosen by the Central as a handle to identify a Coarse Clock Adjustment event. |
| Clk_Adj_Instant | 4 | uint32 | slots | $CLK_{old}$[27:1] at the time of the coarse clock adjustment, based on the value of time_base_offset before the adjustment is made. Only even values are valid. |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Clk_Adj_Mode | 1 | uint8 | | 0: Before Instant<br>1: After Instant<br>2-255: reserved for future use |
| Clk_Adj_Offset | 2 | sint16 | µs | The offset between the old and new slot boundaries. Valid range is –624 to +624. |
| Clk_Adj_Period | 1 | uint8 | slots | Indicates to the Central that adding an integer multiple of this value to Clk_Adj_Slots gives an equally acceptable adjustment. The value of Clk_Adj_Period shall be zero or an even number greater than Clk_Adj_Slots. Only even values are valid. |
| Clk_Adj_Slots | 1 | uint8 | slots | The difference between the clocks at the adjustment instant (i.e. $CLK_{new}[27:1]$ - $CLK_{old}[27:1]$). |
| Clock_Offset | 2 | uint15 | 1.25 ms | ($CLKN_{16\text{-}2\ Peripheral}$ - $CLKN_{16\text{-}2\ Central}$) *mod* $2^{15}$ |
| Commitment_Value | 16 | uint128 | | |
| Company_Identifier | 2 | uint16 | | see Assigned Numbers |
| Confirmation_Value | 16 | uint128 | | |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Data_Rate | 1 | uint8 | | When in Basic Rate mode:<br>bit 0 = 0: use FEC<br>bit 0 = 1: do not use FEC<br>bits 1-2=0: No packet-size preference available<br>bits 1-2=1: use 1-slot packets<br>bits 1-2=2: use 3-slot packets<br>bits 1-2=3: use 5-slot packets<br>When in Enhanced Data Rate mode:<br>bits 3-4=0: use DM1 packets<br>bits 3-4=1: use 2 Mb/s packets<br>bits 3-4=2: use 3 Mb/s packets<br>bits 3-4=3: reserved for future use<br>bits 5-6=0: No packet-size preference available<br>bits 5-6=1: use 1-slot packets<br>bits 5-6=2: use 3-slot packets<br>bits 5-6=3: use 5-slot packets<br>bit 7: reserved for future use |
| $D_{eSCO}$ | 1 | uint8 | slots | Only even values less than $T_{eSCO}$ are valid |
| Drift | 1 | uint8 | ppm | |
| $D_{SAM}$ | 1 | uint8 | slots | Only even values less than $T_{SAM}$ are valid |
| $D_{SCO}$ | 1 | uint8 | slots | Only even values less than $T_{SCO}$ are valid |
| $D_{Sniff}$ | 2 | uint16 | slots | Only even values less than $T_{Sniff}$ are valid |
| Encap_Data | 16 | Multiple bytes | | MSBs zero padded when data is less than 16 bytes.<br>Little-endian format |
| Encap_Major_Type | 1 | uint8 | | See Table 5.4 |
| Encap_Minor_Type | 1 | uint8 | | See Table 5.4 |
| Encap_Payload_Length | 1 | uint8 | | See Table 5.4 |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Encryption_Mode | 1 | uint8 | | 0: no encryption<br>1: encryption<br>2: previously used<br>3-255: reserved for future use |
| Error_Code | 1 | uint8 | | See [Vol 1] Part F |
| Escape_Opcode | 1 | uint8 | | Identifies which Escape_Opcode is being acknowledged: range 124-127 |
| eSCO_Handle | 1 | uint8 | | |
| eSCO_LT_ADDR | 1 | uint3 | | Logical transport address for the eSCO logical transport. Valid range is 1-7. |
| eSCO_Packet_Type | 1 | uint8 | | 0x00 (C → P): POLL<br>0x00 (P → C): NULL<br>0x07: EV3<br>0x0C: EV4<br>0x0D: EV5<br>0x26: 2-EV3<br>0x2C: 2-EV5<br>0x37: 3-EV3<br>0x3D: 3-EV5<br>Other values are reserved for future use |
| Extended_Features | 8 | uint1 [64] | | The $n^{th}$ (numbering from 0) element represents feature number 64P + *n* in Tables 3.2 onwards, where P is the relevant features page number. |
| Extended_Opcode | 1 | uint8 | | Which Extended_Opcode is being acknowledged |
| Features | 8 | uint1 [64] | | The $n^{th}$ (numbering from 0) element represents feature number *n* in Table 3.2 |
| Features_Page | 1 | uint8 | | Identifies which page of extended features is being requested.<br>0 means standard features<br>1-255 other feature pages |
| Hold_Instant | 4 | uint32 | slots | Bits 27:1 of the Central's clock value.<br>Only even values are valid |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Hold_Time | 2 | uint16 | slots | Only even values less than or equal to (*supervisionTO* × 0.999) are valid. Mandatory range 0x0014 to 0x8000. |
| IO_Capabilities | 1 | uint8 | | 0: Display only<br>1: Display YesNo<br>2: KeyboardOnly<br>3: NoInputNoOutput<br>4-255: reserved for future use |
| Jitter | 1 | uint8 | µs | |
| Key | 16 | multiple bytes | | |
| Key_Size | 1 | uint8 | byte | Valid range: 7-16 |
| Key_Size_Mask | 2 | uint1 [16] | | Supported broadcast encryption key sizes: first element is support for length 1, and so on. The bit shall be one if the key size is supported. Bits 0 to 6 shall be zero. |
| Max_Slots | 1 | uint8 | slots | |
| Max_Sniff_Subrate | 1 | uint8 | subrate | Valid range: 1-255 |
| Max_Supported_Page | 1 | uint8 | | Highest page of extended features which contains a non-zero bit for the originating device. Range 0-255 |
| Min_Sniff_Mode_Timeout | 2 | uint16 | slots | Only even values are valid |
| Name_Fragment | 14 | utf8s{14z} | | UTF-8 characters. |
| Name_Length | 1 | uint8 | bytes | |
| Name_Offset | 1 | uint8 | bytes | |
| $N_{BC}$ | 1 | uint8 | | Minimum number of times that an APB broadcast packet should be sent. |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Negotiation_State | 1 | uint8 | | 0: Initiate negotiation<br>1: the latest received set of negotiable parameters were possible but these parameters are preferred.<br>2: the latest received set of negotiable parameters would cause a reserved slot violation.<br>3: the latest received set of negotiable parameters would cause a latency violation.<br>4: the latest received set of negotiable parameters are not supported.<br>Other values are reserved for future use |
| Nonce_Value | 16 | Multiple bytes | | Little-endian Format |
| Notification_Type | 1 | uint8 | | 0=passkey entry started<br>1=passkey digit entered<br>2=passkey digit erased<br>3=passkey cleared<br>4=passkey entry completed<br>5-255: reserved for future use |
| $N_{Poll}$ | 1 | uint8 | | |
| $N_{SAM_SM}$ | 1 | uint8 | submaps | Number of submaps in the SAM slot map; range 0 to 48 |
| OOB_Auth_Data | 1 | uint8 | | 0: No OOB Authentication Data received<br>1: OOB Authentication Data received<br>2-255: reserved for future use |
| Opcode | 1 | uint8 | | |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Packet_Length | 2 | uint16 | bytes | Length of the eSCO payload<br>0 for POLL/NULL<br>1-30 for EV3<br>1-120 for EV4<br>1-180 for EV5<br>1-60 for 2-EV3<br>1-360 for 2-EV5<br>1-90 for 3-EV3<br>1-540 for 3-EV5<br>Other values are invalid |
| Packet_Type_Table | 1 | uint8 | | 0: 1 Mb/s only<br>1: 2/3 Mb/s<br>2-255: reserved for future use |
| Paging_Scheme | 1 | uint8 | | 0: mandatory scheme<br>1-255: reserved for future use |
| Paging_Scheme_Settings | 1 | uint8 | | For mandatory scheme:<br>0: R0<br>1: R1<br>2: R2<br>3-255: reserved for future use |
| Poll_Interval | 2 | uint16 | slots | Only even values are valid. Mandatory range 0x0006 to 0x1000. |
| Power_Adj_Req | 1 | uint8 | | 0: decrement power one step<br>1: increment power one step<br>2: increase to maximum power<br>3-255: reserved for future use |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Power_Adj_Rsp | 1 | uint2 [4] | | element 0: GFSK<br>element 1: π/4-DQPSK<br>element 2: 8DPSK<br>element 3: reserved for future use<br>Each 2-bit value is defined as follows<br>0: not supported<br>1: changed one step, (not min or max)<br>2: max power<br>3: min power |
| Random_Number | 16 | multiple bytes | | |
| Reserved | 1 | uint8 | | Reserved for future use |
| SAM_Index | 1 | uint8 | | Index of the SAM slot map. Mandatory values 0, 1, 2, and 0xFF. |
| SAM_Instant | 4 | uint32 | slots | CLK[27:1] at the instant the SAM slot map is to be activated. Only even values are valid. |
| SAM_Submaps | 12 | uint2 [48] | | This parameter contains 48 2-bit fields. Only the first $N_{SAM_SM}$ fields are significant; the remainder are reserved for future use.<br>The $n^{th}$ (numbering from 0) such field defines the submap type of the $n^{th}$ submap in the map.<br>The meanings of the possible values are:<br>0 = Each slot is individually available or unavailable as configured. Slots may have different availabilities for transmission and reception<br>1 = All slots are available for transmission and reception<br>2 = All slots are unavailable for transmission and reception<br>3 = reserved for future use |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| SAM_Type0_Submap | 14 | uint2 [56] | | This parameter contains 56 2-bit fields.<br>The $n^{th}$ (numbering from 0) such field defines the slot type of the $n^{th}$ slot.<br>The meanings of the possible values are:<br>0 = The slot is not available for either transmission or reception<br>1 = The slot is available for transmission but not reception<br>2 = The slot is available for reception but not transmission<br>3 = The slot is available for both transmission and reception |
| SCO_Handle | 1 | uint8 | | |
| SCO_Packet | 1 | uint8 | | 0: HV1<br>1: HV2<br>2: HV3<br>3-255: reserved for future use |
| Slot_Offset | 2 | uint16 | µs | Valid range is 0 to 1249 |
| Sniff_Attempt | 2 | uint16 | received slots | Number of receive slots. Mandatory range 1 to $T_{sniff}$ ÷2 |
| Sniff_Subrating_Instant | 4 | uint32 | slots | Bits 27:1 of the Central's clock value<br>Only even values are valid |
| Sniff_Timeout | 2 | uint16 | received slots | Number of receive slots. Mandatory range 0 to 0x0028. |
| Subversion | 2 | uint16 | | Defined by each company |
| Supervision_Timeout | 2 | uint16 | slots | Mandatory values 0 and 0x0190 to 0xFFFF. 0 means an infinite timeout |
| Switch_Instant | 4 | uint32 | slots | Bits 27:1 of the Central's clock value<br>Only even values are valid |
| $T_{eSCO}$ | 1 | uint8 | slots | Valid range is 4 – 254 slots Only even values are valid |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Timing_Control_Flags | 1 | uint8 | | bit 0 is ignored by the recipient<br>bit 1 = 0: use initialization 1<br>bit 1 = 1: use initialization 2<br>bit 2 is ignored by the recipient<br>bits 3-7: reserved for future use |
| $T_{SAM_SM}$ | 1 | uint8 | slots | Length of each SAM submap.<br>Range 2 to 56.<br>Only even values are valid |
| $T_{sco}$ | 1 | uint8 | slots | Only even values are valid. Mandatory values 2 and 6. |
| $T_{Sniff}$ | 2 | uint16 | slots | Only even values less than or equal to (*supervisionTO* × 0.999) are valid. Mandatory values: valid values in the range 0x0006 to 0x0540. |
| Update_Mode | 1 | uint8 | | 0: Existing SAM slot maps containing any type 0 submaps are invalidated<br>1: The defined type 0 submap takes effect immediately<br>2:The defined type 0 submap takes effect at the start of the next sub-interval<br>All other values are reserved for future use. |
| Version | 1 | uint8 | | See Assigned Numbers |
| $W_{eSCO}$ | 1 | uint8 | slots | Number of slots in the retransmission window<br>Valid range is 0 – 254 slots Only even values are valid |

*Table 5.2: Parameters in LM PDUs*

If a parameter is described as "only even values are valid" and a device receives an LMP PDU with an odd value for this parameter field, the PDU should be rejected with an error code of *Invalid LMP Parameters* (0x1E).

Where a parameter is described as bits N:M of some other value V, the parameter value in the PDU shall be V shifted right by M bits. Where the space available for the parameter is greater than that needed (N-M+1 bits), the parameter value still occupies the entire space and the remaining bits shall be zero.

For example, if a 16 bit parameter is equal to 0xDEAD and bits 10:6 are stored in a uint8, the resulting parameter will be 0x1A.

The recipient shall accept any of the values for the eSCO parameter listed in Table 5.3 and may accept any other value listed for the parameter in Table 5.2.

| | **Single Slot Packets** | **3-Slot Packets** |
|---|---|---|
| **Packet type and $T_{eSCO}$** | EV3: 6<br>2-EV3: 6-12 (even)<br>3-EV3: 6-18 (even) | EV4: 16<br>EV5: 16<br>2-EV5: 16<br>3-EV5: 16 |
| **$W_{eSCO}$** | 0, 2, and 4 | 0 and 6 |
| **Packet_Length (in each direction)** | 10 × $T_{eSCO}$ ÷ 2 | 10 × $T_{eSCO}$ ÷ 2 |
| **Air_Mode** | At least one of A-law, μ-law, CVSD, transparent | transparent |

*Table 5.3: Mandatory parameter ranges for eSCO packet types*

## 5.3 LMP encapsulated

| **Name** | **Major Type** | **Minor Type** | **Payload Length** | **Detailed** |
|---|---|---|---|---|
| P-192 Public Key | 1 | 1 | 48 | X, Y format<br>Bytes 23-0: X co-ordinate<br>Bytes 47-24: Y co-ordinate<br>Little-endian Format |
| P-256 Public Key | 1 | 2 | 64 | X, Y format<br>Bytes 31-0: X co-ordinate<br>Bytes 63-32: Y co-ordinate<br>Little-endian Format |

*Table 5.4: LMP encapsulated*

All other combinations of major and minor type are reserved for future use.

## 5.4 Default values

Devices shall use these values before anything else has been negotiated:

| **Parameter** | **Value** |
|---|---|
| AFH_Mode | disabled |
| AFH_Reporting_Mode | disabled |

| Parameter | Value |
|---|---|
| Drift | 250 |
| Jitter | 10 |
| Max_Slots | 1 |
| Poll_Interval | 40 |

*Table 5.5: Device default values*

# Appendix A Changes to parameter names

Previous versions of this specification used different names for some of the parameters listed in Section 5.2. Table A.1 shows the previous and current names where a name has been changed; Table A.2 shows those names that have not changed.

| Previous name | Current name |
|---|---|
| access scheme | Access_Scheme |
| AFH_channel_classification | AFH_Channel_Classification |
| AFH_channel_map | AFH_Channel_Map |
| AFH_instant | AFH_Instant |
| AFH_max_interval | AFH_Max_Interval |
| AFH_min_interval | AFH_Min_Interval |
| AFH_mode | AFH_Mode |
| AFH_reporting_mode | AFH_Reporting_Mode |
| air mode | Air_Mode |
| authentication response | Authentication_Rsp |
| clk_adj_clk | Clk_Adj_Clk |
| clk_adj_id | Clk_Adj_ID |
| clk_adj_instant | Clk_Adj_Instant |
| clk_adj_mode | Clk_Adj_Mode |
| clk_adj_period | Clk_Adj_Period |
| clk_adj_slots | Clk_Adj_Slots |
| clk_adj_us | Clk_Adj_Offset |
| clock offset | Clock_Offset |
| Commitment value | Commitment_Value |
| CompId | Company_Identifier |
| Confirmation value | Confirmation_Value |
| data rate | Data_Rate |
| drift | Drift |
| $D_{sniff}$ | $D_{Sniff}$ |
| encapsulated data | Encap_Data |
| encapsulated major type | Encap_Major_Type |
| encapsulated minor type | Encap_Minor_Type |

| Previous name | Current name |
|---|---|
| encapsulated payload length | Encap_Payload_Length |
| encryption mode | Encryption_Mode |
| error code | Error_Code |
| escape op code | Escape_Opcode |
| eSCO handle | eSCO_Handle |
| eSCO LT_ADDR | eSCO_LT_ADDR |
| eSCO packet type | eSCO_Packet_Type |
| extended features | Extended_Features |
| extended op code | Extended_Opcode |
| features | Features |
| features page | Features_Page |
| for future use | Reserved |
| hold instant | Hold_Instant |
| hold time | Hold_Time |
| hopping mode | Hopping_Mode |
| jitter | Jitter |
| key | Key |
| key size | Key_Size |
| key size mask | Key_Size_Mask |
| length of test data | Test_Data_Length |
| max slots | Max_Slots |
| max supported page | Max_Supported_Page |
| max_sniff_subrate | Max_Sniff_Subrate |
| min_sniff_mode_timeout | Min_Sniff_Mode_Timeout |
| name fragment | Name_Fragment |
| name length | Name_Length |
| name offset | Name_Offset |
| negotiation state | Negotiation_State |
| Nonce Value | Nonce_Value |
| Notification Type | Notification_Type |
| $N_{poll}$ | $N_{Poll}$ |
| $N_{SAM-SM}$ | $N_{SAM_SM}$ |
| OOB Authentication Data | OOB_Auth_Data |

| Previous name | Current name |
|---|---|
| op code | Opcode |
| packet length | Packet_Length |
| packet type | Packet_Type |
| packet type table | Packet_Type_Table |
| paging scheme | Paging_Scheme |
| paging scheme settings | Paging_Scheme_Settings |
| poll interval | Poll_Interval |
| poll period | Poll_Period |
| power_adjustment_request | Power_Adj_Req |
| power_adjustment_response | Power_Adj_Rsp |
| power control mode | Power_Mode |
| random number | Random_Number |
| RX frequency | RX_Frequency |
| SAM_Type0-Submap | SAM_Type0_Submap |
| SCO handle | SCO_Handle |
| SCO packet | SCO_Packet |
| slot offset | Slot_Offset |
| sniff attempt | Sniff_Attempt |
| sniff timeout | Sniff_Timeout |
| sniff_subrating_instant | Sniff_Subrating_Instant |
| SubVersNr | Subversion |
| supervision timeout | Supervision_Timeout |
| switch instant | Switch_Instant |
| test scenario | Test_Scenario |
| timing control flags | Timing_Control_Flags |
| $T_{SAM-SM}$ | $T_{SAM_SM}$ |
| $T_{sniff}$ | $T_{Sniff}$ |
| TX frequency | TX_Frequency |
| Update Mode | Update_Mode |
| VersNr | Version |

*Table A.1: Changes to parameter names*

| Parameter name |
|---|
| Authentication_Requirements |
| BD_ADDR |
| $D_{eSCO}$ |
| $D_{SAM}$ |
| $D_{SCO}$ |
| IO_Capabilities |
| $N_{BC}$ |
| SAM_Index |
| SAM_Instant |
| SAM_Submaps |
| $T_{eSCO}$ |
| $T_{SCO}$ |
| $W_{eSCO}$ |

*Table A.2: Unchanged parameter names*

# [THIS PART IS NO LONGER USED]

*Controller Error Codes are located in [Vol 1] Part F*

# [THIS PART IS NO LONGER USED]

*Host Controller Interface Functional Specification is located in [Vol 4] Part E*

BR/EDR Controller
Part F

# MESSAGE SEQUENCE CHARTS

*Examples of interactions between Host Controller Interface commands and events and Link Manager Protocol Data Units are represented in the form of message sequence charts. These charts show typical interactions and do not indicate all possible protocol behavior.*

# CONTENTS

**1 Introduction ..... 779**
1.1 Notation ..... 779
1.2 Flow of control ..... 780
1.3 Example MSC ..... 780
1.4 Forward compatibility ..... 780

**2 Services without connection request ..... 782**
2.1 Remote Name Request ..... 782
2.2 One-time inquiry ..... 784
2.3 Periodic inquiry ..... 786

**3 ACL connection establishment and detachment ..... 789**
3.1 Connection setup ..... 790

**4 Optional activities after ACL connection establishment ..... 798**
4.1 Authentication requested ..... 798
4.2 Secure Simple Pairing message sequence charts ..... 800
4.2.1 Optional OOB information collection ..... 801
4.2.2 Enable Secure Simple Pairing and Secure Connections ..... 802
4.2.3 Connection establishment ..... 803
4.2.4 L2CAP connection request for a secure service ..... 803
4.2.5 Optional OOB information transfer ..... 804
4.2.6 Start Secure Simple Pairing ..... 804
4.2.7 IO capability exchange ..... 806
4.2.8 Public key exchange ..... 806
4.2.9 Authentication ..... 807
4.2.10 Numeric Comparison ..... 808
4.2.11 Numeric Comparison failure on initiating side ..... 809
4.2.12 Numeric Comparison failure on responding side ..... 810
4.2.13 Passkey Entry ..... 810
4.2.14 Passkey Entry failure on responding side ..... 812
4.2.15 Passkey Entry failure on initiator side ..... 813
4.2.16 Out of Band ..... 814
4.2.17 OOB failure on initiator side ..... 816
4.2.18 DHKey checks ..... 816
4.2.19 Calculate link key ..... 818
4.2.20 Enable encryption ..... 819
4.2.21 L2CAP connection response ..... 819
4.2.22 LMP ping ..... 820

4.3 Link Supervision Timeout Changed event ..... 822
4.4 Set Connection Encryption ..... 823
4.5 Change connection link key ..... 825
4.6 Change connection link key with encryption pause and resume ..... 825
4.7 Temporary Link Key ..... 827
4.8 Read remote supported features ..... 828
4.9 Read remote extended features ..... 829
4.10 Read clock offset ..... 830
4.11 Role switch on an encrypted link using encryption pause and resume ..... 831
4.12 Refreshing encryption keys ..... 833
4.13 Read remote version information ..... 834
4.14 QoS setup ..... 835
4.15 Switch role ..... 836
4.16 [This section is no longer used] ..... 838
4.17 [This section is no longer used] ..... 838
4.18 Slot availability mask ..... 838
4.19 LMP transaction collision ..... 842

**5 Synchronous connection establishment and detachment ..... 843**
5.1 Synchronous connection setup ..... 843
5.2 Synchronous connection setup with enhanced synchronous commands ..... 850

**6 Sniff and Hold modes ..... 857**
6.1 Sniff mode ..... 857
6.2 Hold mode ..... 858
6.3 [This section is no longer used] ..... 860

**7 Buffer management, flow control ..... 861**

**8 Loopback mode ..... 863**
8.1 Local Loopback mode ..... 863
8.2 Remote Loopback mode ..... 865

**9 Connectionless Peripheral Broadcast services ..... 867**

# 1 INTRODUCTION

This Part shows typical interactions between Host Controller Interface (HCI) commands and events and Link Manager (LM) Protocol Data Units (PDU) on the BR/EDR Controller. It provides message sequence charts (MSCs) for a range of common Link Manager procedures and the associated Host commands.

This Part illustrates only the most useful scenarios, it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or Host interface. In all message sequence charts it is assumed that all events are not masked, so the Controller will not filter out any events.

The sequence of messages in these message sequence charts is for illustrative purposes. The messages may be sent in a different order where allowed by the Link Manager or HCI. If any of these charts differ with text in the Baseband, Link Manager, or HCI Parts, the text in those Parts overrides these charts.

## 1.1 Notation

The notation used in the message sequence charts (MSCs) consists of ovals, elongated hexagons, boxes, lines, and arrows. The vertical lines terminated on the top by a shadow box and at the bottom by solid oval indicate a protocol entity that resides in a device. MSCs describe interactions between these entities and states those entities may be in.

The following symbols represent interactions and states:

| | |
|---|---|
| **Oval** | Defines the context for the message sequence chart. |
| **Hexagon** | Indicates a condition needed to start the transactions below this hexagon. The location and width of the Hexagon indicates which entity or entities make this decision. |
| **Box** | Replaces a group of transactions. May indicate a user action, or a procedure in the Baseband. |
| **Dashed Box** | Optional group of transactions. |
| **Solid Arrow** | Represents a message, signal or transaction. Can be used to show LMP and HCI traffic.<br>Some Baseband packet traffic is also shown. These are prefixed by BB followed by either the type of packet, or an indication that there is an ACK signal in a packet. |
| **Dashed Arrow** | Represents an optional message, signal or transaction. Can be used to show LMP and HCI traffic. |

## 1.2 Flow of control

Some message sequences are split into several charts. In these charts, numbers indicate normal or required ordering and letters represent alternative paths. For example, Step 4 is after Step 3, and Step 5a could be executed instead of Step 5b.

## 1.3 Example MSC

The protocol entities represented in the example shown in Figure 1.1 illustrate the interactions of two devices named A and B. Each device includes a Host and a LM entity in this example. Other MSCs in this Part may show the interactions of more than two devices.

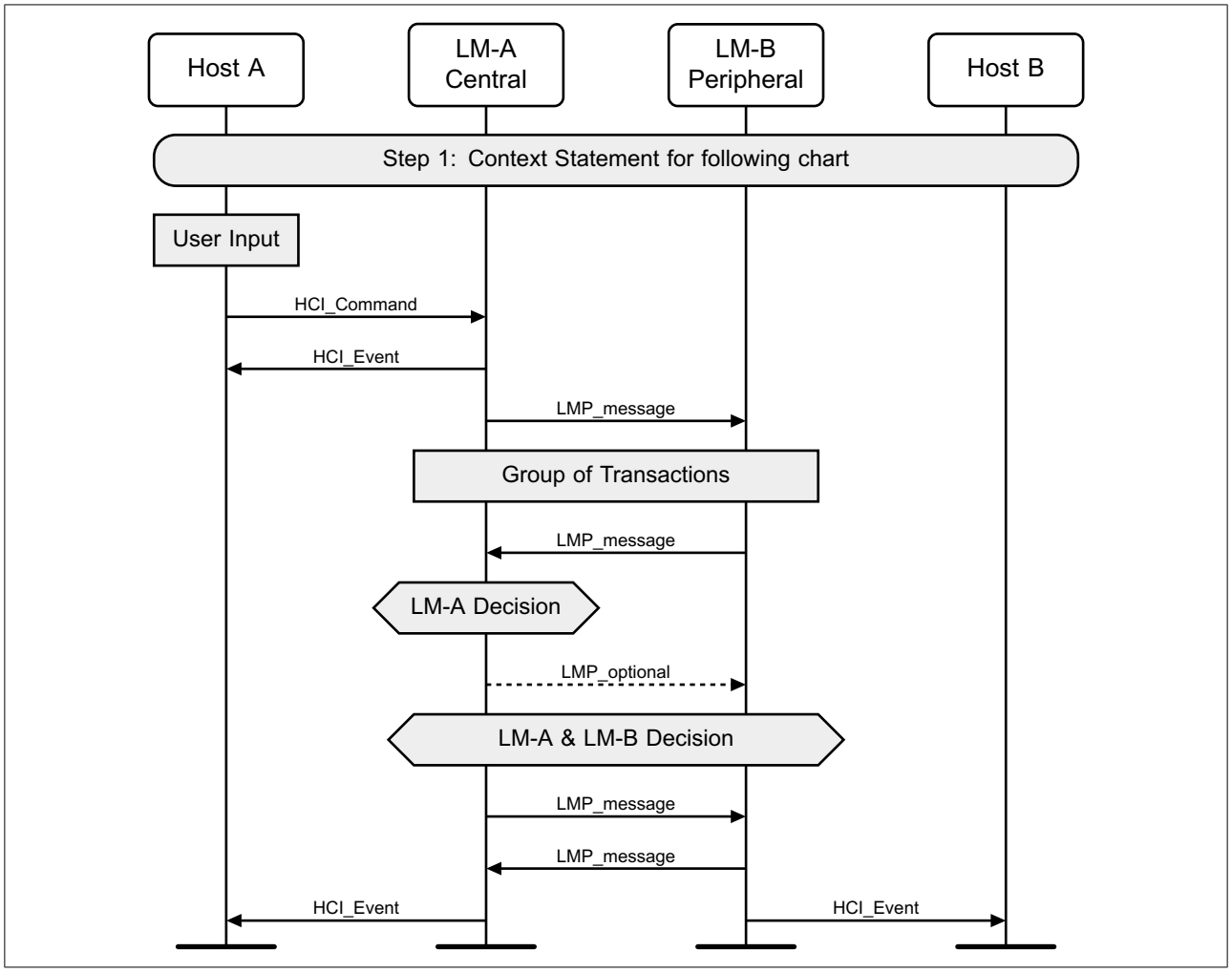



*Figure 1.1: Example MSC*

## 1.4 Forward compatibility

Many of the message sequences in this Part use HCI commands or events that have enhanced or extended variants that were added to the specification later than the

relevant sequence. Such variants can be related commands or events with different names (e.g., HCI_Flush and HCI_Enhanced_Flush commands) or commands or events with multiple versions (e.g., HCI_Encryption_Change event). In some instances (for example, see [Vol 4] Part E, Section 3.1.1), a Host is required to use the new variant rather than the one shown in the MSC. Even when this is not a requirement, Host implementers may prefer to use the newer variants.

In these circumstances, the MSCs have not been rewritten to use newer features but have been left unchanged. In general, the new commands and events will directly replace the old ones, but this is not always the case and readers should not assume it.

# 2 SERVICES WITHOUT CONNECTION REQUEST

## 2.1 Remote Name Request

The service Remote Name Request is used to find out the name of the remote device without requiring an explicit ACL connection.

**Step 1:** The Host sends an HCI_Set_Event_Mask with the bit of Remote Host Supported Features Notification event (bit 60) set and an HCI_Remote_Name_Request command expecting that its local device will automatically try to connect to the remote device. (See Figure 2.1.)

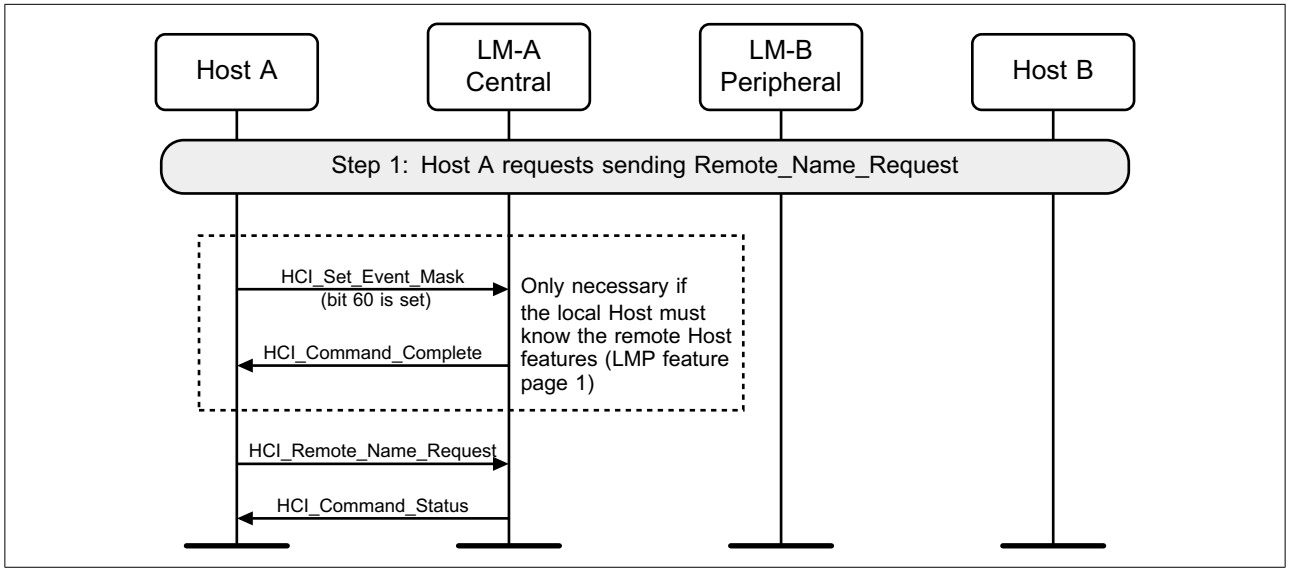



*Figure 2.1: Remote name request*

**Step 2a:** If an ACL connection does not exist device A pages device B. After the Baseband Paging procedure, the local device attempts to get the remote device's extended features, send an HCI_Remote_Host_Supported_Features_Notification event, get the remote name, disconnect, and return the name of the remote device to the Host. (See Figure 2.2.)

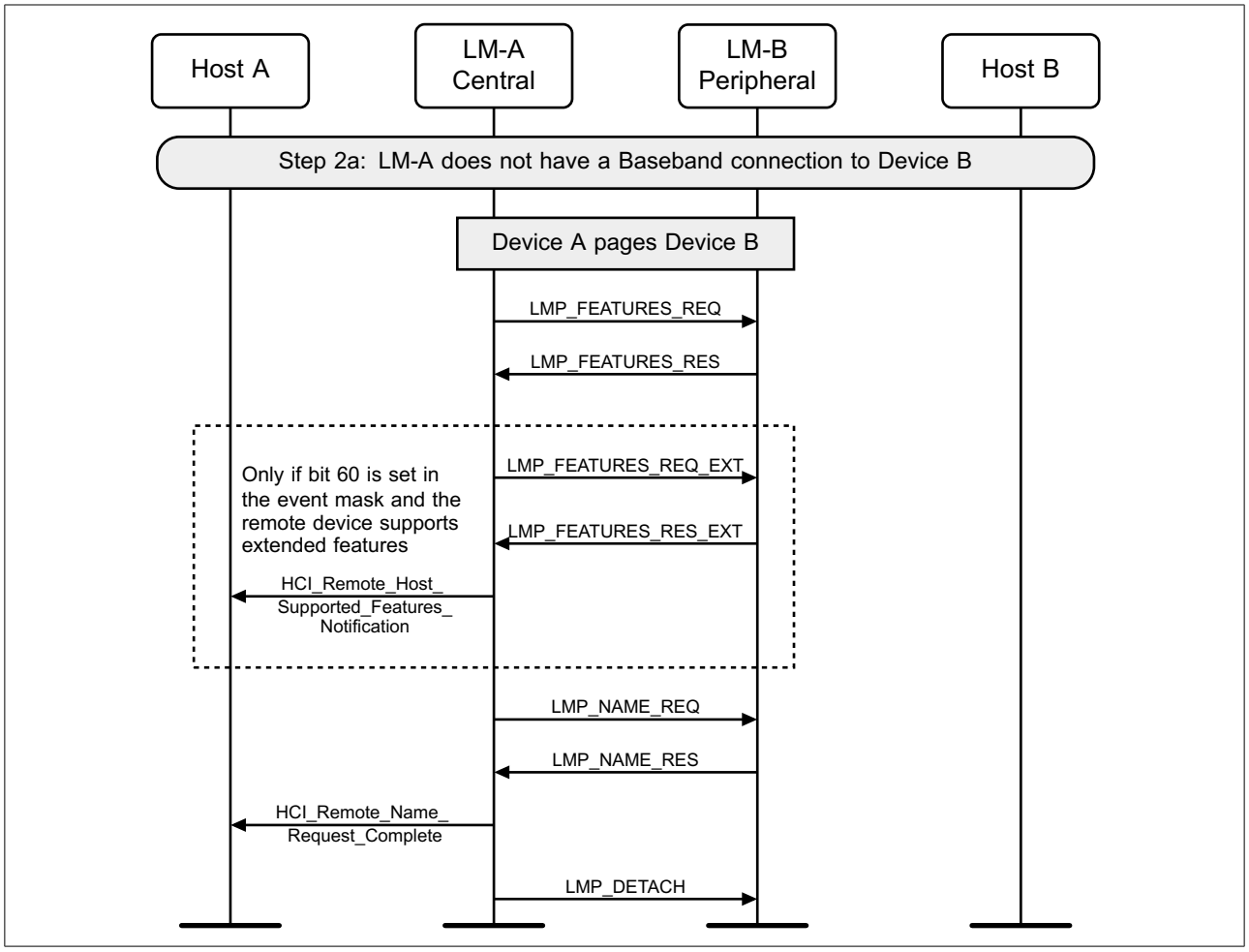



*Figure 2.2: Remote name request if no current Baseband connection*

**Step 2b:** If an ACL connection exists when the request is made, then the Remote Name Request procedure will be executed like an optional service. No Paging and no ACL disconnect is done. (See Figure 2.3.)

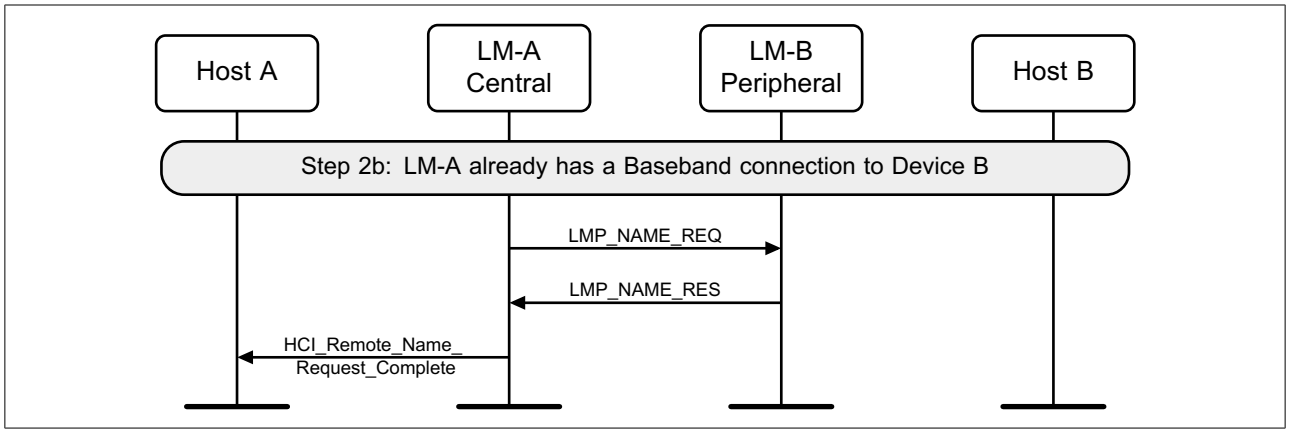



*Figure 2.3: Remote name request with Baseband connection*

## 2.2 One-time inquiry

Inquiry is used to detect and collect nearby devices.

Device A is the inquiring device and devices B and C are discoverable devices.

**Step 1:** The Host sends an HCI_Inquiry command. (See Figure 2.4.)

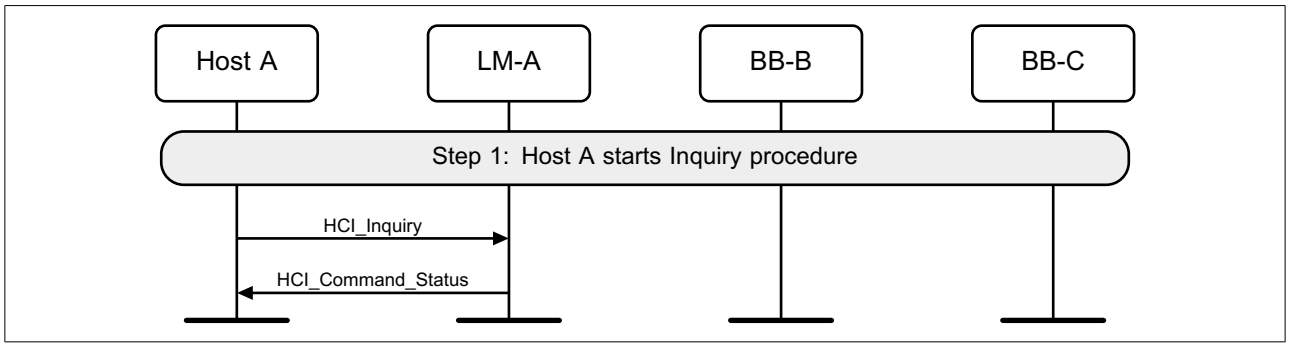



*Figure 2.4: Host A starts inquiry procedure*

**Step 2:** The Controller will start the Baseband Inquiry procedure with the specified Inquiry Access Code and Inquiry Length. When inquiry responses are received, the Controller extracts the required information and returns the information related to

the found devices using one or more HCI_Inquiry_Result events to the Host. (See Figure 2.5.)

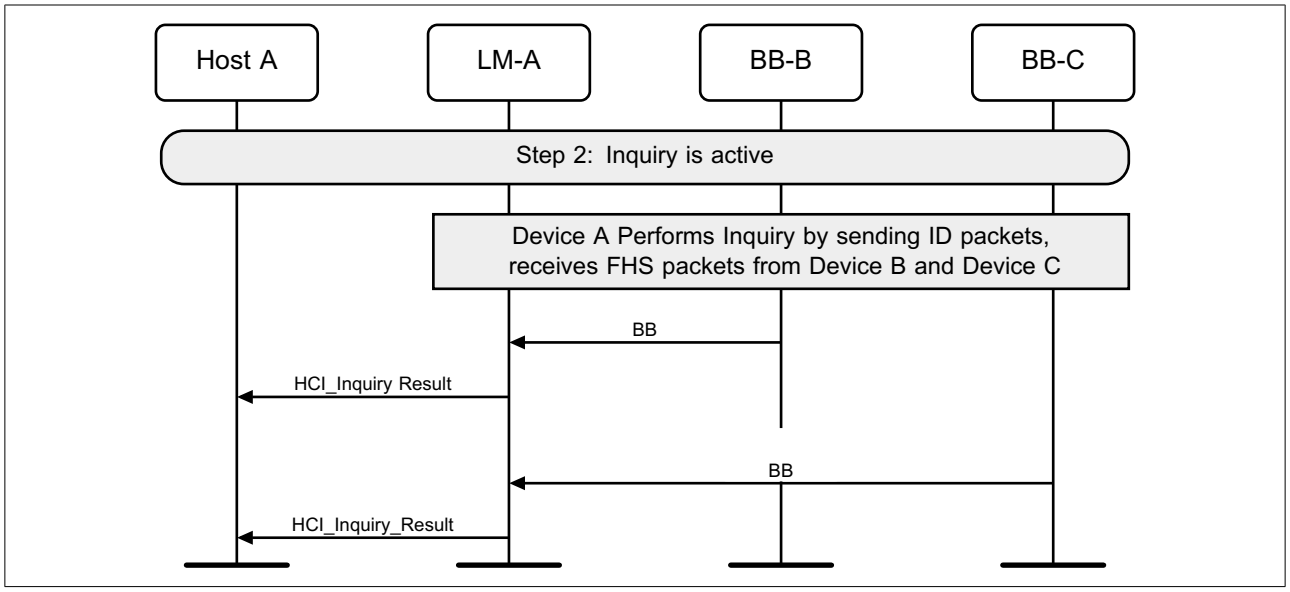



*Figure 2.5: LM-A performs inquiry and reports result*

**Step 3a:** If the Host needs to terminate an Inquiry, the HCI_Inquiry_Cancel command is used to immediately stop the inquiry procedure. (See Figure 2.6.)

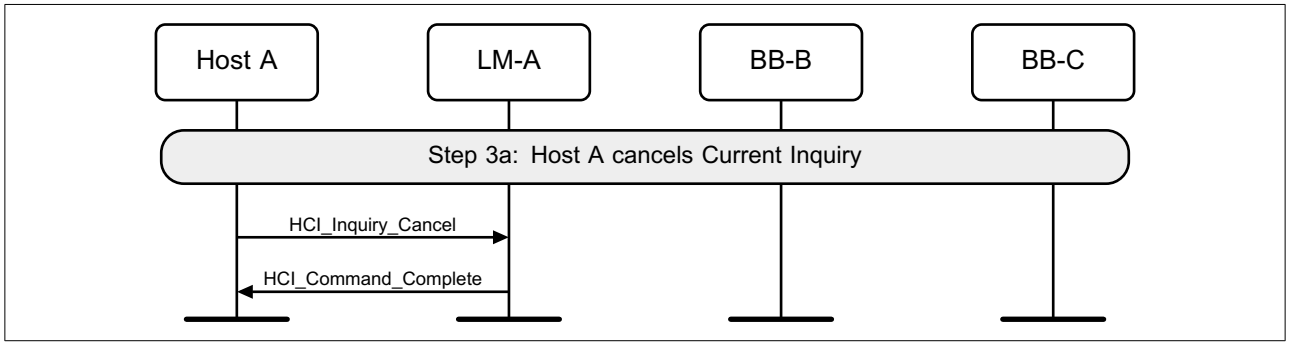



*Figure 2.6: Host A cancels inquiry*

**Step 3b:** If the Inquiry procedure is completed due to the number of results obtained, or the Inquiry Length has expired, an HCI_Inquiry_Complete event is returned to the Host. (See Figure 2.7.)

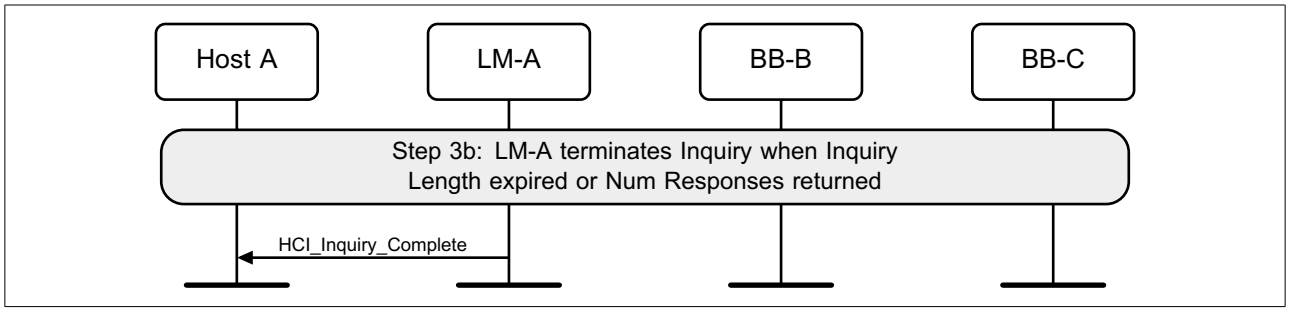



*Figure 2.7: LM-A terminates current inquiry*

## 2.3 Periodic inquiry

Periodic inquiry is used when the inquiry procedure is to be repeated periodically.

Device A is the inquiring device and devices B and C are discoverable devices.

**Step 1:** The Hosts sends an HCI_Periodic_Inquiry_Mode command. (See Figure 2.8.)

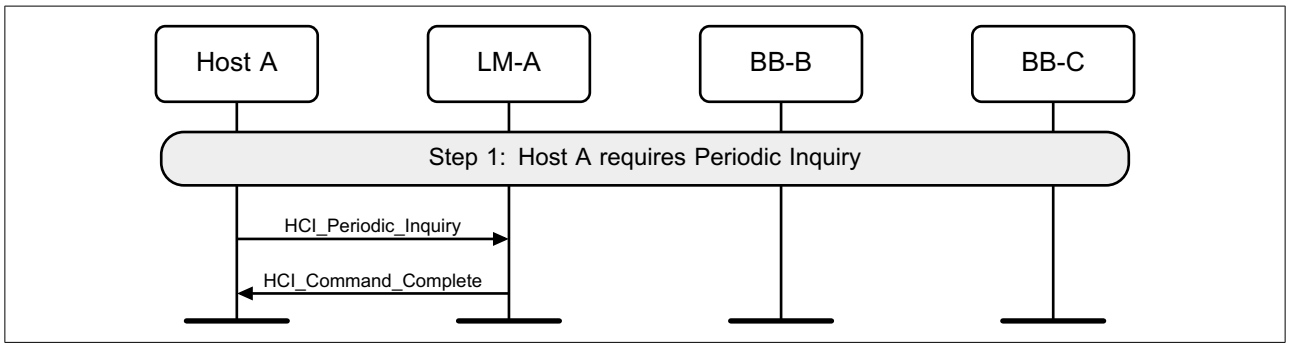



*Figure 2.8: Host A starts periodic inquiry*

**Step 2:** The Controller will start a periodic Inquiry. In the inquiry cycle, one or several HCI_Inquiry_Result events will be returned. (See Figure 2.9.)

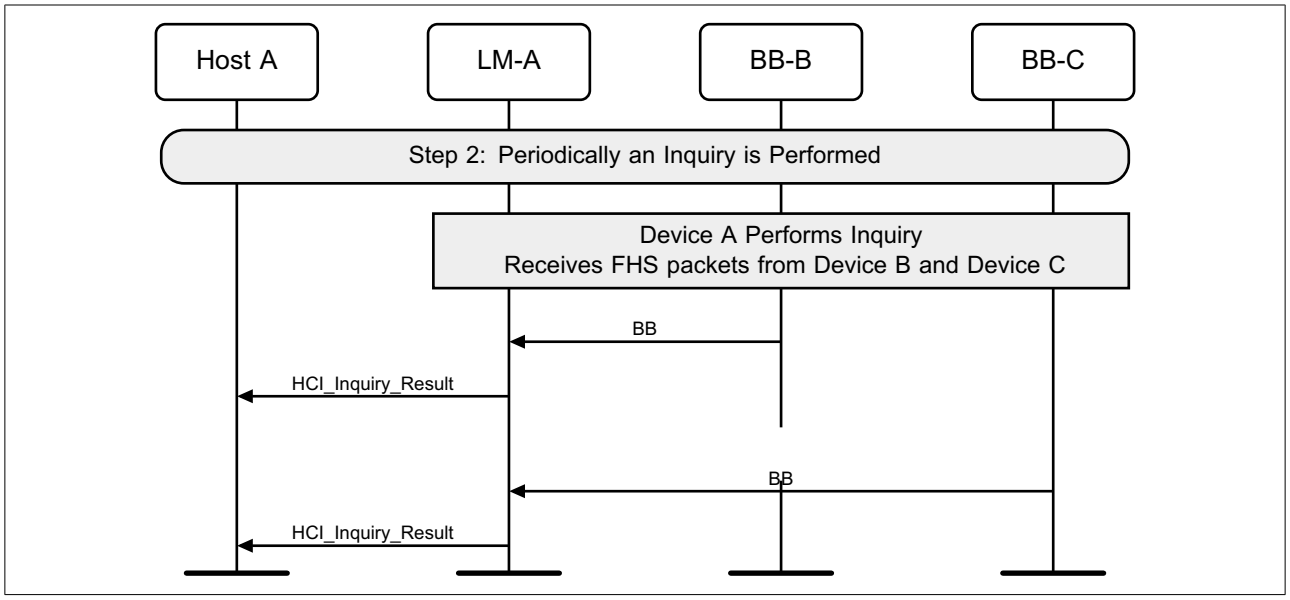



*Figure 2.9: LM-A periodically performs an inquiry and reports result*

**Step 3:** An HCI_Inquiry_Complete event will be returned to the Host when the current periodic inquiry has finished. (See Figure 2.10.)

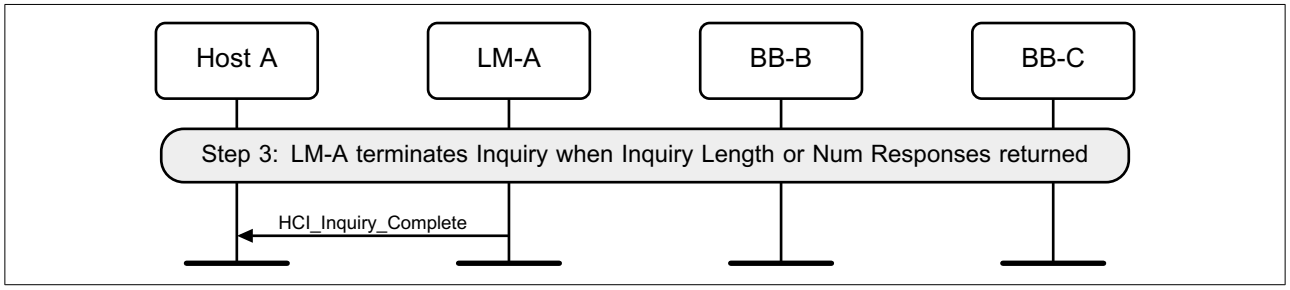



*Figure 2.10: LM-A terminates current inquiry*

**Step 4:** The periodic Inquiry can be stopped using the HCI_Exit_Periodic_Inquiry_Mode command. (See Figure 2.11.)

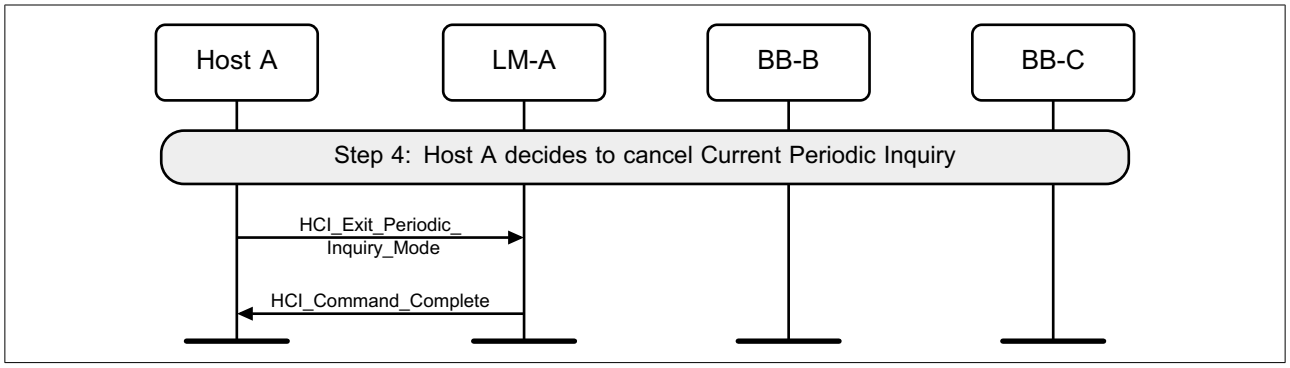



*Figure 2.11: Host A decides to exit periodic inquiry*

# 3 ACL CONNECTION ESTABLISHMENT AND DETACHMENT

A flow diagram of the establishment and detachment of a connection between two devices is shown in Figure 3.1. The process is illustrated in 9 distinct steps. A number of these steps may be performed, such as authentication and encryption. Some steps are required, such as the Connection Request and Setup Complete steps. The steps in the overview diagram directly relate to the steps in the following message sequence charts.

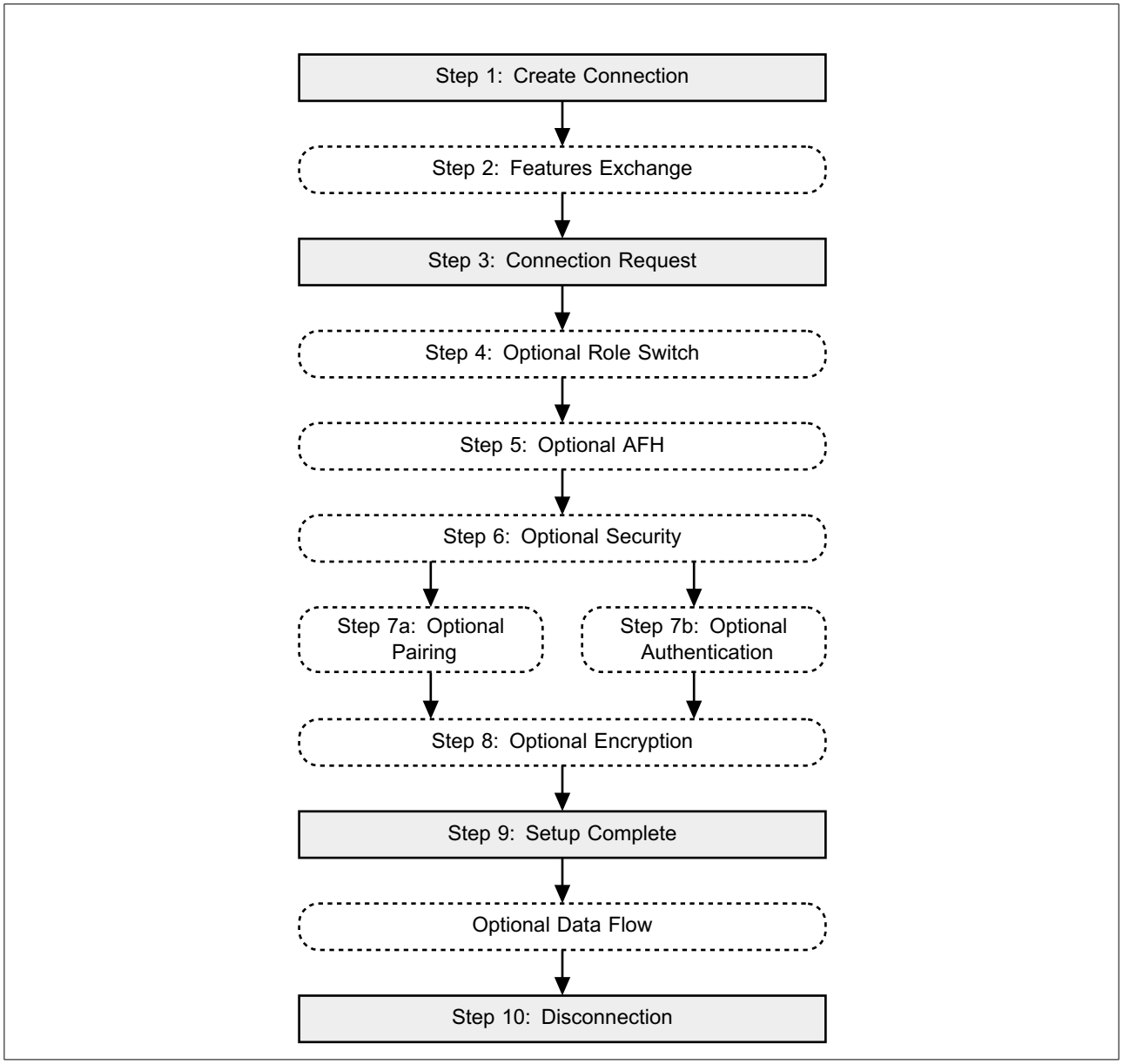



*Figure 3.1: Overview diagram for connection setup*

## 3.1 Connection setup

**Step 1:** The Host sends an HCI_Create_Connection command to the Controller. The Controller then performs a Baseband Paging procedure with the specified BD_ADDR. (See Figure 3.2.)

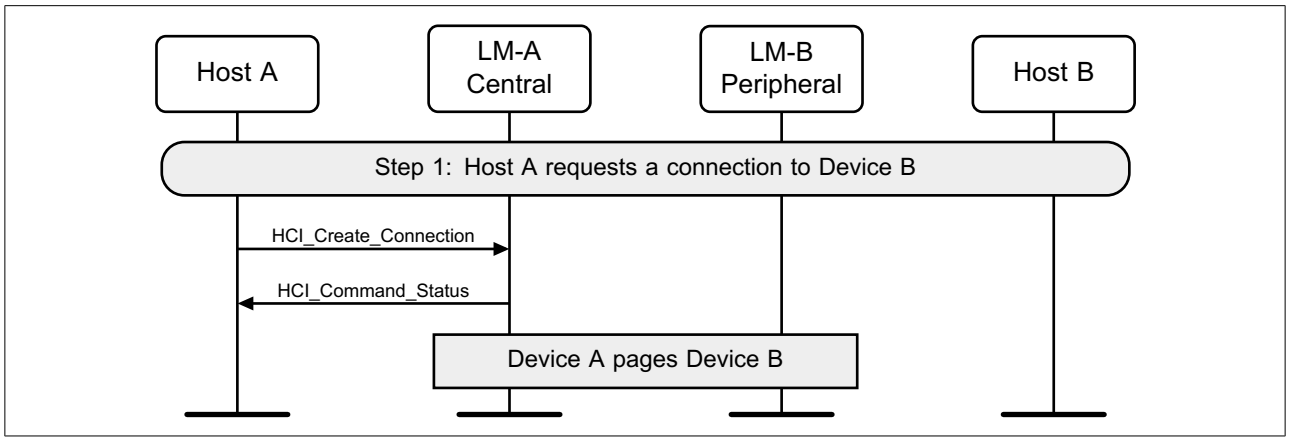



*Figure 3.2: Host A requests connection with device B*

**Step 2:** The LM may decide to exchange features.

(See Figure 3.3.)

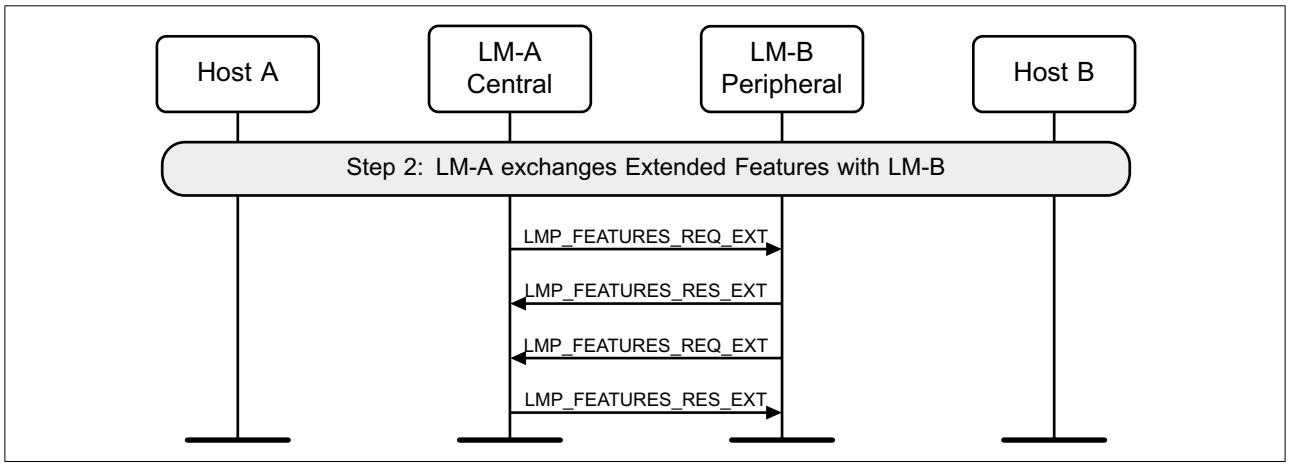



*Figure 3.3: LM-A and LM-B exchange features*

**Step 3:** The LM on the Central will request an LMP_HOST_CONNECTION_REQ PDU. The LM on the Peripheral will then confirm that a connection is OK, and if so, what role is preferred. (See Figure 3.4)

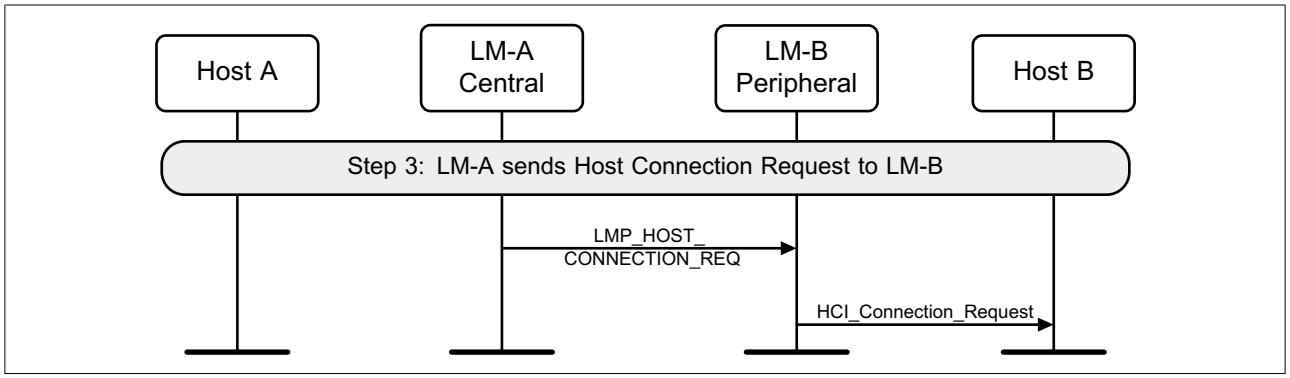



*Figure 3.4: LM-A requests Host connection*

**Step 4a:** The remote Host rejects this connection, and the link is terminated. (See Figure 3.5.)

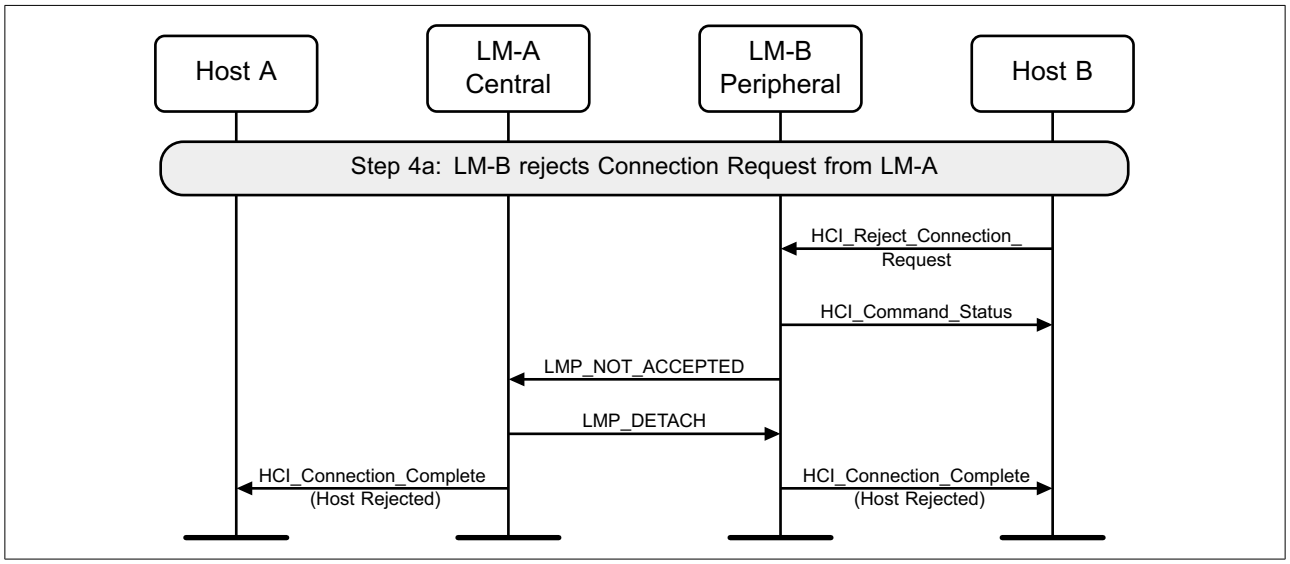



*Figure 3.5: Device B rejects connection request*

**Step 4b:** The remote Host accepts this connection. (See Figure 3.6.)

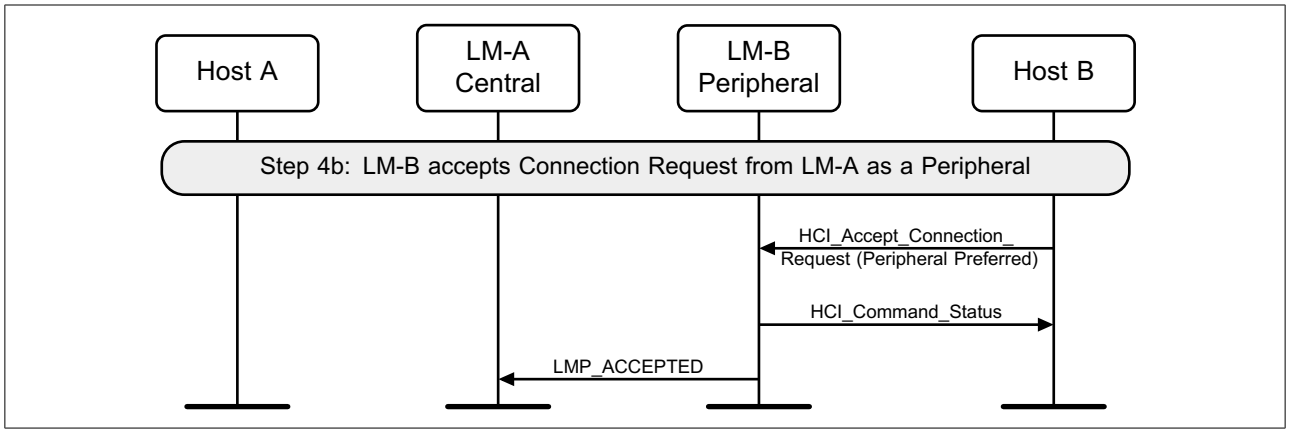



*Figure 3.6: Device B accepts connection request*

**Step 4c:** The remote Host accepts this connection but with the preference of being a Central. This will cause a role switch to occur before the LMP_ACCEPTED for the LMP_HOST_CONNECTION_REQ PDU is sent. (See Figure 3.7.)

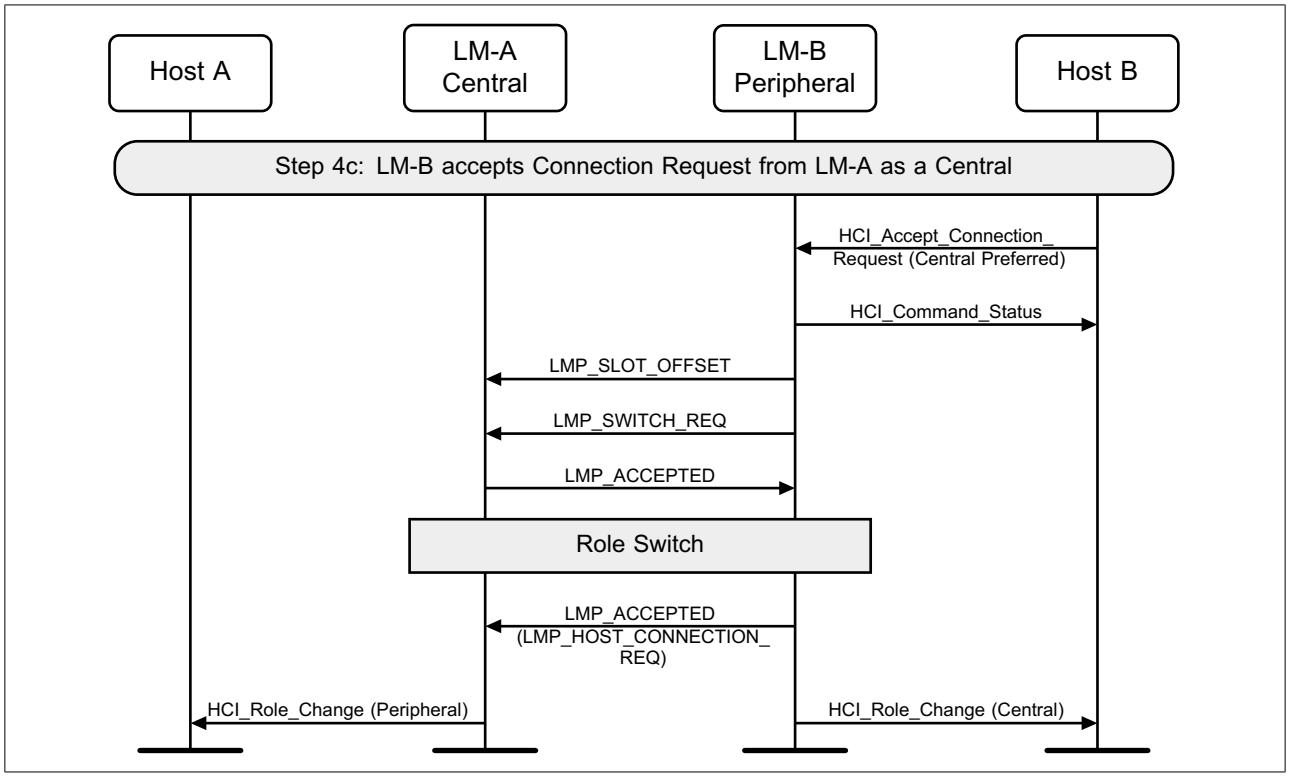



*Figure 3.7: Device B accepts connection requests as Central*

**Step 5:** After the features have been exchanged and AFH support is determined to be available, the Central may at any time send an LMP_SET_AFH and LMP_CHANNEL_CLASSIFICATION_REQ PDU. (See Figure 3.8.)

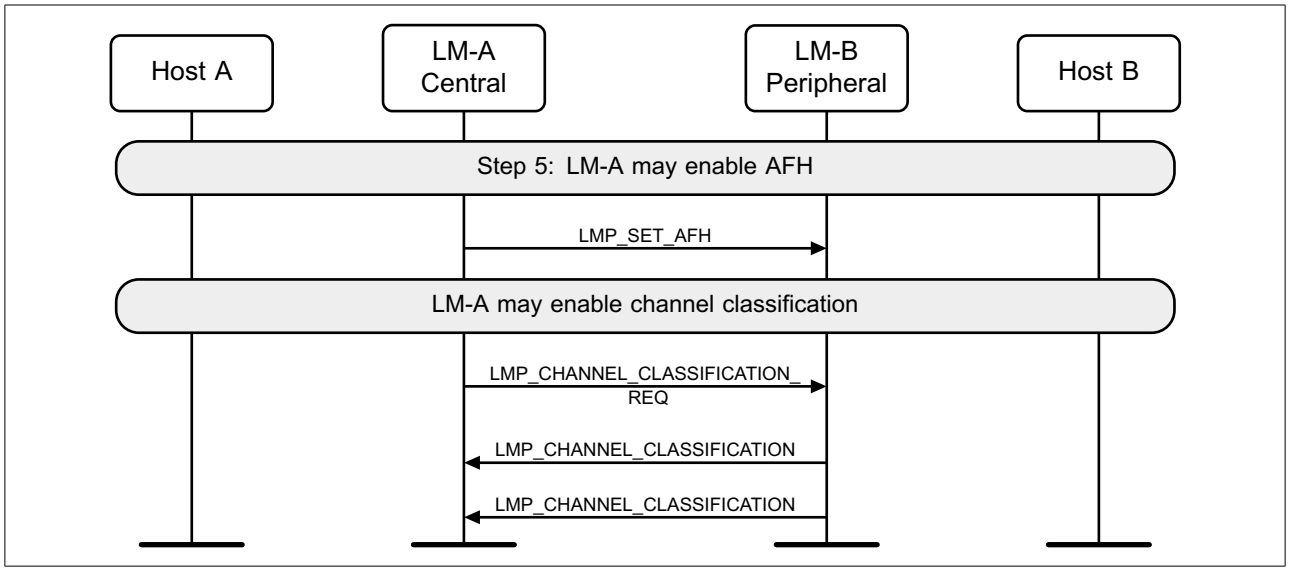



*Figure 3.8: LM-A starts Adaptive Frequency Hopping*

**Step 6:** The LM will request if authentication is required. It does this by requesting the Link Key for this connection from the Host. (See Figure 3.9.)

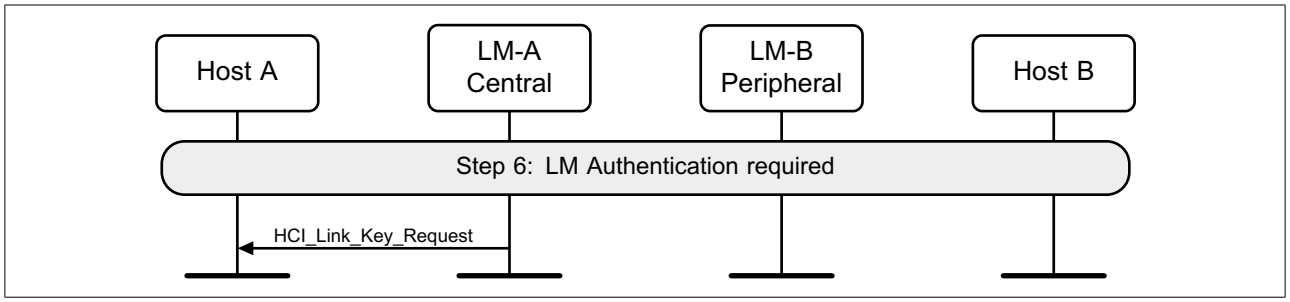



*Figure 3.9: Authentication initiated*

**Step 7a:** If authentication is required by the higher layers and the devices to be connected do not have a common link key, a pairing procedure will be used. The LM will have requested a link key from the Host for this connection. If there is a negative reply, then a PIN code will be requested. This PIN code will be requested on both sides of the connection, and authentication performed based on this PIN code. The last step is for the new link key for this connection to be passed to the Host so that it may store it for future connections. (See Figure 3.10.)

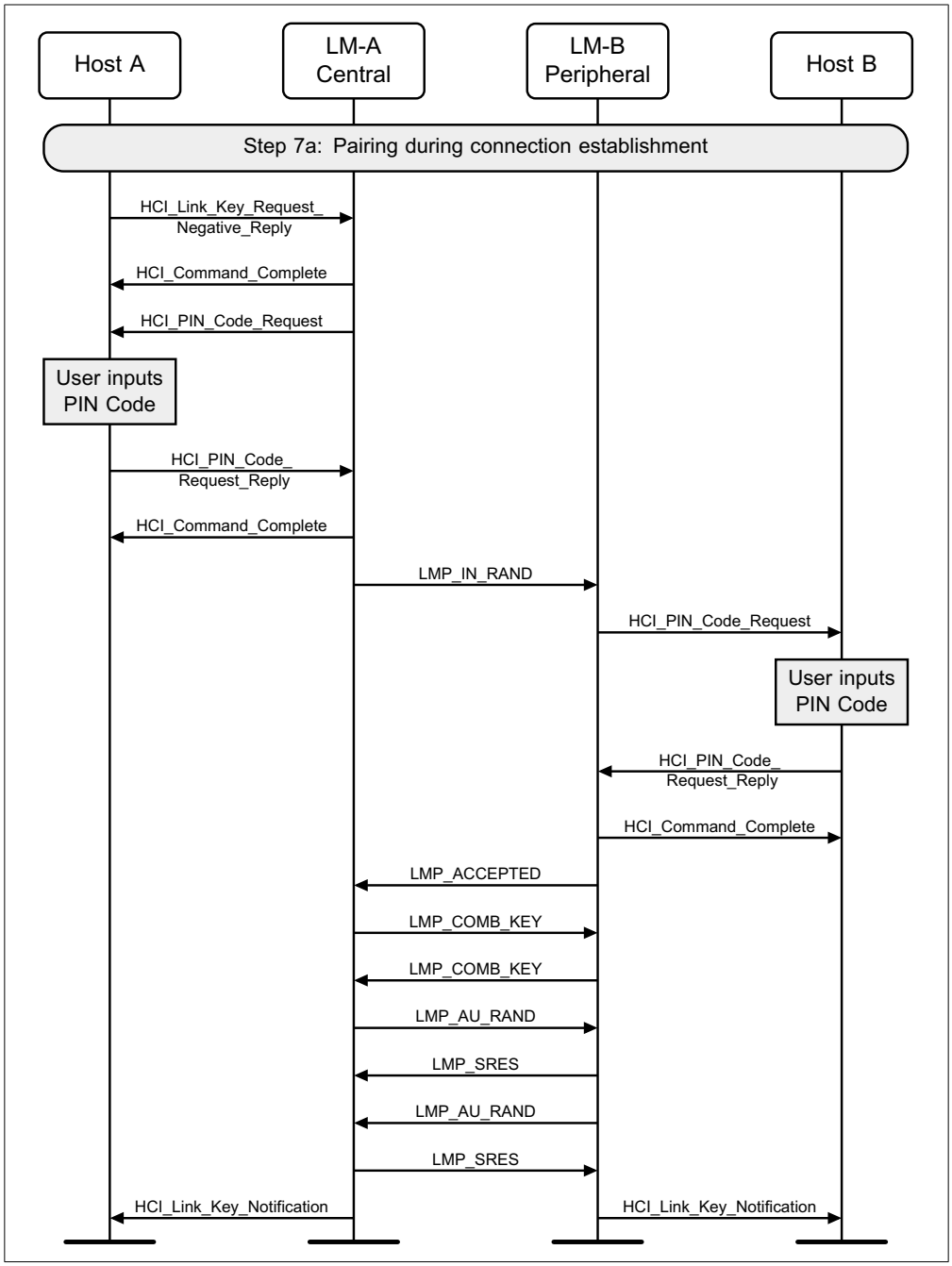



*Figure 3.10: Pairing during connection setup*

**Step 7b:** If a common link key exists between the devices, then pairing is not needed. The LM will have asked for a link key from the Host for this connection. If this is a positive reply, then the link key is used for authentication. If the configuration parameter Authentication_Enable is set, then the authentication procedure is executed. This MSC only shows the case when Authentication_Enable is set on both sides. (See Figure 3.11.)

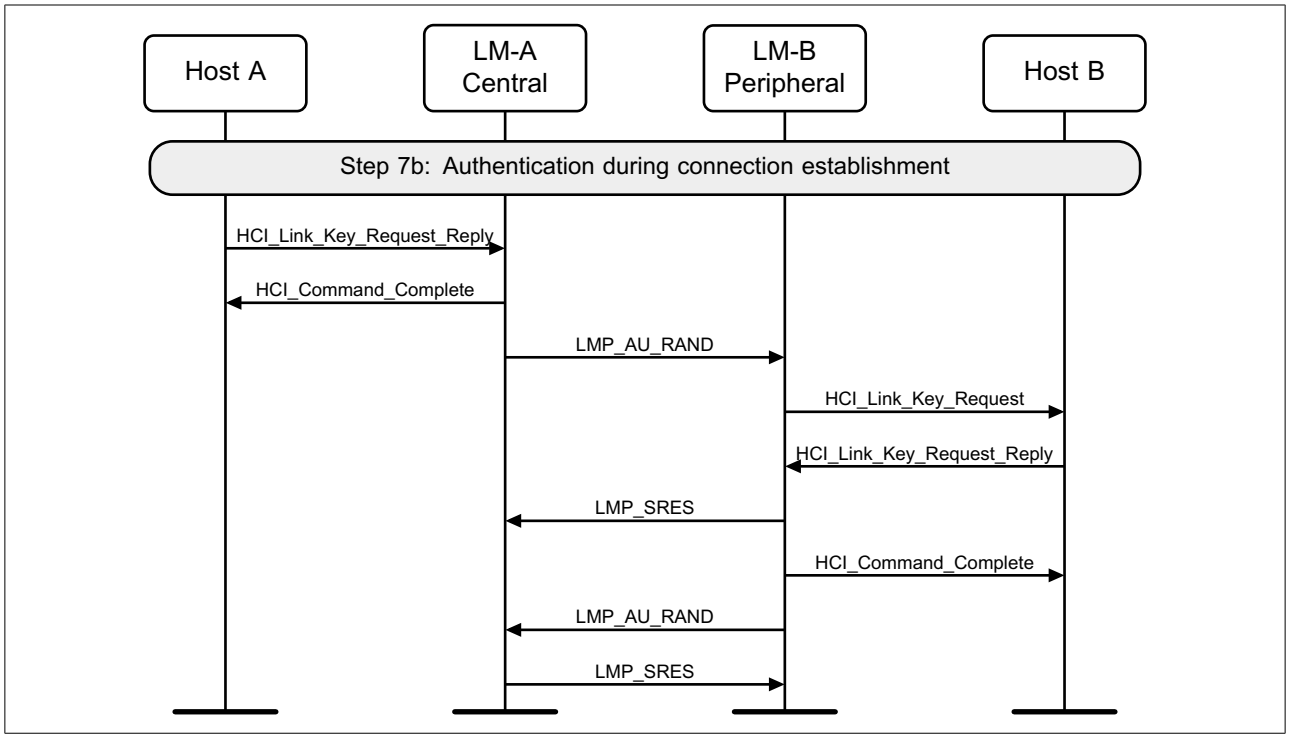



*Figure 3.11: Authentication during connection setup*

**Step 8:** Once the pairing or authentication procedure is successful, the encryption procedure may be started. This MSC only shows the set up of an encrypted point-to-point connection. (See Figure 3.12.)

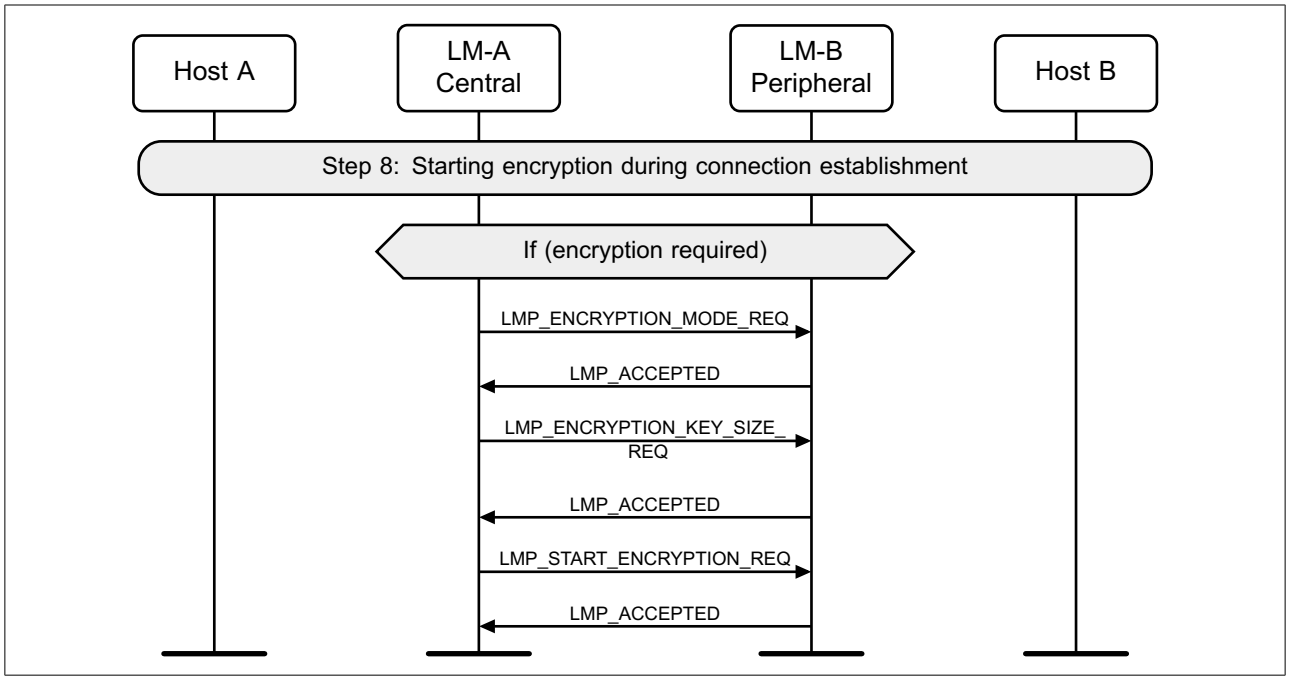



*Figure 3.12: Starting encryption during connection setup*

**Step 9:** The LMs indicate that the connection is setup by sending LMP_SETUP_COMPLETE PDU. This will cause the Host to be notified of the new Connection_Handle, and this connection may be used to send higher layer data such as L2CAP information. (See Figure 3.13.)

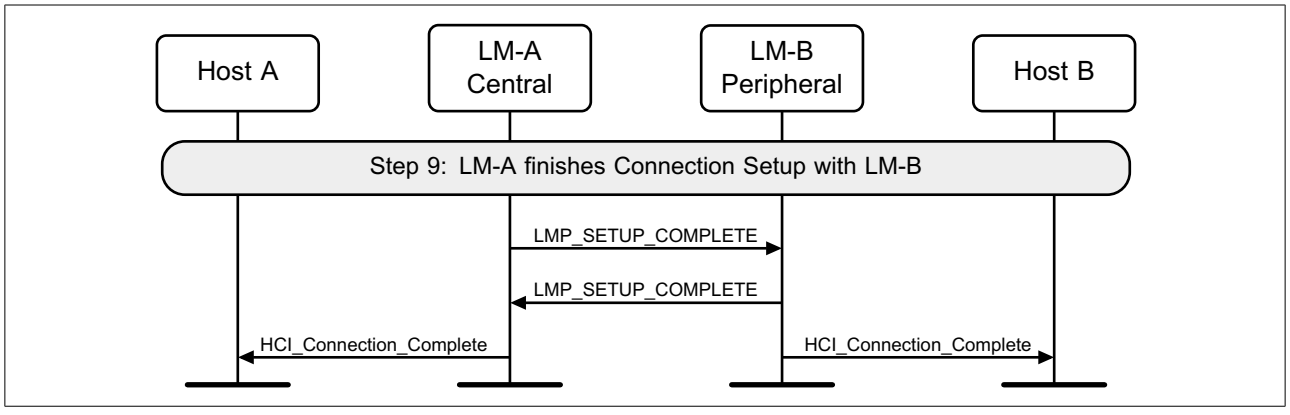



*Figure 3.13: LM-A and LM-B finishes connection setup*

**Step 10:** Once the connection is no longer needed, either device may terminate the connection using the HCI_Disconnect command and LMP_DETACH message PDU. The disconnection procedure is one-sided and does not need an explicit acknowledgment from the remote LM. The use of ARQ Acknowledgment from the Baseband indicates that the remote LM has received the LMP_DETACH PDU. (See Figure 3.14.)

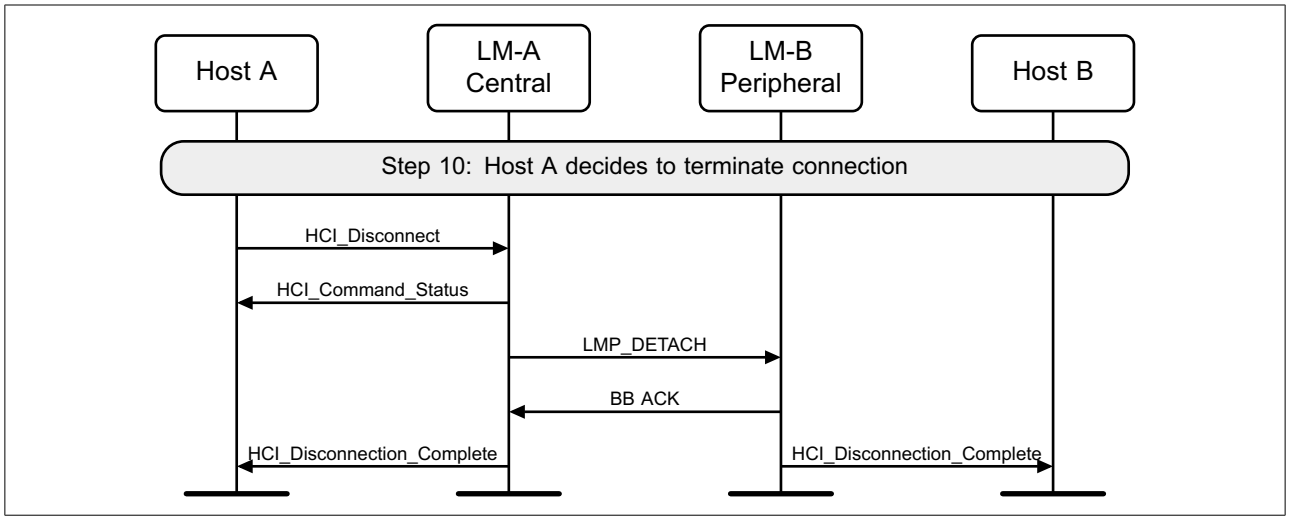



*Figure 3.14: Host A decides to disconnect*

# 4 OPTIONAL ACTIVITIES AFTER ACL CONNECTION ESTABLISHMENT

## 4.1 Authentication requested

**Step 1:** Authentication can be explicitly executed at any time after a connection has been established. If no Link Key is available then the Link Key is required from the Host. (See Figure 4.1.)

If the Controller or LM and the Host do not have the Link Key, the devices can use a procedure such as that in Section 3.1 Step 7a or that in Section 4.2.

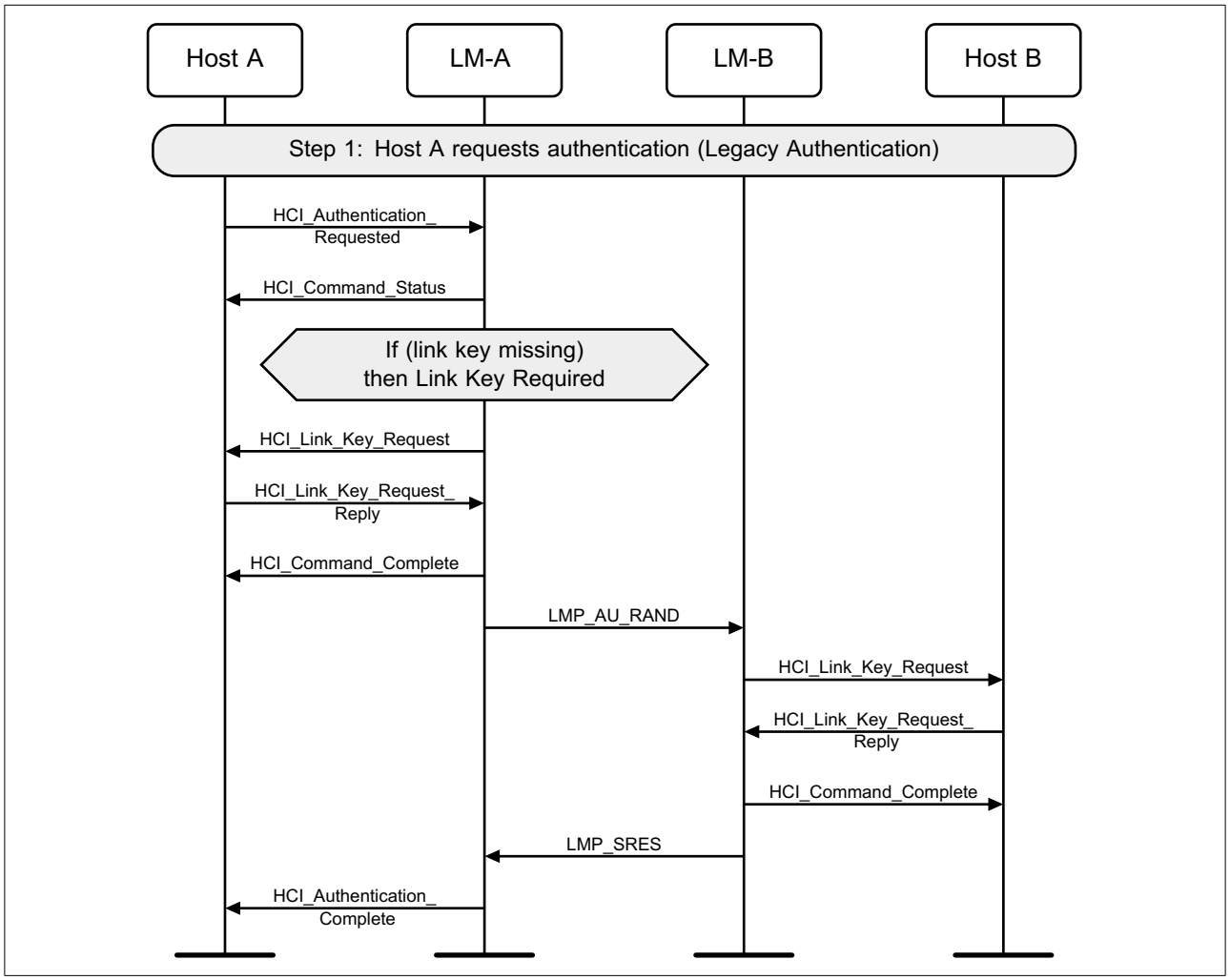



*Figure 4.1: Authentication requested (legacy authentication)*

When both devices support Secure Connections, Secure Authentication is used.

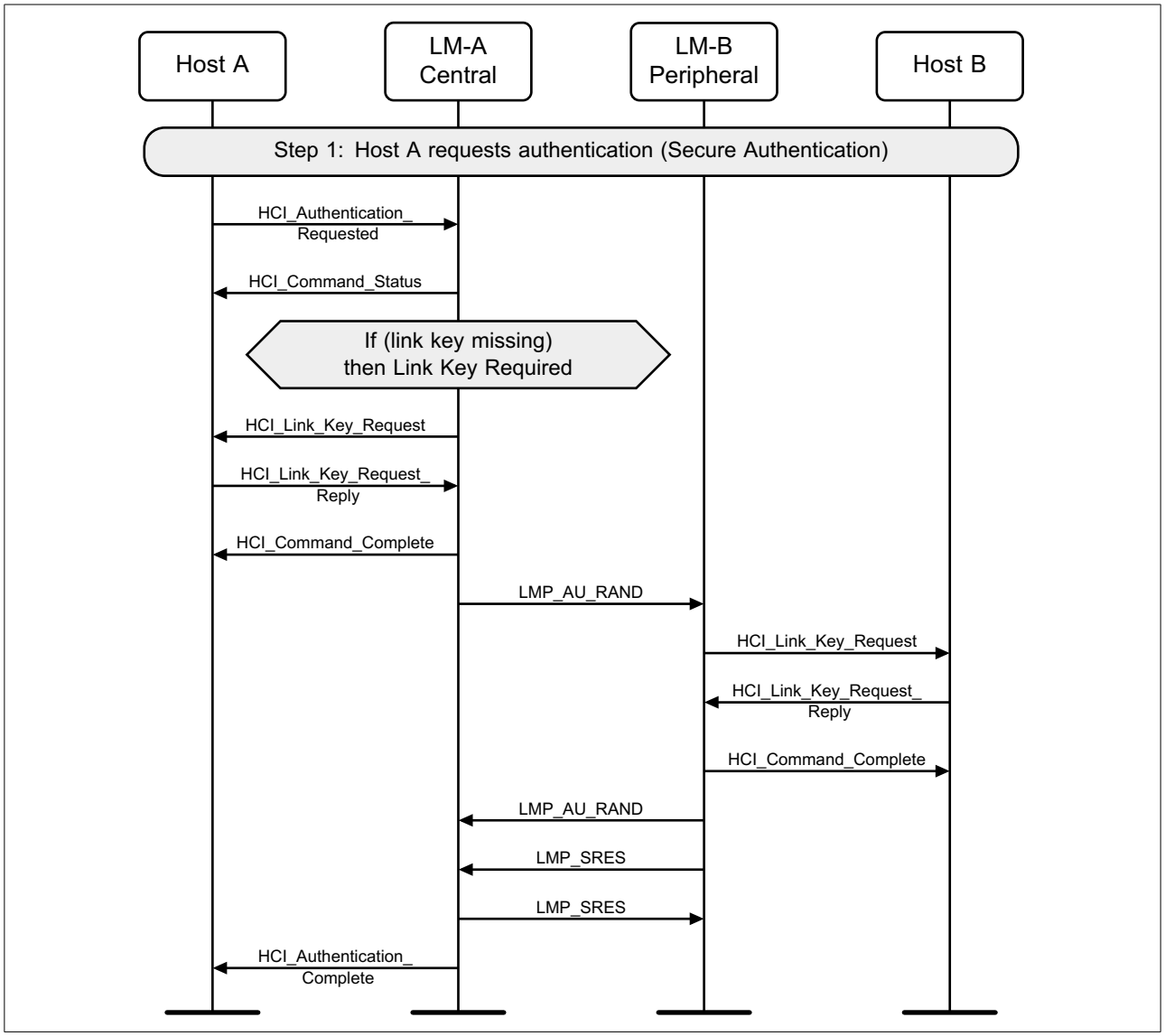



*Figure 4.2: Authentication requested (secure authentication)*

## 4.2 Secure Simple Pairing message sequence charts

A flow diagram of Secure Simple Pairing between two devices is shown in Figure 4.3. The process is illustrated in 11 distinct steps. A number of these steps have a number of different options.

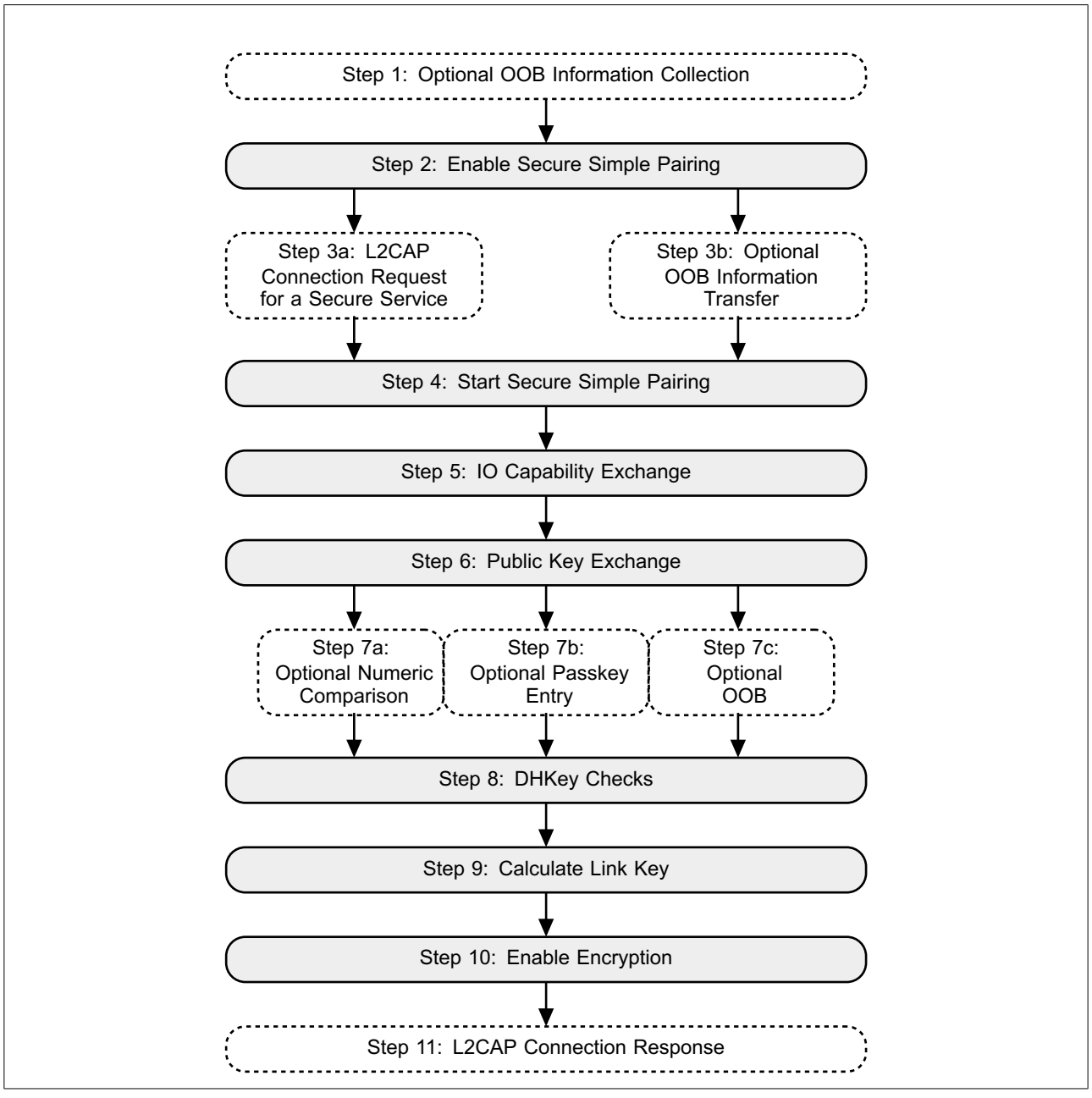



*Figure 4.3: Secure Simple Pairing - flow diagram*

### 4.2.1 Optional OOB information collection

If a device supports OOB information exchange, then the Host should request the C and R values from the Controller that need to be sent by OOB. It is then assumed that the Host transfers this information to the OOB system. This could occur a long time before, for example at the factory for a passive tag.

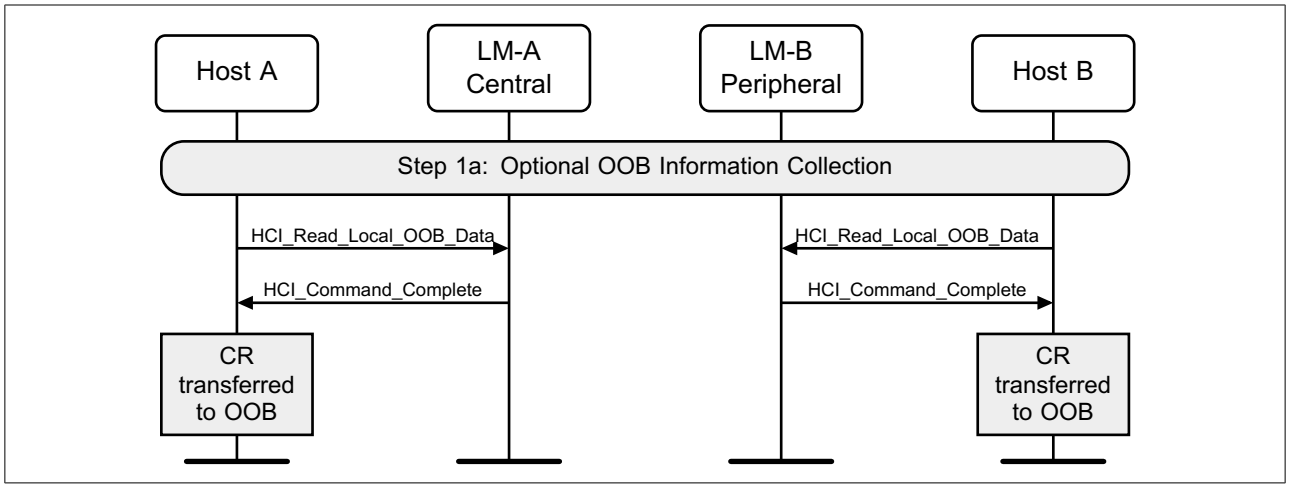



*Figure 4.4: Optional OOB information collection (P-192 only)*

When the Controller and Host support Secure Connections, the HCI_Read_Local_OOB_Extended_Data command is used instead of HCI_Read_Local_OOB_Data.

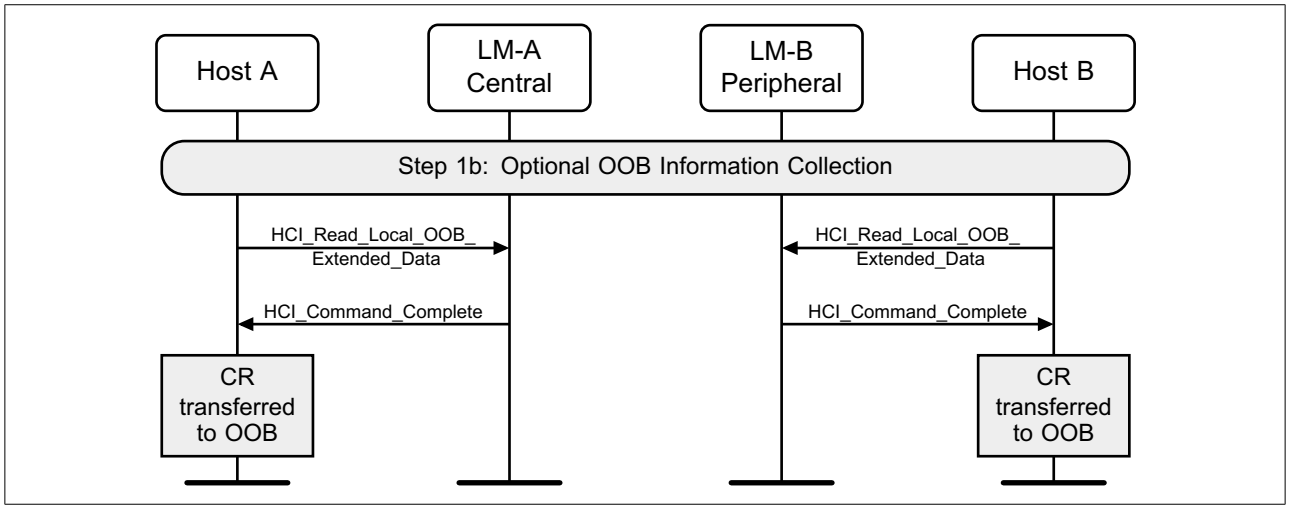



*Figure 4.5: Optional OOB information collection (P-192 and P-256)*

### 4.2.2 Enable Secure Simple Pairing and Secure Connections

To enable Secure Simple Pairing, a device must use the HCI_Write_Simple_Pairing_Mode command. This must be done before any connections that use Secure Simple Pairing are created.

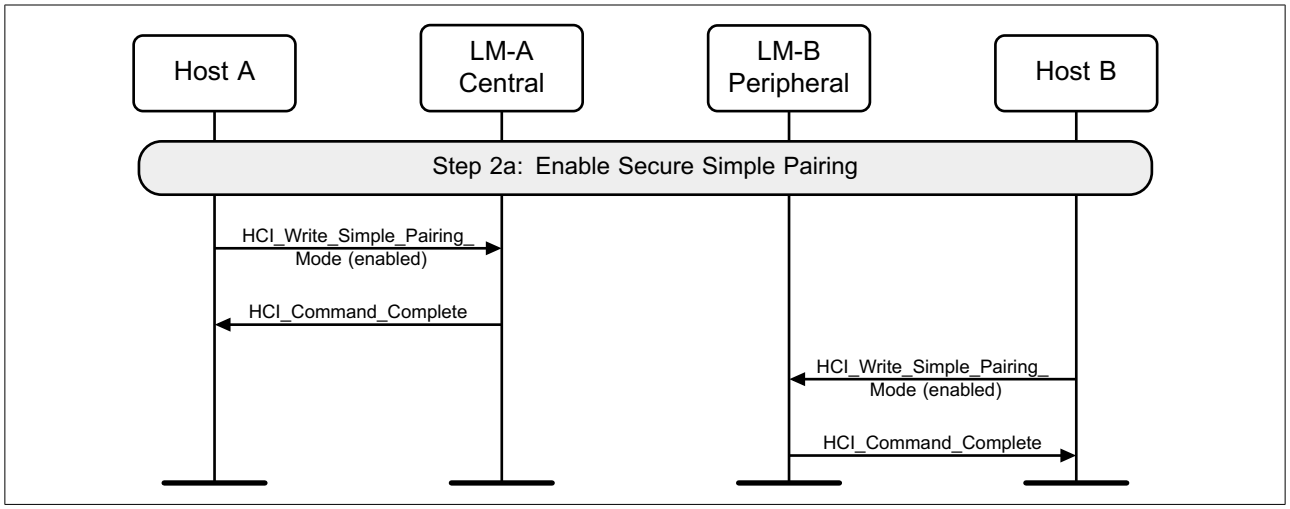



*Figure 4.6: Enable Secure Simple Pairing*

To configure the Controller to use Secure Connections HCI commands and events, a device must use the HCI_Write_Secure_Connections_Host_Support command. This must be done when no ACL connections are present.

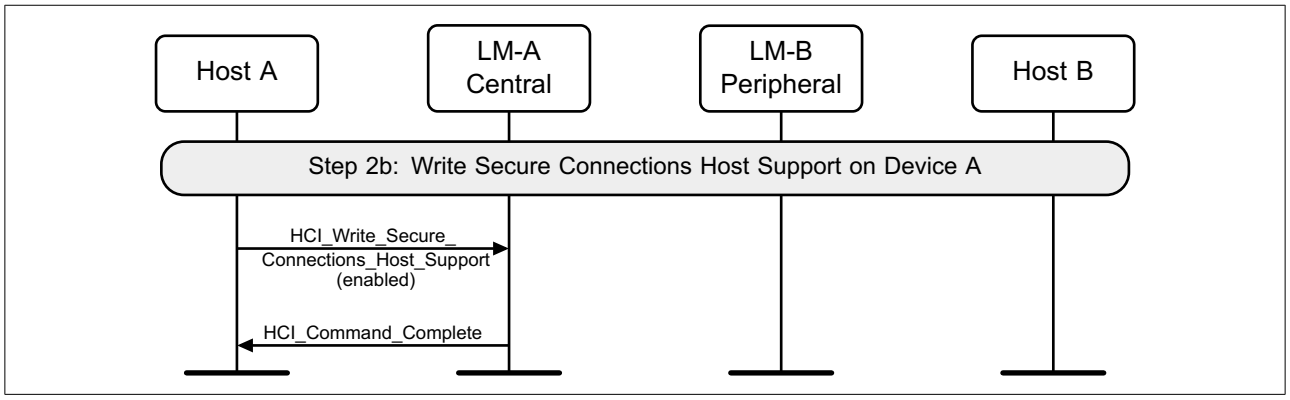



*Figure 4.7: Enable Secure Connections Host Support*

To configure the Controller to enforce a maximum interval between packets containing a MIC (when AES-CCM encryption is used), a device must use the HCI_Write_Authenticated_Payload_Timeout command.

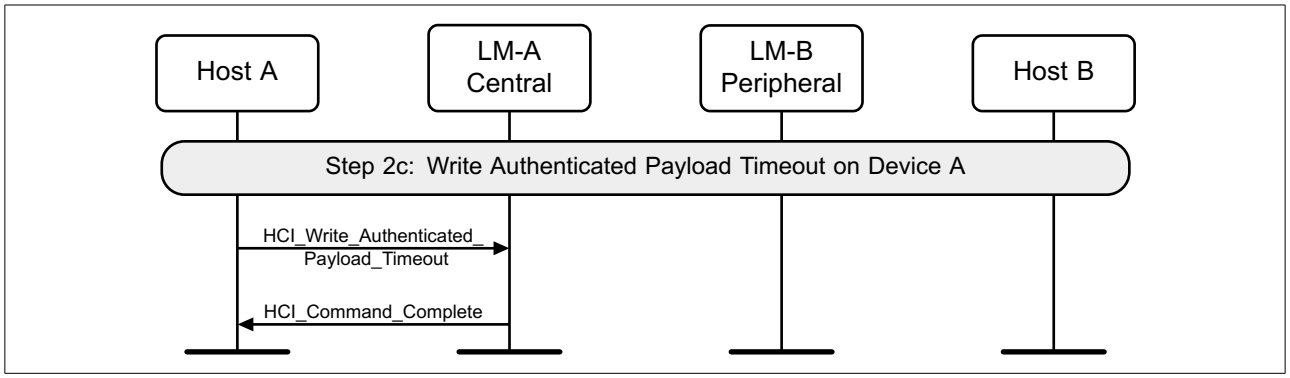



*Figure 4.8: Set Authenticated_Payload_Timeout*

### 4.2.3 Connection establishment

Secure Simple Pairing, once it is enabled, is triggered by one of two possible actions. It could be triggered by an L2CAP connection request to a service that requires security, or it could be triggered by an OOB transfer of information.

### 4.2.4 L2CAP connection request for a secure service

Once a connection has been established between two devices, if a device requests an L2CAP connection to a service that requires authentication and encryption, then the device will start Secure Simple Pairing.

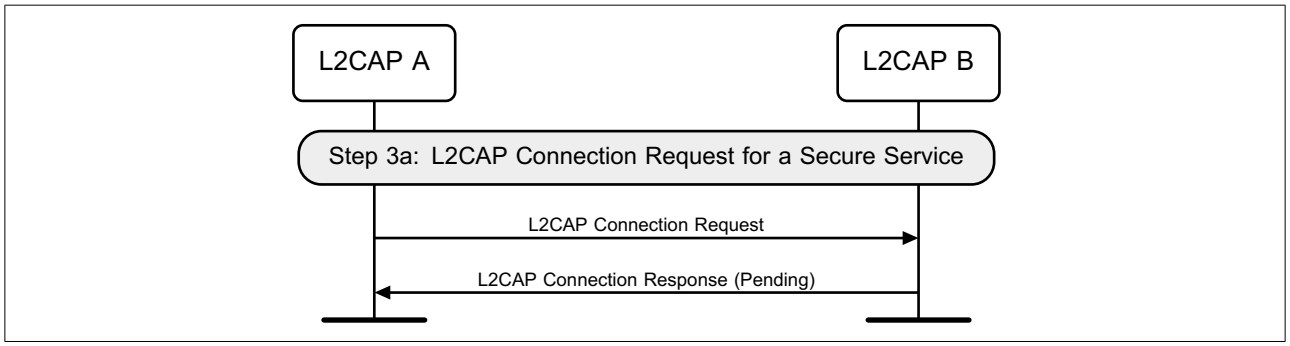



*Figure 4.9: L2CAP connection request for a secure service*

### 4.2.5 Optional OOB information transfer

Even if a Bluetooth connection has not been established between two devices, an OOB transfer can occur that transfers the Bluetooth Device Address of the device, and other OOB information for authentication. If an OOB transfer occurs, then the Host can start Secure Simple Pairing.

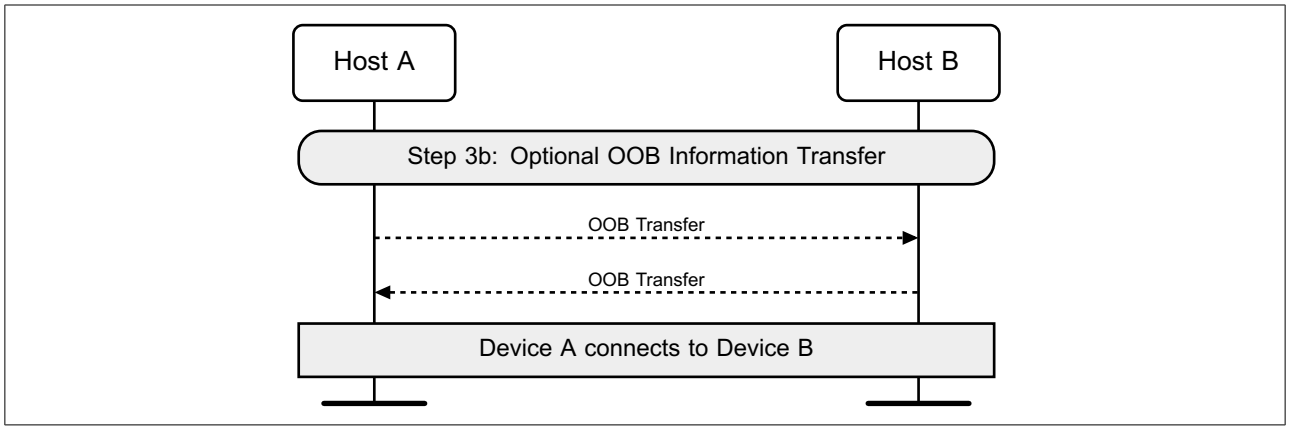



*Figure 4.10: Optional OOB information transfer*

### 4.2.6 Start Secure Simple Pairing

Once the Host has determined that Secure Simple Pairing should start, it issues an HCI_Authentication_Requested command to the Controller. This will cause the Controller to generate a request for a link key. If the Host has a link key for this connection, then pairing is not required, and the link key can be used immediately once it has been authenticated. Secure Simple Pairing will only be used if an

HCI_Link_Key_Request_Negative_Reply command is sent from the Host to the Controller on the initiating side.

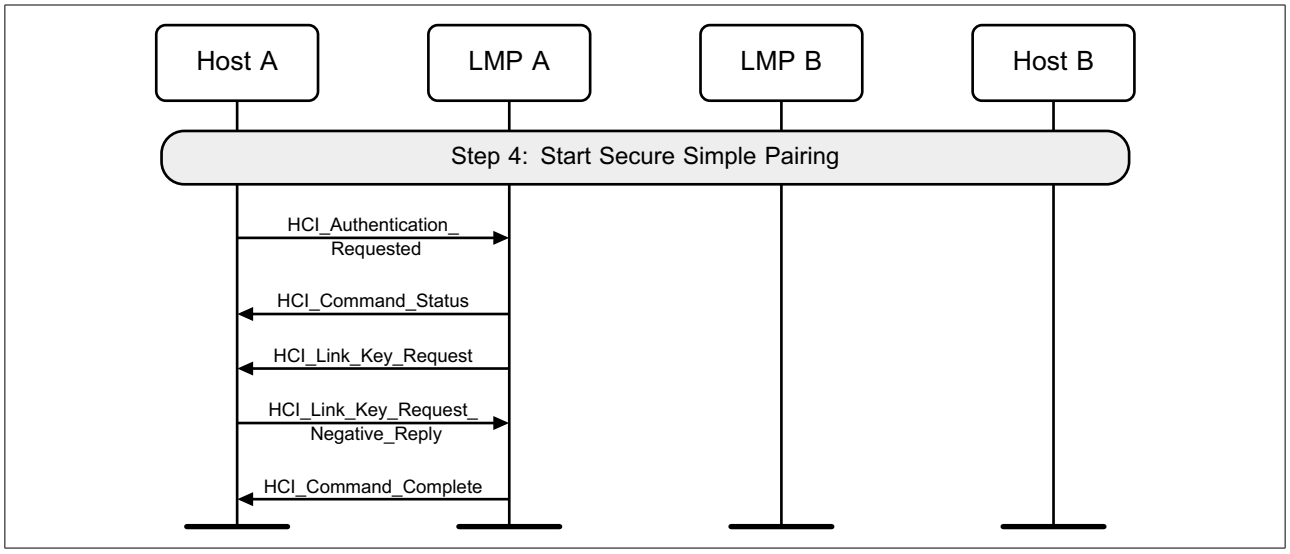



*Figure 4.11: Start Secure Simple Pairing*

### 4.2.7 IO capability exchange

To be able to determine the correct authentication algorithm to use, the input / output capabilities of the two devices are exchanged.

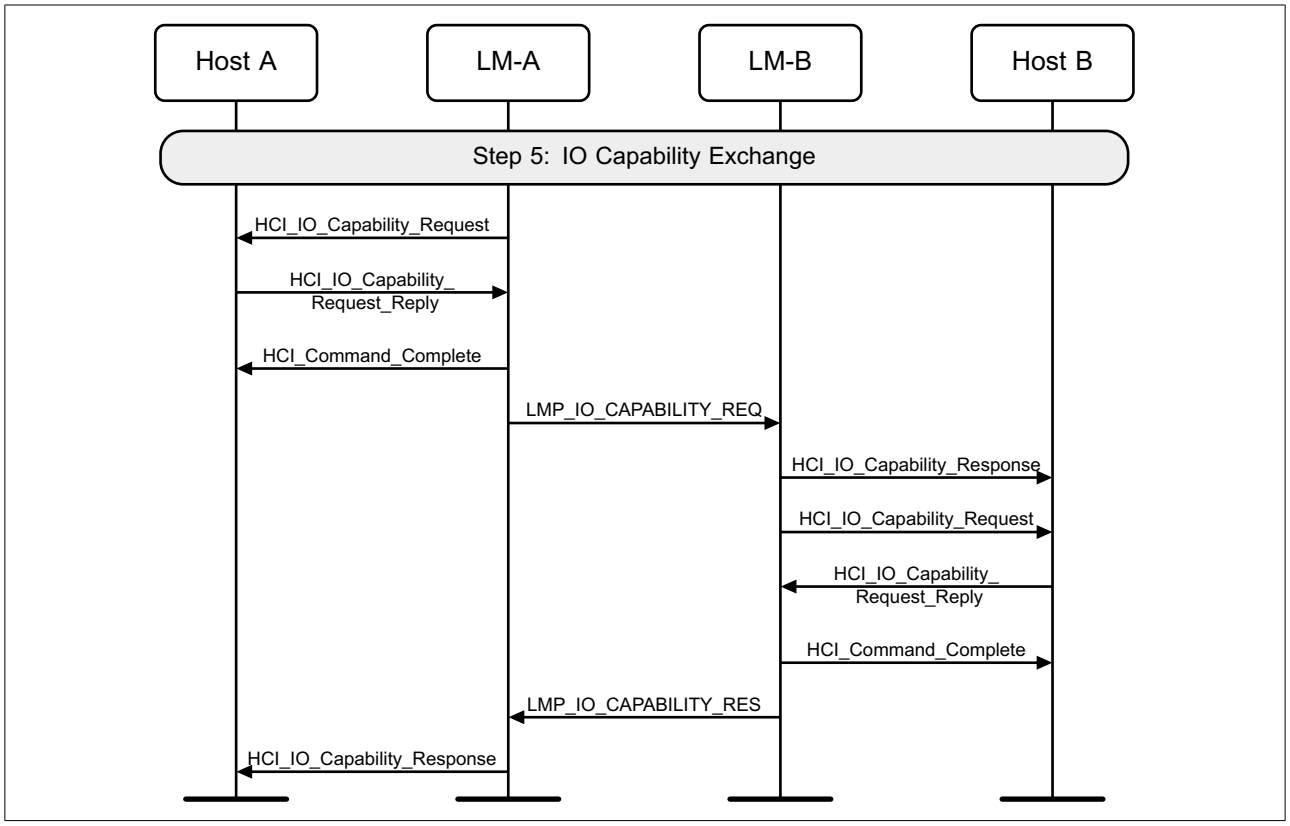



*Figure 4.12: IO capability exchange*

### 4.2.8 Public key exchange

Next the public keys are exchanged between the two devices. Once a device has received the public key of the peer device, it can start to calculate the Diffie Hellman

Key (DHKey). This may take a long time, and should be started early, so that user interaction can hide the calculation time. The DHKey is not required until step 8.

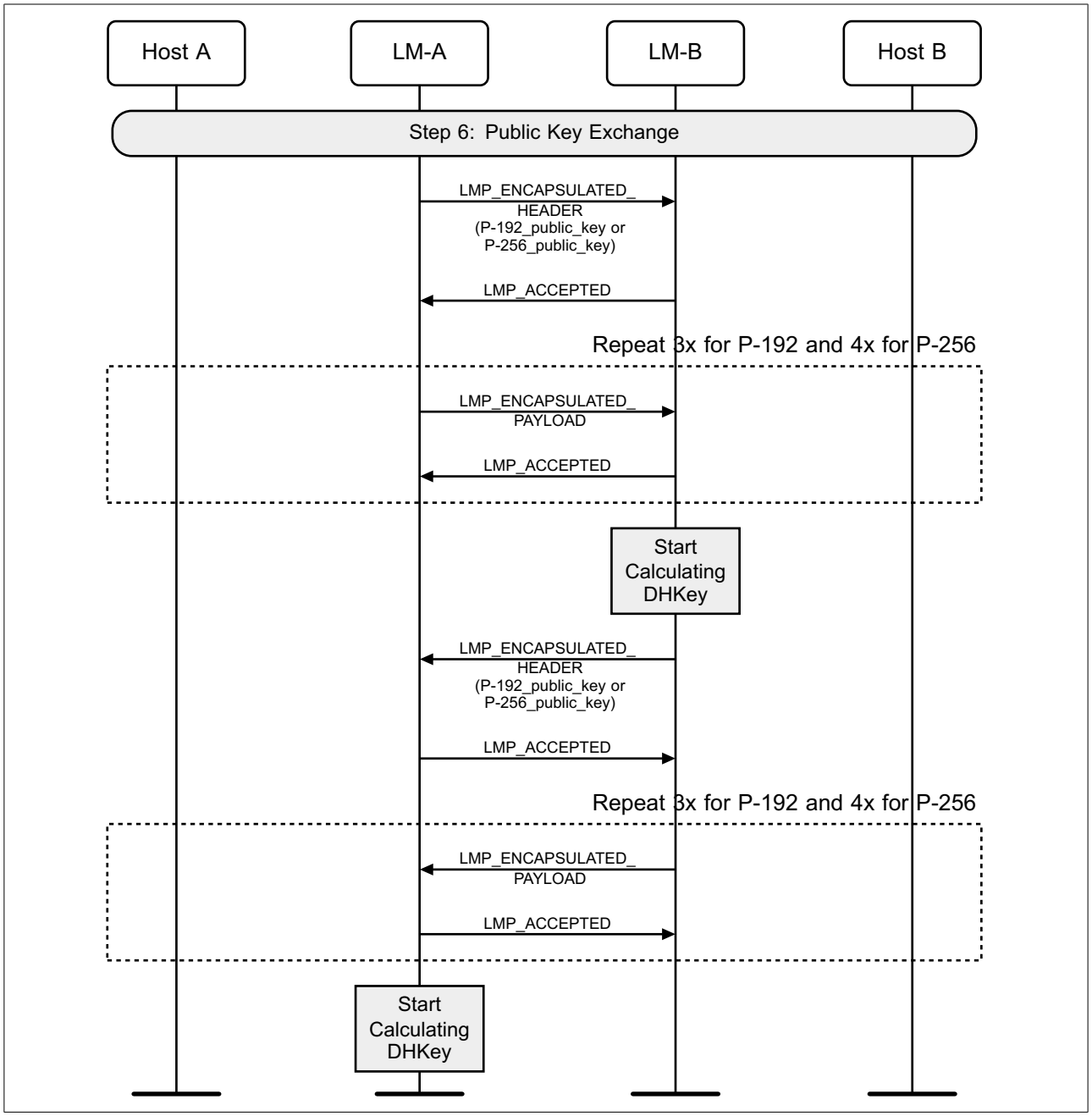



*Figure 4.13: Public key exchange*

### 4.2.9 Authentication

A device can be authenticated by using one of three algorithms. The choice of algorithm is determined by the combination of the IO capabilities of the two devices.

### 4.2.10 Numeric Comparison

The numeric comparison step will be done when both devices have output capabilities, or if one of the devices has no input or output capabilities. If both devices have output capabilities, this step requires the displaying of a user confirmation value. This value should be displayed until the end of step 8. If one or both devices do not have output capabilities, the same protocol is used but the Hosts will skip the step asking for the user confirmation. The sequence for Just Works is identical to that of Numeric Comparison with the exception that the Host will not show the numbers to the user.

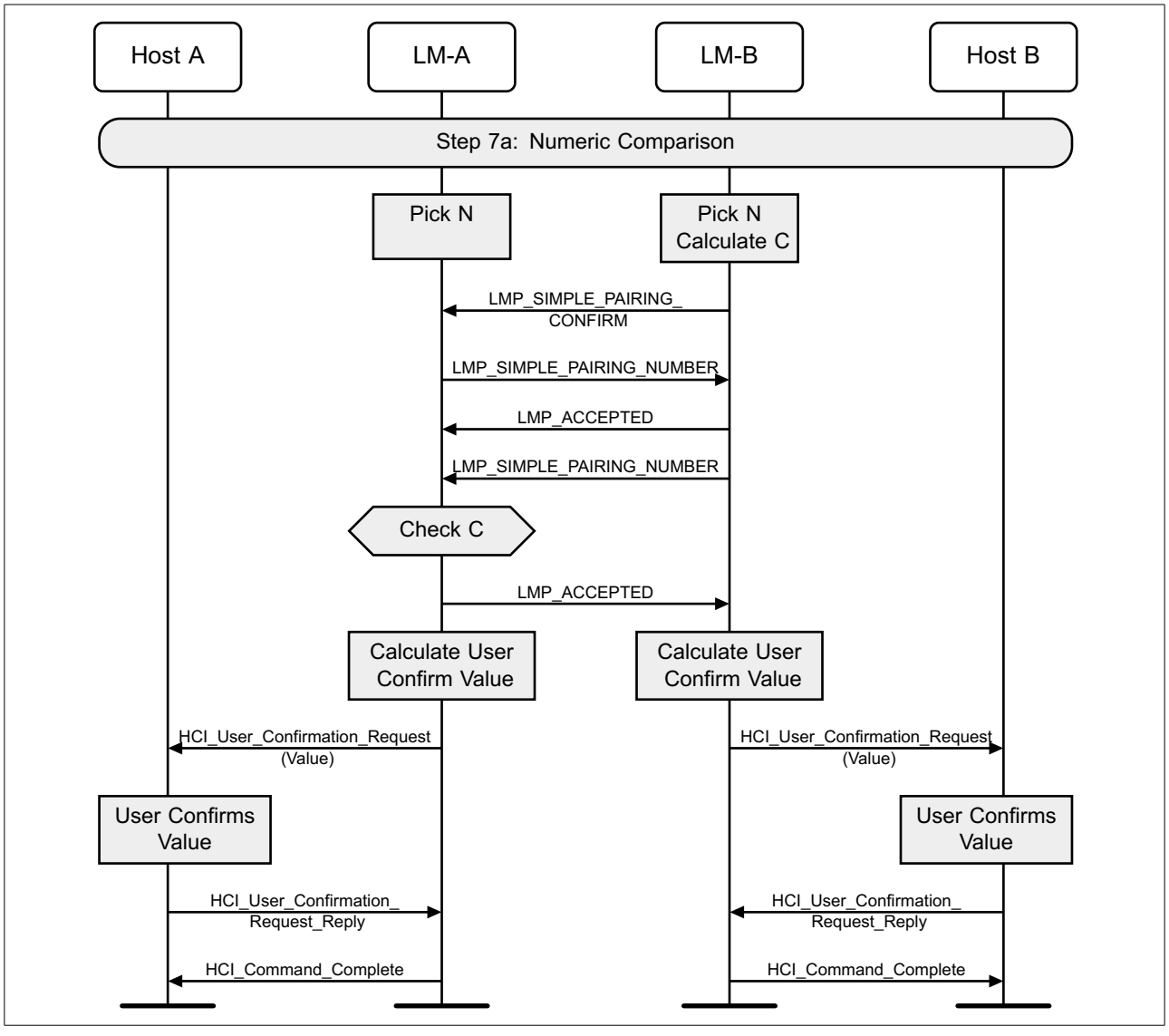



*Figure 4.14: Numeric Comparison authentication*

### 4.2.11 Numeric Comparison failure on initiating side

If the numeric comparison fails on the initiating side due to the user indicating that the confirmation values do not match, Secure Simple Pairing is terminated.

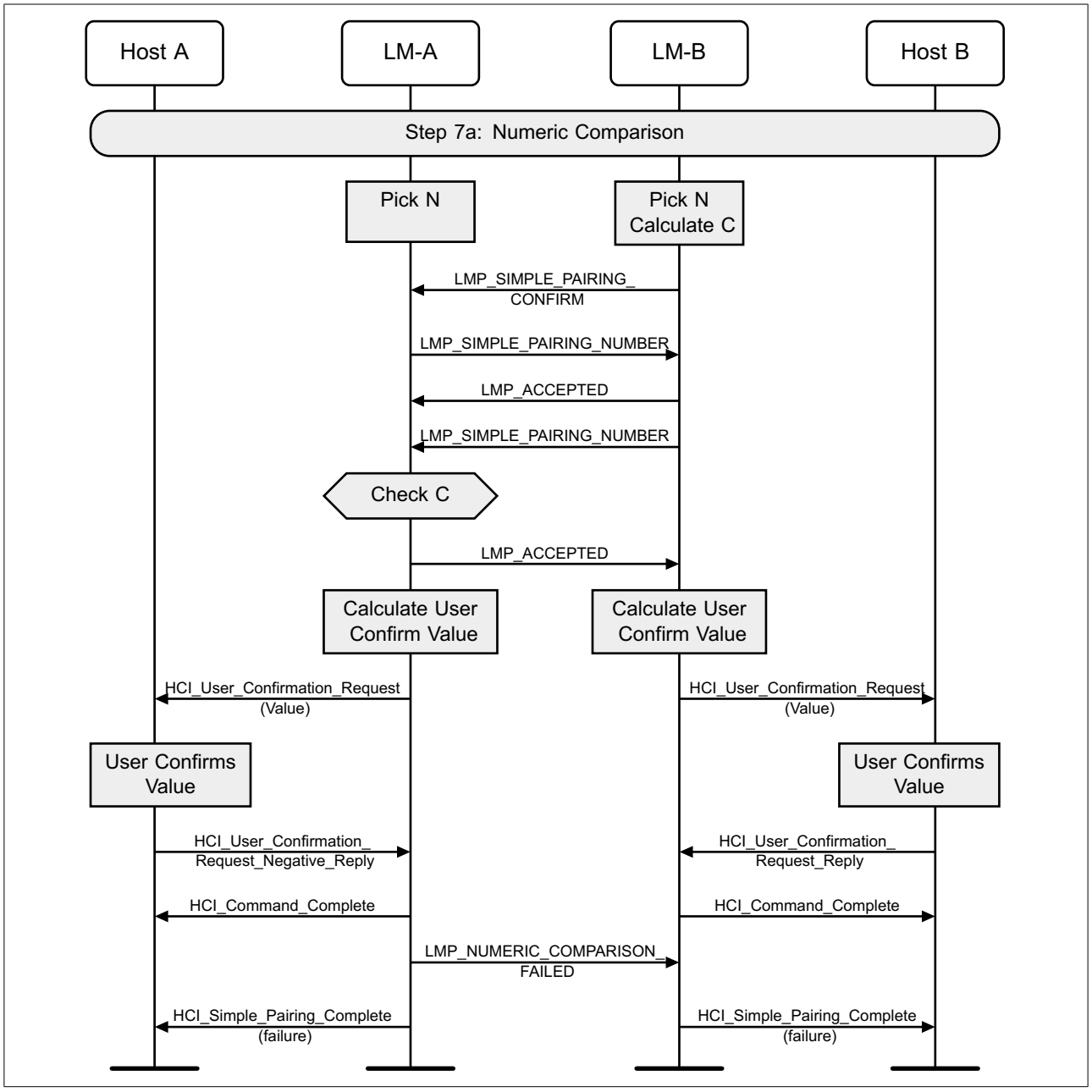



*Figure 4.15: Numeric Comparison authentication (failure on initiating side)*

### 4.2.12 Numeric Comparison failure on responding side

If the numeric comparison fails on the responding side due to the user indicating that the confirmation values do not match, Secure Simple Pairing is terminated.

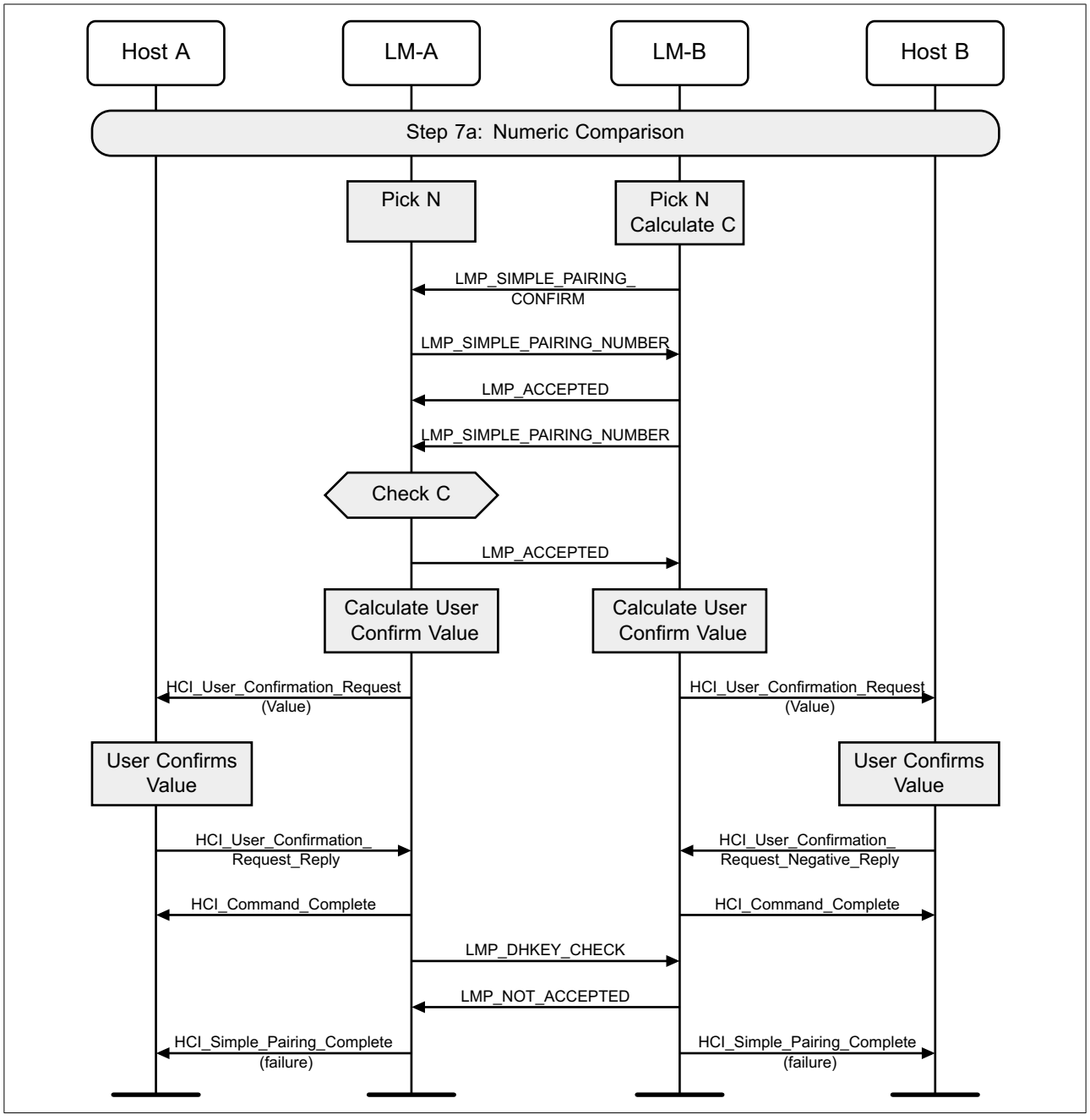



*Figure 4.16: Numeric Comparison failure on responding side*

### 4.2.13 Passkey Entry

The Passkey Entry step is used in two cases: when one device has numeric input only and the other device has either a display or numeric input capability or when both

devices only have numeric input capability. In this step, one device display a number to be entered by the other device or the user enters a number on both devices. This number should be displayed until the end of step 8. Key press notification messages are shown during the user input phase.

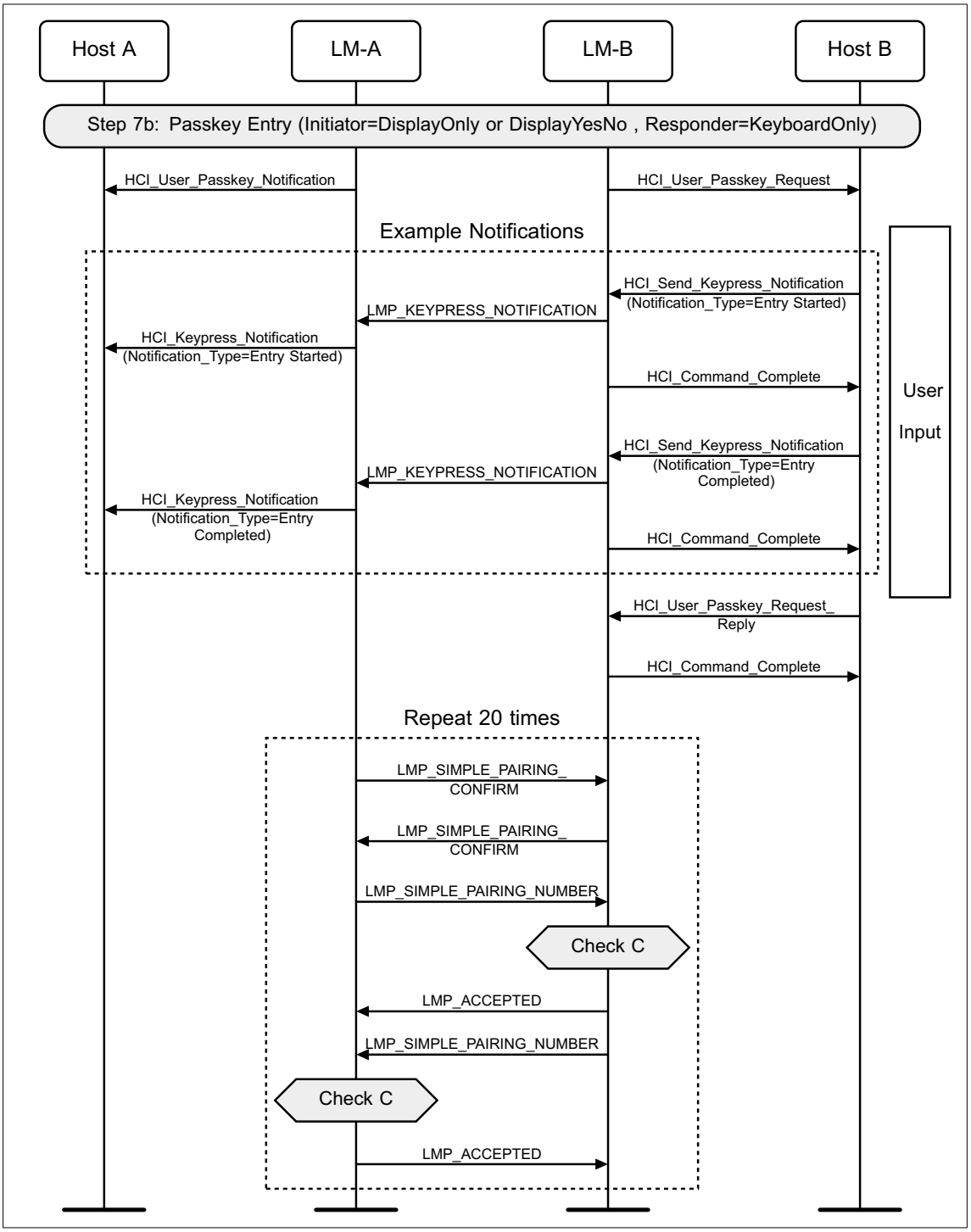



*Figure 4.17: Passkey Entry authentication*

### 4.2.14 Passkey Entry failure on responding side

If the Passkey Entry fails on the responding side, Secure Simple Pairing is terminated.

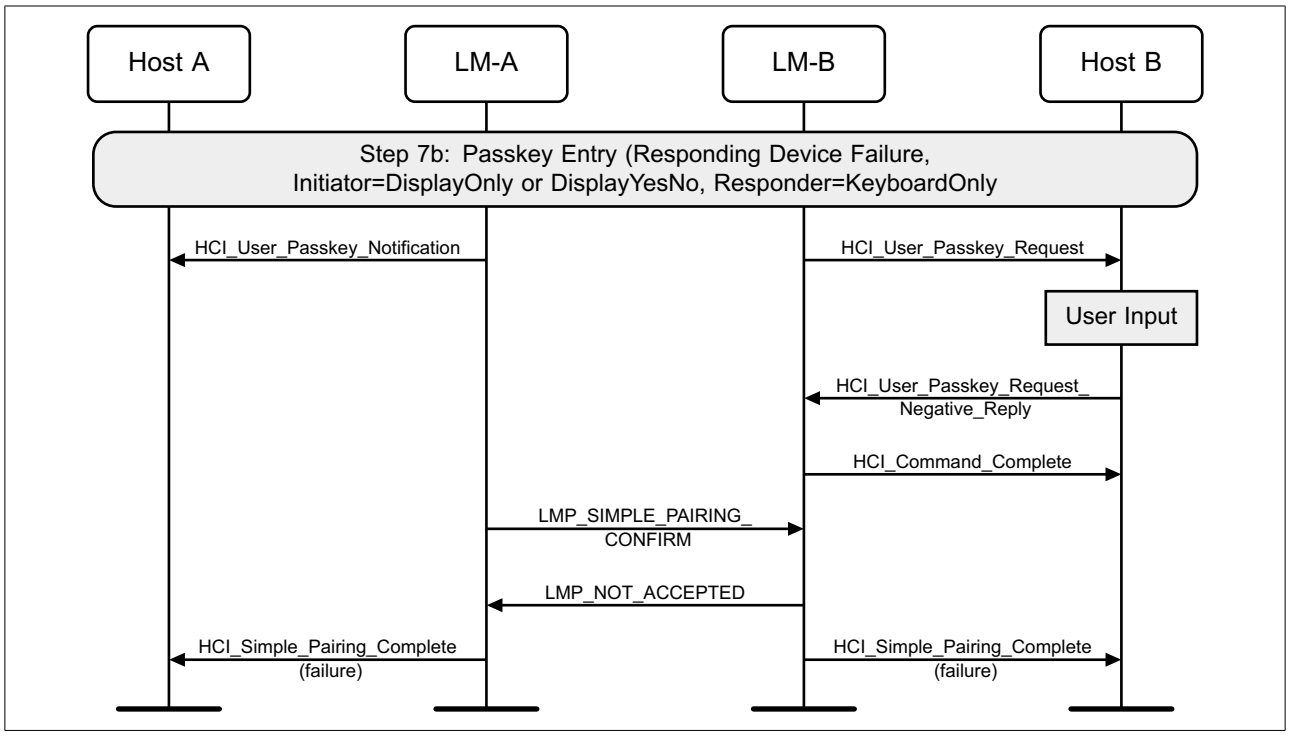



*Figure 4.18: Passkey Entry failure on responding side*

### 4.2.15 Passkey Entry failure on initiator side

If the Passkey Entry fails on the initiating side, Secure Simple Pairing is terminated. This is only possible if the initiating LM side sends an HCI_User_Passkey_Request event.

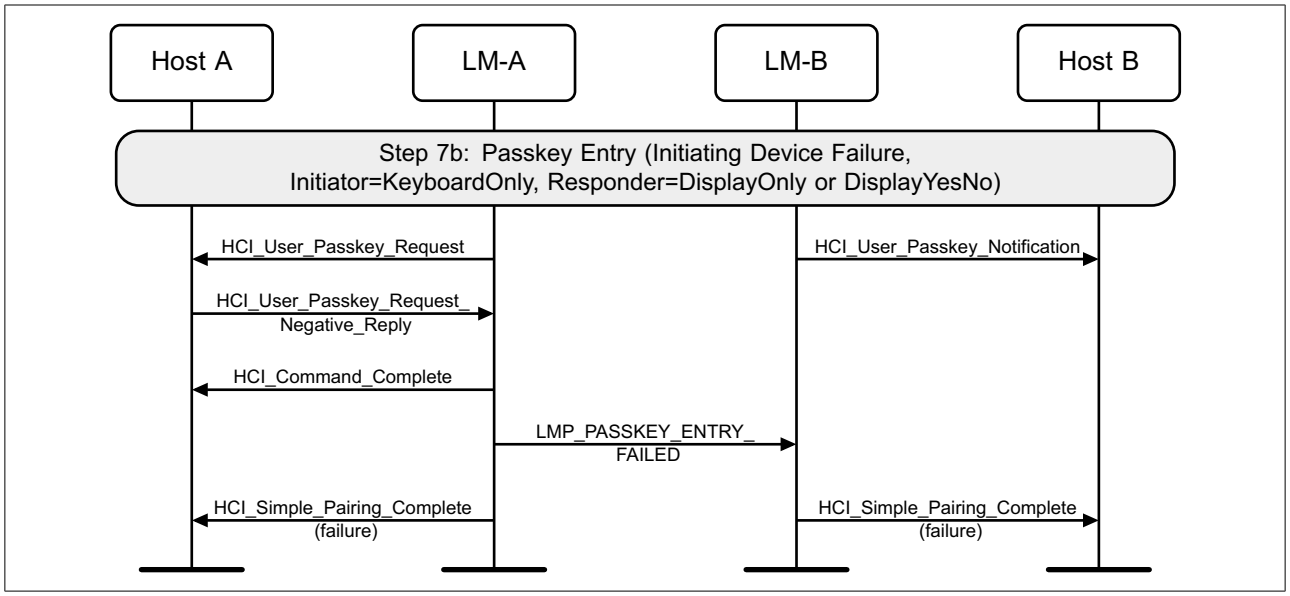



*Figure 4.19: Passkey Entry failure on initiating side*

### 4.2.16 Out of Band

The OOB authentication will only be done when both devices have some OOB information to use. This step requires no user interaction.

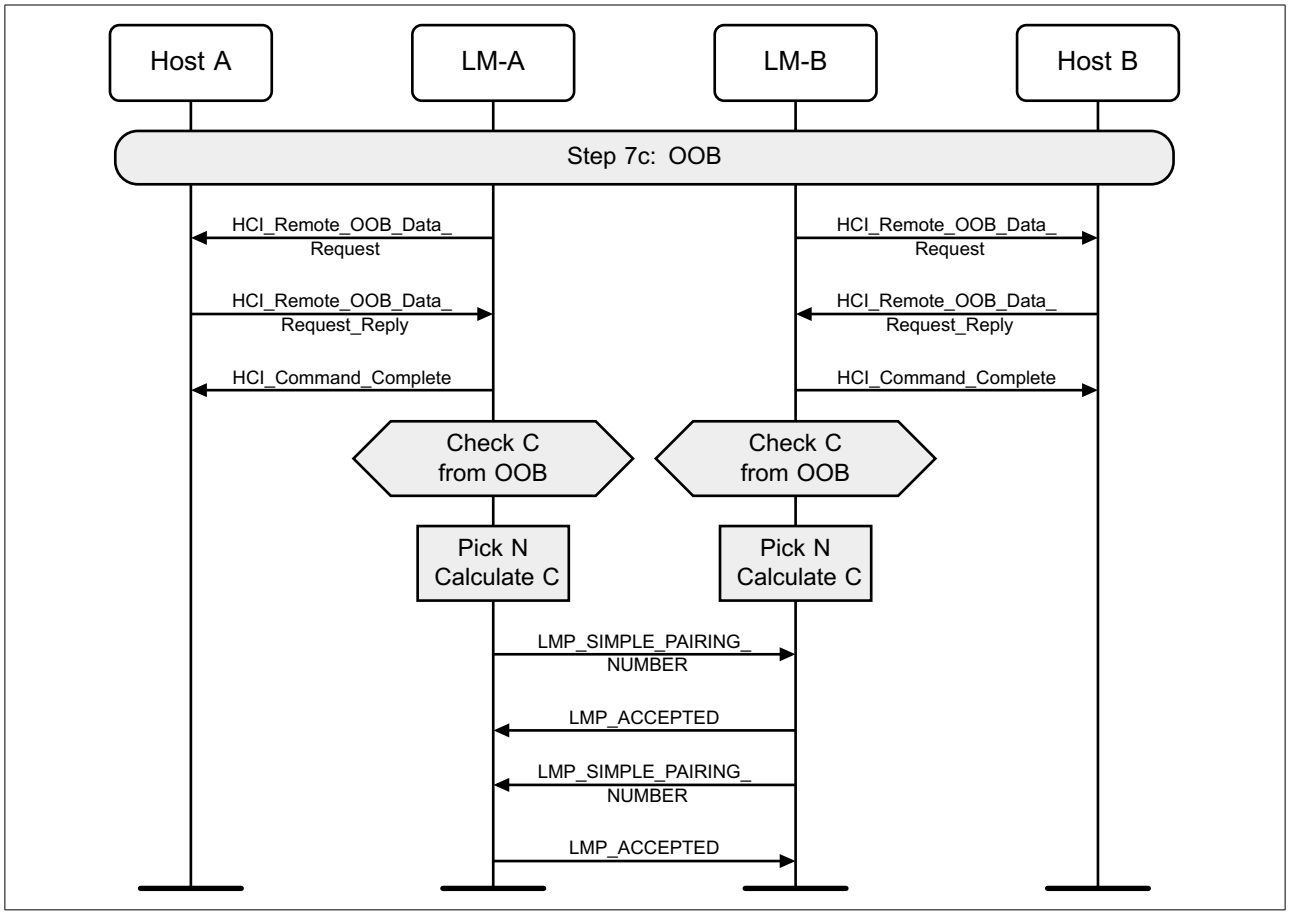



*Figure 4.20: OOB authentication (P-192)*

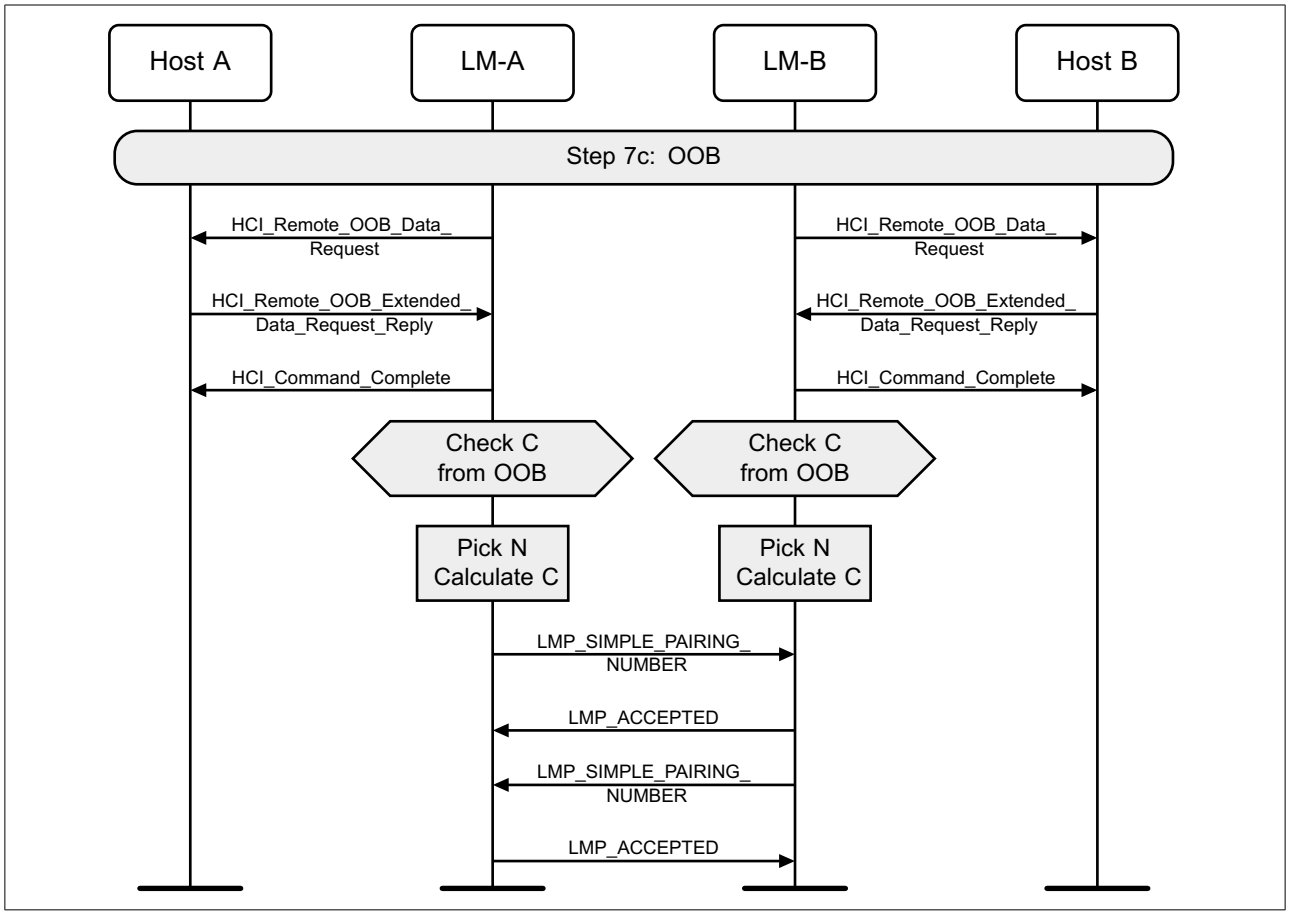



*Figure 4.21: OOB authentication (P-256)*

### 4.2.17 OOB failure on initiator side

If the initiating side does not have OOB information, Secure Simple Pairing is terminated.

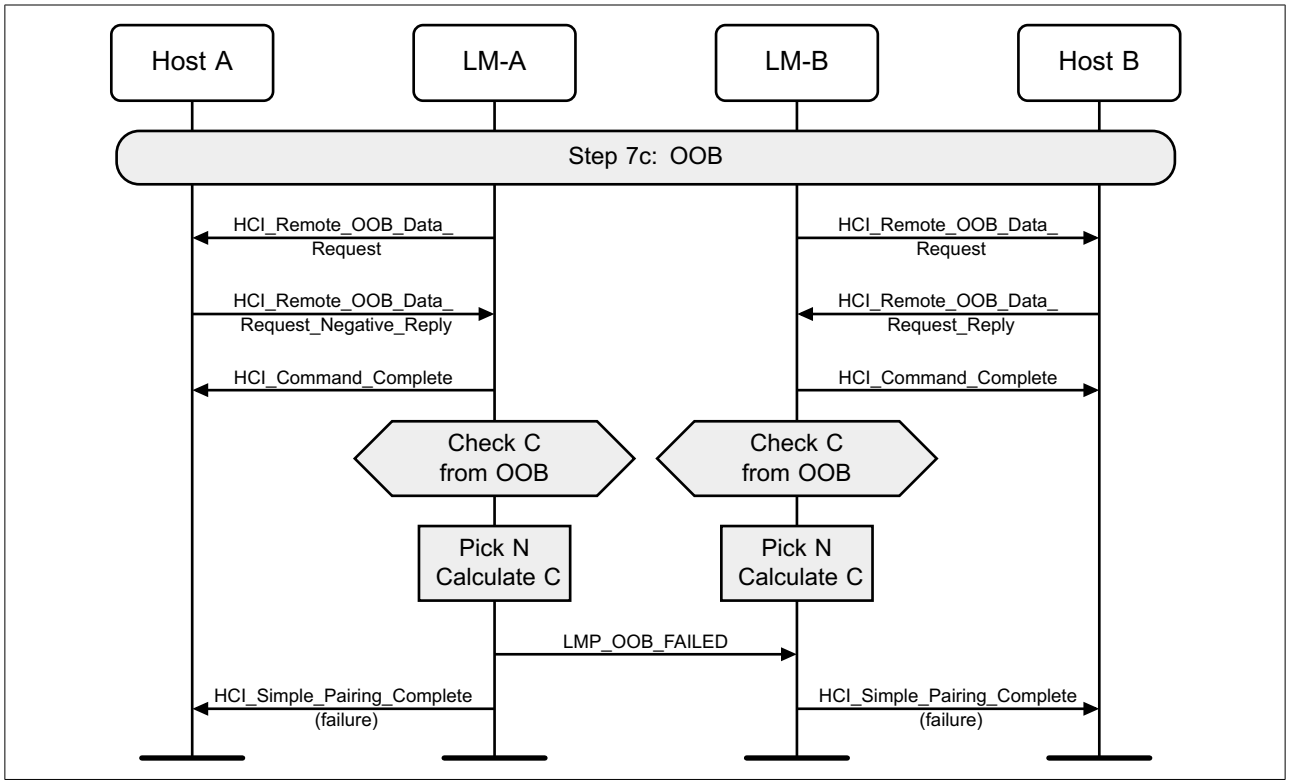



*Figure 4.22: OOB authentication failure on initiating side*

### 4.2.18 DHKey checks

Once the devices have been authenticated, and the DHKey calculation has completed, the DHKey value generated is checked. If this succeeds, then both devices would have

finished displaying information to the user about the process, and therefore a message is sent from the Controller to the Host to notify it to stop displaying this information.

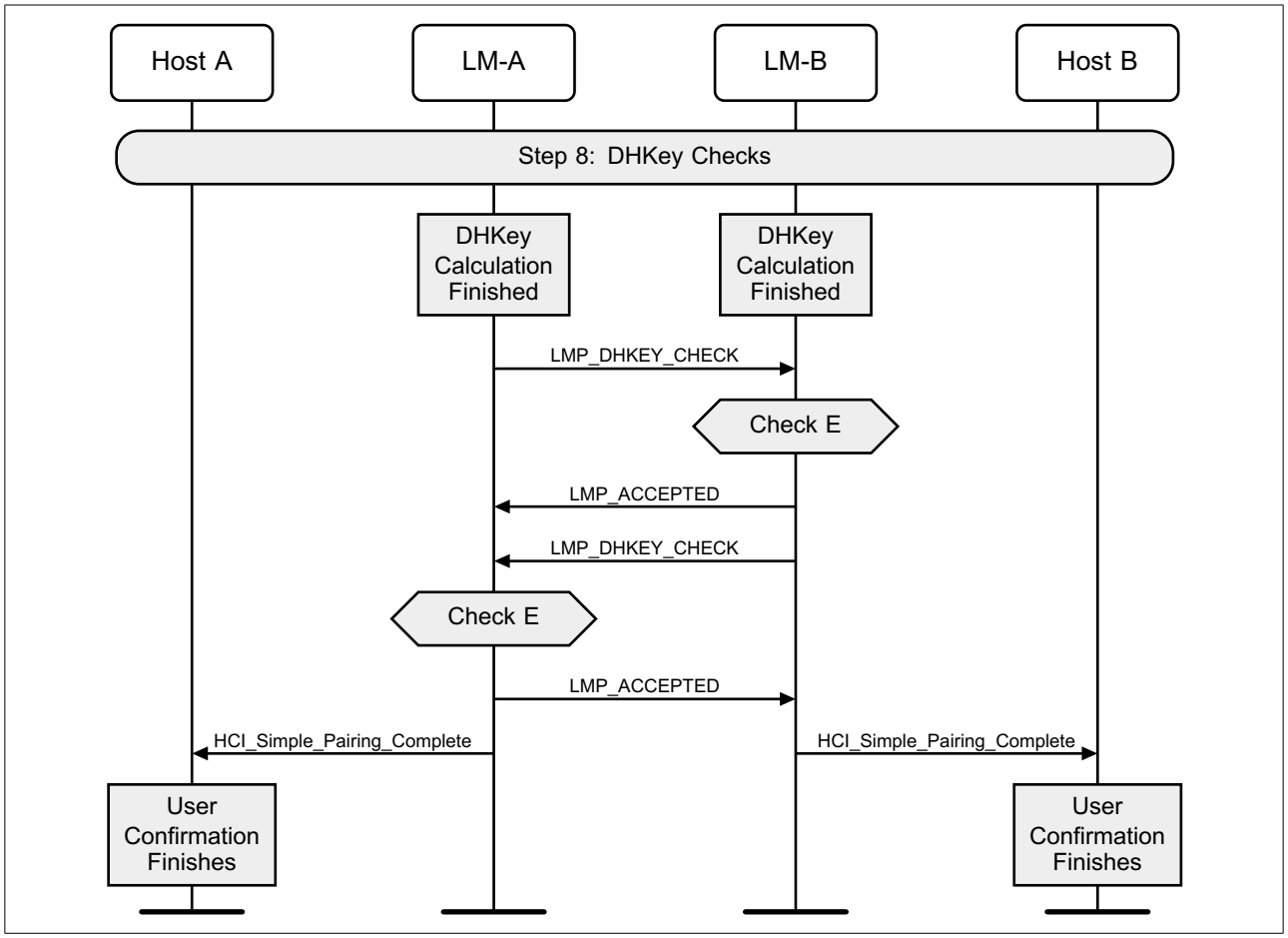



*Figure 4.23: DHKey checks*

### 4.2.19 Calculate link key

Once Secure Simple Pairing has been completed, the link key is calculated from the DHKey and used as input to a standard mutual authentication. Once the authentication has completed successfully, an HCI_Link_Key_Notification event will be generated.

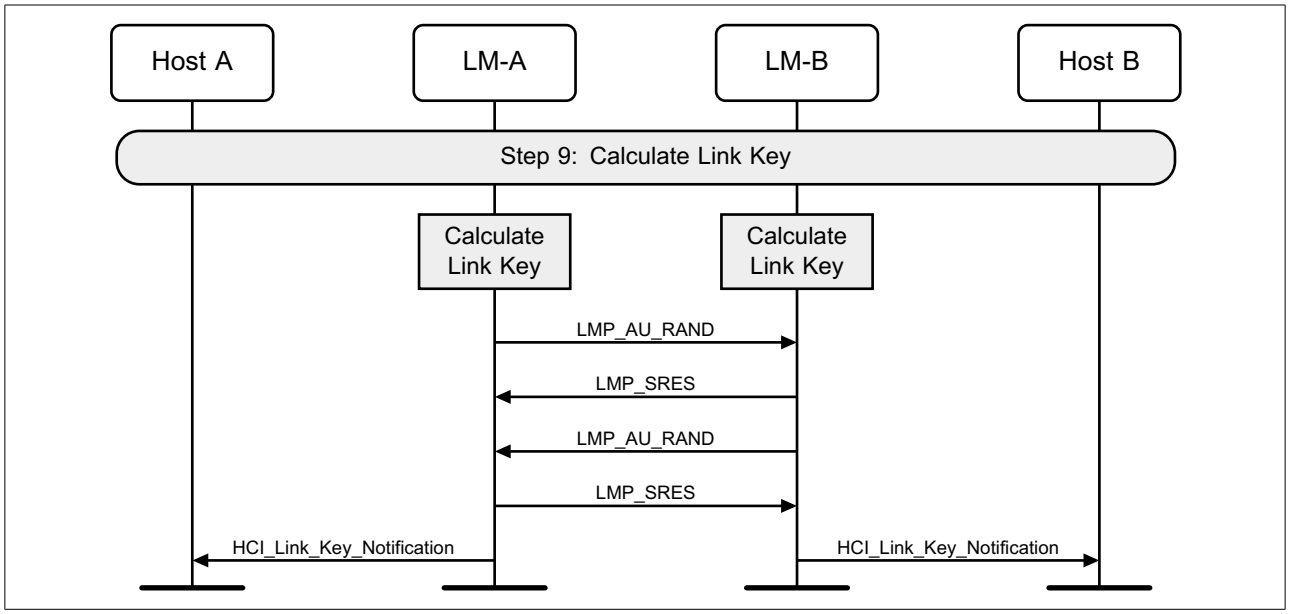



*Figure 4.24: Calculate link key*

### 4.2.20 Enable encryption

Once the link key has been notified to the Host, the HCI_Authentication_Requested command will complete with an HCI_Authentication_Complete event. The Host can then turn on encryption using the standard methods.

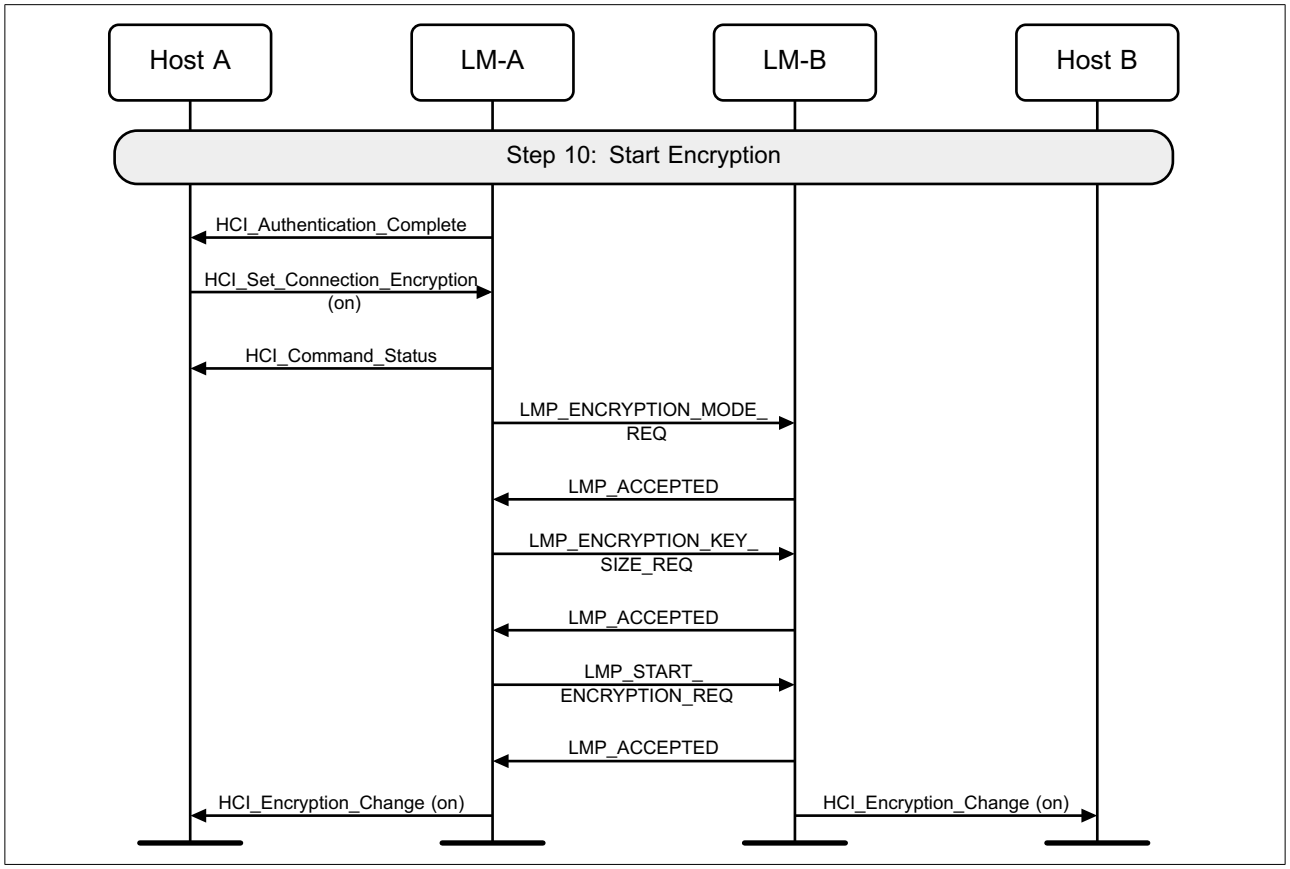



*Figure 4.25: Start encryption*

### 4.2.21 L2CAP connection response

If this Secure Simple Pairing was triggered by an L2CAP Connection Request, then only after all of the above steps have completed can the L2CAP Connection Response message be sent.

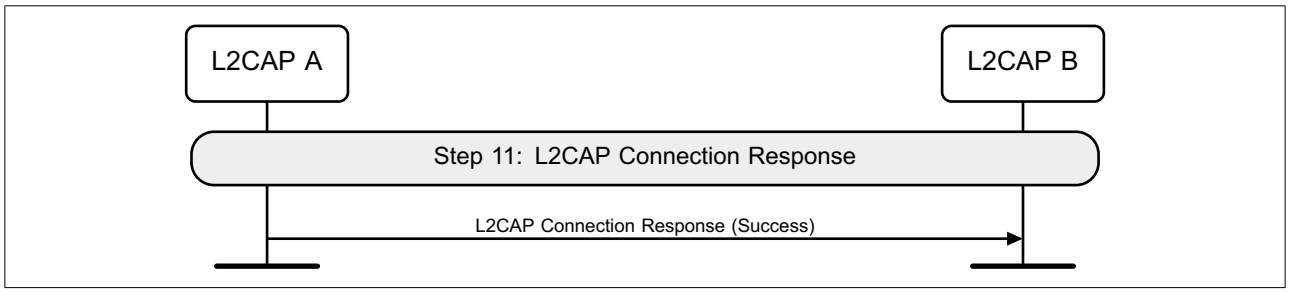



*Figure 4.26: L2CAP Connection Response*

### 4.2.22 LMP ping

When the Authenticated Payload Timeout has nearly expired, the Link Manager will force the remote device to send a packet containing a MIC by sending the LMP_PING_REQ PDU.

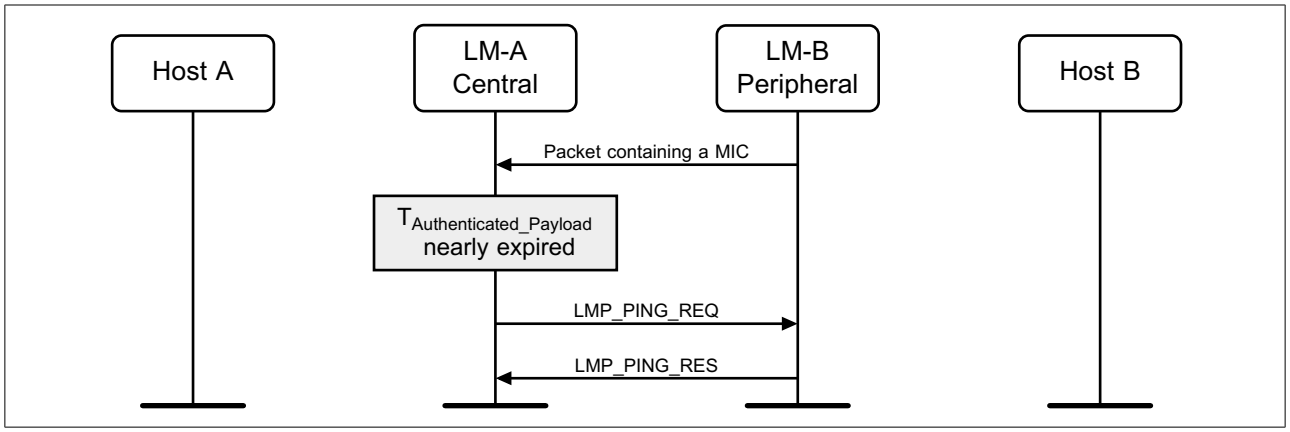



*Figure 4.27: Successful ping*

When a packet with a MIC has not been received within the Authenticated Payload Timeout, the Host is notified that the timer has expired.

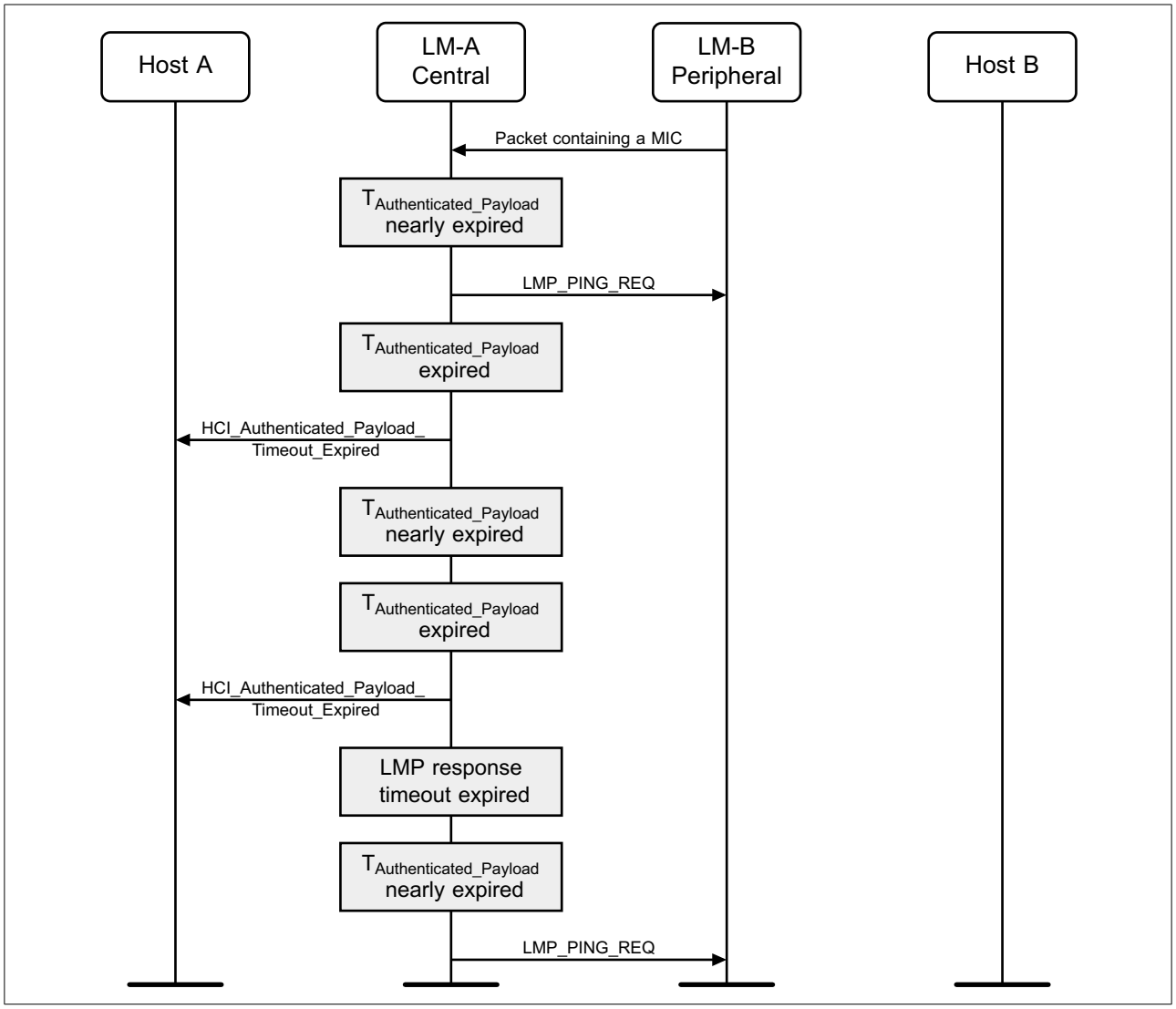



*Figure 4.28: Unsuccessful ping*

## 4.3 Link Supervision Timeout Changed event

When enabled by the Host, the Peripheral generates an HCI_Link_Supervision_Timeout_Changed event after the LMP_SUPERVISION_TIMEOUT PDU is received.

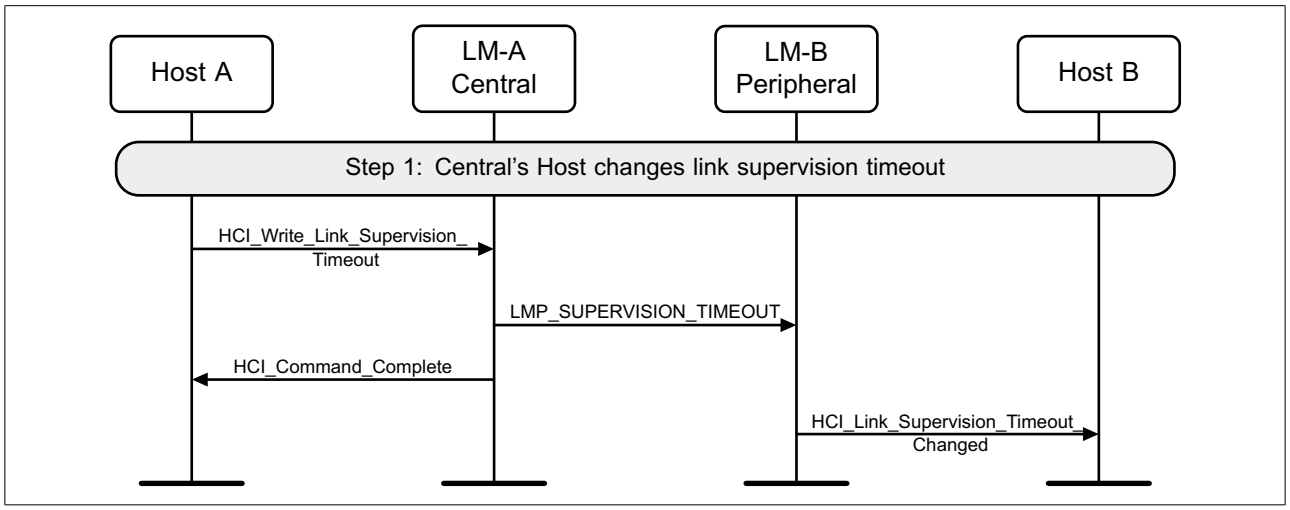



*Figure 4.29: Link supervision timeout event*

## 4.4 Set Connection Encryption

**Step 1:** The Host may at any time turn on encryption using the HCI_Set_Connection_Encryption command. This command can be originated from either the Central or Peripheral sides. Only the Central side is shown in Figure 4.30. If this command is sent from a Peripheral, the only difference is that the LMP_ENCRYPTION_MODE_REQ PDU will be sent from the Peripheral. The LMP_ENCRYPTION_KEY_SIZE_REQ and LMP_START_ENCRYPTION_REQ PDUs will always be requested from the Central. (See Figure 4.30.)

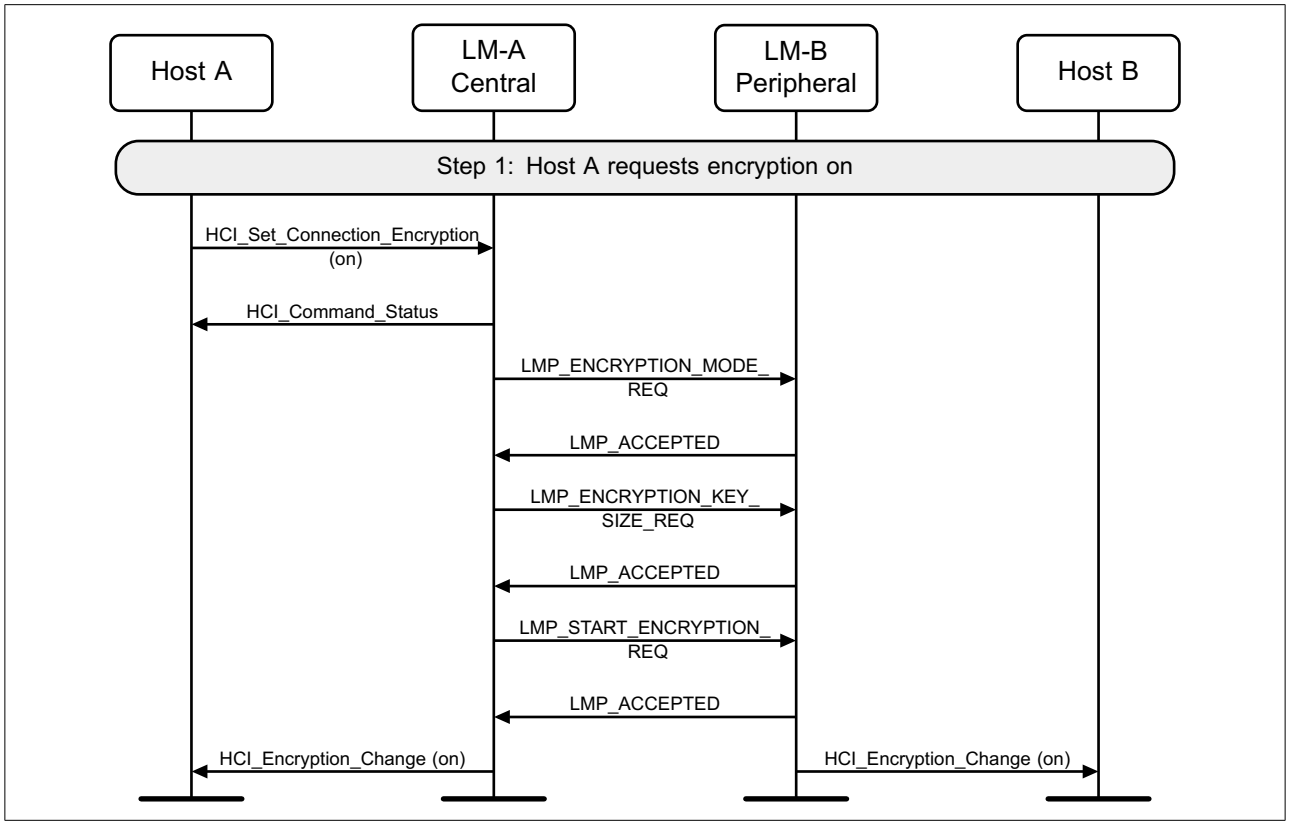



*Figure 4.30: Encryption requested*

**Step 2:** To terminate the use of encryption, the HCI_Set_Connection_Encryption command is used. (See Figure 4.31.)

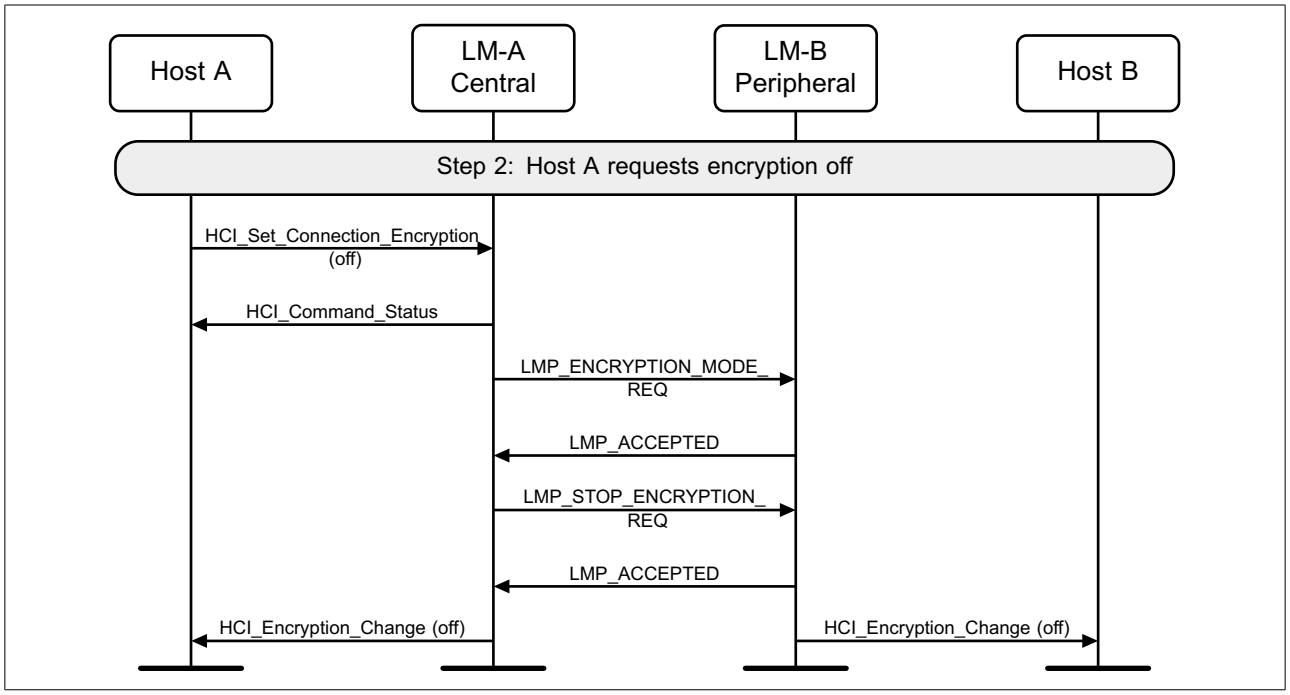



*Figure 4.31: Encryption off requested*

## 4.5 Change connection link key

**Step 1:** The Central's Host (Host A) may change the connection link key using the HCI_Change_Connection_Link_Key command. A new link key will be generated and the Hosts will be notified of this new link key. (See Figure 4.32.)

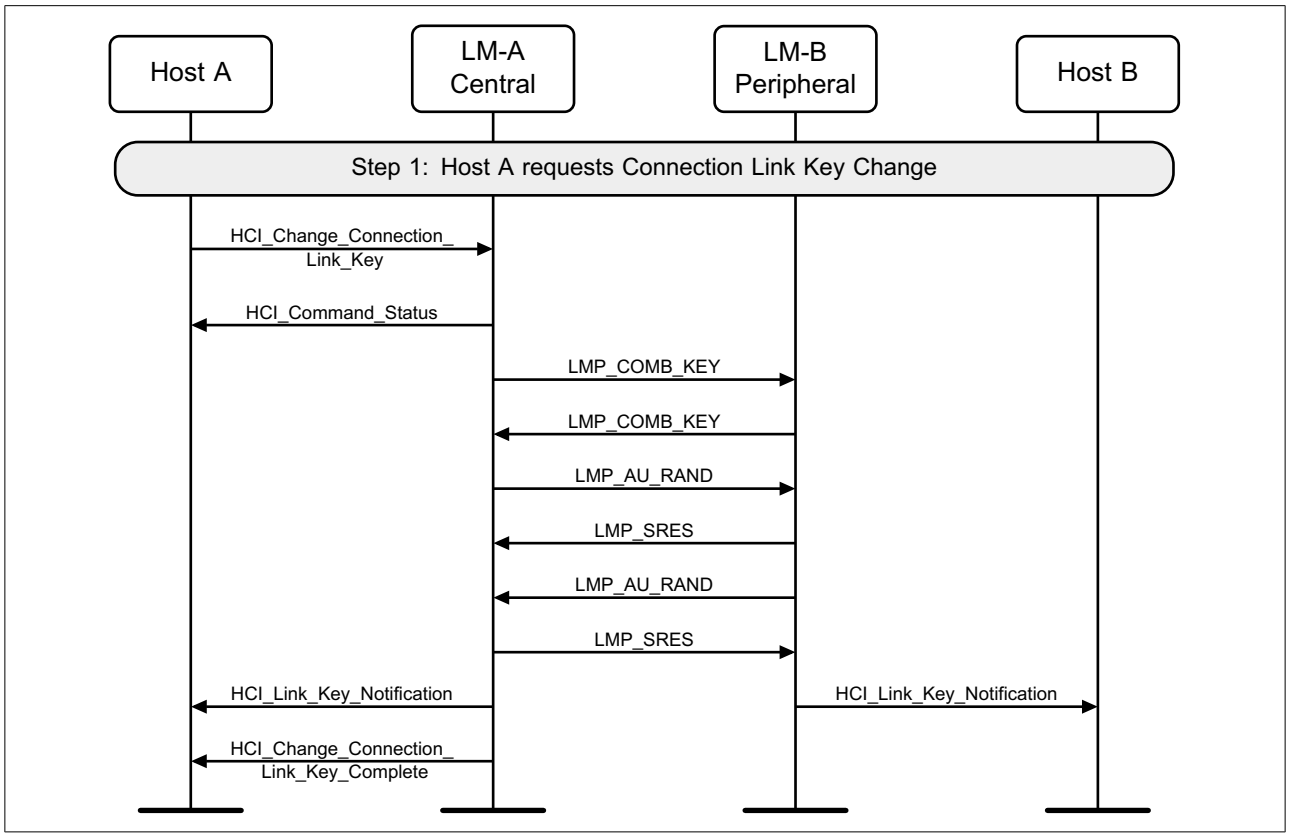



*Figure 4.32: Change connection link key*

## 4.6 Change connection link key with encryption pause and resume

**Step 1:** The Central's Host (Host A) may change the connection link key using the HCI_Change_Connection_Link_Key command. A new link key will be generated and the Hosts will be notified of this new link key. Encryption will then be paused and

resumed, immediately using this new link key to generate a new encryption key. (See Figure 4.33.)

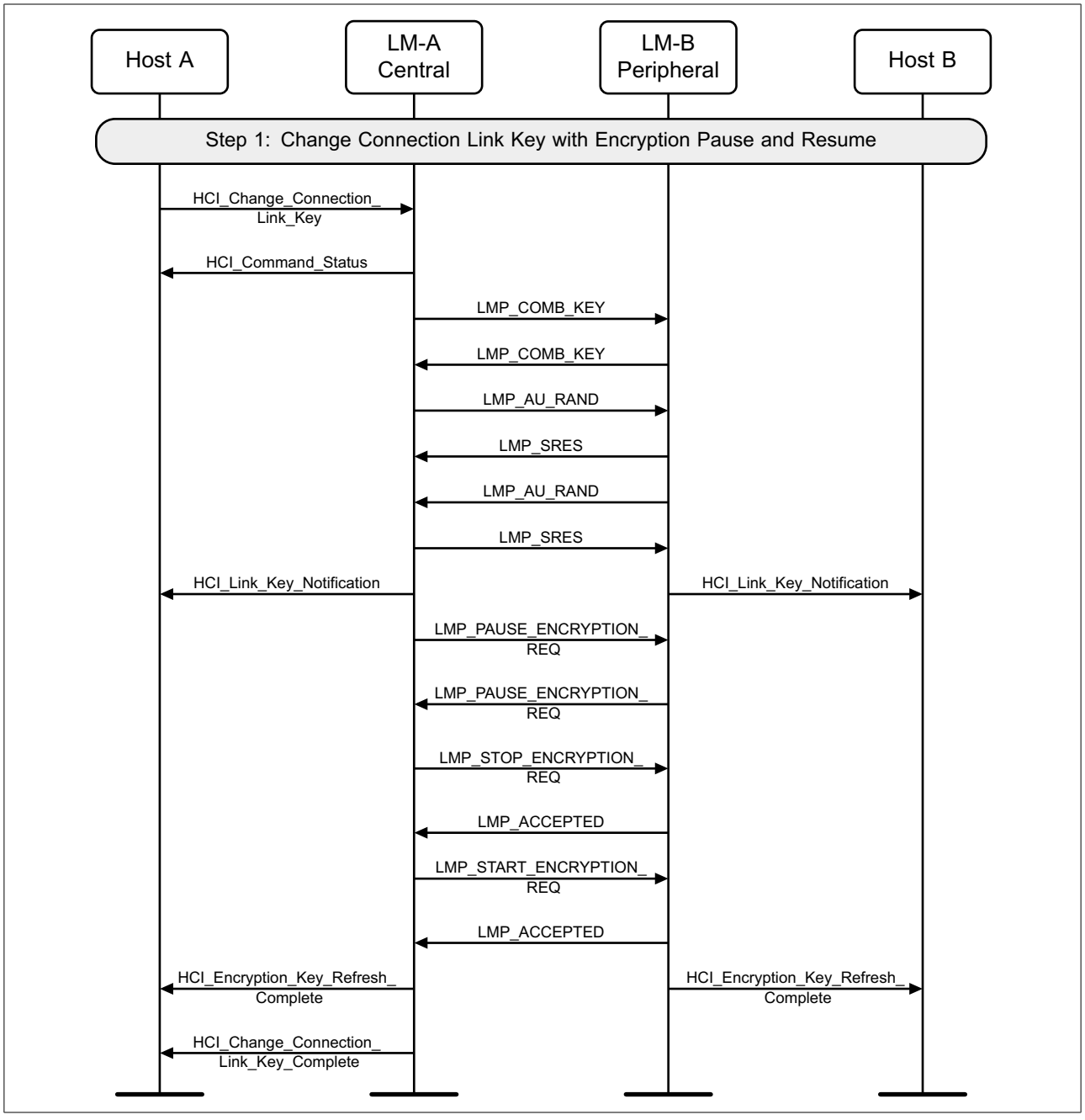



*Figure 4.33: Change connection link key with encryption pause and resume*

## 4.7 Temporary Link Key

**Step 1:** The Host changes to a Temporary Link Key from a Semi-permanent Link Key using the HCI_Link_Key_Selection command when at least one device does not support Encryption Pause and Resume. (See Figure 4.34.)

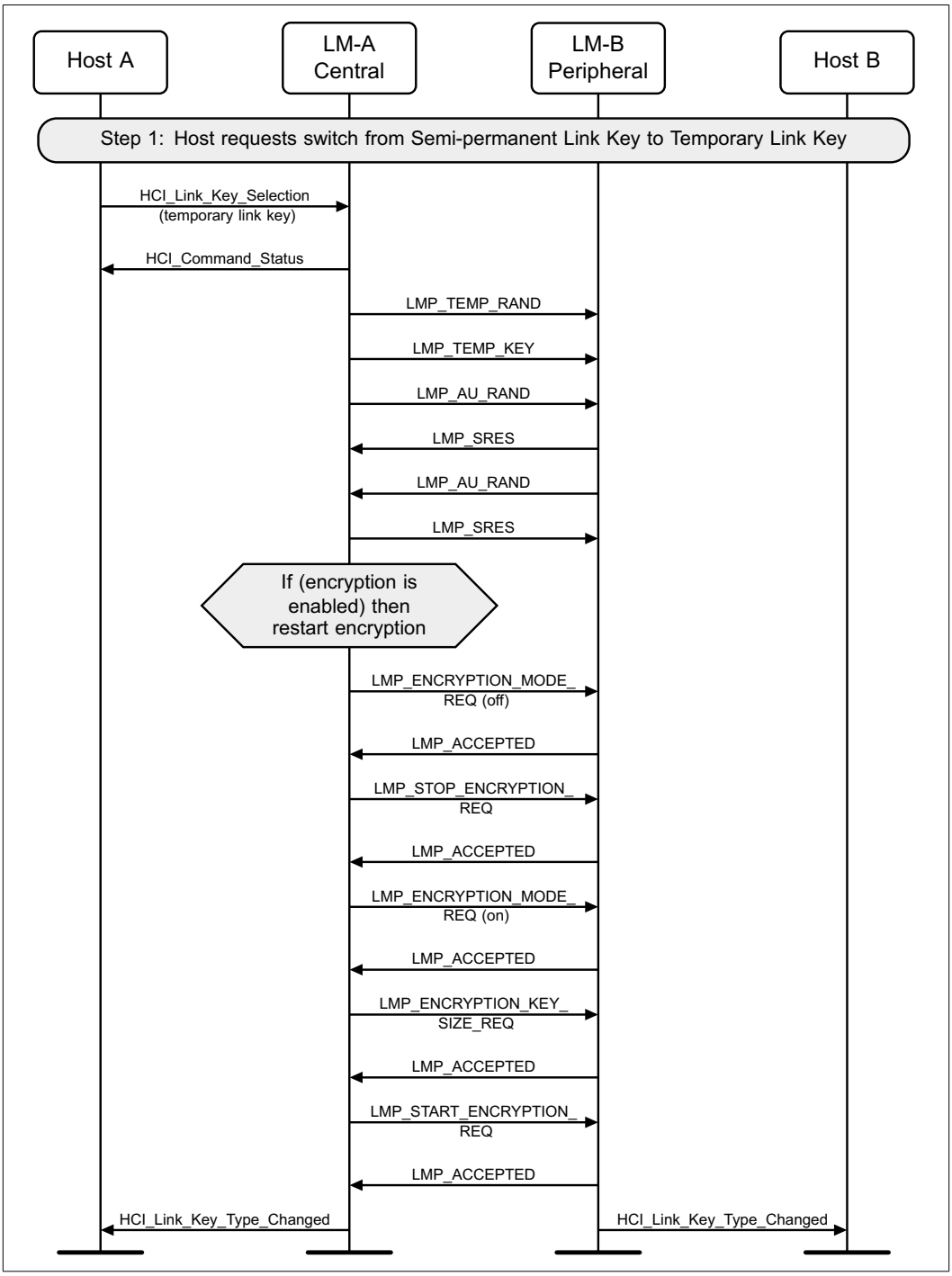



*Figure 4.34: Change to temporary link key*

**Step 2:** The Host changes to a Semi-permanent Link Key from a Temporary Link Key using the HCI_Link_Key_Selection command. (See Figure 4.35.)

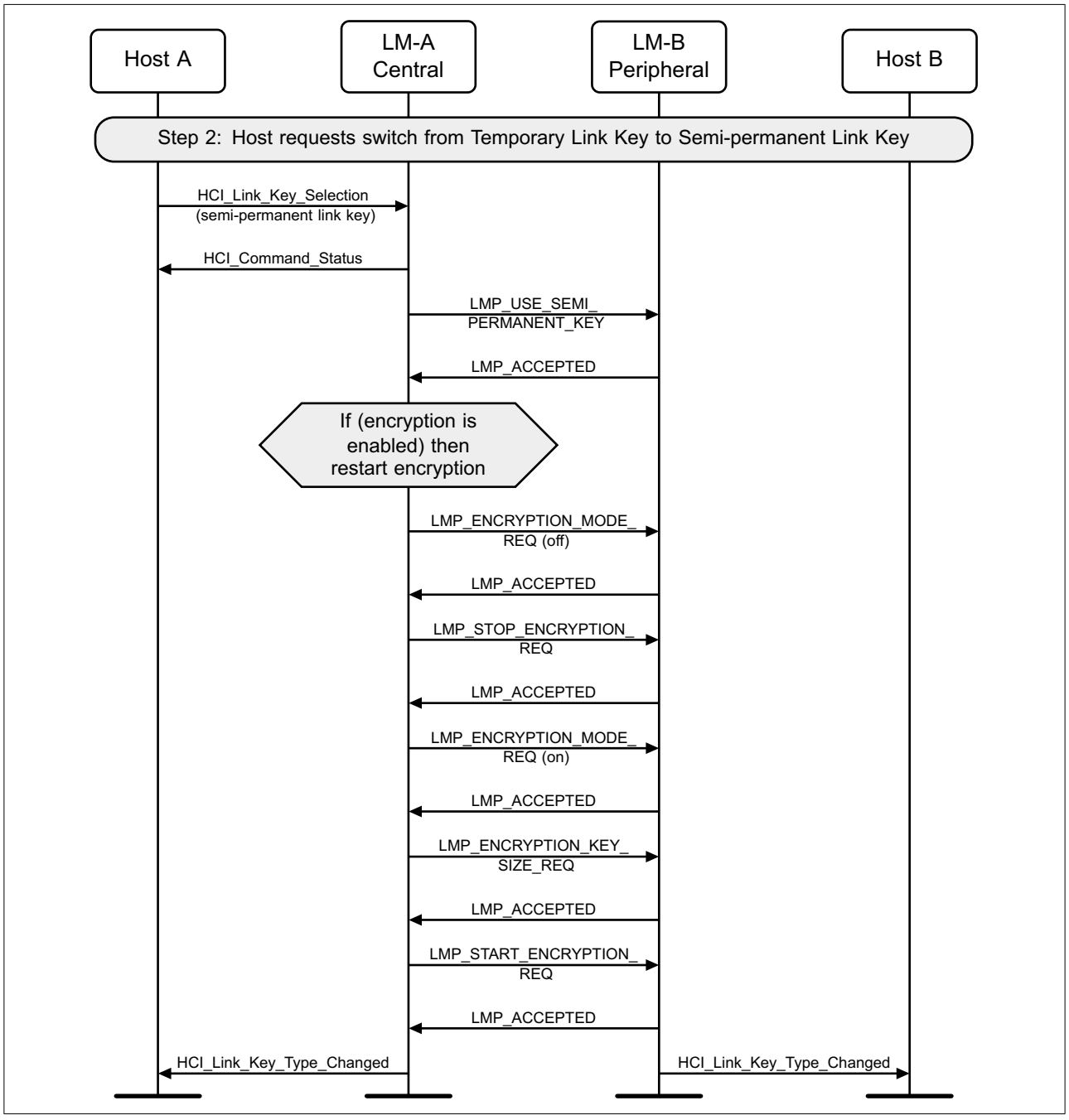



*Figure 4.35: Change to semi-permanent link key*

## 4.8 Read remote supported features

Using the HCI_Read_Remote_Supported_Features command the supported LMP Features of a remote device can be read. (See Figure 4.36.)

If the remote supported features have been obtained previously then the Controller may return them without sending any LMP PDUs.

**Step 1:** The Host requests the supported features of a remote device.

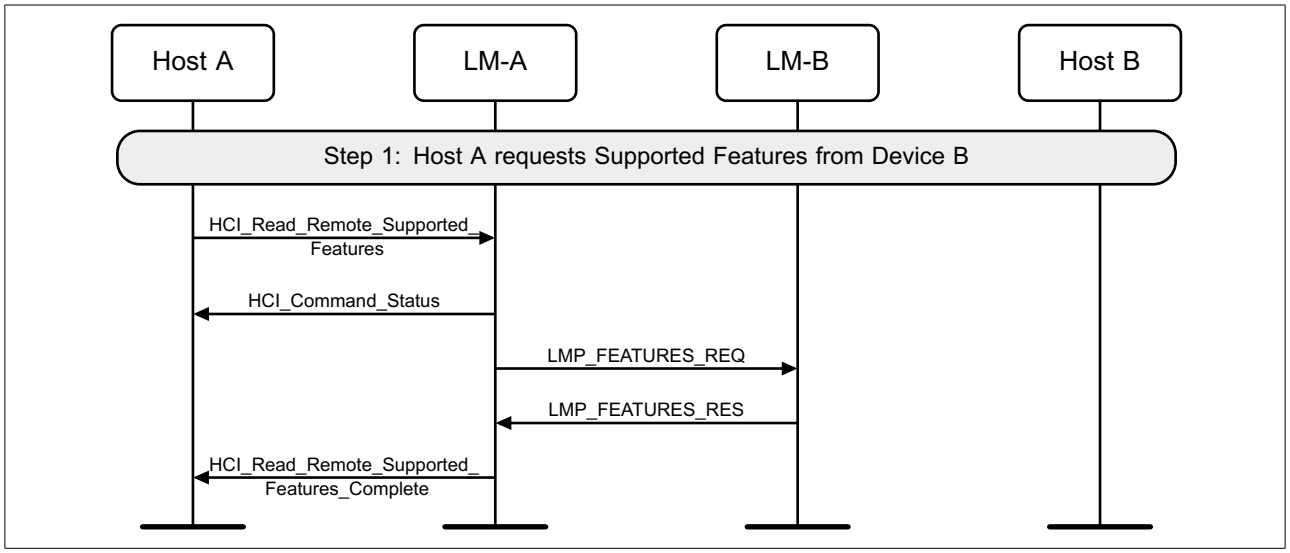



*Figure 4.36: Read remote supported features*

## 4.9 Read remote extended features

Using the HCI_Read_Remote_Extended_Features command the extended LMP features of a remote device can be read. (See Figure 4.37.)

If the remote extended features have been obtained previously then the Controller may return them without sending any LMP PDUs.

**Step 1:** The Host requests the extended features of a remote device.

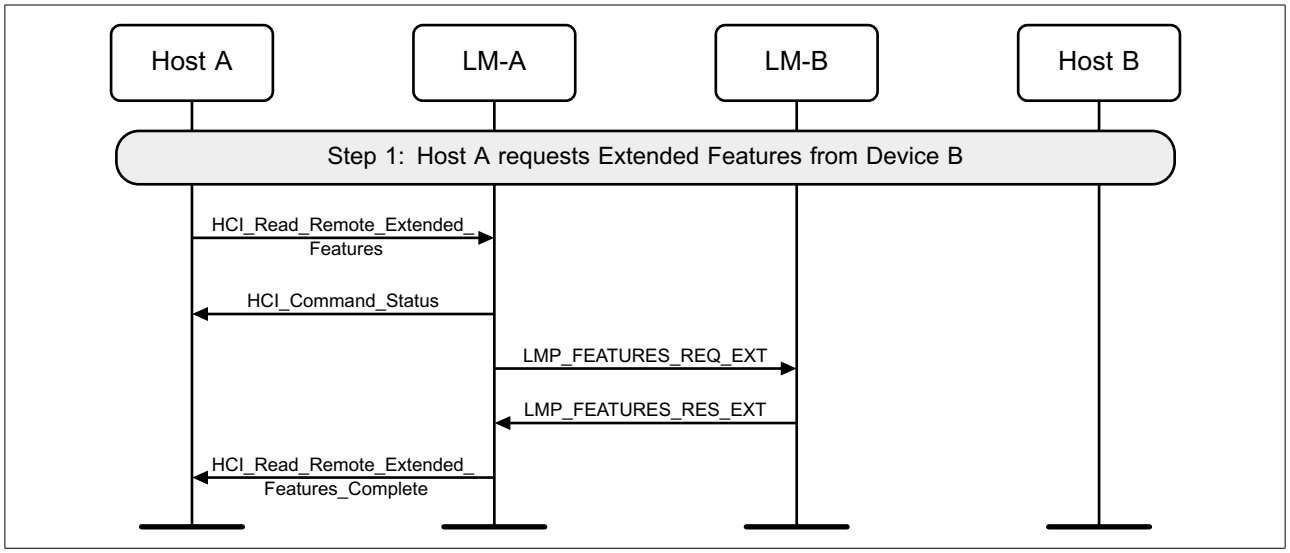



*Figure 4.37: Read remote extended features*

## 4.10 Read clock offset

Using the HCI_Read_Clock_Offset command the device acting as the Central can read the Clock Offset of a Peripheral (see Figure 4.38). The Clock Offset can be used to speed up the paging procedure in a later connection attempt.

**Step 1:** The Host requests the clock offset of a remote device.

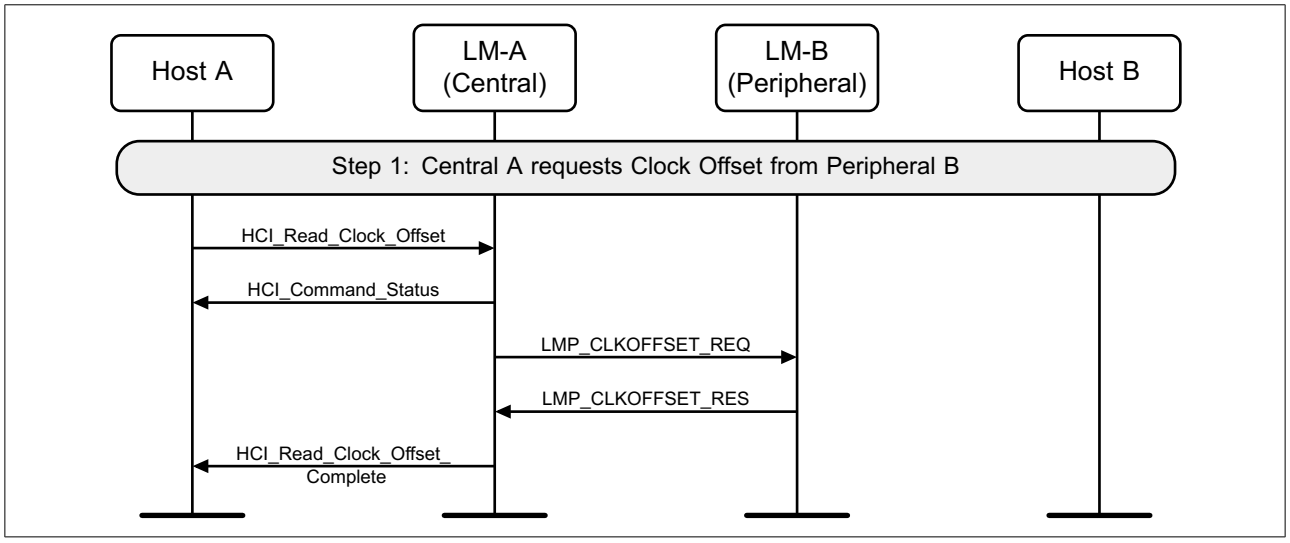



*Figure 4.38: Read clock offset (Central)*

If the command is requested from the Peripheral, the Controller will directly return an HCI_Command_Status event and an HCI_Read_Clock_Offset_Complete event without sending any LMP PDUs (see Figure 4.39).

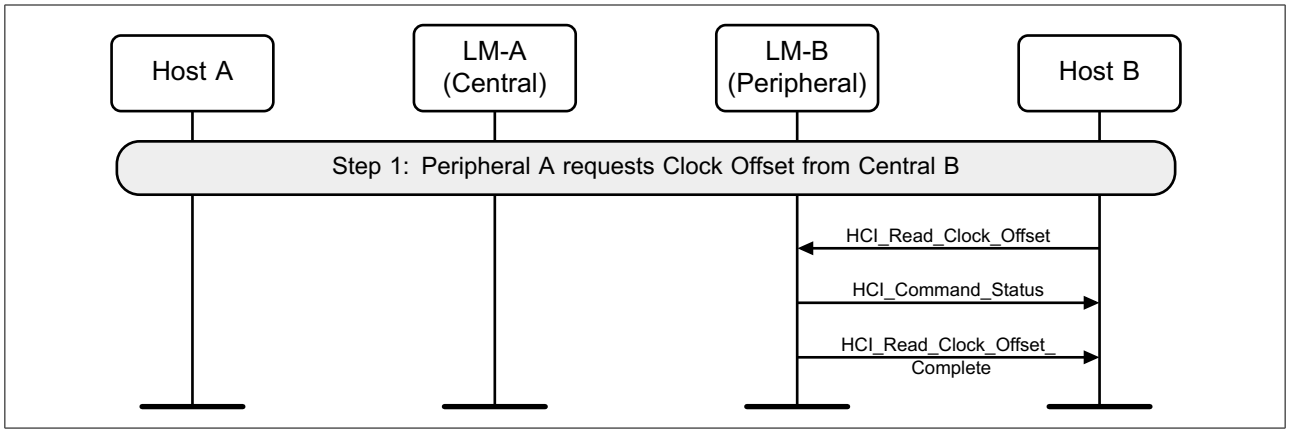



*Figure 4.39: Read clock offset (Peripheral)*

## 4.11 Role switch on an encrypted link using encryption pause and resume

The HCI_Switch_Role command can be used to explicitly switch the current Central / Peripheral role of the local device with the specified device. The Central's Host (A) requests a role switch with a Peripheral. This will first pause encryption, and then send the switch request, and the Peripheral will respond with the slot offset and accepted. The role switch is performed by doing the TDD switch and piconet switch.

Encryption is resumed, and finally an HCI_Role_Change event is sent on both sides. (See Figure 4.40.)

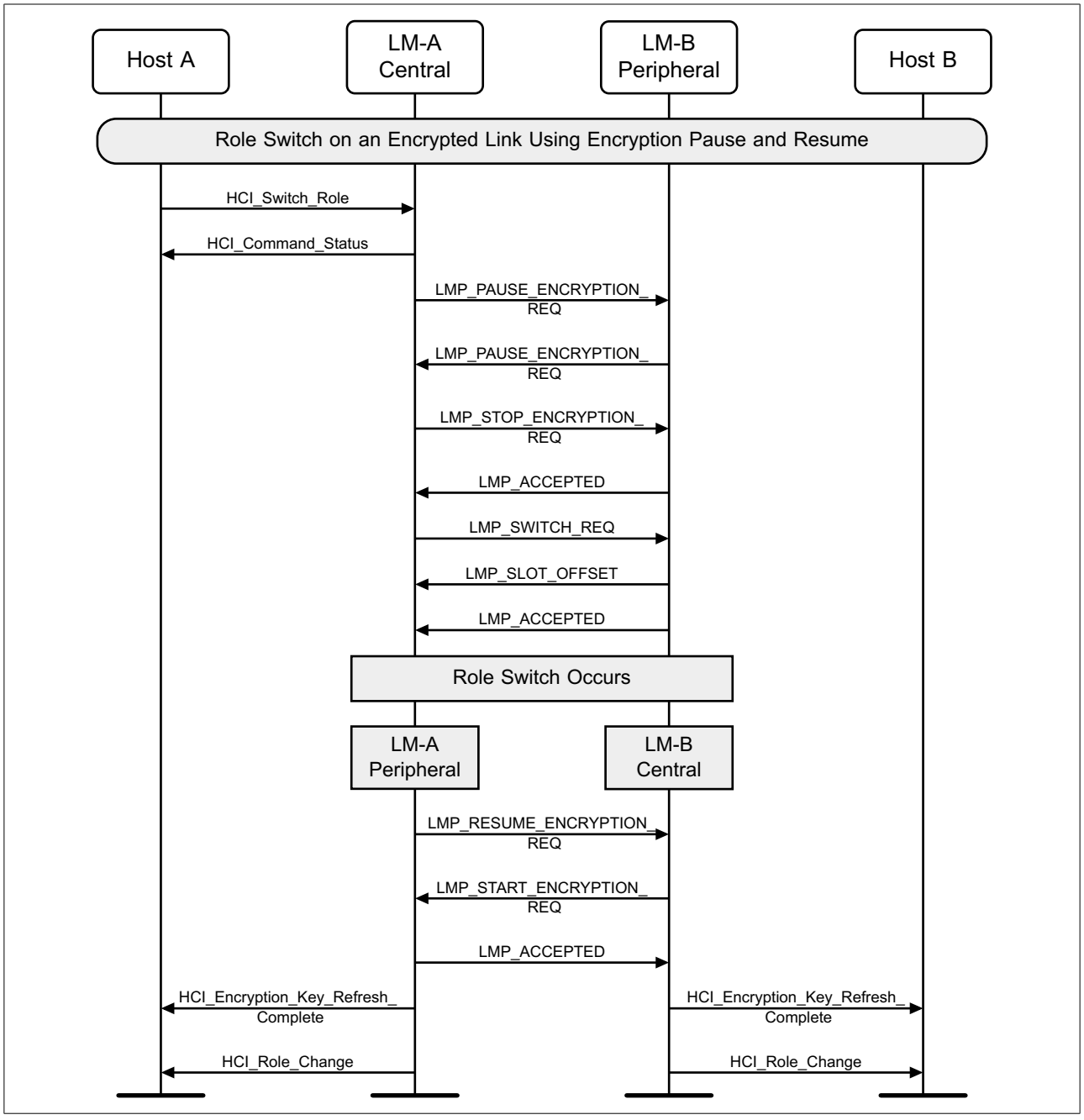



*Figure 4.40: Role switch on an encrypted link using encryption pause and resume*

## 4.12 Refreshing encryption keys

The HCI_Refresh_Encryption_Key command may be used by the Central's Host to explicitly pause and resuming encryption to refresh the encryption key. After encryption is resumed an HCI_Encryption_Key_Refresh_Complete event is sent on both sides. (See Figure 4.41).

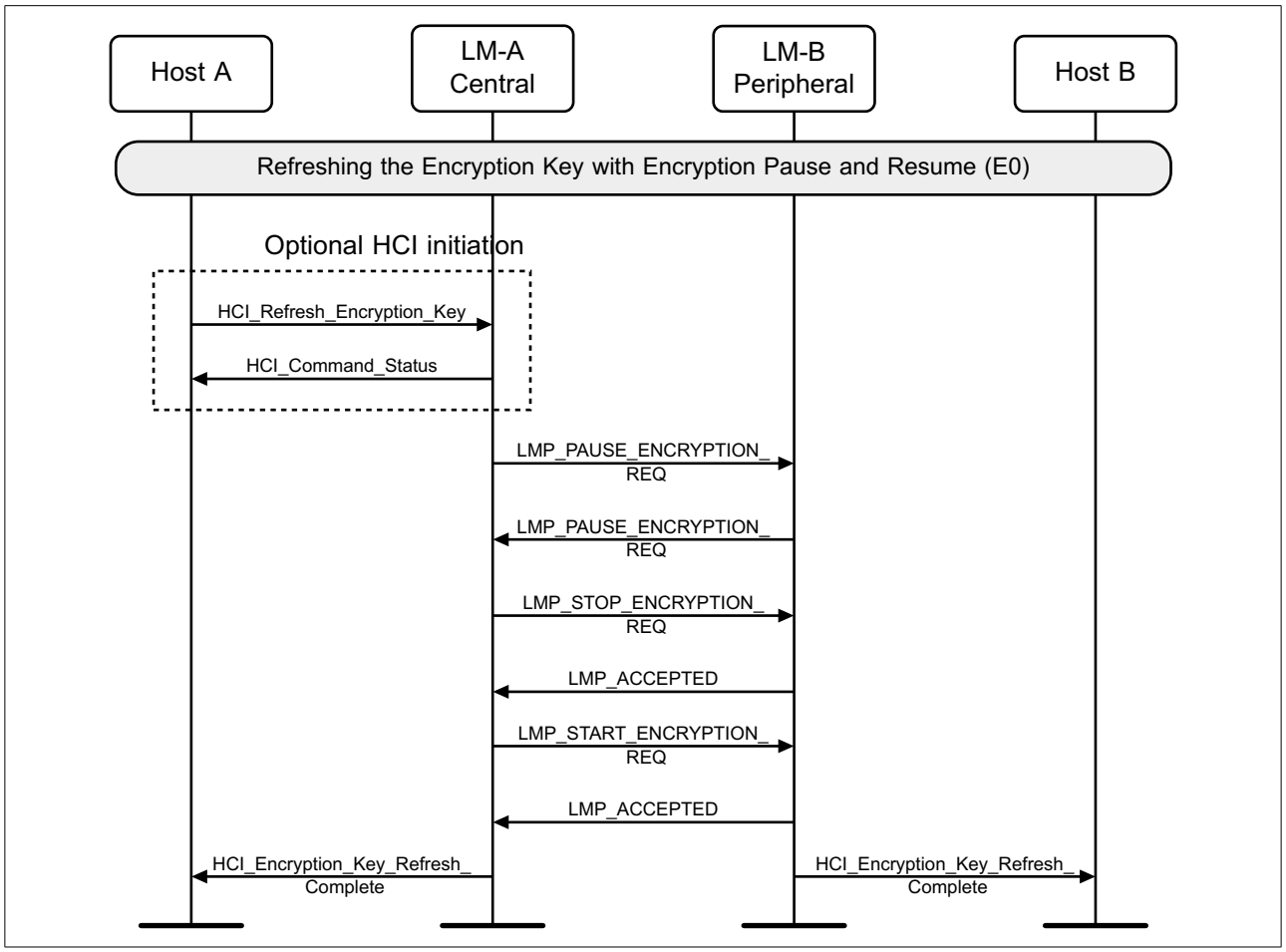



*Figure 4.41: Refreshing encryption keys (E0)*

When both devices support Secure Connections, the encryption key refresh sequence is performed as follows.

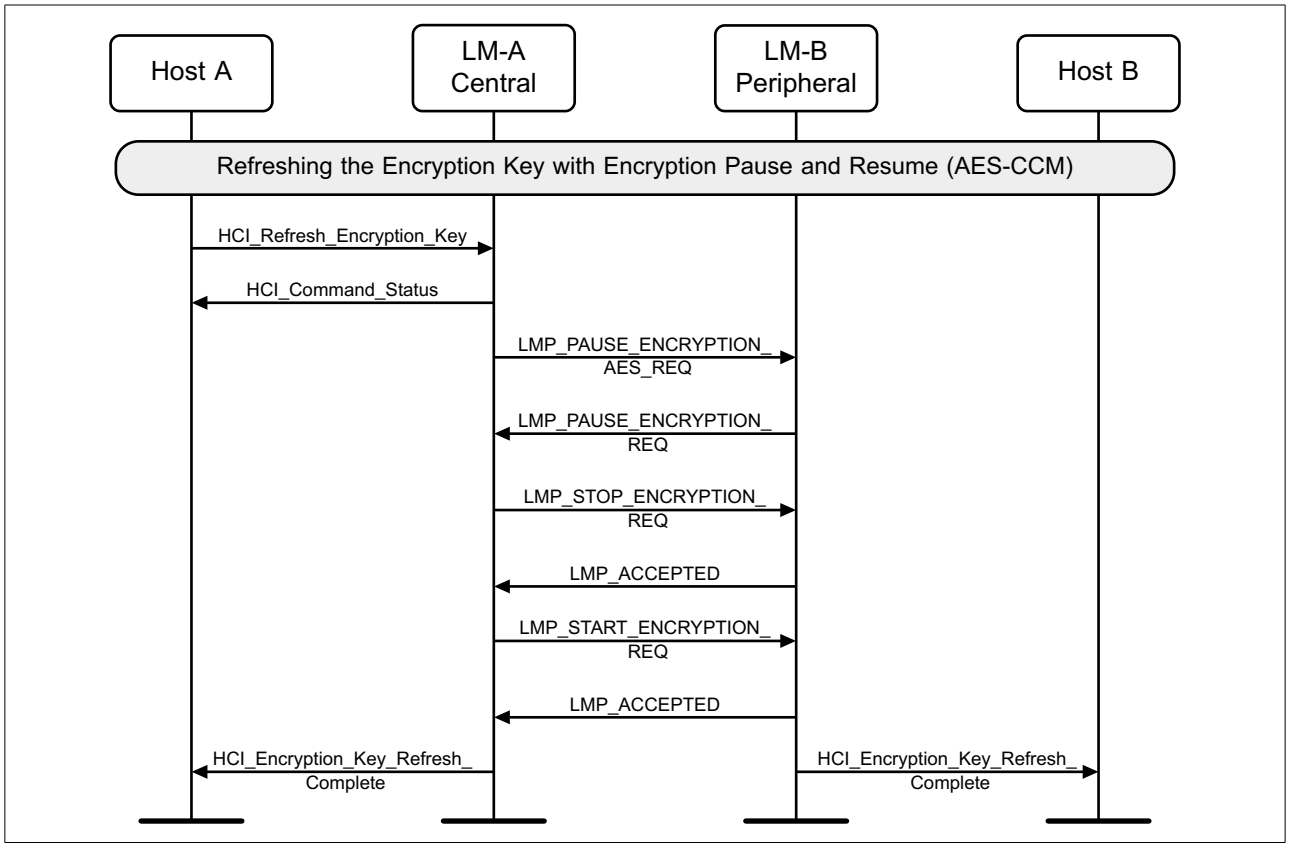



*Figure 4.42: Refreshing encryption keys (AES-CCM)*

## 4.13 Read remote version information

Using the HCI_Read_Remote_Version_Information command the version information of a remote device can be read. (See Figure 4.43.)

If the remote version information has been obtained previously then the Controller may return them without sending any LMP PDUs.

**Step 1:** The Host requests the version information of a remote device.

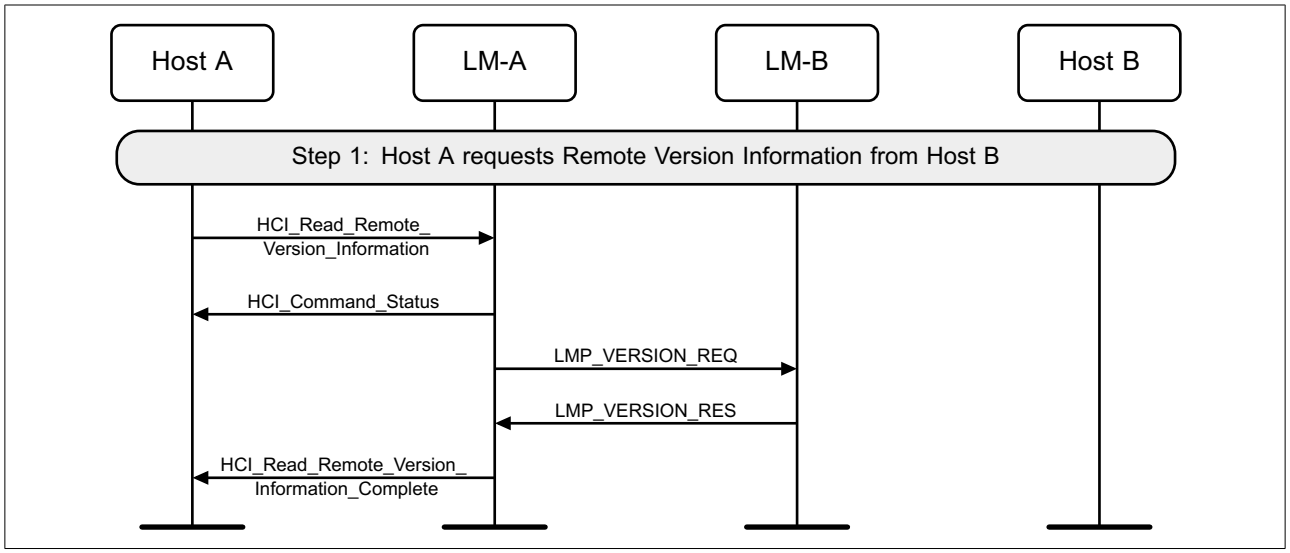



*Figure 4.43: Read remote version information*

## 4.14 QoS setup

Using the HCI_Flow_Specification command the Quality of Service (QoS) and Flow Specification requirements of a connection can be notified to a Controller. The Controller may then change the quality of service parameters with a remote device. (See Figure 4.44.)

**Step 1:** The Host sends QoS parameters to a remote device.

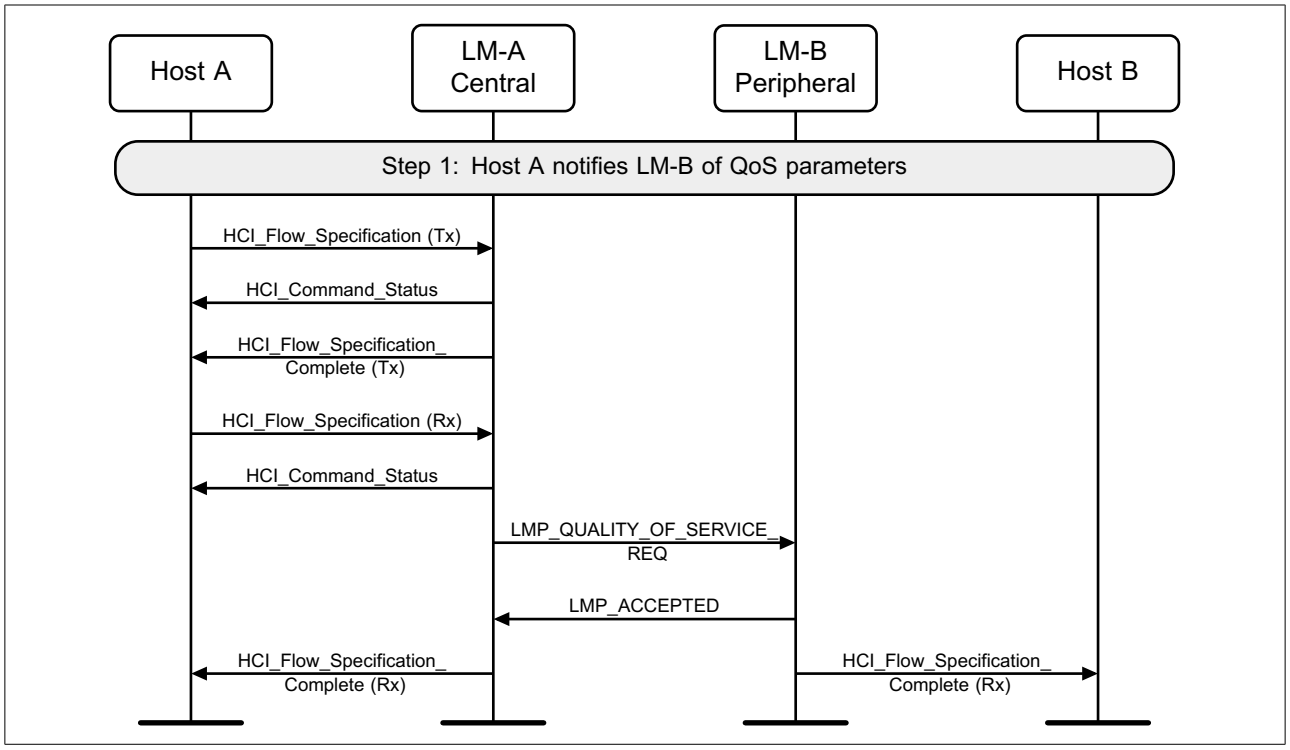



*Figure 4.44: QoS flow specification*

## 4.15 Switch role

The HCI_Switch_Role command can be used to explicitly switch the current Central / Peripheral role of the local device with the specified device.

**Step 1a:** The Central's Host (A) requests a role switch with a Peripheral. This will send the switch request, and the Peripheral will respond with the slot offset and accepted. (See Figure 4.45.)

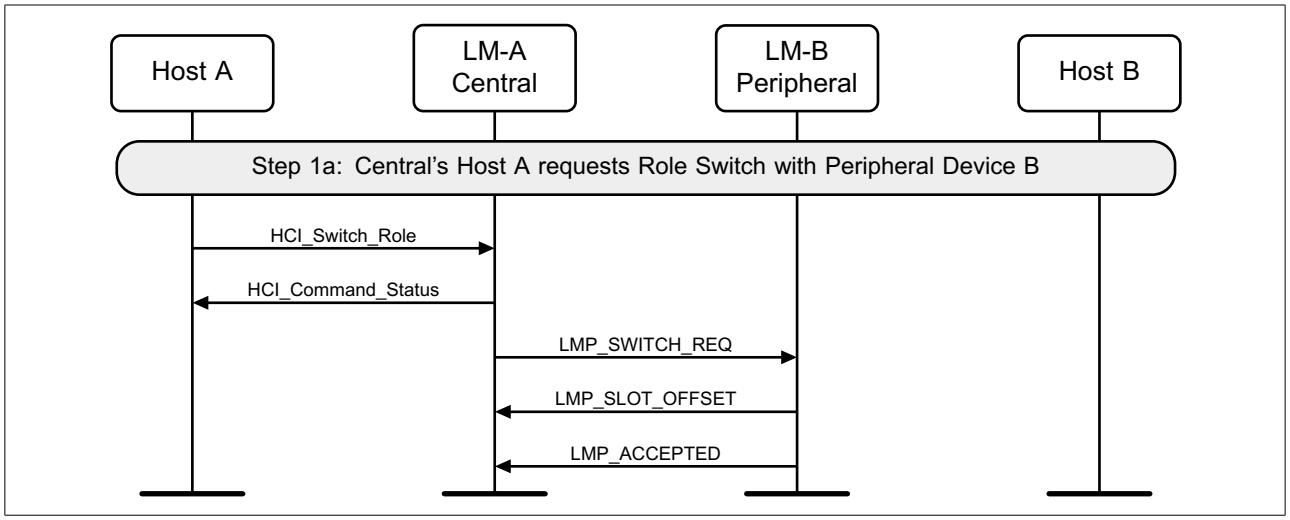



*Figure 4.45: Central requests role switch*

**Step 1b:** The Peripheral's Host (B) requests a role switch with a Central. This will send the slot offset and the switch request, and the Central will respond with a LMP_ACCEPTED PDU. (See Figure 4.46.)

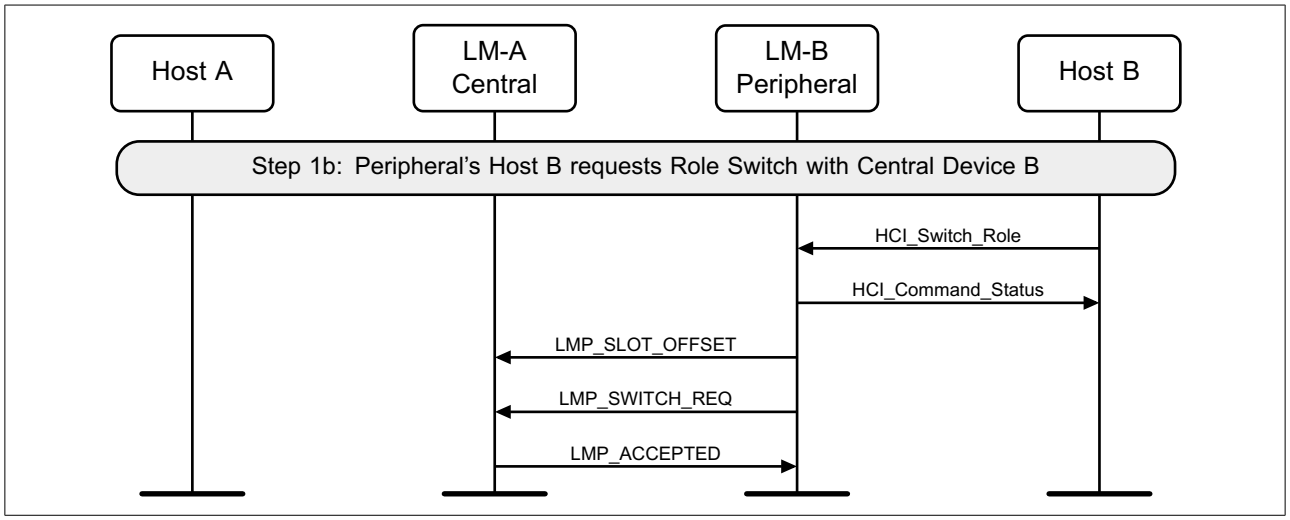



*Figure 4.46: Peripheral requests role switch*

**Step 2:** The role switch is performed by doing the TDD switch and piconet switch. Finally an HCI_Role_Change event is sent on both sides. (See Figure 4.47.)

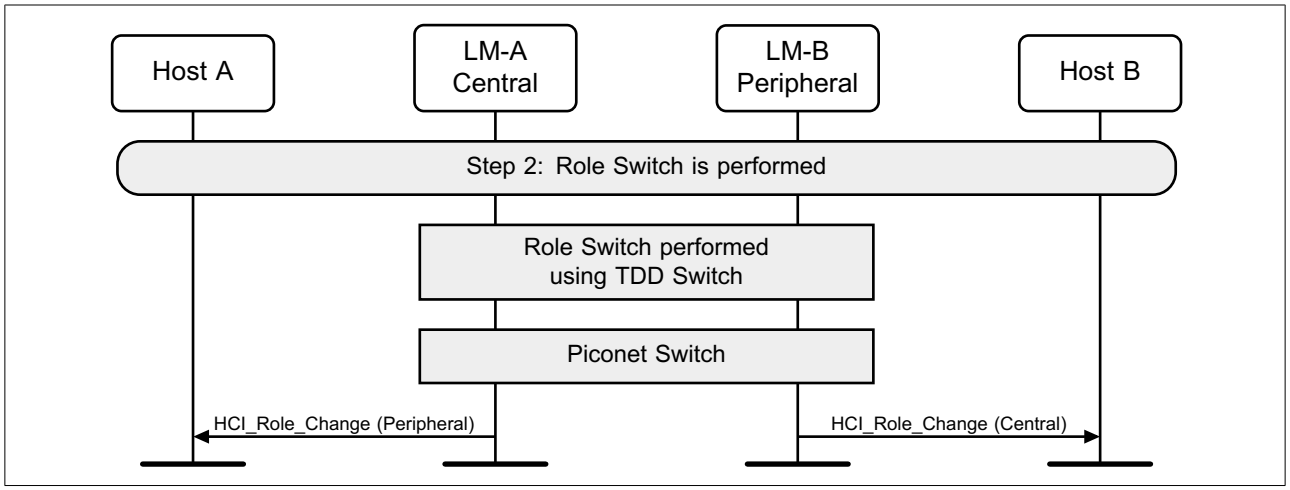



*Figure 4.47: Role switch is performed*

## 4.16 [This section is no longer used]

## 4.17 [This section is no longer used]

## 4.18 Slot availability mask

**Step 1:** The setup of SAM may be triggered by the Link Manager for MWS coexistence or topology management purposes. SAM setup may be triggered by HCI commands

as shown in Figure 4.48 or by the real-time signals (e.g. MWS_PATTERN_Index, FRAME_SYNC, etc.) from the Coexistence Logical Interface (see [Vol 7] Part A ).

Piconet Clock Adjustment may be performed to create more available slot pairs per MWS frame.

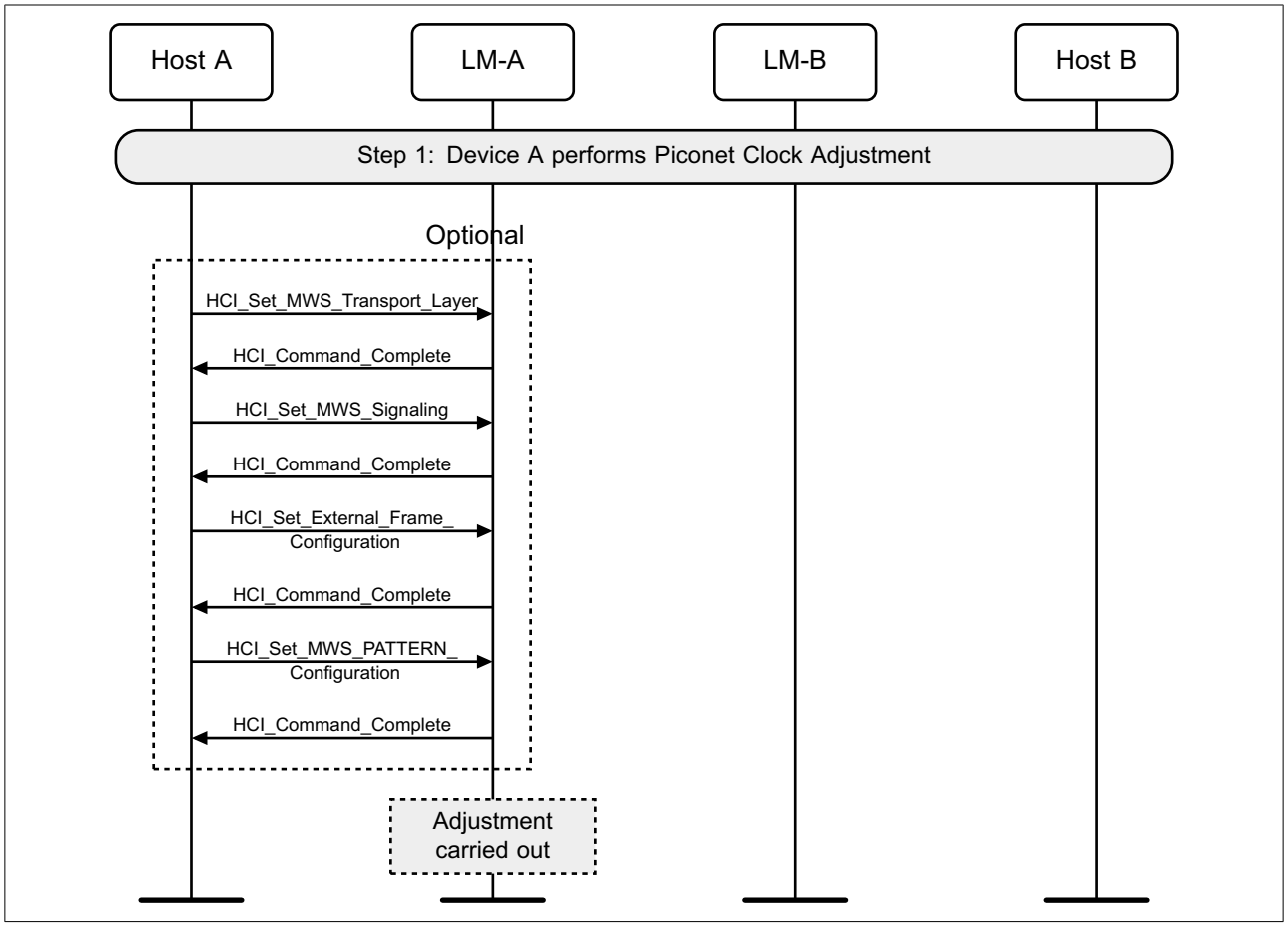



*Figure 4.48: SAM configuration setup on device A*

**Step 2:** The Link Manager on device A sends the SAM configuration to device B and then selects a SAM map (see Figure 4.49).

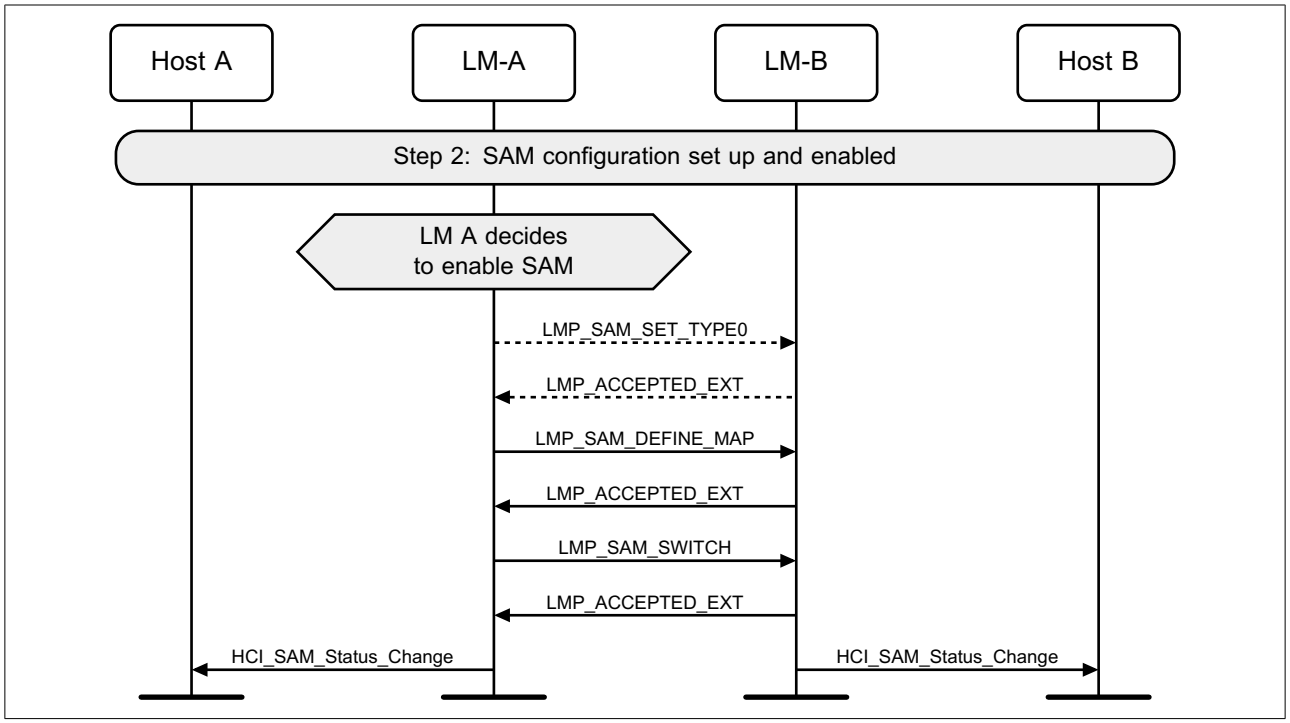



*Figure 4.49: SAM configuration transmitted to device B*

**Step 3:** Device A receives a new configuration (possibly over HCI), sends it to device B, and switches to using it (see Figure 4.50).

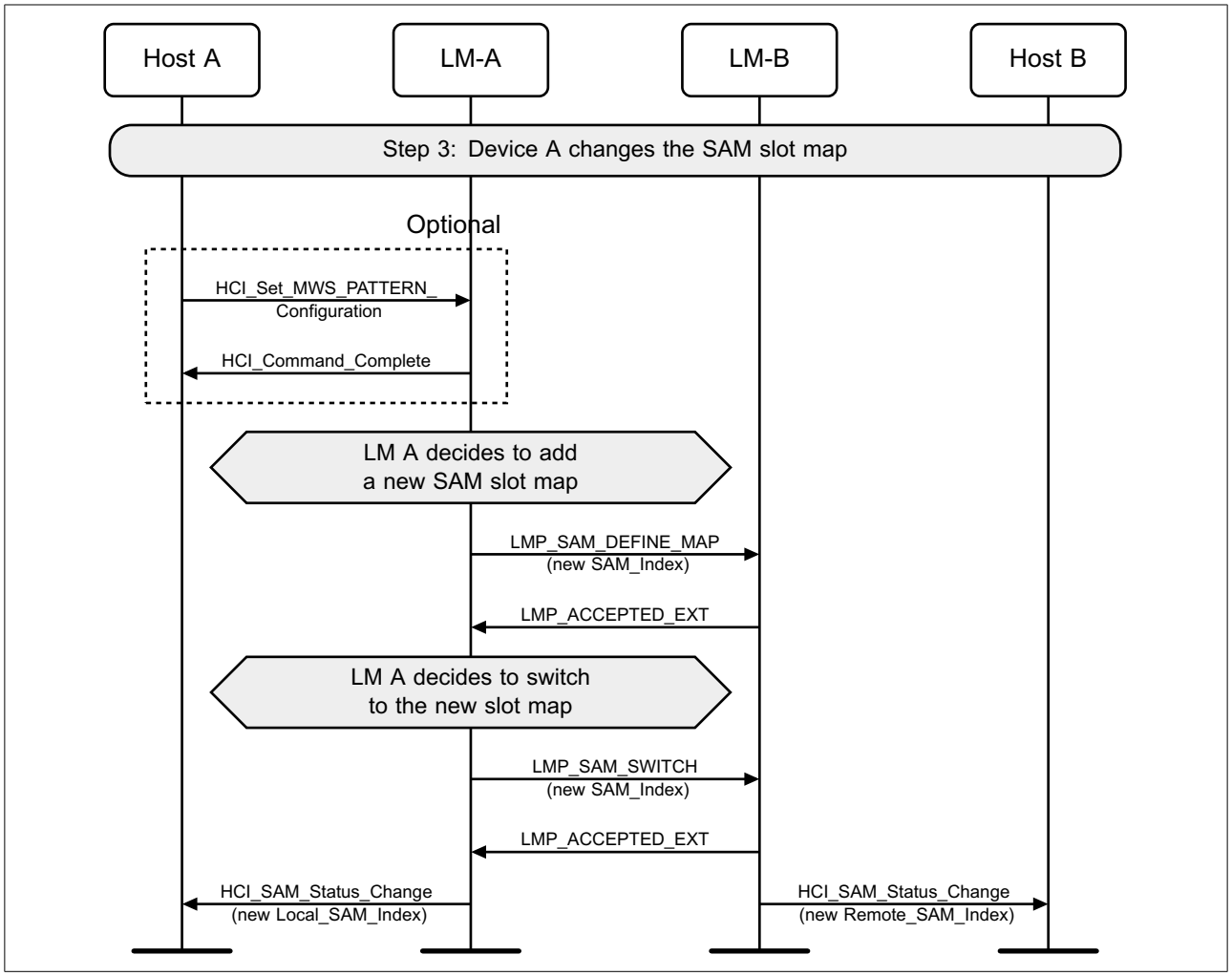



*Figure 4.50: SAM configuration change*

## 4.19 LMP transaction collision

The Link Managers of both the Central and Peripheral may initiate the same LMP transaction at the same time.

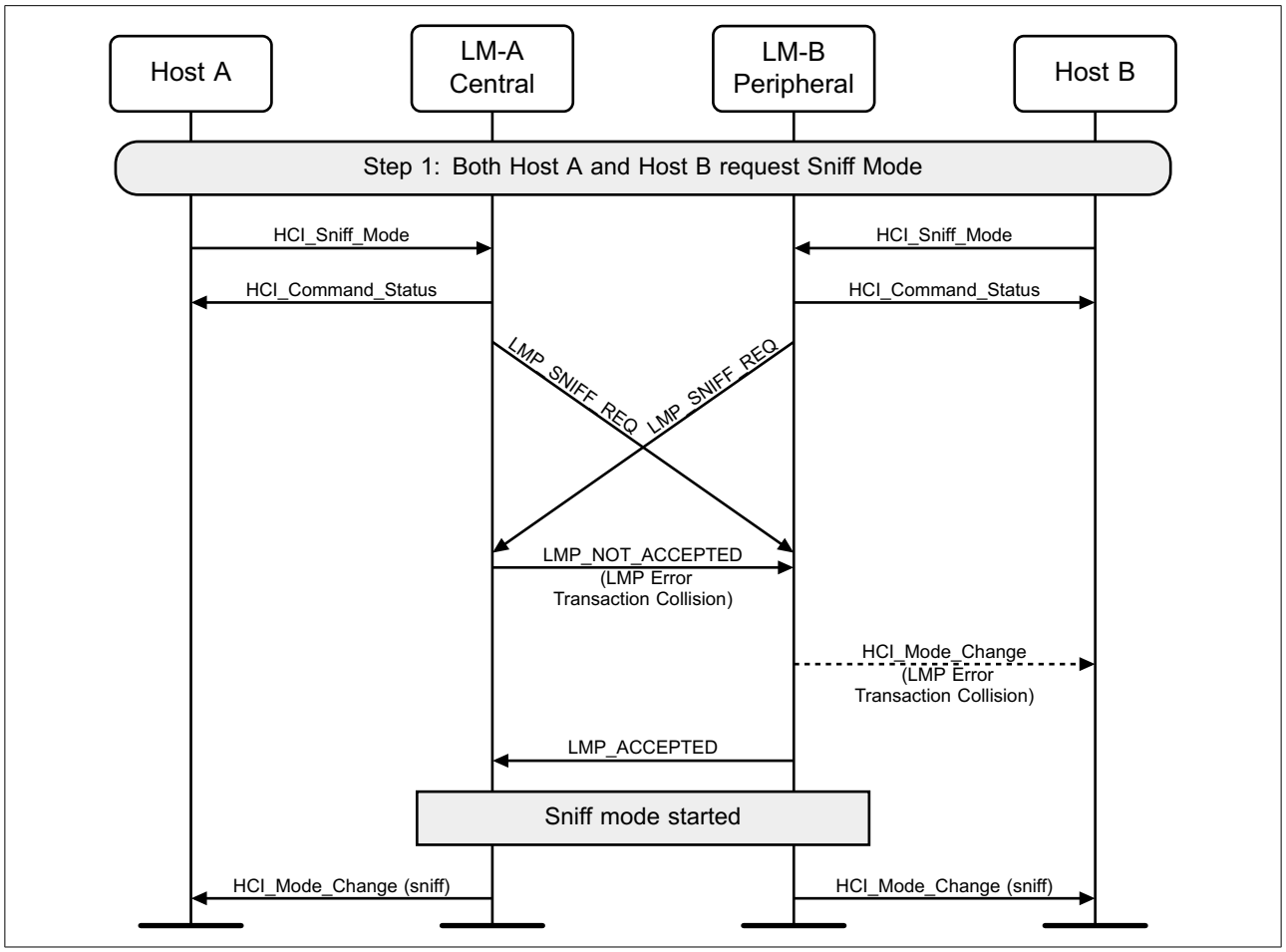



*Figure 4.51: LMP transaction collision*

# 5 SYNCHRONOUS CONNECTION ESTABLISHMENT AND DETACHMENT

## 5.1 Synchronous connection setup

Using the HCI_Setup_Synchronous_Connection command, a Host can add a synchronous logical channel to the link. A synchronous logical link can be provided by creating a SCO or an eSCO logical transport.

An ACL connection must be established before a synchronous connection can be created.

**Step 1a:** Central requests a synchronous connection with a device. (See Figure 5.1.)

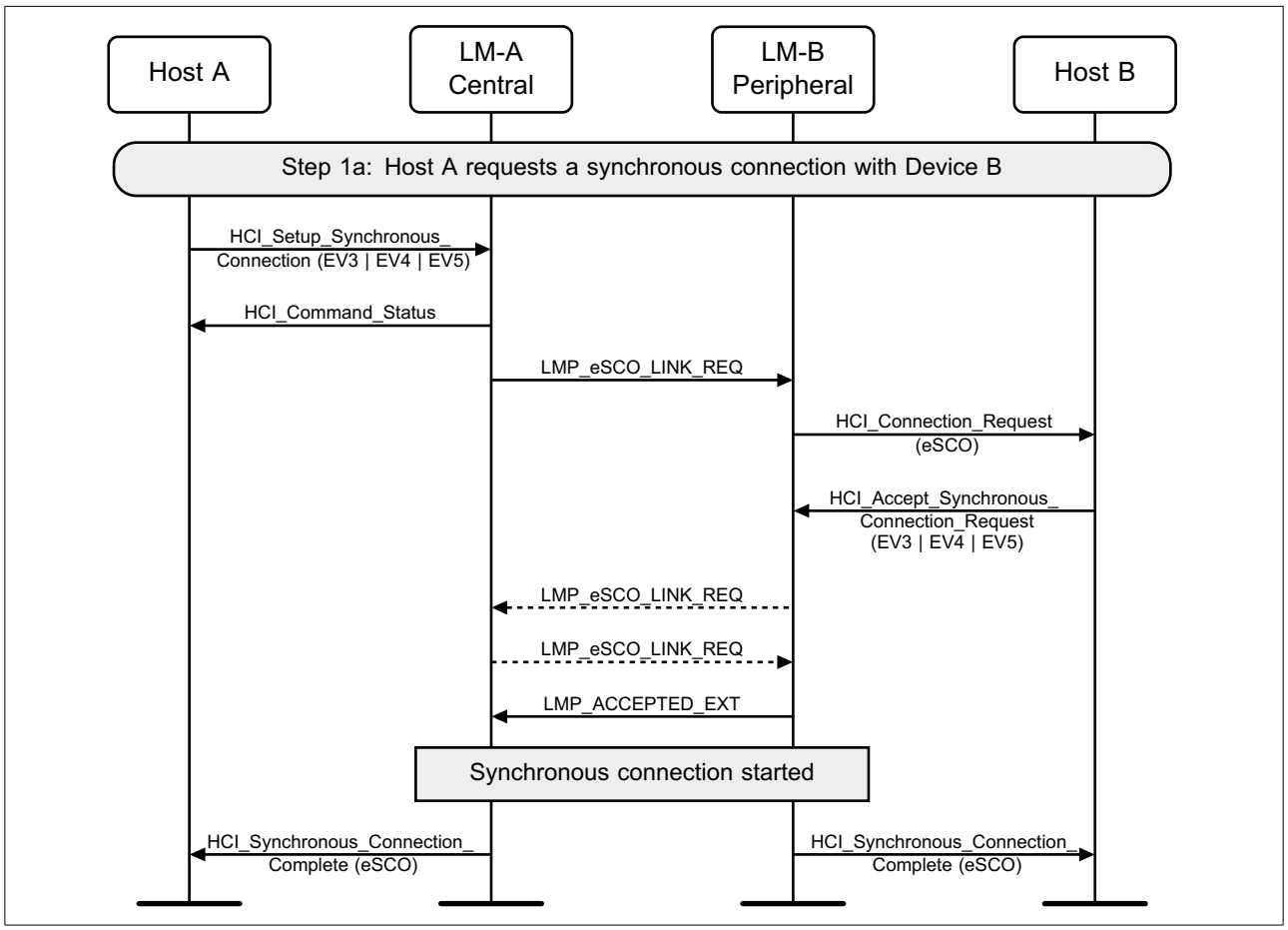



*Figure 5.1: Central requests synchronous EV3, EV4, or EV5 connection*

**Step 1b:** Peripheral requests a synchronous connection with a device. (See Figure 5.2.)

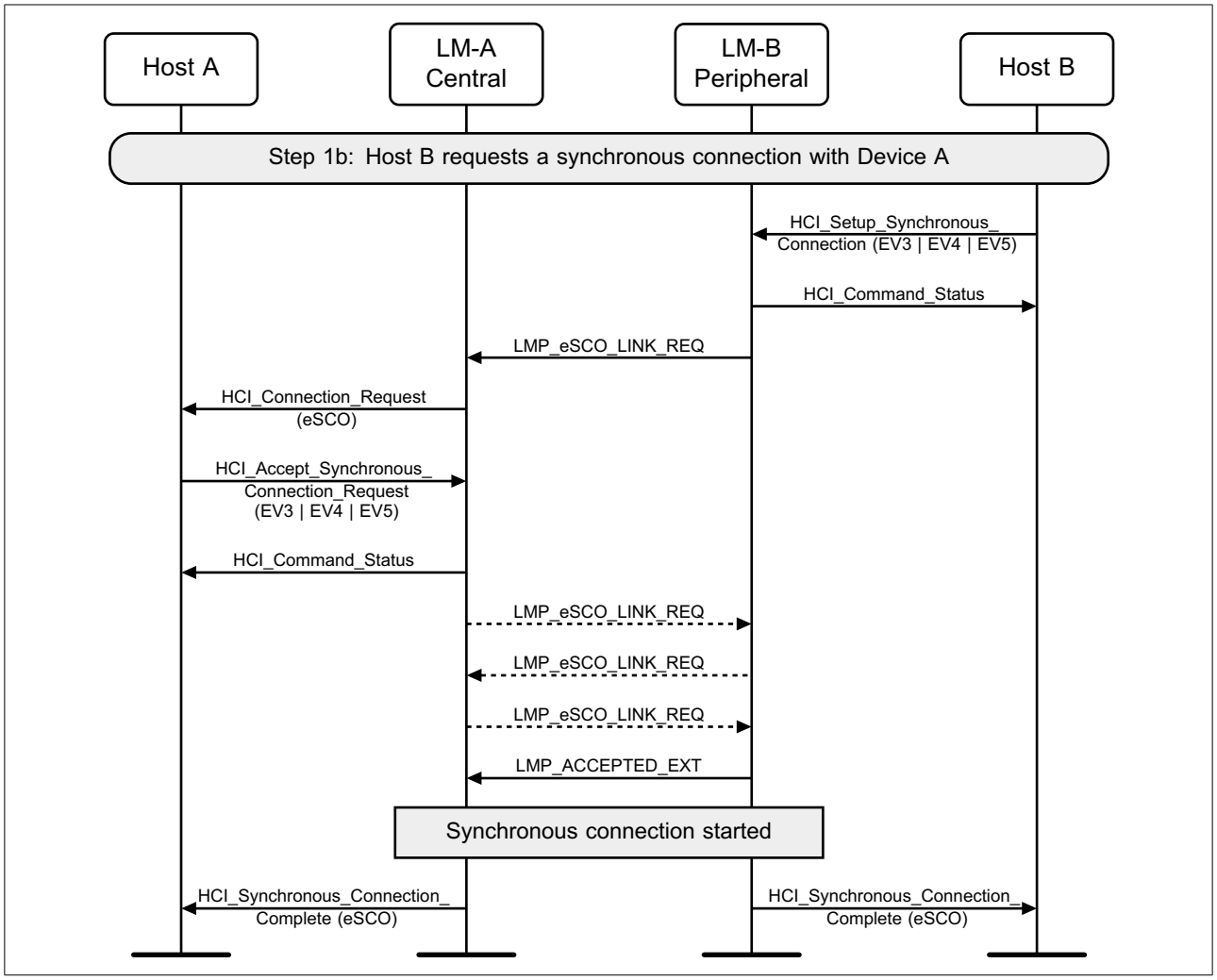



*Figure 5.2: Peripheral requests synchronous EV3, EV4, or EV5 connection*

**Step 1c:** Central requests a SCO connection with a device. (See Figure 5.3.)

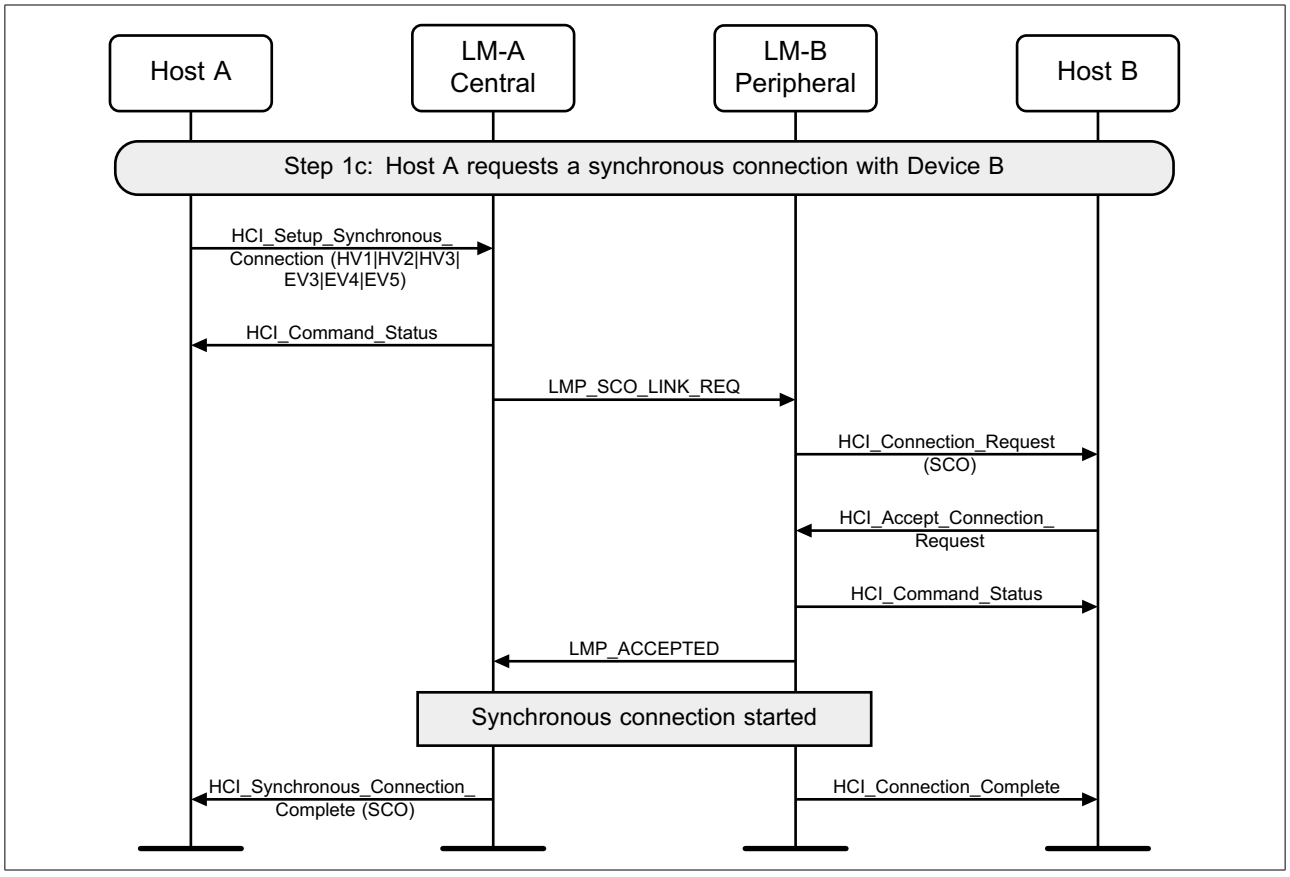



*Figure 5.3: Central requests synchronous connection using SCO*

**Step 1d:** Central requests a SCO connection with a device. (See Figure 5.4.)

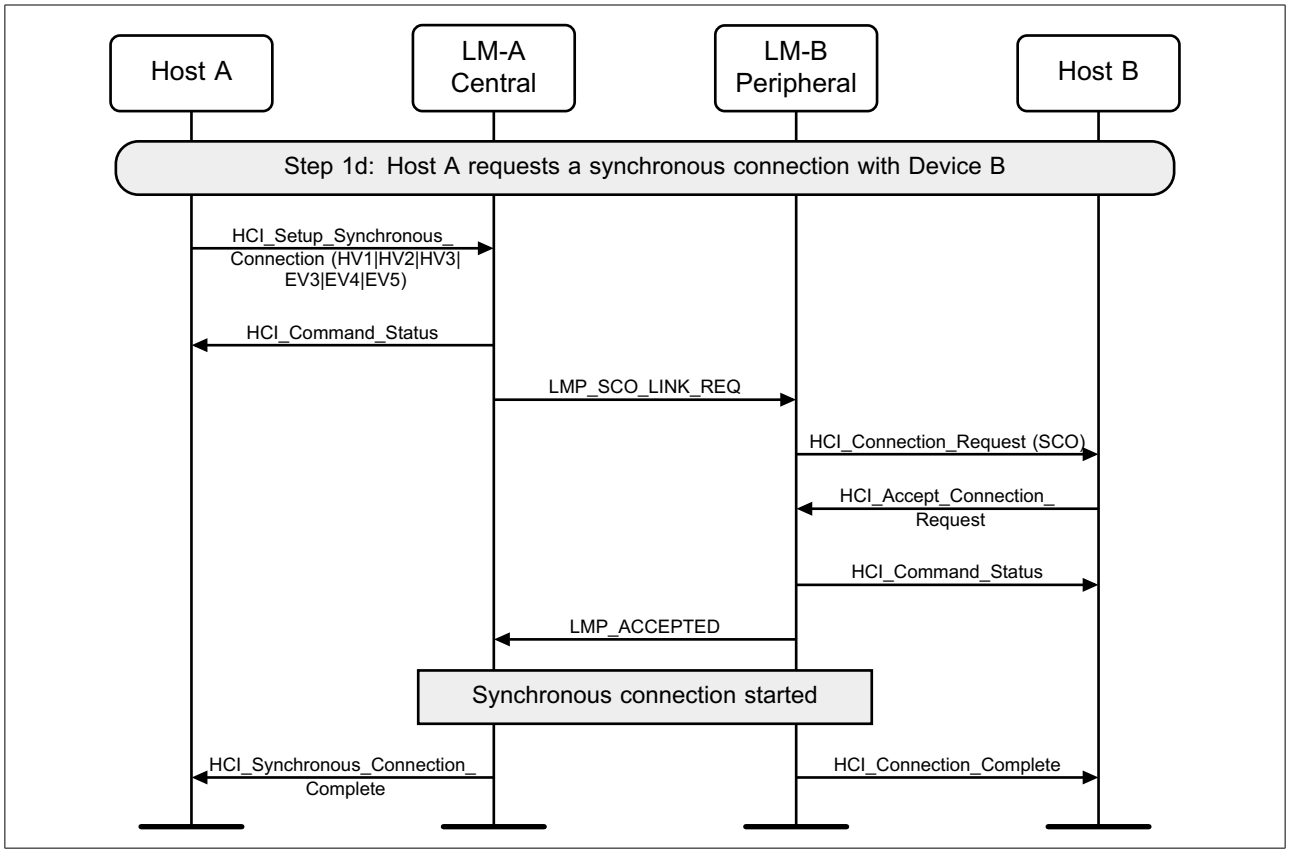



*Figure 5.4: Central requests synchronous connection with legacy Peripheral*

**Step 1e:** Host device requests a SCO connection with a device. (See Figure 5.5.)

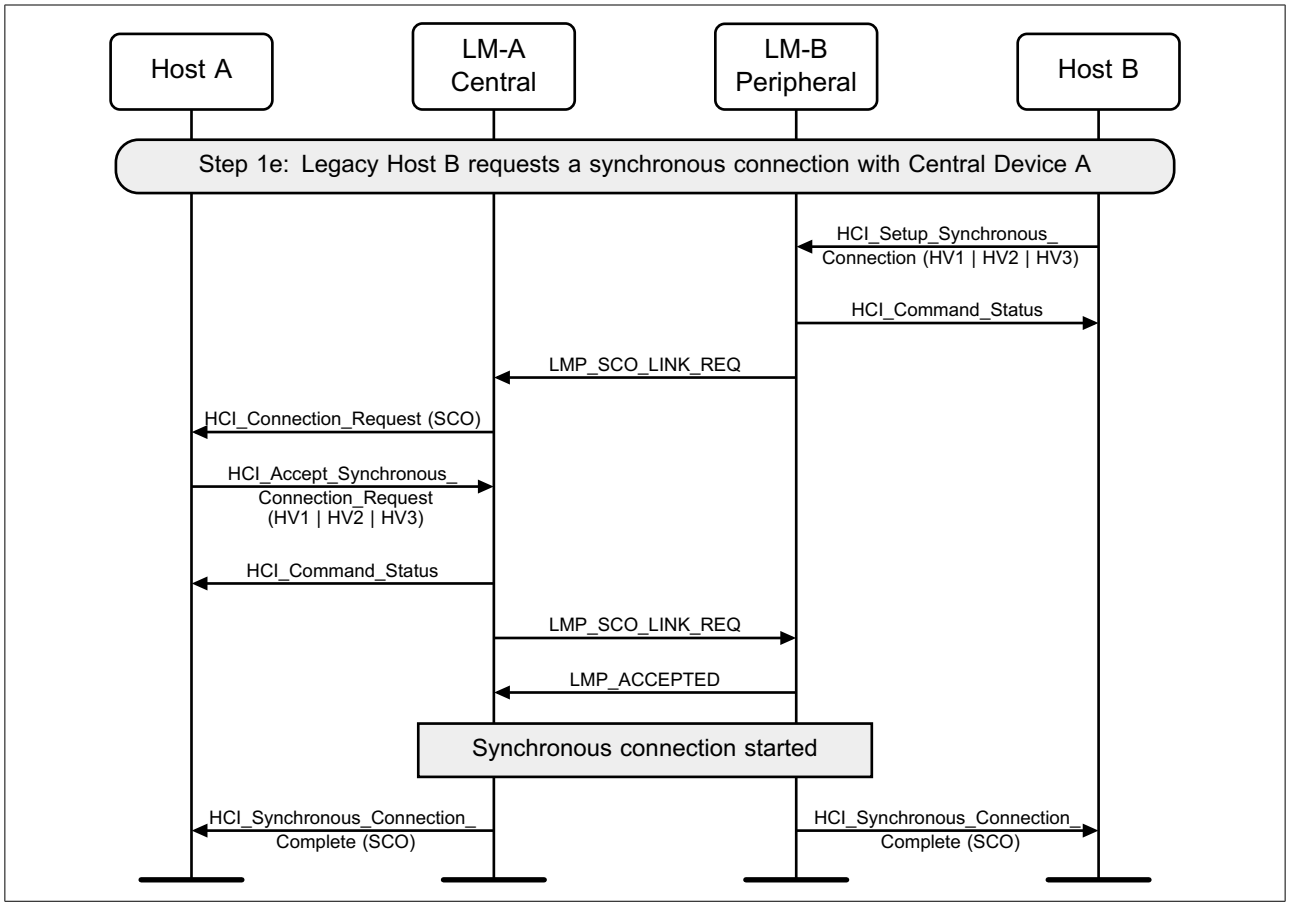



*Figure 5.5: Any device that supports only SCO connections requests a synchronous connection with a device*

**Step 2a:** Central renegotiates eSCO connection (see Figure 5.6.)

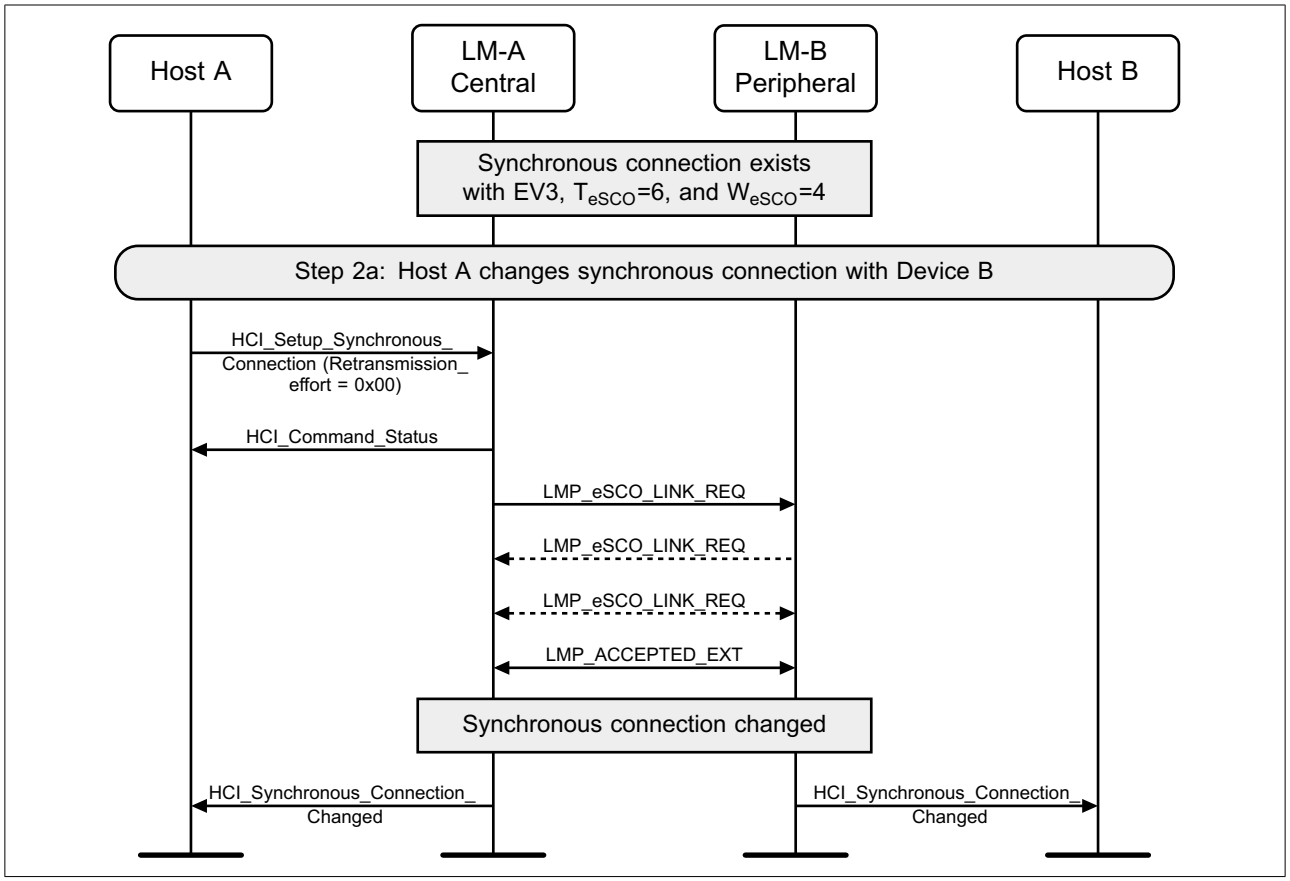



*Figure 5.6: Central renegotiates eSCO connection*

**Step 2b:** Peripheral renegotiates eSCO connection (see Figure 5.7.)

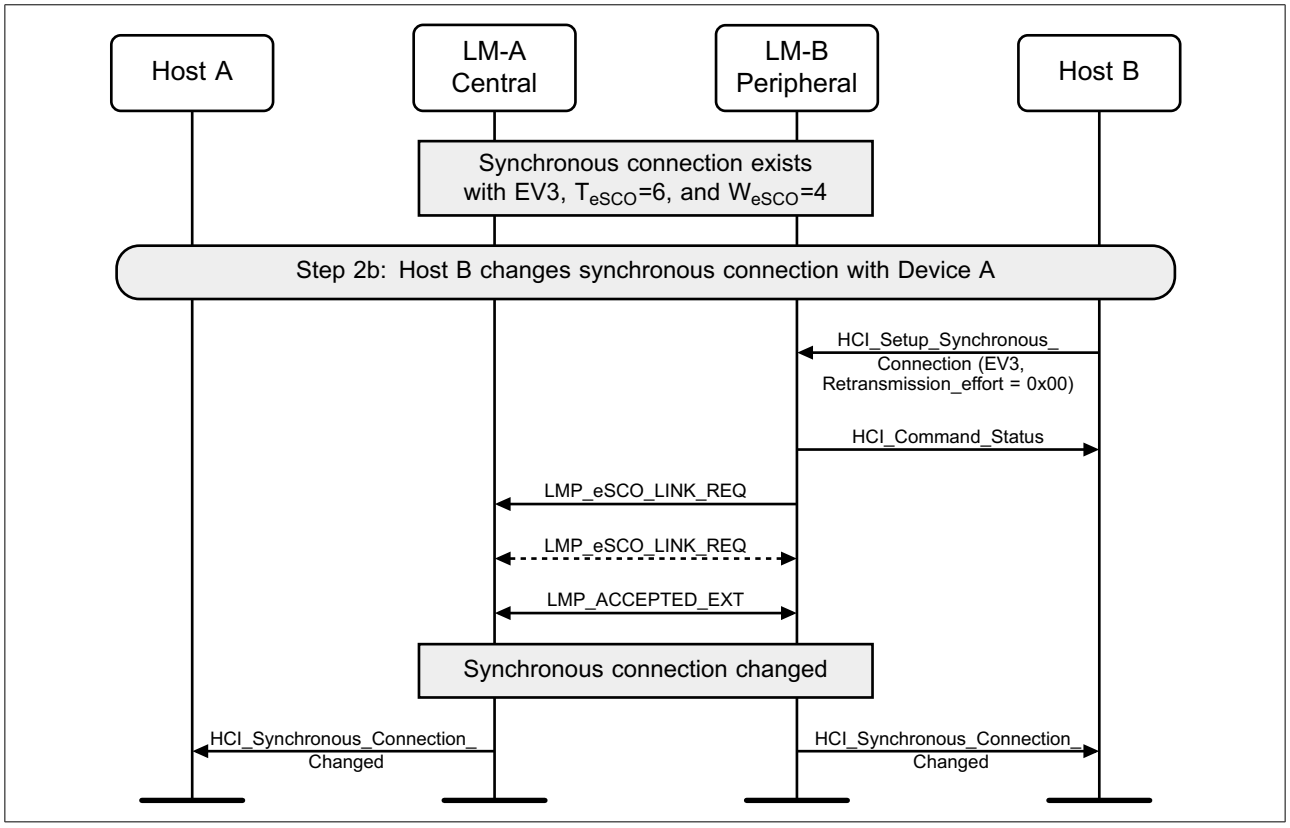



*Figure 5.7: Peripheral renegotiates eSCO connection*

**Step 3a:** eSCO disconnection. (See Figure 5.8.)

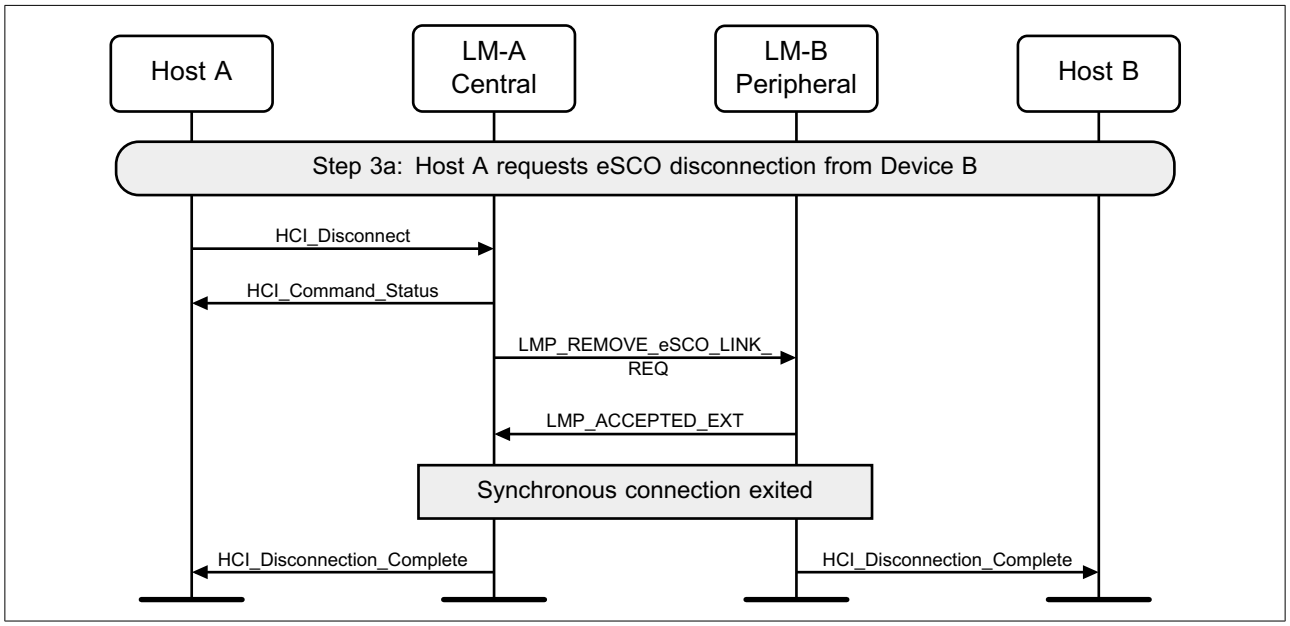



*Figure 5.8: Synchronous disconnection of eSCO connection*

**Step 3b:** SCO disconnection. (See Figure 5.9.)

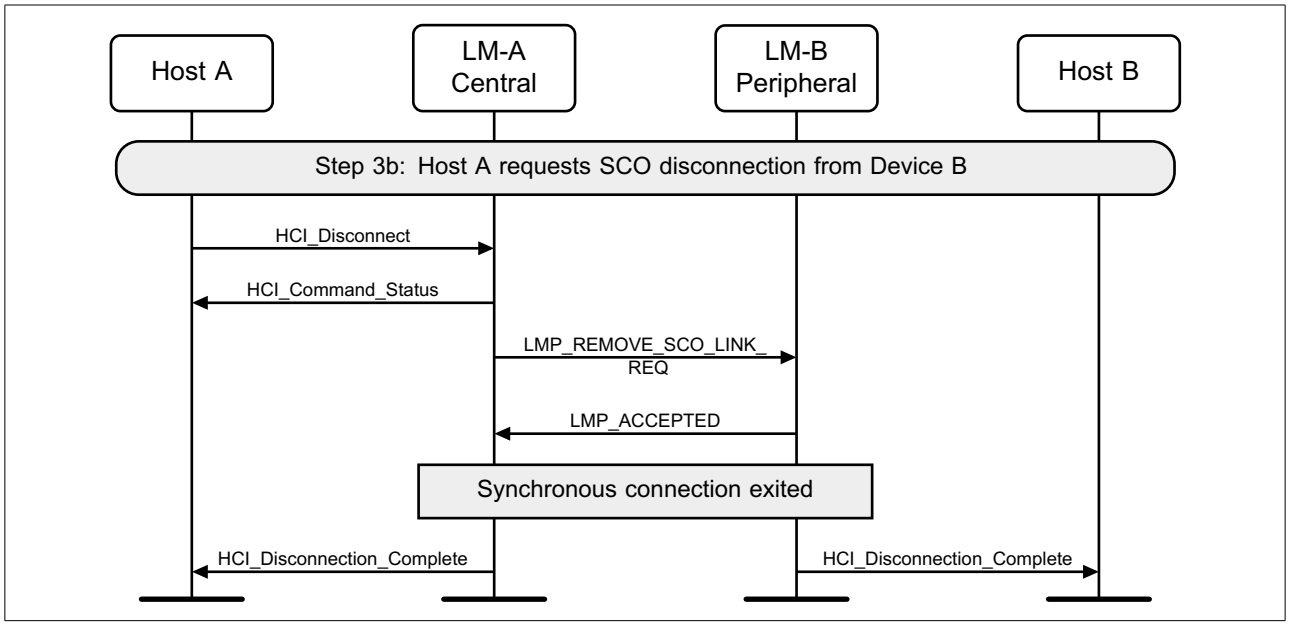



*Figure 5.9: Synchronous disconnection of SCO connection*

## 5.2 Synchronous connection setup with enhanced synchronous commands

Using the HCI_Enhanced_Setup_Synchronous_Connection command, a Host can add a synchronous logical channel to the link. A synchronous logical link can be provided by creating a SCO or an eSCO logical transport.

An ACL connection must be established before a synchronous connection can be created.

**Step 1a:** Central requests a synchronous connection with a Peripheral.

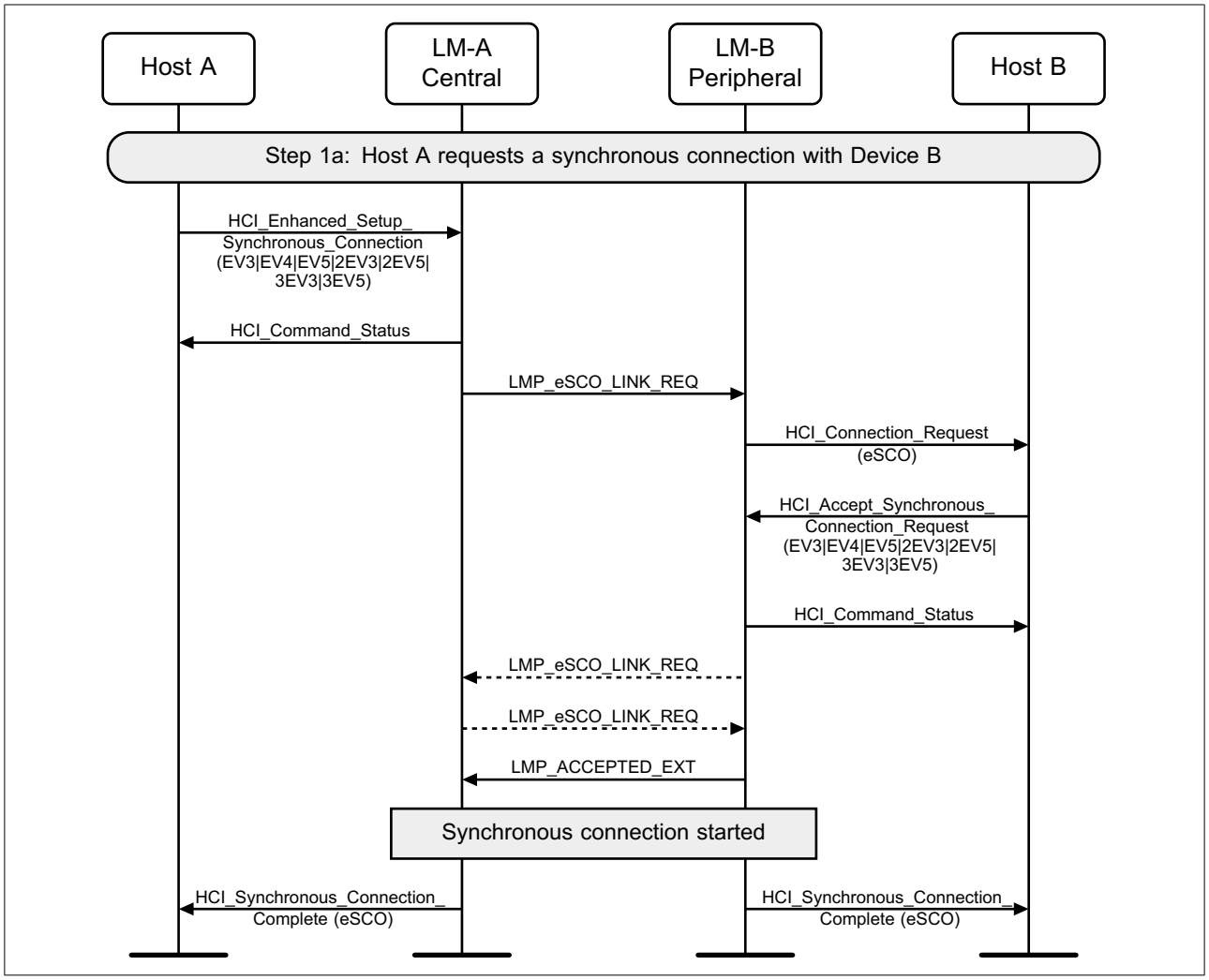



*Figure 5.10: Central requests synchronous connection (EV3, EV4, EV5, 2-EV3, 2-EV5, 3-EV3, or 3-EV5)*

**Step 1b:** Peripheral requests a synchronous connection with a Central.

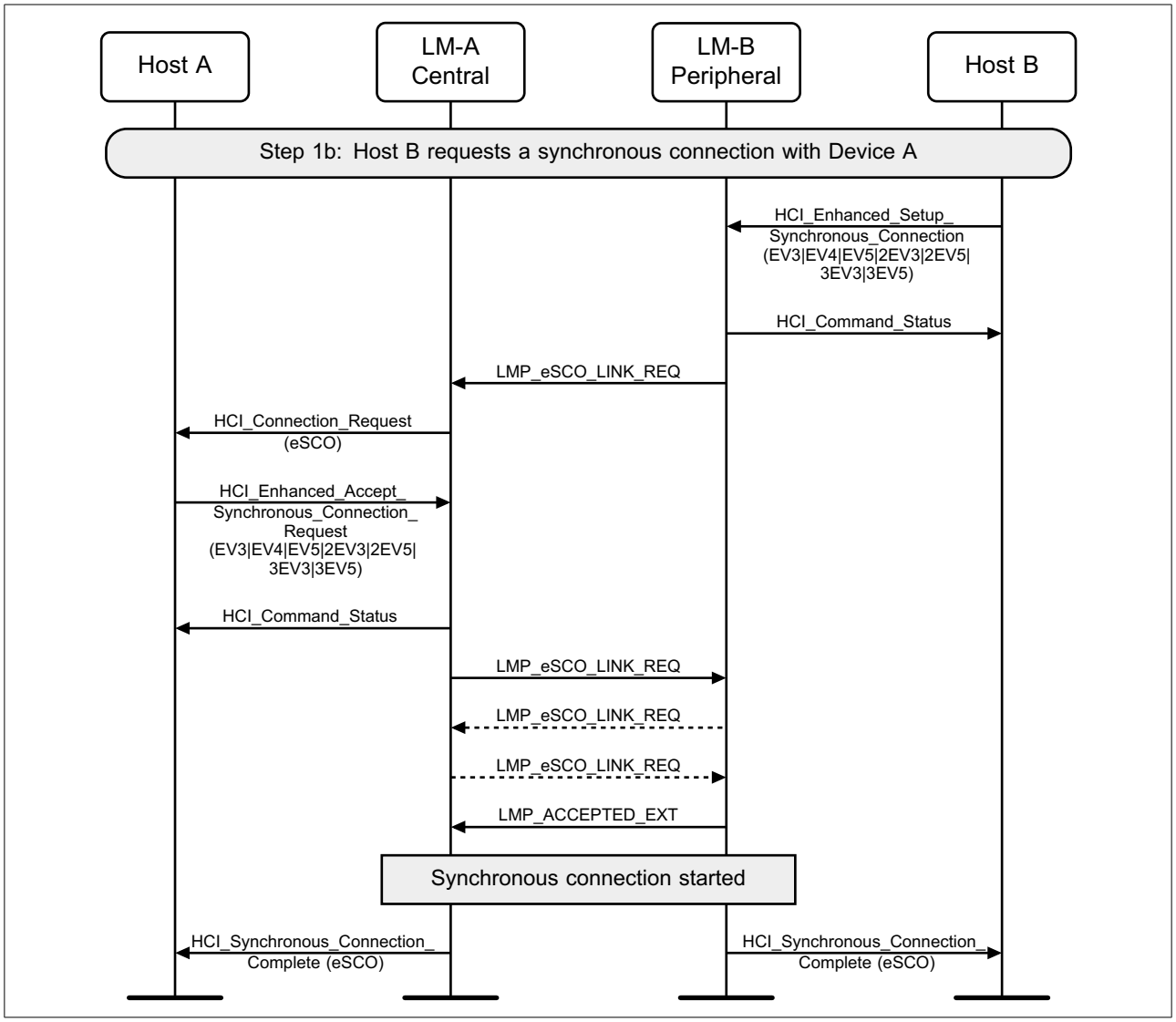



*Figure 5.11: Peripheral requests synchronous connection (EV3, EV4, EV5, 2-EV3, 2-EV5, 3-EV3, or 3-EV5)*

**Step 1c:** Central requests a SCO connection with a Peripheral.

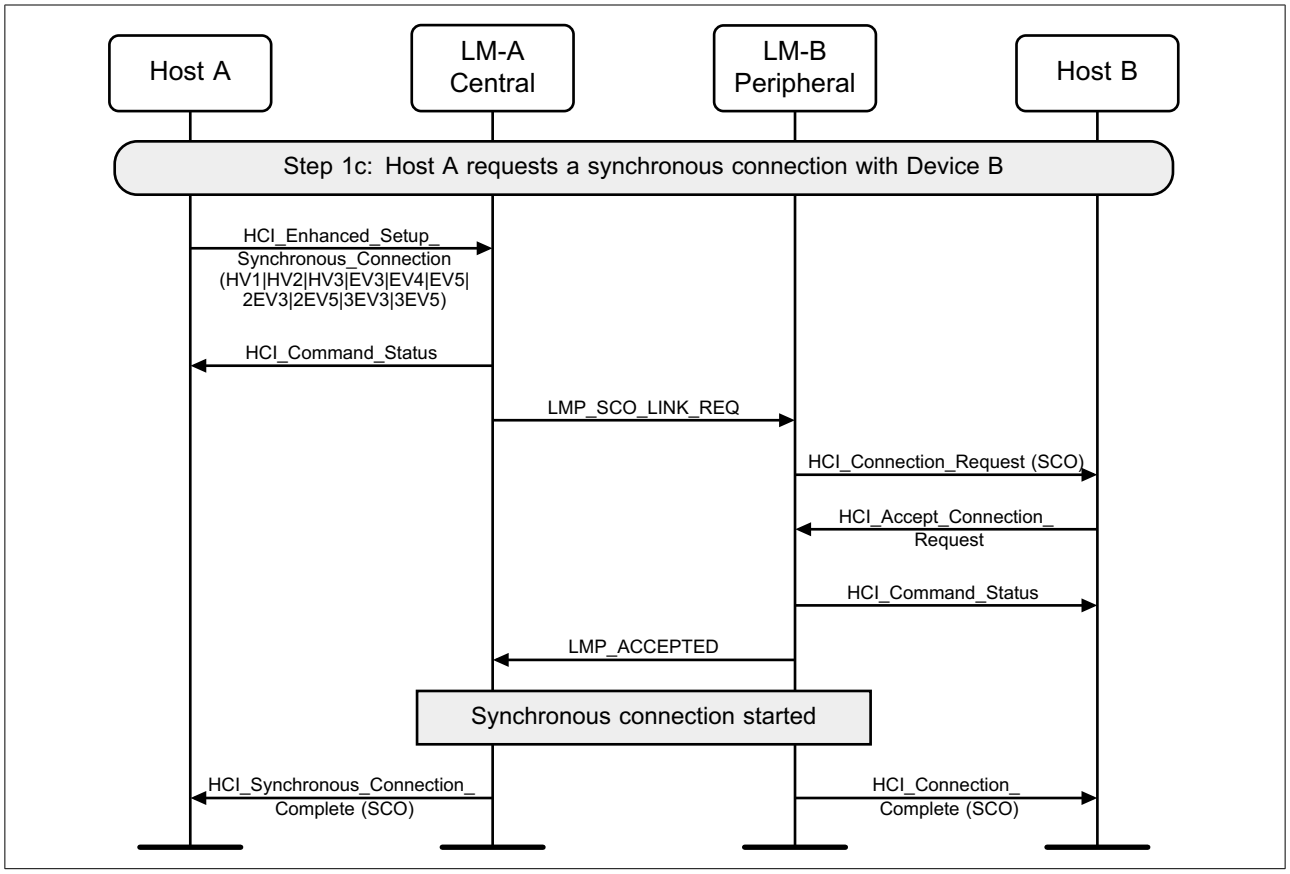



*Figure 5.12: Central requests synchronous connection (HV1, HV2, HV3, EV3, EV4, EV5, 2EV3, 2EV5, 3EV3, or 3EV5)*

**Step 1d:** Peripheral requests a SCO connection with a Central.

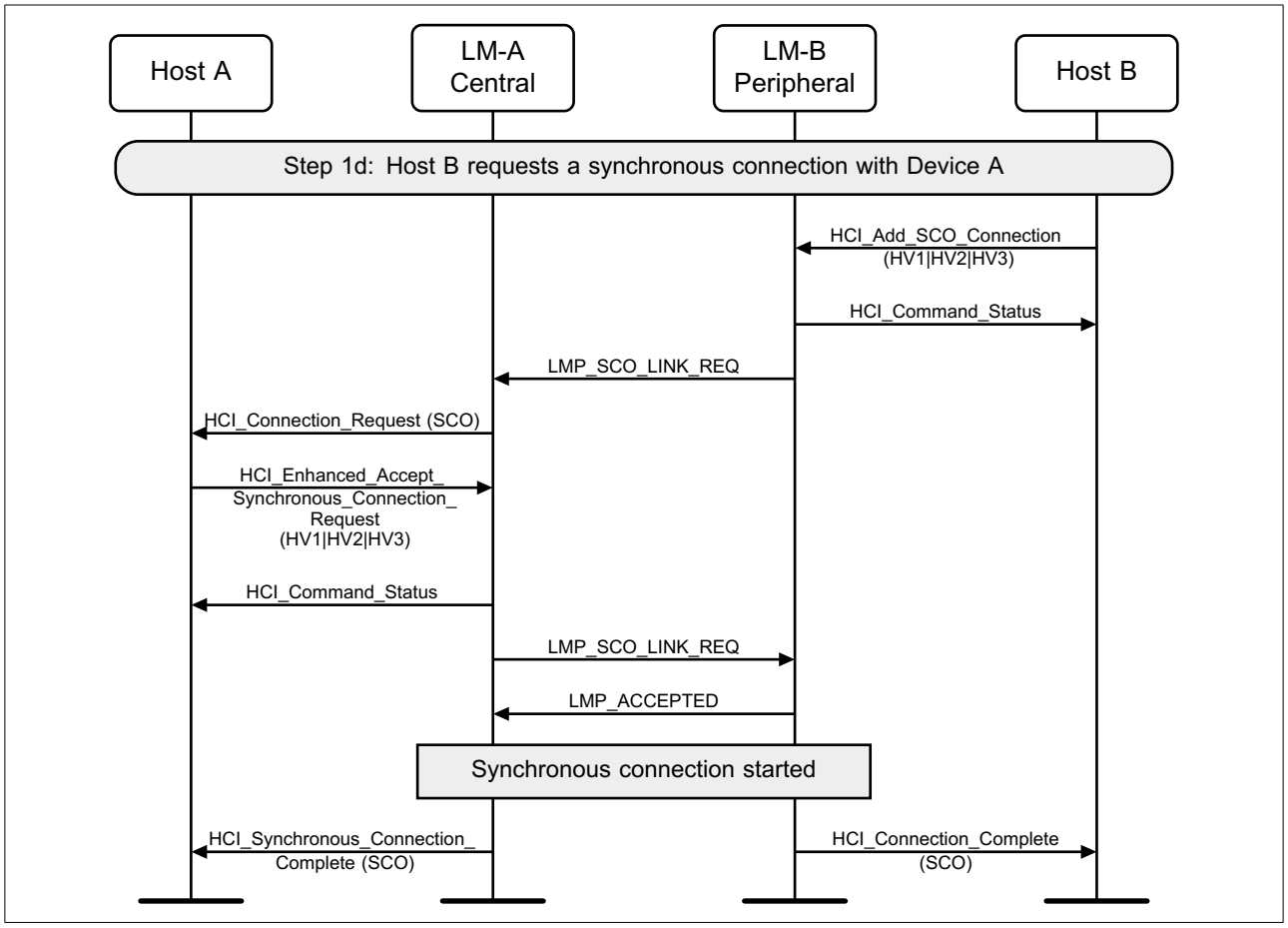



*Figure 5.13: Peripheral requests synchronous connection (HV1, HV2, or HV3)*

**Step 2a:** Central renegotiates eSCO connection.

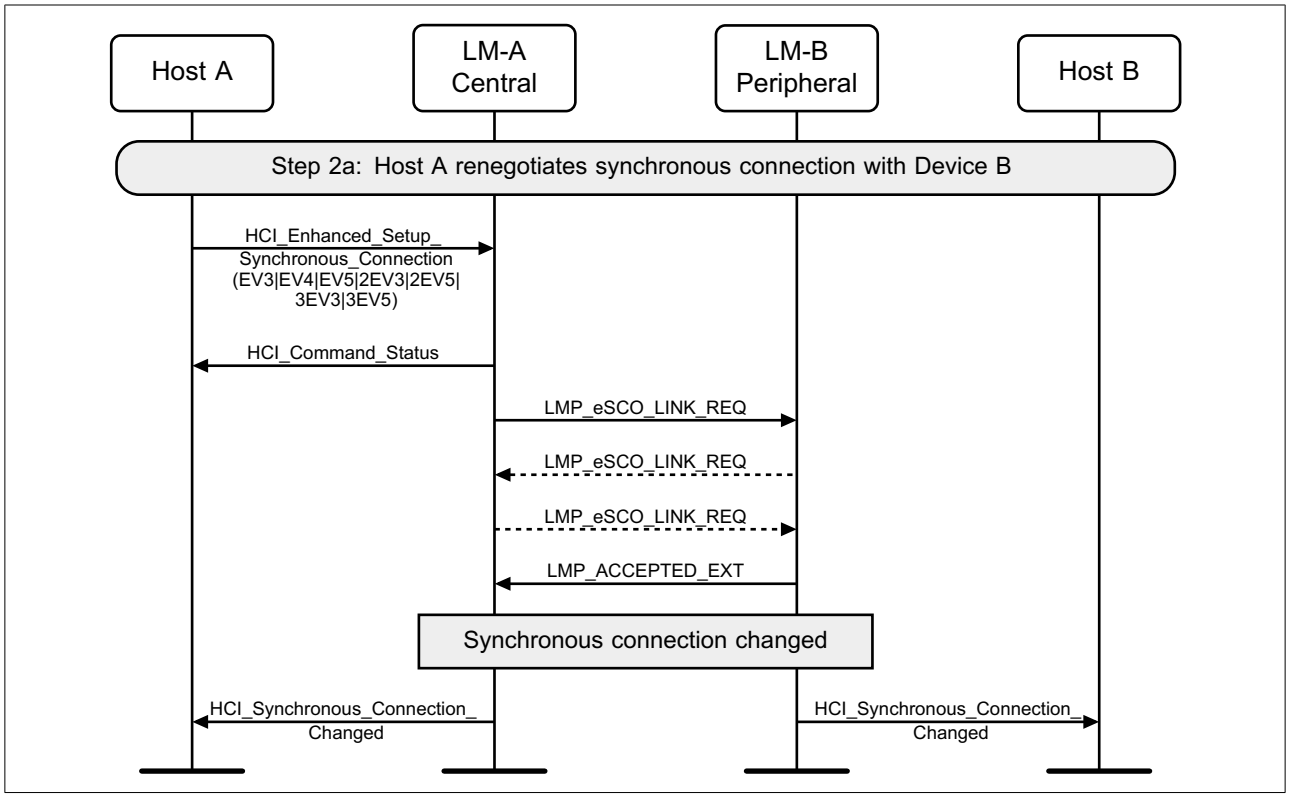



*Figure 5.14: Central renegotiates synchronous connection parameter change*

**Step 2b:** Peripheral renegotiates eSCO connection.

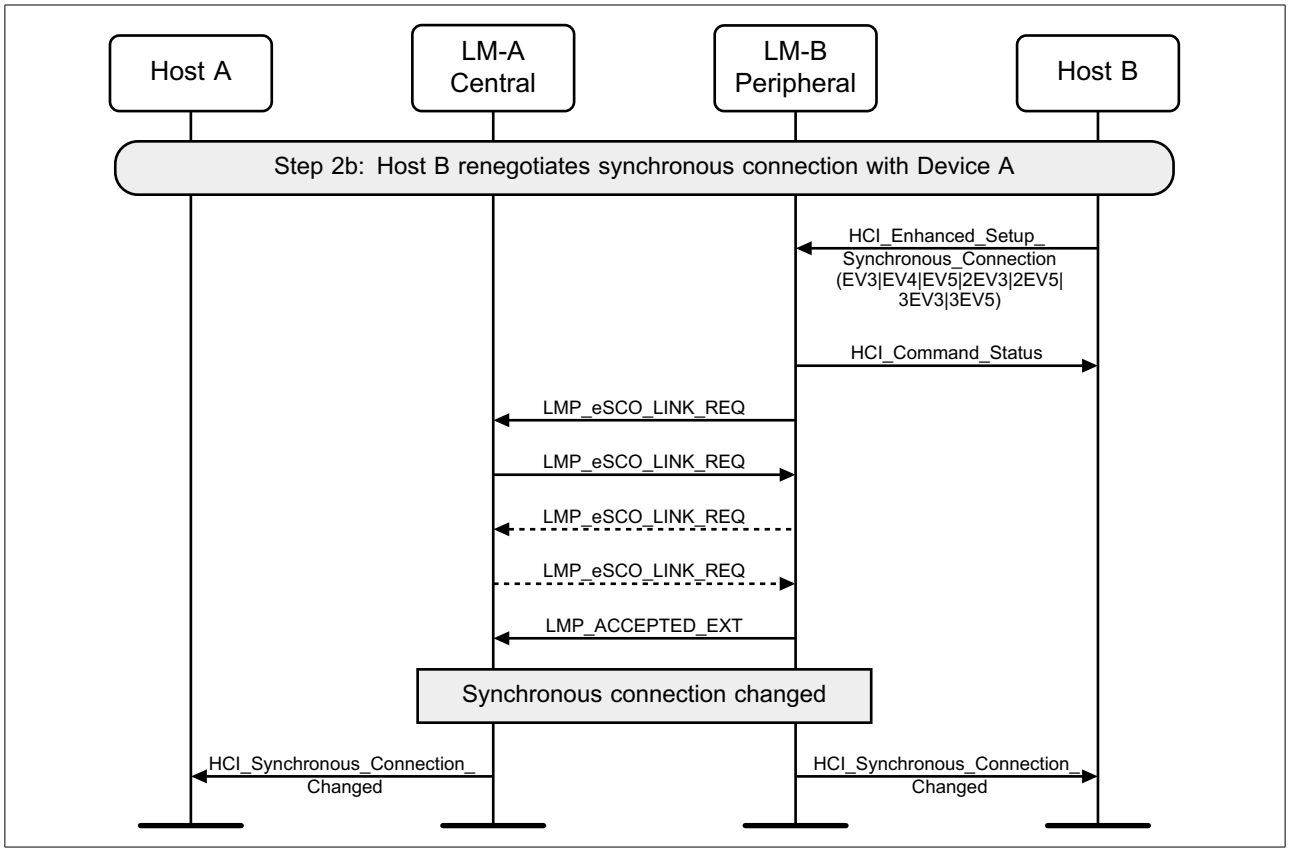



*Figure 5.15: Peripheral renegotiates synchronous connection parameter change*

# 6 SNIFF AND HOLD MODES

Entry into Sniff mode or Hold mode requires an established ACL connection.

## 6.1 Sniff mode

The HCI_Sniff_Mode command is used to enter Sniff mode. The HCI_Exit_Sniff_Mode command is used to exit Sniff mode.

**Step 1:** Host requests to enter Sniff mode. Multiple LMP_SNIFF_REQ PDUs may be sent as the parameters for Sniff mode are negotiated. (See Figure 6.1.)

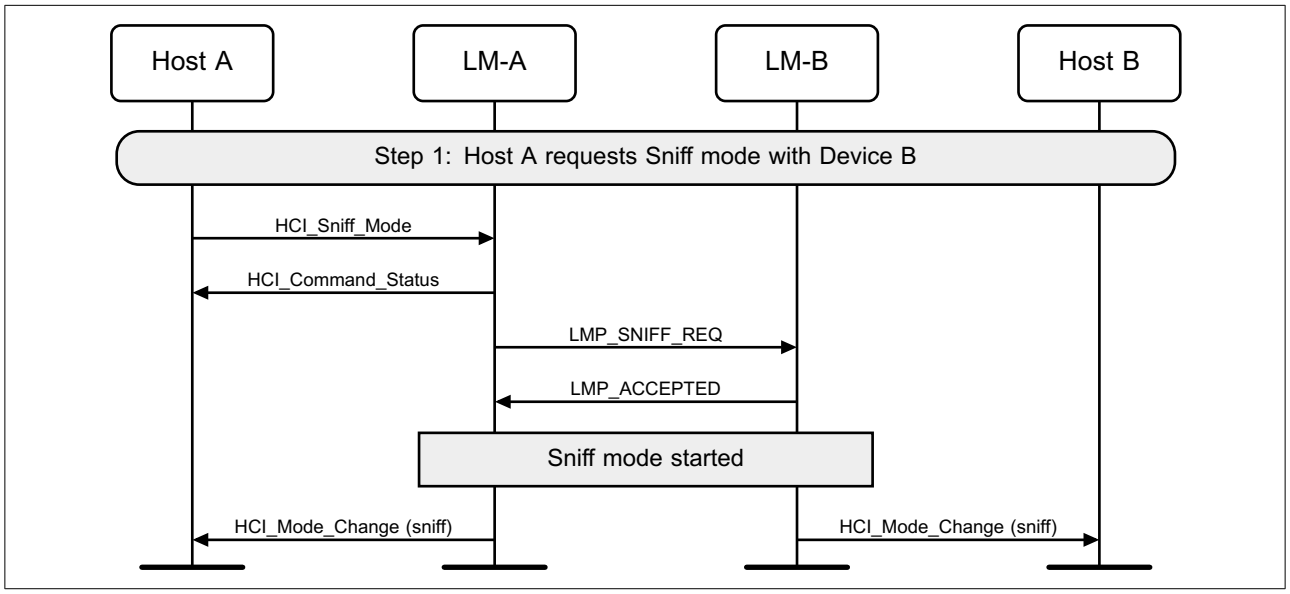



*Figure 6.1: Sniff mode request*

**Step 2:** Host requests to exit Sniff mode. (See Figure 6.2.)

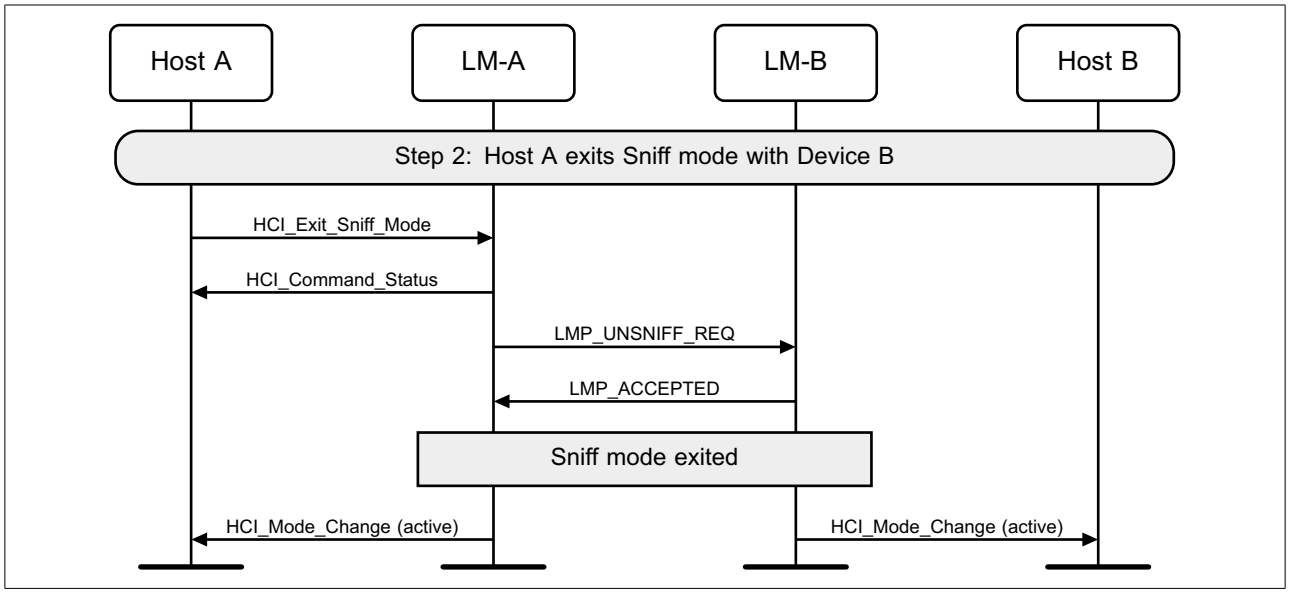



*Figure 6.2: Exit Sniff mode request*

## 6.2 Hold mode

The HCI_Hold_Mode command can be used to place a device into Hold mode. The Controller may do this by either negotiating the Hold mode parameters or forcing Hold mode. Hold mode will automatically end after the negotiated length of time.

**Step 1a:** A Host requests Hold mode. (See Figure 6.3.)

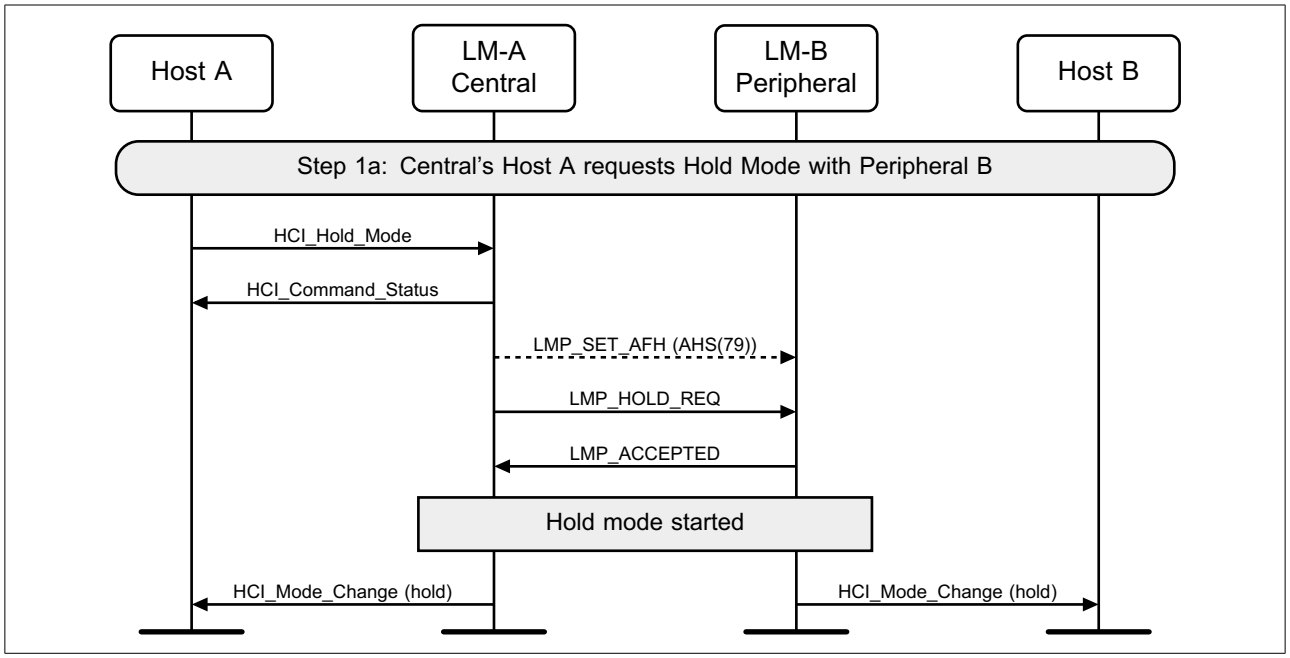



*Figure 6.3: Hold request*

**Step 1b:** A Host may force Hold mode. (See Figure 6.4.)

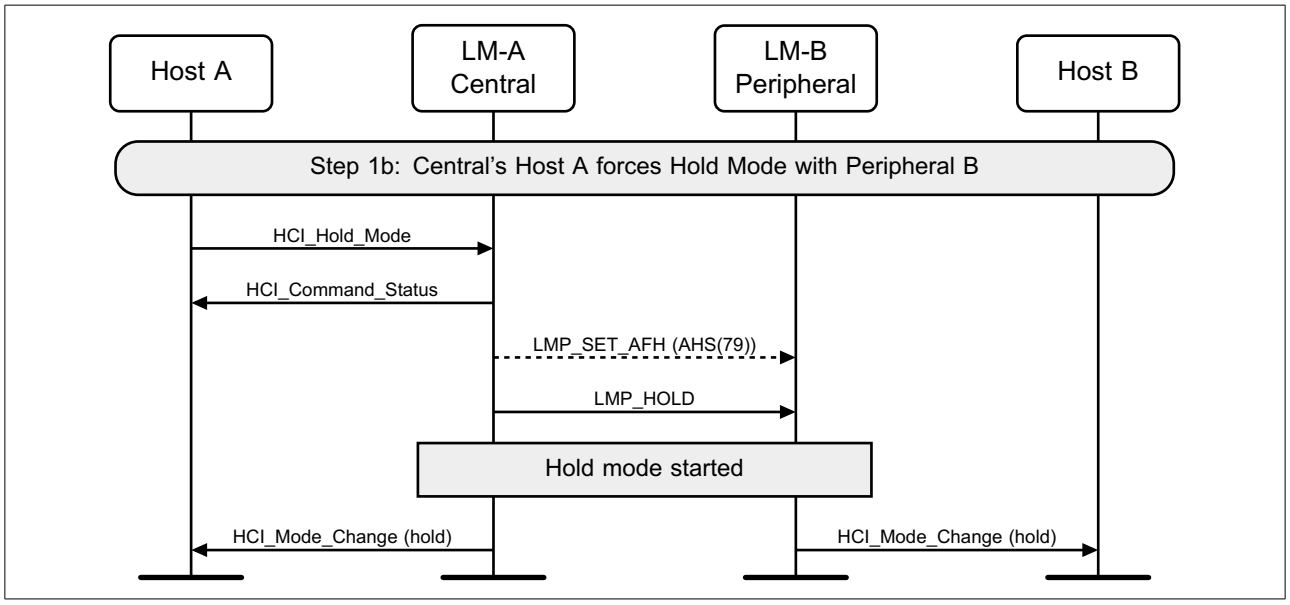



*Figure 6.4: Central forces Hold mode*

**Step 1c:** A Peripheral requests Hold mode. (See Figure 6.5.)

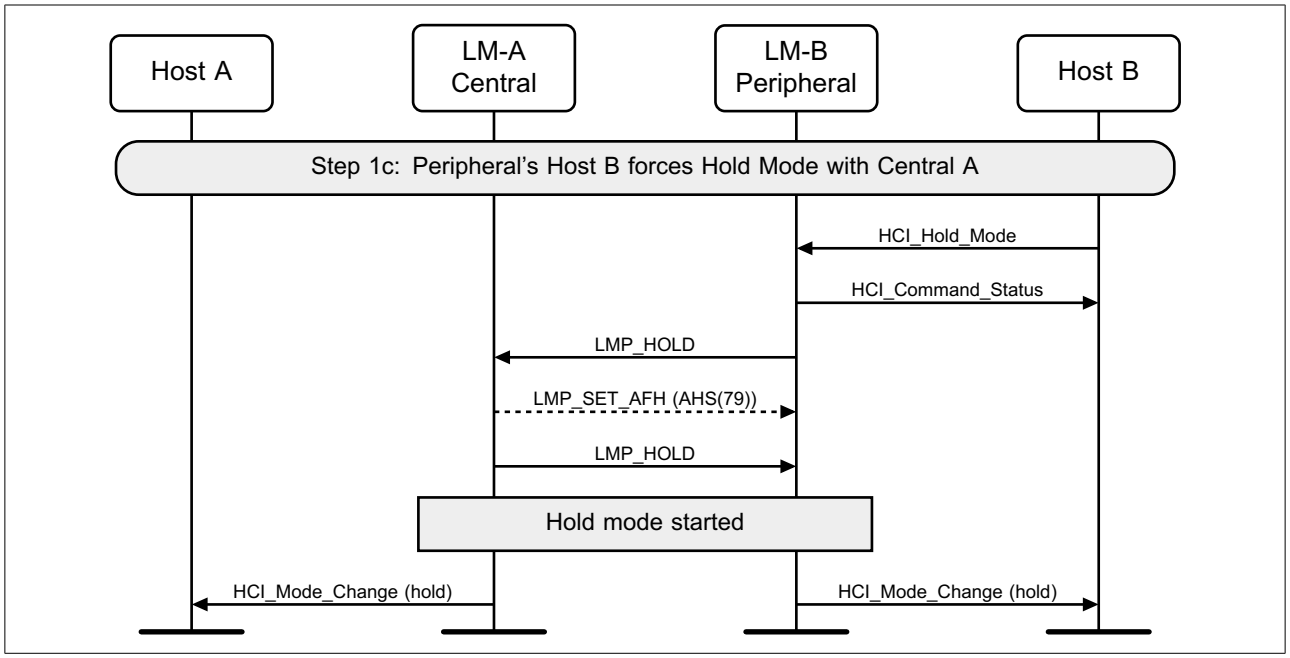



*Figure 6.5: Peripheral forces Hold mode*

**Step 2:** When Hold mode completes the Hosts are notified using the HCI_Mode_Change event. (See Figure 6.6.)

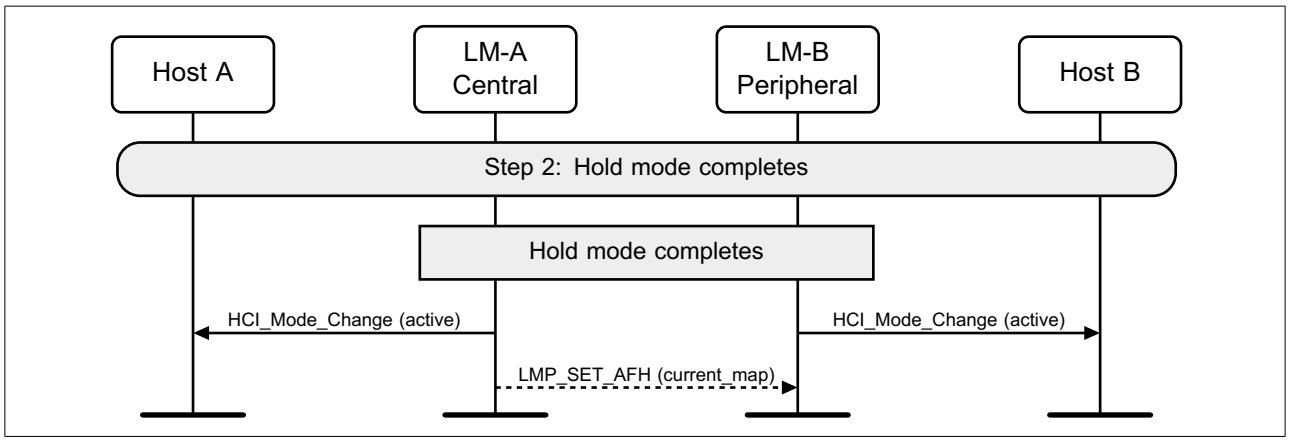



*Figure 6.6: Hold mode completes*

## 6.3 [This section is no longer used]

# 7 BUFFER MANAGEMENT, FLOW CONTROL

Buffer management is very important for resource limited devices.
This can be achieved on the Host Controller interface using the HCI_Read_Buffer_Size command, and the HCI_Number_Of_Completed_Packets event, and the HCI_Set_Controller_To_Host_Flow_Control, HCI_Host_Buffer_Size and HCI_Host_Number_Of_Completed_Packets commands.

**Step 1:** During initialization, the Host reads the buffer sizes available in the Controller. When an HCI Data packet has been transferred to the remote device, and a Baseband acknowledgment has been received for this data, then an HCI_Number_Of_Completed_Packets event will be generated. (See Figure 7.1.)

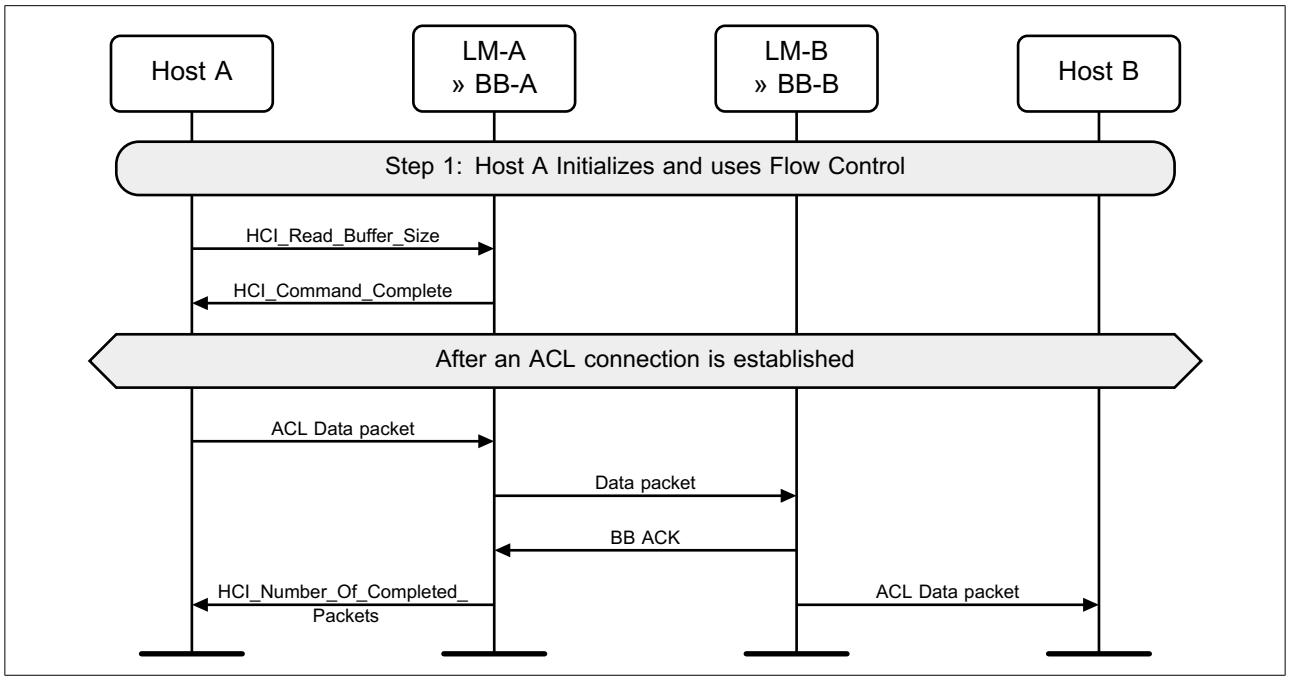



*Figure 7.1: Host to Controller flow control*

**Step 2:** During initialization, the Host may notify the Controller that Host flow control is being used and the Host buffer sizes available. When a data packet has been received from a remote device, an HCI Data packet is sent to the Host from the Controller and the Host acknowledges its receipt by sending an HCI_Host_Number_Of_Completed_Packets command. (See Figure 7.2.)

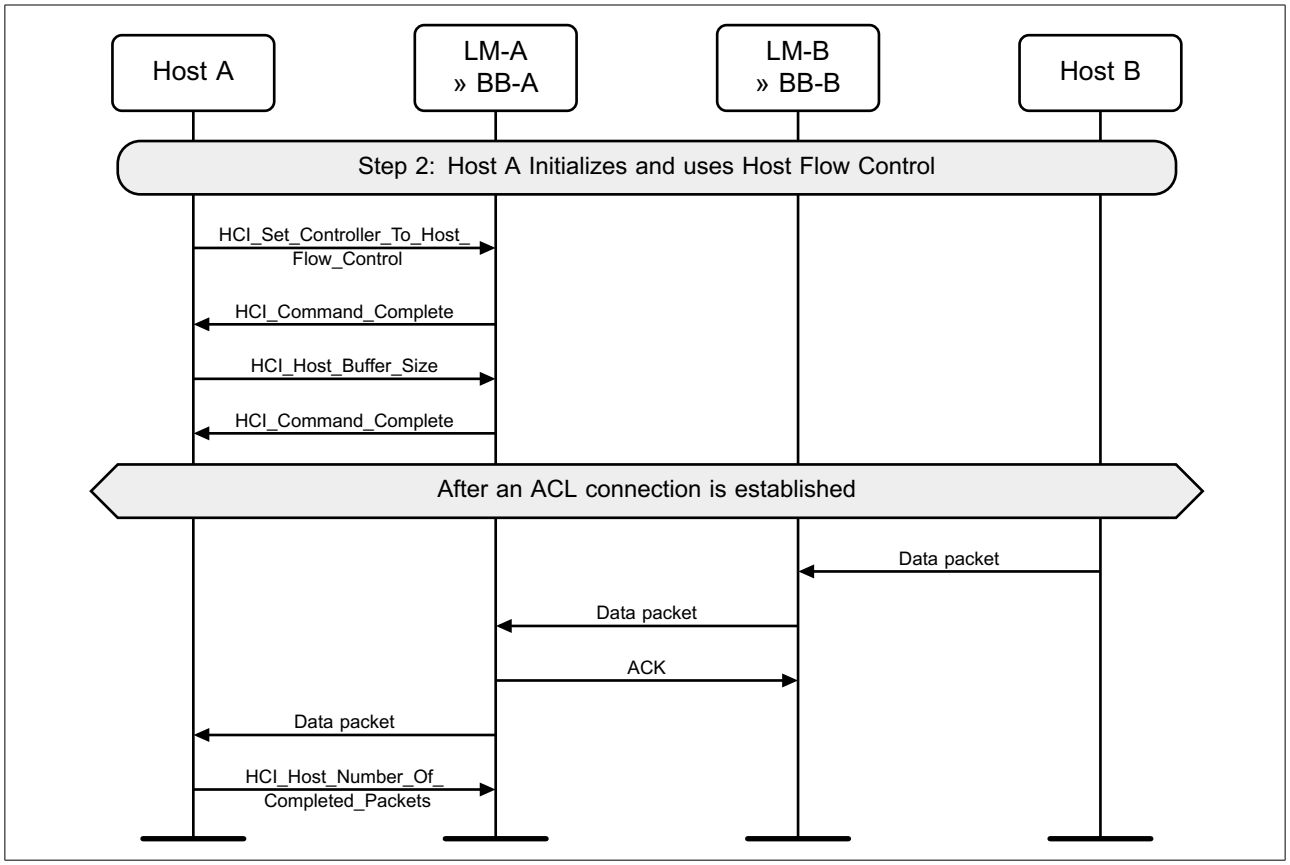



*Figure 7.2: Controller to Host flow control*

# 8 LOOPBACK MODE

The loopback modes are used for testing of a device only.

## 8.1 Local Loopback mode

The Local Loopback mode is used to loopback received HCI commands, and HCI ACL Data and HCI Synchronous Data packets sent from the Host to the Controller.

**Step 1:** The Host enters Local Loopback mode. Four HCI_Connection_Complete events are generated and then an HCI_Command_Complete event. (See Figure 8.1.)

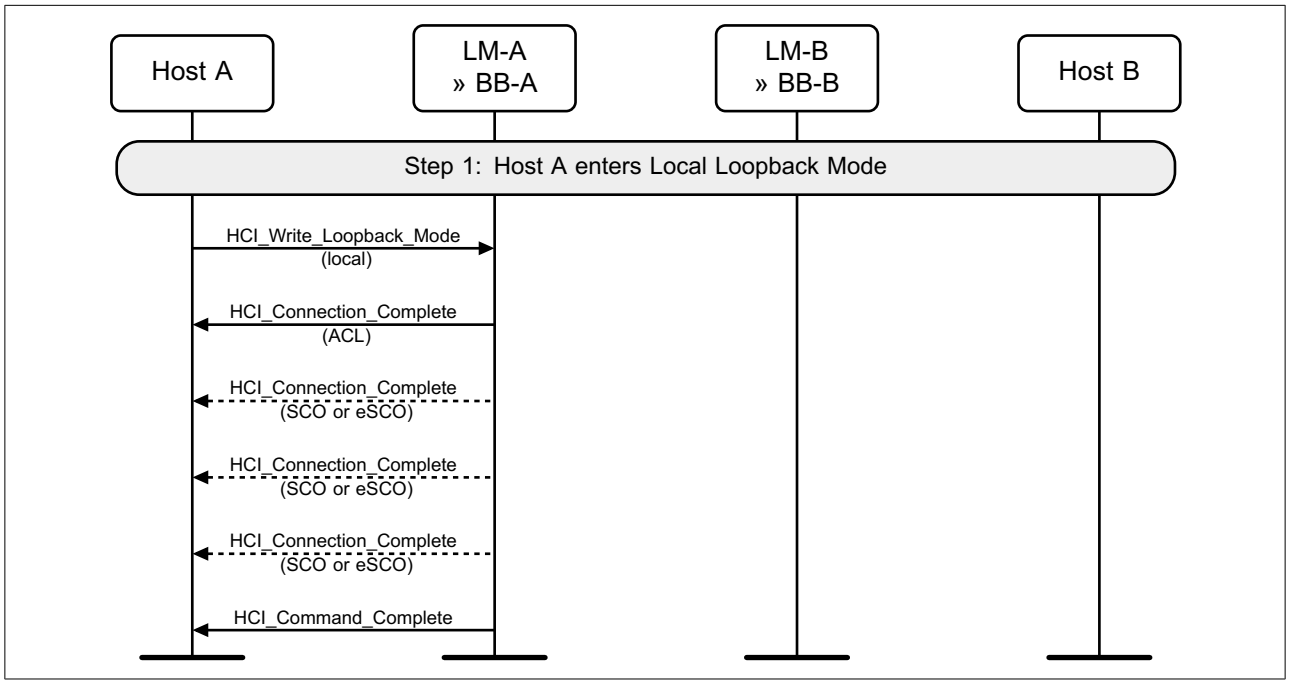



*Figure 8.1: Entering Local Loopback mode*

**Step 2a:** The Host sending HCI Data packet will receive the exact same data back in HCI Data packets from the Controller. (See Figure 8.2.)

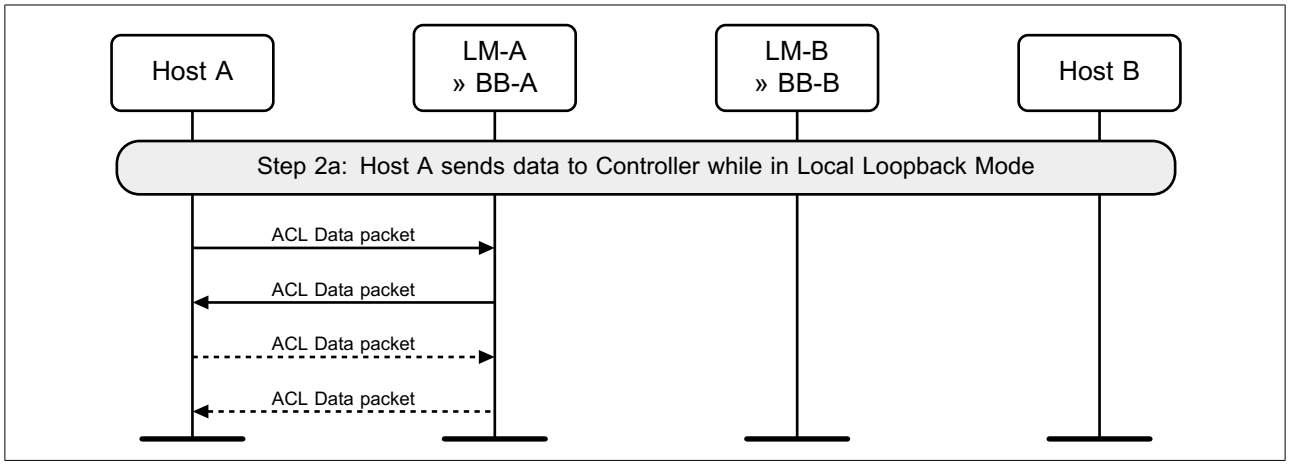



*Figure 8.2: Looping back data in Local Loopback mode*

**Step 2b:** The Host sending most HCI Command packets to the Controller will receive an HCI_Loopback_Command event with the contents of the HCI Command packet in the payload. (See Figure 8.3.)

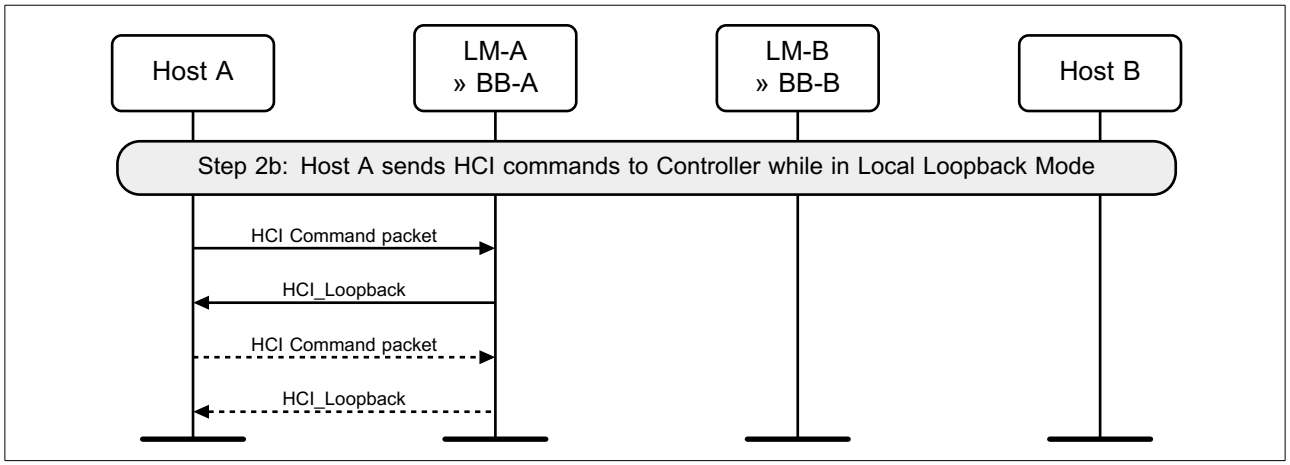



*Figure 8.3: Looping back commands in Local Loopback mode*

**Step 3:** The Host exits Local Loopback mode. Multiple HCI_Disconnection_Complete events are generated before the HCI_Command_Complete event. (See Figure 8.4.)

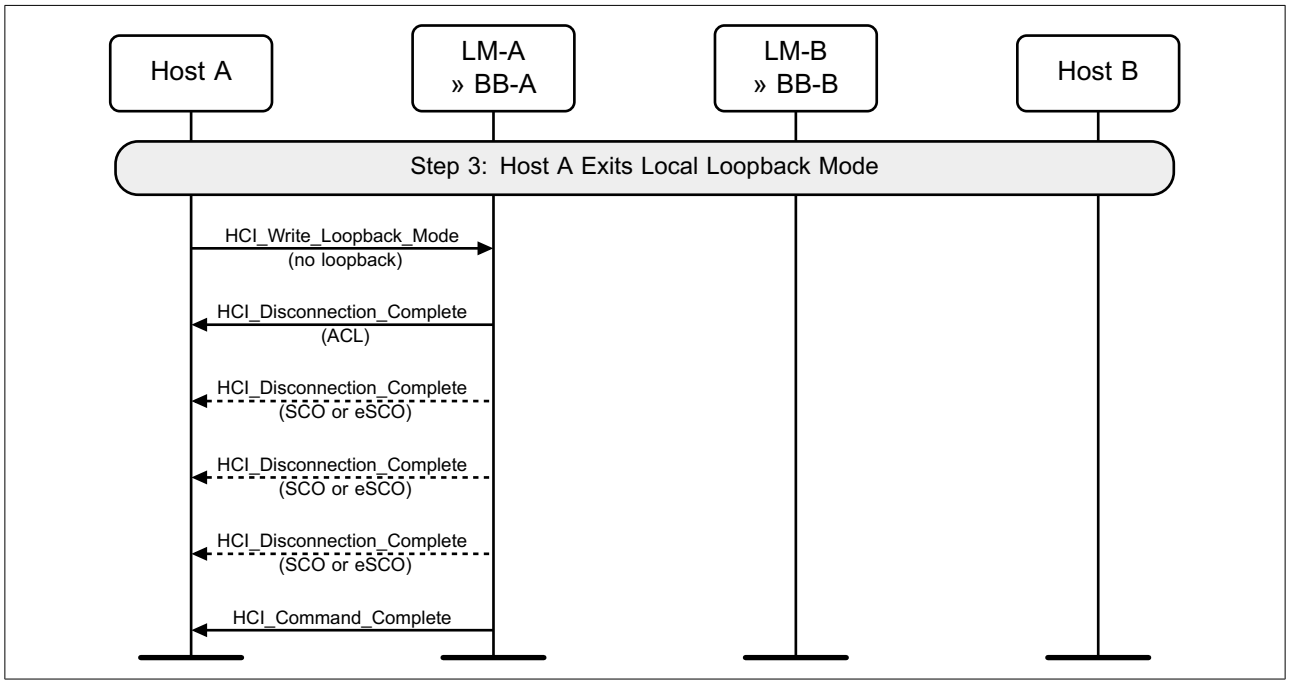



*Figure 8.4: Exiting Local Loopback mode*

## 8.2 Remote Loopback mode

The Remote Loopback mode is used to loopback data to a remote device over the air.

**Step 1:** The local device first enables remote loopback. The remote Host then sets up a connection to the local device. (See Figure 8.5.)

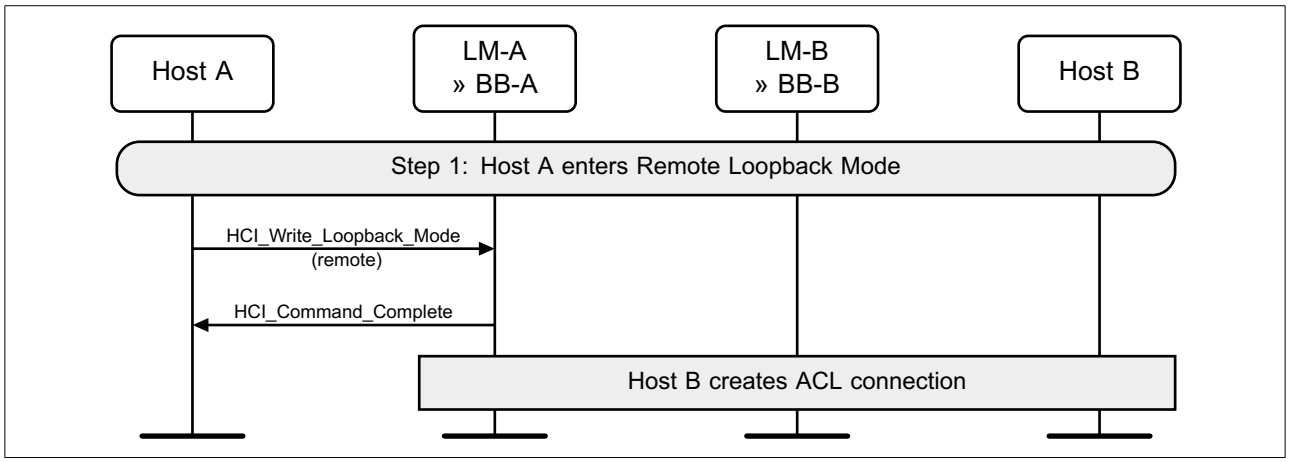



*Figure 8.5: Entering Remote Loopback mode*

**Step 2:** Any data received from the remote Host will be looped back in the Controller of the local device. (See Figure 8.6.)

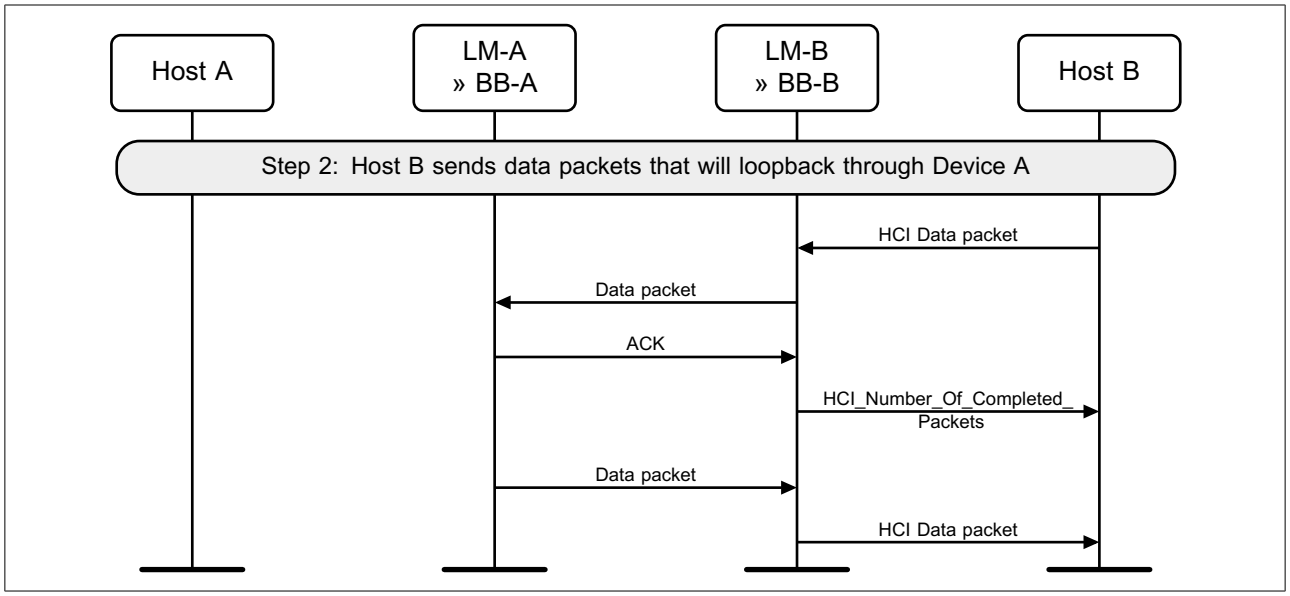



*Figure 8.6: Looping back data in Remote Loopback mode*

**Step 3:** The local Host exits Remote Loopback mode. Any connections can then be disconnected by the remote device. (See Figure 8.7.)

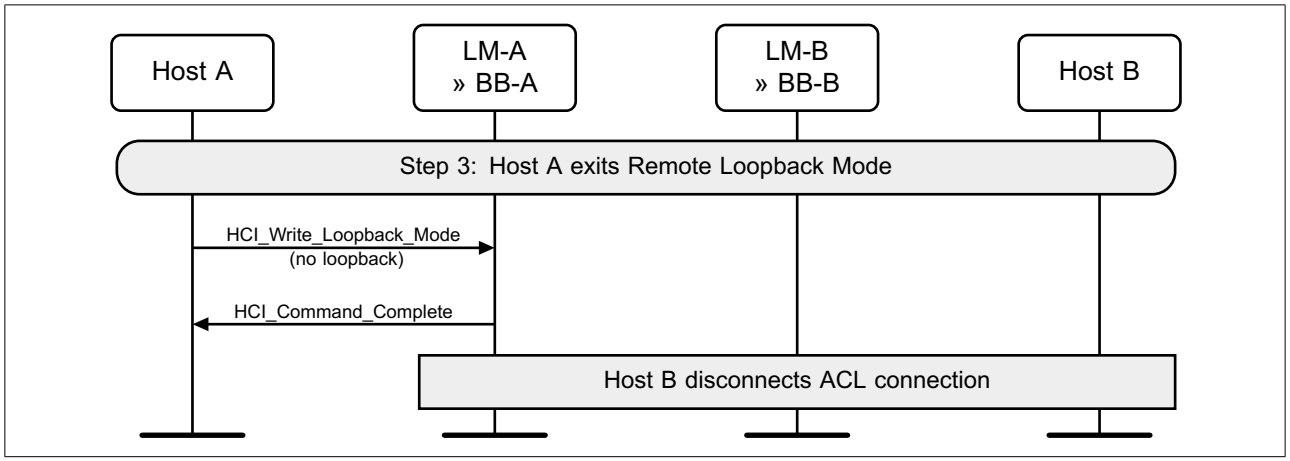



*Figure 8.7: Exiting Remote Loopback mode*

# 9 CONNECTIONLESS PERIPHERAL BROADCAST SERVICES

Figure 9.1 illustrates the Truncated Page procedure.

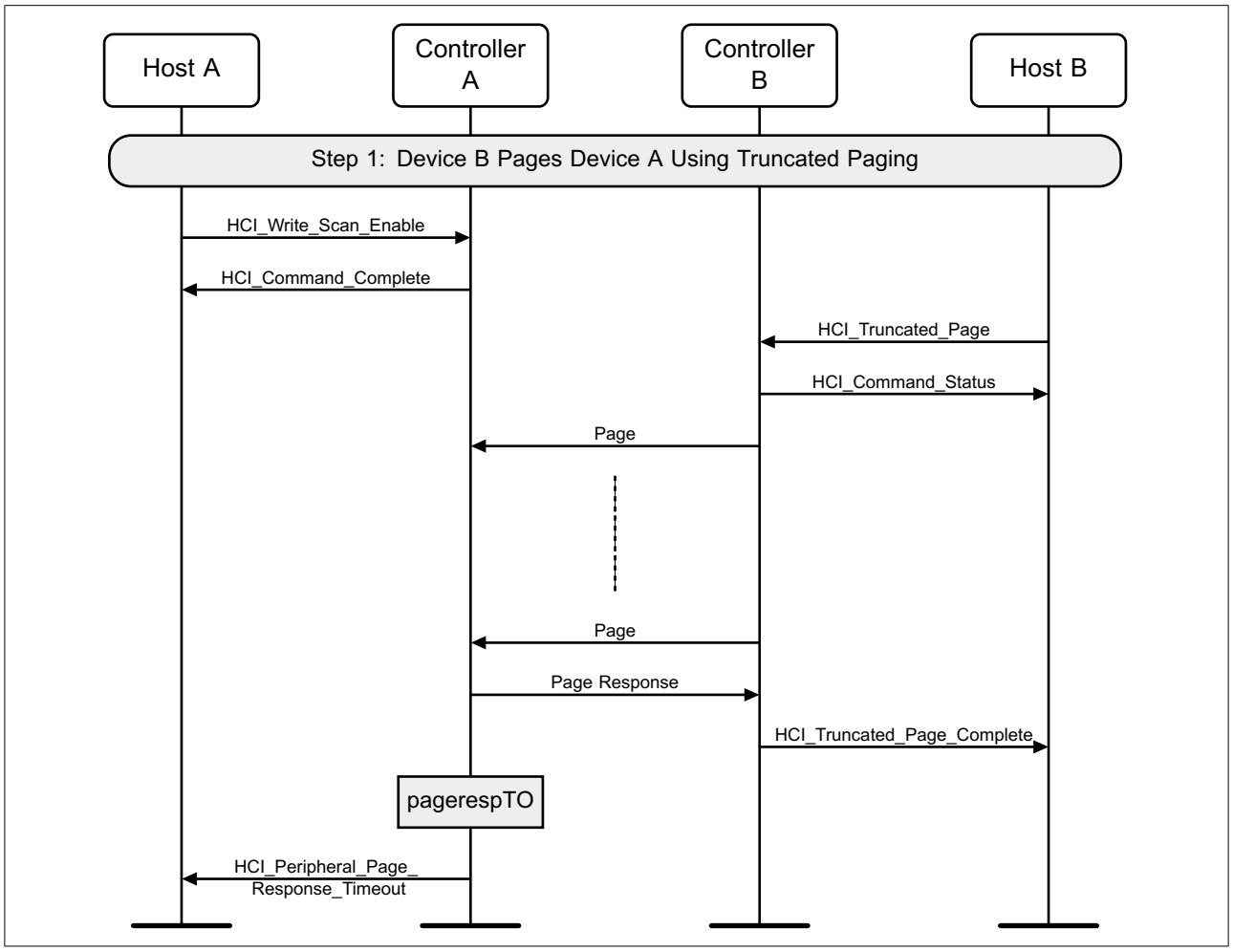



*Figure 9.1: Truncated paging*

Figure 9.2 illustrates how Device A starts transmitting Connectionless Peripheral Broadcast packets to Device B.

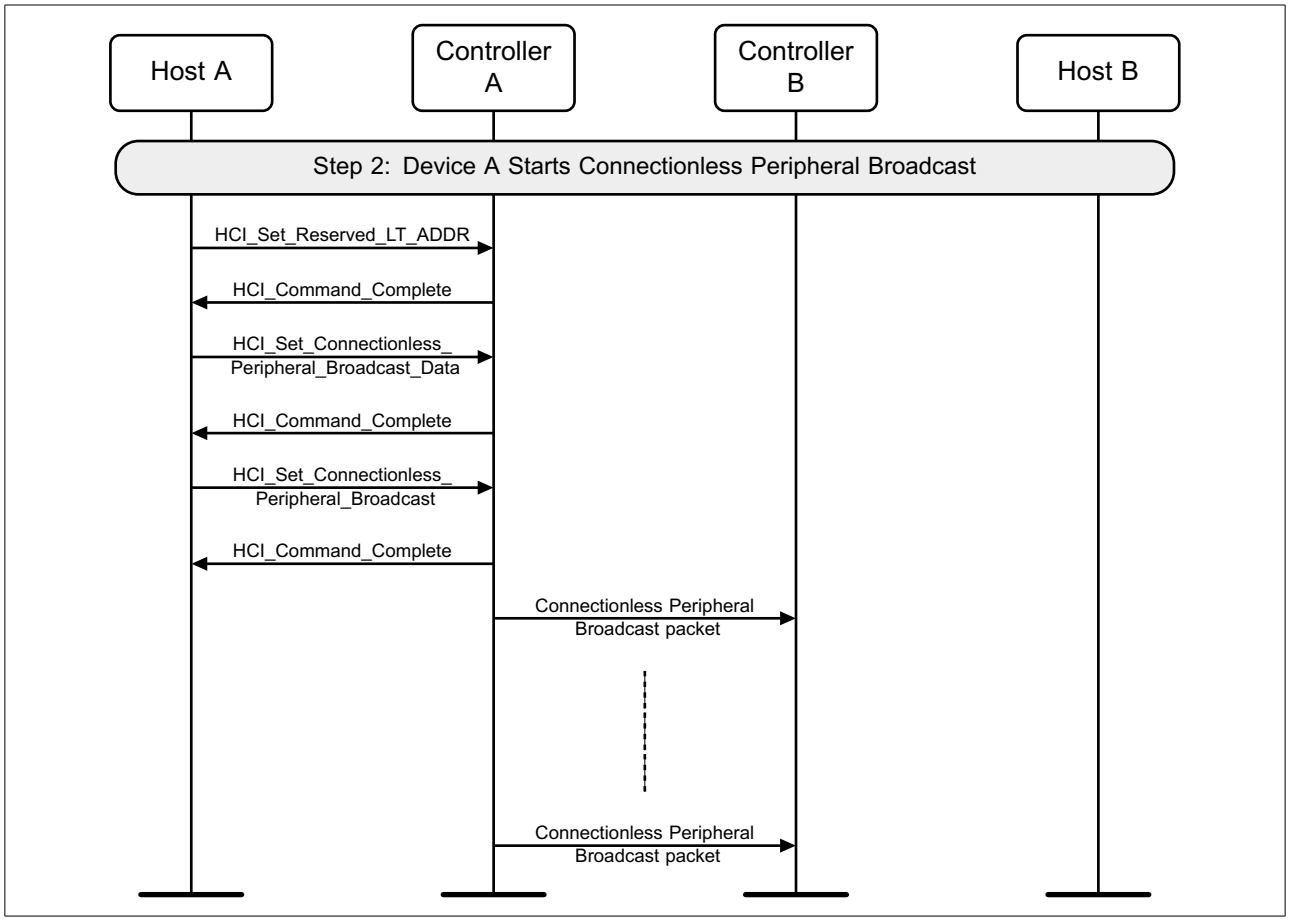



*Figure 9.2: Connectionless Peripheral Broadcast transmitter start*

Figure 9.3 shows the Synchronization Train feature. Device A is the Connectionless Peripheral Broadcast Transmitter. Device B is the Connectionless Peripheral Broadcast Receiver.

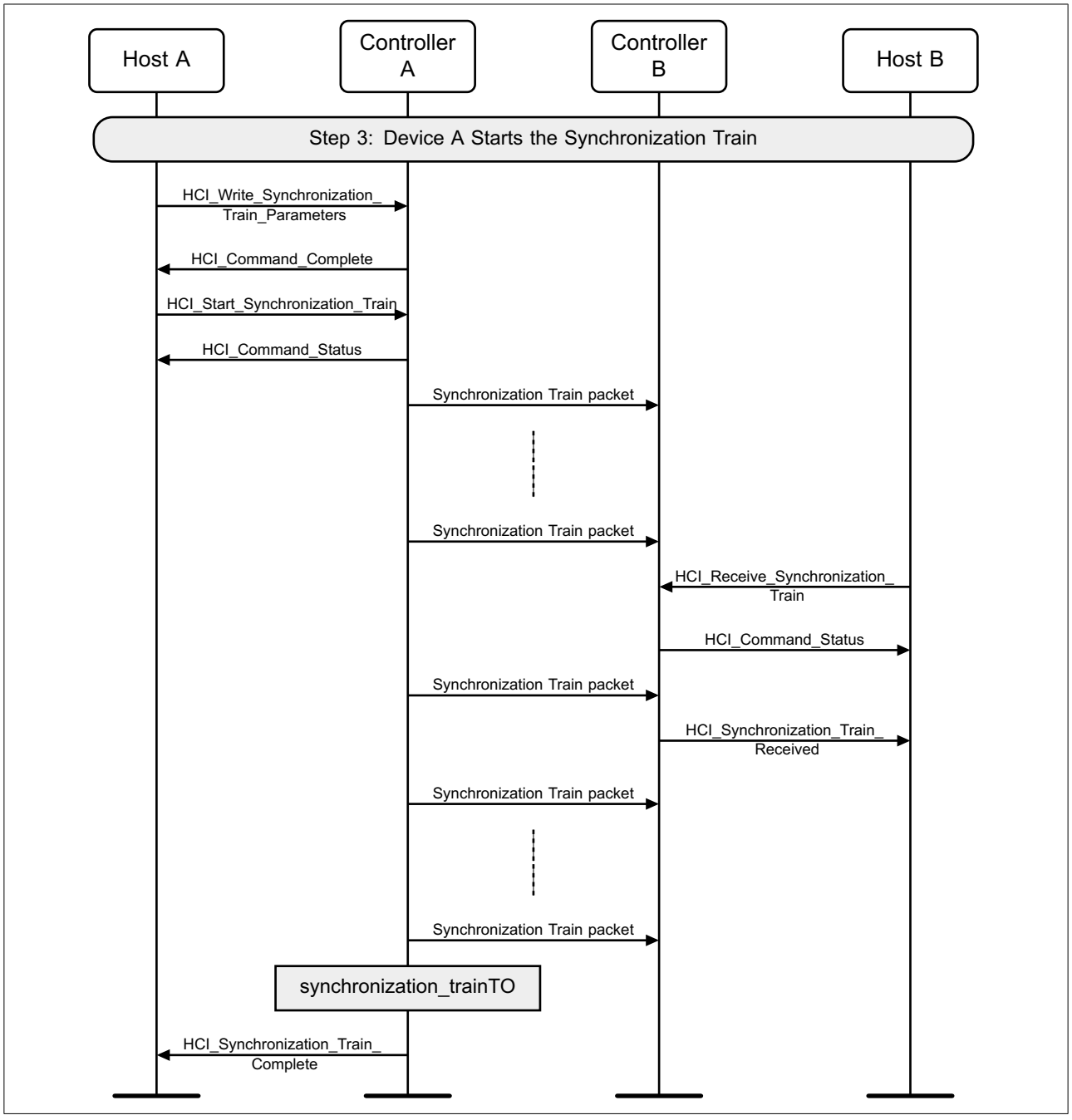



*Figure 9.3: Synchronization Train*

Figure 9.4 illustrates how Device B starts receiving Connectionless Peripheral Broadcast packets from Device A.

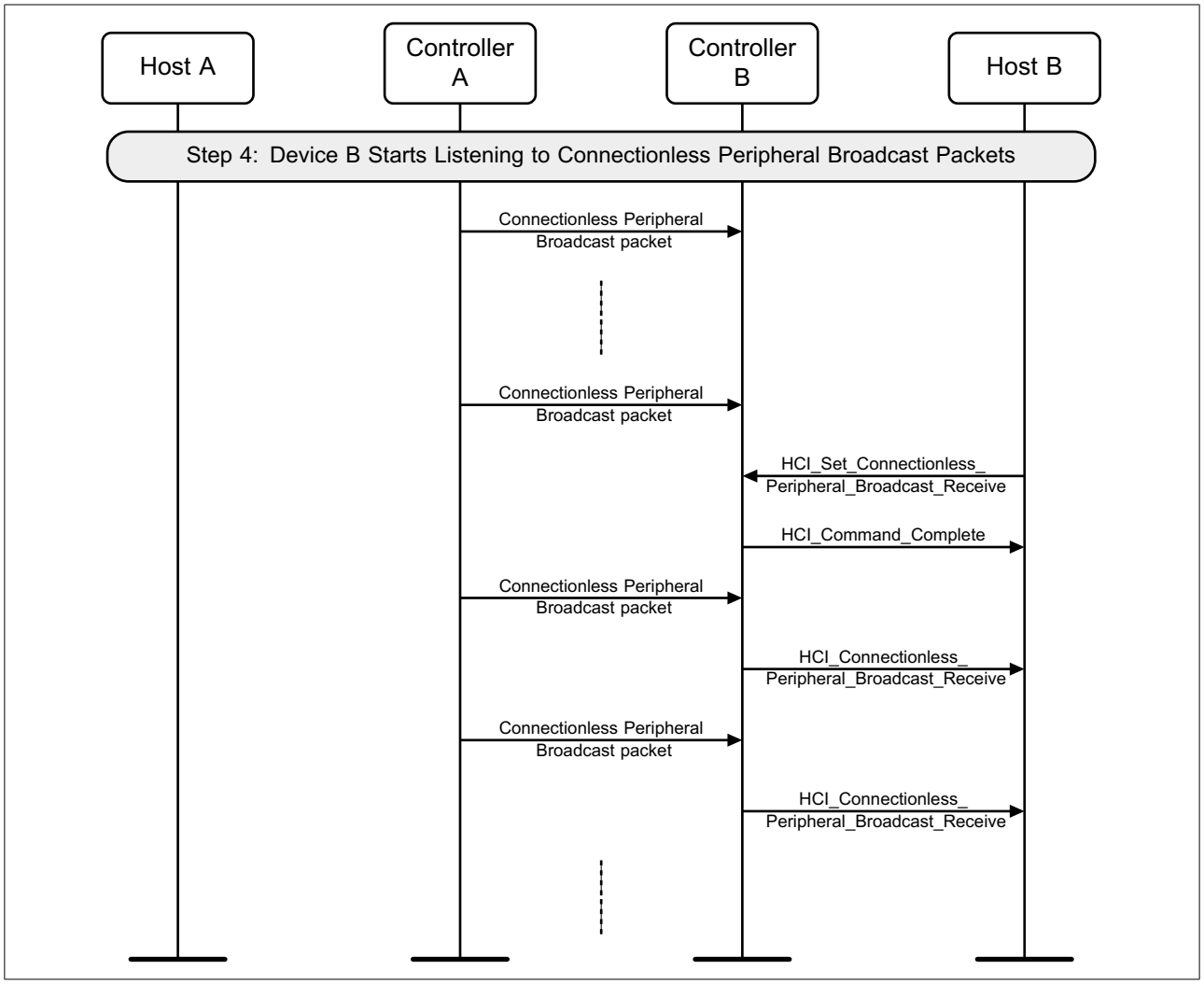



*Figure 9.4: Connectionless Peripheral Broadcast receiver start*

# SAMPLE DATA

*This Part contains sample data for the BR/EDR Controller; see [Vol 1] Part E, Section 4.*

# CONTENTS

**1 Encryption sample data ..... 874**
1.1 E0 encryption sample data ..... 874
1.1.1 Generating K_session from K_enc ..... 874
1.1.2 First set of sample data ..... 876
1.1.3 Second set of sample data ..... 886
1.1.4 Third set of samples ..... 896
1.1.5 Fourth set of samples ..... 906
1.2 AES-CCM encryption sample data ..... 916
1.2.1 Sample data 1 (DM1, Central → Peripheral) ..... 916
1.2.2 Sample data 2 (DM1, Central → Peripheral) ..... 917
1.2.3 Sample data 3 (DM1, Peripheral → Central) ..... 918
1.2.4 Sample data 4 (DM1, Central → Peripheral) ..... 919
1.2.5 Sample data 5 (DM1, Peripheral → Central) ..... 920
1.2.6 Sample data 6 (DH1, Central → Peripheral) ..... 921
1.2.7 Sample data 7 (DH1, Peripheral → Central) ..... 922
1.2.8 Sample data 8 (DH1, Central → Peripheral) ..... 923
1.2.9 Sample data 9 (DH1, Peripheral → Central) ..... 924
1.2.10 Sample data 10 (2-DH3, Central → Peripheral) ..... 925
1.2.11 Sample data 11 (2-DH3, Peripheral → Central) ..... 929
1.2.12 Sample data 12 (3-DH5, Central → Peripheral) ..... 933
1.2.13 Sample data 13 (3-DH5, Peripheral → Central) ..... 943
1.2.14 Sample data 14 (EV3) ..... 953

**2 Frequency hopping sample data ..... 954**
2.1 First set ..... 954
2.2 Second set ..... 960
2.3 Third set ..... 966

**3 Access code sample data ..... 973**

**4 HEC and packet header sample data ..... 976**

**5 CRC sample data ..... 977**

**6 Complete sample packets ..... 978**
6.1 Example of DH1 packet ..... 978
6.2 Example of DM1 packet ..... 980

**7 Secure Simple Pairing sample data ..... 982**
7.1 Elliptic curve sample data ..... 982

7.1.1 P-192 sample data ..... 982
7.1.1.1 P-192 data set 1 ..... 982
7.1.1.2 P-192 data set 2 ..... 982
7.1.1.3 P-192 data set 3 ..... 982
7.1.1.4 P-192 data set 4 ..... 982
7.1.1.5 P-192 data set 5 ..... 982
7.1.1.6 P-192 data set 6 ..... 983
7.1.1.7 P-192 data set 7 ..... 983
7.1.1.8 P-192 data set 8 ..... 983
7.1.1.9 P-192 data set 9 ..... 983
7.1.1.10 P-192 data set 10 ..... 983
7.1.2 P-256 sample data ..... 983
7.1.2.1 P-256 data set 1 ..... 983
7.1.2.2 P-256 data set 2 ..... 984
7.2 Hash functions sample data ..... 984
7.2.1 f1() ..... 984
7.2.1.1 f1() with P-192 inputs ..... 984
7.2.1.2 f1() with P-256 inputs ..... 985
7.2.2 g() ..... 986
7.2.2.1 g() with P-192 inputs ..... 986
7.2.2.2 g() with P-256 inputs ..... 986
7.2.3 f2() ..... 986
7.2.3.1 f2() with P-192 inputs ..... 986
7.2.3.2 f2() with P-256 inputs ..... 986
7.2.4 f3() ..... 987
7.2.4.1 f3() with P-192 inputs ..... 987
7.2.4.2 f3() with P-256 inputs ..... 993
7.2.5 [This section is no longer used] ..... 993
7.2.6 h4() ..... 993
7.2.7 h5() ..... 994
7.2.8 h3() ..... 994

**8 Whitening sequence sample data ..... 995**

**9 FEC sample data ..... 998**

**10 Encryption key sample data ..... 999**
10.1 Four tests of E1 ..... 999
10.2 Four tests of E21 ..... 1004
10.3 Three tests of E22 ..... 1007
10.4 Tests of E22 with Pin augmenting ..... 1009
10.5 Four tests of E3 ..... 1020

**11 Connectionless Peripheral Broadcast sample data ..... 1025**

# 1 ENCRYPTION SAMPLE DATA

## 1.1 E0 encryption sample data

This section contains four sets of sample data for the encryption process.

With respect to the functional description of the $E_0$ encryption algorithm in the Bluetooth Baseband specification, the contents of registers and resulting concurrent values are listed as well. This does not prohibit other implementations that produce the same encryption stream, but is intended to describe the functional behavior.

### 1.1.1 Generating K_session from K_enc

where K_session(x) = g2(x)(K_enc(x) *mod* g1(x)).

All polynomials are in hexadecimal notation.

'L' is the effective key length in bytes.

The notation 'p: [m]' implies that deg(p(x)) = m.

```
                                MSB                                 LSB
L = 7
g1:                  [56]       00000000 00000000 01000000 00000095
g2:                  [71]       00000000 000000b3 f7fffce2 79f3a073
K_enc:                          05454e03 8ddcfbe3 ed024b2d 92b7f54c
K_enc mod g1:        [55]       00000000 00000000 0095b8a4 8eb816da
g2(K_enc mod g1):    [126]      50f9c0d4 e3178da9 4a09fe0d 34f67b0e
-----------------------------------------------------------------
L = 8
g1:                  [64]       00000000 00000001 00000000 0000001b
g2:                  [63]       00000000 00000000 a1ab815b c7ec8025
K_enc:                          7ce149fc f4b38ad7 2a5d8a41 eb15ba31
K_enc mod g1:        [63]       00000000 00000000 8660806c 1865deec
g2(K_enc mod g1):    [126]      532c36d4 5d0954e0 922989b6 826f78dc
-----------------------------------------------------------------
L = 9
g1:                  [72]       00000000 00000100 00000000 00000609
g2:                  [49]       00000000 00000000 0002c980 11d8b04d
K_enc:                          5eeff7ca 84fc2782 9c051726 3df6f36e
K_enc mod g1:        [71]       00000000 00000083 58ccb7d0 b95d3c71
g2(K_enc mod g1):    [120]      016313f6 0d3771cf 7f8e4bb9 4aa6827d
-----------------------------------------------------------------
L = 10
g1:                  [80]       00000000 00010000 00000000 00000215
g2:                  [42]       00000000 00000000 0000058e 24f9a4bb
K_enc:                          7b13846e 88beb4de 34e7160a fd44dc65
```

*Sample Data*

```
K_enc mod g1:       [79]       00000000 0000b4de 34171767 f36981c3
g2(K_enc mod g1):   [121]      023bc1ec 34a0029e f798dcfb 618ba58d
-----------------------------------------------------------------
L = 11
g1:                 [88]       00000000 01000000 00000000 0000013b
g2:                 [35]       00000000 00000000 0000000c a76024d7
K_enc:                         bda6de6c 6e7d757e 8dfe2d49 9a181193
K_enc mod g1:       [86]       00000000 007d757e 8dfe88aa 2fcee371
g2(K_enc mod g1):   [121]      022e08a9 3aa51d8d 2f93fa78 85cc1f87
-----------------------------------------------------------------
L = 12
g1:                 [96]       00000001 00000000 00000000 000000dd
g2:                 [28]       00000000 00000000 00000000 1c9c26b9
K_enc:                         e6483b1c 2cdb1040 9a658f97 c4efd90d
K_enc mod g1:       [93]       00000000 2cdb1040 9a658fd7 5b562e41
g2(K_enc mod g1):   [121]      030d752b 216fe29b b880275c d7e6f6f9
-----------------------------------------------------------------
L = 13
g1:                 [104]      00000100 00000000 00000000 0000049d
g2:                 [21]       00000000 00000000 00000000 0026d9e3
K_enc:                         d79d281d a2266847 6b223c46 dc0ab9ee
K_enc mod g1:       [100]      0000001d a2266847 6b223c45 e1fc5fa6
g2(K_enc mod g1):   [121]      03f11138 9cebf919 00b93808 4ac158aa
-----------------------------------------------------------------
L = 14
g1:                 [112]      00010000 00000000 00000000 0000014f
g2:                 [14]       00000000 00000000 00000000 00004377
K_enc:                         cad9a65b 9fca1c1d a2320fcf 7c4ae48e
K_enc mod g1:       [111]      0000a65b 9fca1c1d a2320fcf 7cb6a909
g2(K_enc mod g1):   [125]      284840fd f1305f3c 529f5703 76adf7cf
-----------------------------------------------------------------
L = 15
g1:                 [120]      01000000 00000000 00000000 000000e7
g2:                 [7]        00000000 00000000 00000000 00000089
K_enc:                         21f0cc31 049b7163 d375e9e1 06029809
K_enc mod g1:       [119]      00f0cc31 049b7163 d375e9e1 0602840e
g2(K_enc mod g1):   [126]      7f10b53b 6df84b94 f22e566a 3754a37e
-----------------------------------------------------------------
L = 16
g1:                 [128]   1  00000000 00000000 00000000 00000000
g2:                 [0]        00000000 00000000 00000000 00000001
K_enc:                         35ec8fc3 d50ccd32 5f2fd907 bde206de
K_enc mod g1:       [125]      35ec8fc3 d50ccd32 5f2fd907 bde206de
g2(K_enc mod g1):   [125]      35ec8fc3 d50ccd32 5f2fd907 bde206de
-----------------------------------------------------------------
```

*Sample Data*

### 1.1.2 First set of sample data

In Section 1.1.2 to Section 1.1.5 the notation X[j] means the jth octet of X rather than the jth bit of X, counting from the LSB and an asterisk appended to an LFSR value indicates that the switch for that LFSR is open (see step 2 of [Vol 2] Part H, Section 4.5).

```
Initial values for the key, BD_ADDR and clock

K_session[0]  = 00  K_session[1]  = 00  K_session[2]  = 00  K_session[3]  = 00
K_session[4]  = 00  K_session[5]  = 00  K_session[6]  = 00  K_session[7]  = 00
K_session[8]  = 00  K_session[9]  = 00  K_session[10] = 00  K_session[11] = 00
K_session[12] = 00  K_session[13] = 00  K_session[14] = 00  K_session[15] = 00

BD_ADDR_C[0] = 00  BD_ADDR_C[1] = 00  BD_ADDR_C[2] = 00
BD_ADDR_C[3] = 00  BD_ADDR_C[4] = 00  BD_ADDR_C[5] = 00

CL[0]  = 00  CL[1]  = 00  CL[2]  = 00  CL[3]  = 00

The corresponding values of CLK are 0x000_0000 and 0x800_0000.

================================================================================================
    Fill LFSRs with initial data
================================================================================================

t  clk#  LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

0     0  0000000* 00000000* 000000000* 0000000000*    0  0  0  0    0  00     00   00
1     1  0000000* 00000001* 000000000* 0000000001*    0  0  0  0    0  00     00   00
2     2  0000000* 00000002* 000000000* 0000000003*    0  0  0  0    0  00     00   00
3     3  0000000* 00000004* 000000000* 0000000007*    0  0  0  0    0  00     00   00
4     4  0000000* 00000008* 000000000* 000000000E*    0  0  0  0    0  00     00   00
5     5  0000000* 00000010* 000000000* 000000001C*    0  0  0  0    0  00     00   00
6     6  0000000* 00000020* 000000000* 0000000038*    0  0  0  0    0  00     00   00
7     7  0000000* 00000040* 000000000* 0000000070*    0  0  0  0    0  00     00   00
8     8  0000000* 00000080* 000000000* 00000000E0*    0  0  0  0    0  00     00   00
9     9  0000000* 00000100* 000000000* 00000001C0*    0  0  0  0    0  00     00   00
10   10  0000000* 00000200* 000000000* 0000000380*    0  0  0  0    0  00     00   00
11   11  0000000* 00000400* 000000000* 0000000700*    0  0  0  0    0  00     00   00
12   12  0000000* 00000800* 000000000* 0000000E00*    0  0  0  0    0  00     00   00
13   13  0000000* 00001000* 000000000* 0000001C00*    0  0  0  0    0  00     00   00
14   14  0000000* 00002000* 000000000* 0000003800*    0  0  0  0    0  00     00   00
15   15  0000000* 00004000* 000000000* 0000007000*    0  0  0  0    0  00     00   00
16   16  0000000* 00008000* 000000000* 000000E000*    0  0  0  0    0  00     00   00
```

*Sample Data*

```
17   17  0000000* 00010000* 000000000* 000001C000*    0  0  0  0    0  00      00   00
18   18  0000000* 00020000* 000000000* 0000038000*    0  0  0  0    0  00      00   00
19   19  0000000* 00040000* 000000000* 0000070000*    0  0  0  0    0  00      00   00
20   20  0000000* 00080000* 000000000* 00000E0000*    0  0  0  0    0  00      00   00
21   21  0000000* 00100000* 000000000* 00001C0000*    0  0  0  0    0  00      00   00
22   22  0000000* 00200000* 000000000* 0000380000*    0  0  0  0    0  00      00   00
23   23  0000000* 00400000* 000000000* 0000700000*    0  0  0  0    0  00      00   00
24   24  0000000* 00800000* 000000000* 0000E00000*    0  1  0  0    1  00      00   00
25   25  0000000* 01000000* 000000000* 0001C00000*    0  0  0  0    0  00      00   00
26   26  0000000  02000000* 000000000* 0003800000*    0  0  0  0    0  00      00   00
27   27  0000000  04000000* 000000000* 0007000000*    0  0  0  0    0  00      00   00
28   28  0000000  08000000* 000000000* 000E000000*    0  0  0  0    0  00      00   00
29   29  0000000  10000000* 000000000* 001C000000*    0  0  0  0    0  00      00   00
30   30  0000000  20000000* 000000000* 0038000000*    0  0  0  0    0  00      00   00
31   31  0000000  40000000* 000000000* 0070000000*    0  0  0  0    0  00      00   00
32   32  0000000  00000001  000000000* 00E0000000*    0  0  0  1    1  00      00   00
33   33  0000000  00000002  000000000* 01C0000000*    0  0  0  1    1  00      00   00
34   34  0000000  00000004  000000000  0380000000*    0  0  0  1    1  00      00   00
35   35  0000000  00000008  000000000  0700000000*    0  0  0  0    0  00      00   00
36   36  0000000  00000010  000000000  0E00000000*    0  0  0  0    0  00      00   00
37   37  0000000  00000020  000000000  1C00000000*    0  0  0  0    0  00      00   00
38   38  0000000  00000040  000000000  3800000000*    0  0  0  0    0  00      00   00
39   39  0000000  00000080  000000000  7000000000*    0  0  0  0    0  00      00   00
================================================================================================
    Start clocking Summation Combiner
================================================================================================
40    1  0000000  00000100  000000000  6000000001     0  0  0  0    0  00      00   00
41    2  0000000  00000200  000000000  4000000003     0  0  0  0    0  00      00   00
42    3  0000000  00000400  000000000  0000000007     0  0  0  0    0  00      00   00
43    4  0000000  00000800  000000000  000000000E     0  0  0  0    0  00      00   00
44    5  0000000  00001001  000000000  000000001D     0  0  0  0    0  00      00   00
45    6  0000000  00002002  000000000  000000003B     0  0  0  0    0  00      00   00
46    7  0000000  00004004  000000000  0000000077     0  0  0  0    0  00      00   00
47    8  0000000  00008008  000000000  00000000EE     0  0  0  0    0  00      00   00
48    9  0000000  00010011  000000000  00000001DD     0  0  0  0    0  00      00   00
49   10  0000000  00020022  000000000  00000003BB     0  0  0  0    0  00      00   00
50   11  0000000  00040044  000000000  0000000777     0  0  0  0    0  00      00   00
51   12  0000000  00080088  000000000  0000000EEE     0  0  0  0    0  00      00   00
52   13  0000000  00100110  000000000  0000001DDD     0  0  0  0    0  00      00   00
53   14  0000000  00200220  000000000  0000003BBB     0  0  0  0    0  00      00   00
54   15  0000000  00400440  000000000  0000007777     0  0  0  0    0  00      00   00
55   16  0000000  00800880  000000000  000000EEEE     0  1  0  0    1  00      00   00
56   17  0000000  01001100  000000000  000001DDDD     0  0  0  0    0  00      00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 57 | 18 | 0000000 | 02002200 | 000000000 | 000003BBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 58 | 19 | 0000000 | 04004400 | 000000000 | 0000077777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 59 | 20 | 0000000 | 08008800 | 000000000 | 00000EEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 60 | 21 | 0000000 | 10011000 | 000000000 | 00001DDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 61 | 22 | 0000000 | 20022000 | 000000000 | 00003BBBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 62 | 23 | 0000000 | 40044000 | 000000000 | 0000777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 63 | 24 | 0000000 | 00088001 | 000000000 | 0000EEEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 64 | 25 | 0000000 | 00110003 | 000000000 | 0001DDDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 65 | 26 | 0000000 | 00220006 | 000000000 | 0003BBBBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 66 | 27 | 0000000 | 0044000C | 000000000 | 0007777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 67 | 28 | 0000000 | 00880018 | 000000000 | 000EEEEEEE | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 68 | 29 | 0000000 | 01100031 | 000000000 | 001DDDDDDC | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 69 | 30 | 0000000 | 02200062 | 000000000 | 003BBBBBB8 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 70 | 31 | 0000000 | 044000C4 | 000000000 | 0077777770 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 71 | 32 | 0000000 | 08800188 | 000000000 | 00EEEEEEE0 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 72 | 33 | 0000000 | 11000311 | 000000000 | 01DDDDDDC1 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 73 | 34 | 0000000 | 22000622 | 000000000 | 03BBBBBB83 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 74 | 35 | 0000000 | 44000C44 | 000000000 | 0777777707 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 75 | 36 | 0000000 | 08001888 | 000000000 | 0EEEEEEE0E | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 76 | 37 | 0000000 | 10003111 | 000000000 | 1DDDDDDC1D | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 77 | 38 | 0000000 | 20006222 | 000000000 | 3BBBBBB83B | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 78 | 39 | 0000000 | 4000C444 | 000000000 | 7777777077 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 79 | 40 | 0000000 | 00018888 | 000000000 | 6EEEEEE0EF | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 80 | 41 | 0000000 | 00031110 | 000000000 | 5DDDDDC1DE | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 81 | 42 | 0000000 | 00062220 | 000000000 | 3BBBBB83BC | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 82 | 43 | 0000000 | 000C4440 | 000000000 | 7777770779 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 83 | 44 | 0000000 | 00188880 | 000000000 | 6EEEEE0EF2 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 84 | 45 | 0000000 | 00311100 | 000000000 | 5DDDDC1DE5 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 85 | 46 | 0000000 | 00622200 | 000000000 | 3BBBB83BCB | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 86 | 47 | 0000000 | 00C44400 | 000000000 | 7777707797 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 87 | 48 | 0000000 | 01888801 | 000000000 | 6EEEE0EF2F | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 88 | 49 | 0000000 | 03111003 | 000000000 | 5DDDC1DE5E | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 89 | 50 | 0000000 | 06222006 | 000000000 | 3BBB83BCBC | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 90 | 51 | 0000000 | 0C44400C | 000000000 | 7777077979 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 91 | 52 | 0000000 | 18888018 | 000000000 | 6EEE0EF2F2 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 92 | 53 | 0000000 | 31110030 | 000000000 | 5DDC1DE5E5 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 93 | 54 | 0000000 | 62220060 | 000000000 | 3BB83BCBCB | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 94 | 55 | 0000000 | 444400C1 | 000000000 | 7770779797 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 11 |
| 95 | 56 | 0000000 | 08880183 | 000000000 | 6EE0EF2F2F | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 96 | 57 | 0000000 | 11100307 | 000000000 | 5DC1DE5E5F | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 97 | 58 | 0000000 | 2220060E | 000000000 | 3B83BCBCBF | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 98 | 59 | 0000000 | 44400C1C | 000000000 | 770779797E | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 99 | 60 | 0000000 | 08801838 | 000000000 | 6E0EF2F2FC | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 61 | 0000000 | 11003070 | 000000000 | 5C1DE5E5F8 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 101 | 62 | 0000000 | 220060E0 | 000000000 | 383BCBCBF0 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 102 | 63 | 0000000 | 4400C1C0 | 000000000 | 70779797E0 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 103 | 64 | 0000000 | 08018380 | 000000000 | 60EF2F2FC1 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 104 | 65 | 0000000 | 10030701 | 000000000 | 41DE5E5F82 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 105 | 66 | 0000000 | 20060E02 | 000000000 | 03BCBCBF04 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 106 | 67 | 0000000 | 400C1C05 | 000000000 | 0779797E09 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 107 | 68 | 0000000 | 0018380A | 000000000 | 0EF2F2FC12 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 108 | 69 | 0000000 | 00307015 | 000000000 | 1DE5E5F825 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 109 | 70 | 0000000 | 0060E02A | 000000000 | 3BCBCBF04B | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 110 | 71 | 0000000 | 00C1C055 | 000000000 | 779797E097 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 111 | 72 | 0000000 | 018380AA | 000000000 | 6F2F2FC12F | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 112 | 73 | 0000000 | 03070154 | 000000000 | 5E5E5F825E | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 113 | 74 | 0000000 | 060E02A8 | 000000000 | 3CBCBF04BC | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 114 | 75 | 0000000 | 0C1C0550 | 000000000 | 79797E0979 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 115 | 76 | 0000000 | 18380AA0 | 000000000 | 72F2FC12F2 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 116 | 77 | 0000000 | 30701541 | 000000000 | 65E5F825E5 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 117 | 78 | 0000000 | 60E02A82 | 000000000 | 4BCBF04BCB | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 118 | 79 | 0000000 | 41C05505 | 000000000 | 1797E09796 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 119 | 80 | 0000000 | 0380AA0A | 000000000 | 2F2FC12F2C | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 120 | 81 | 0000000 | 07015415 | 000000000 | 5E5F825E59 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 121 | 82 | 0000000 | 0E02A82A | 000000000 | 3CBF04BCB2 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 122 | 83 | 0000000 | 1C055054 | 000000000 | 797E097964 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 123 | 84 | 0000000 | 380AA0A8 | 000000000 | 72FC12F2C9 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 124 | 85 | 0000000 | 70154151 | 000000000 | 65F825E593 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 125 | 86 | 0000000 | 602A82A3 | 000000000 | 4BF04BCB26 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 126 | 87 | 0000000 | 40550546 | 000000000 | 17E097964C | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 127 | 88 | 0000000 | 00AA0A8D | 000000000 | 2FC12F2C99 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 128 | 89 | 0000000 | 0154151A | 000000000 | 5F825E5932 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 129 | 90 | 0000000 | 02A82A34 | 000000000 | 3F04BCB264 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 130 | 91 | 0000000 | 05505468 | 000000000 | 7E097964C9 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 131 | 92 | 0000000 | 0AA0A8D0 | 000000000 | 7C12F2C992 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 132 | 93 | 0000000 | 154151A1 | 000000000 | 7825E59324 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 133 | 94 | 0000000 | 2A82A342 | 000000000 | 704BCB2648 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 134 | 95 | 0000000 | 55054684 | 000000000 | 6097964C91 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 135 | 96 | 0000000 | 2A0A8D09 | 000000000 | 412F2C9923 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 136 | 97 | 0000000 | 54151A12 | 000000000 | 025E593246 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 137 | 98 | 0000000 | 282A3424 | 000000000 | 04BCB2648D | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 138 | 99 | 0000000 | 50546848 | 000000000 | 097964C91A | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 139 | 100 | 0000000 | 20A8D090 | 000000000 | 12F2C99235 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 140 | 101 | 0000000 | 4151A120 | 000000000 | 25E593246A | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 141 | 102 | 0000000 | 02A34240 | 000000000 | 4BCB2648D5 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 142 | 103 | 0000000 | 05468481 | 000000000 | 17964C91AB | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 143 | 104 | 0000000 | 0A8D0903 | 000000000 | 2F2C992357 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 144 | 105 | 0000000 | 151A1206 | 000000000 | 5E593246AE | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 145 | 106 | 0000000 | 2A34240C | 000000000 | 3CB2648D5C | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 146 | 107 | 0000000 | 54684818 | 000000000 | 7964C91AB8 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 147 | 108 | 0000000 | 28D09030 | 000000000 | 72C9923571 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 148 | 109 | 0000000 | 51A12060 | 000000000 | 6593246AE2 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 149 | 110 | 0000000 | 234240C0 | 000000000 | 4B2648D5C5 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 150 | 111 | 0000000 | 46848180 | 000000000 | 164C91AB8A | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 151 | 112 | 0000000 | 0D090301 | 000000000 | 2C99235714 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 152 | 113 | 0000000 | 1A120602 | 000000000 | 593246AE28 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 153 | 114 | 0000000 | 34240C04 | 000000000 | 32648D5C51 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 154 | 115 | 0000000 | 68481809 | 000000000 | 64C91AB8A2 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 155 | 116 | 0000000 | 50903012 | 000000000 | 4992357144 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 156 | 117 | 0000000 | 21206024 | 000000000 | 13246AE288 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 157 | 118 | 0000000 | 4240C048 | 000000000 | 2648D5C511 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 158 | 119 | 0000000 | 04818090 | 000000000 | 4C91AB8A23 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 159 | 120 | 0000000 | 09030120 | 000000000 | 1923571446 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 160 | 121 | 0000000 | 12060240 | 000000000 | 3246AE288D | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 161 | 122 | 0000000 | 240C0480 | 000000000 | 648D5C511B | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 162 | 123 | 0000000 | 48180900 | 000000000 | 491AB8A237 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 163 | 124 | 0000000 | 10301200 | 000000000 | 123571446F | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 164 | 125 | 0000000 | 20602400 | 000000000 | 246AE288DF | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 165 | 126 | 0000000 | 40C04800 | 000000000 | 48D5C511BE | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 166 | 127 | 0000000 | 01809001 | 000000000 | 11AB8A237D | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 167 | 128 | 0000000 | 03012002 | 000000000 | 23571446FA | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 168 | 129 | 0000000 | 06024004 | 000000000 | 46AE288DF5 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 169 | 130 | 0000000 | 0C048008 | 000000000 | 0D5C511BEA | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 170 | 131 | 0000000 | 18090011 | 000000000 | 1AB8A237D5 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 171 | 132 | 0000000 | 30120022 | 000000000 | 3571446FAA | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 172 | 133 | 0000000 | 60240044 | 000000000 | 6AE288DF55 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 173 | 134 | 0000000 | 40480089 | 000000000 | 55C511BEAA | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 174 | 135 | 0000000 | 00900113 | 000000000 | 2B8A237D54 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 175 | 136 | 0000000 | 01200227 | 000000000 | 571446FAA8 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 176 | 137 | 0000000 | 0240044E | 000000000 | 2E288DF550 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 177 | 138 | 0000000 | 0480089C | 000000000 | 5C511BEAA0 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 178 | 139 | 0000000 | 09001138 | 000000000 | 38A237D540 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 179 | 140 | 0000000 | 12002270 | 000000000 | 71446FAA81 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 180 | 141 | 0000000 | 240044E0 | 000000000 | 6288DF5503 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 181 | 142 | 0000000 | 480089C0 | 000000000 | 4511BEAA06 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 182 | 143 | 0000000 | 10011381 | 000000000 | 0A237D540D | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 183 | 144 | 0000000 | 20022702 | 000000000 | 1446FAA81A | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 184 | 145 | 0000000 | 40044E04 | 000000000 | 288DF55035 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 185 | 146 | 0000000 | 00089C08 | 000000000 | 511BEAA06A | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 186 | 147 | 0000000 | 00113810 | 000000000 | 2237D540D5 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 187 | 148 | 0000000 | 00227021 | 000000000 | 446FAA81AA | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 188 | 149 | 0000000 | 0044E042 | 000000000 | 08DF550355 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 189 | 150 | 0000000 | 0089C085 | 000000000 | 11BEAA06AA | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 190 | 151 | 0000000 | 0113810A | 000000000 | 237D540D54 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 191 | 152 | 0000000 | 02270215 | 000000000 | 46FAA81AA9 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 192 | 153 | 0000000 | 044E042A | 000000000 | 0DF5503553 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 193 | 154 | 0000000 | 089C0854 | 000000000 | 1BEAA06AA7 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 194 | 155 | 0000000 | 113810A8 | 000000000 | 37D540D54E | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 195 | 156 | 0000000 | 22702150 | 000000000 | 6FAA81AA9D | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 196 | 157 | 0000000 | 44E042A0 | 000000000 | 5F5503553A | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 197 | 158 | 0000000 | 09C08540 | 000000000 | 3EAA06AA75 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 198 | 159 | 0000000 | 13810A80 | 000000000 | 7D540D54EA | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |
| 199 | 160 | 0000000 | 27021500 | 000000000 | 7AA81AA9D5 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 200 | 161 | 0000000 | 4E042A00 | 000000000 | 75503553AB | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 201 | 162 | 0000000 | 1C085400 | 000000000 | 6AA06AA756 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 202 | 163 | 0000000 | 3810A800 | 000000000 | 5540D54EAC | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 203 | 164 | 0000000 | 70215000 | 000000000 | 2A81AA9D58 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 204 | 165 | 0000000 | 6042A001 | 000000000 | 5503553AB0 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 205 | 166 | 0000000 | 40854002 | 000000000 | 2A06AA7561 | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |
| 206 | 167 | 0000000 | 010A8004 | 000000000 | 540D54EAC3 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 207 | 168 | 0000000 | 02150009 | 000000000 | 281AA9D586 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 208 | 169 | 0000000 | 042A0012 | 000000000 | 503553AB0C | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 209 | 170 | 0000000 | 08540024 | 000000000 | 206AA75618 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 210 | 171 | 0000000 | 10A80048 | 000000000 | 40D54EAC30 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 211 | 172 | 0000000 | 21500091 | 000000000 | 01AA9D5861 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 212 | 173 | 0000000 | 42A00122 | 000000000 | 03553AB0C3 | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 213 | 174 | 0000000 | 05400244 | 000000000 | 06AA756186 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 214 | 175 | 0000000 | 0A800488 | 000000000 | 0D54EAC30D | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 215 | 176 | 0000000 | 15000911 | 000000000 | 1AA9D5861A | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 216 | 177 | 0000000 | 2A001223 | 000000000 | 3553AB0C35 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 217 | 178 | 0000000 | 54002446 | 000000000 | 6AA756186A | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 218 | 179 | 0000000 | 2800488D | 000000000 | 554EAC30D5 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 219 | 180 | 0000000 | 5000911B | 000000000 | 2A9D5861AA | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 220 | 181 | 0000000 | 20012236 | 000000000 | 553AB0C355 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 221 | 182 | 0000000 | 4002446C | 000000000 | 2A756186AA | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 222 | 183 | 0000000 | 000488D9 | 000000000 | 54EAC30D54 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 223 | 184 | 0000000 | 000911B2 | 000000000 | 29D5861AA8 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 224 | 185 | 0000000 | 00122364 | 000000000 | 53AB0C3550 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 225 | 186 | 0000000 | 002446C8 | 000000000 | 2756186AA0 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 226 | 187 | 0000000 | 00488D90 | 000000000 | 4EAC30D540 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 227 | 188 | 0000000 | 00911B20 | 000000000 | 1D5861AA81 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 228 | 189 | 0000000 | 01223640 | 000000000 | 3AB0C35502 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |

*Sample Data*

```
229 190  0000000  02446C80  000000000  756186AA05     0  0  0  0    1  01     01   00
230 191  0000000  0488D901  000000000  6AC30D540B     0  1  0  1    1  11     01   01
231 192  0000000  0911B203  000000000  55861AA817     0  0  0  1    0  10     11   01
232 193  0000000  12236407  000000000  2B0C35502F     0  0  0  0    0  01     10   11
233 194  0000000  2446C80E  000000000  56186AA05F     0  0  0  0    1  00     01   10
234 195  0000000  488D901C  000000000  2C30D540BF     0  1  0  0    1  11     00   01
235 196  0000000  111B2039  000000000  5861AA817E     0  0  0  0    1  10     11   00
236 197  0000000  22364072  000000000  30C35502FD     0  0  0  1    1  01     10   11
237 198  0000000  446C80E4  000000000  6186AA05FB     0  0  0  1    0  01     01   10
238 199  0000000  08D901C8  000000000  430D540BF6     0  1  0  0    0  11     01   01
239 200  0000000  11B20391  000000000  061AA817EC     0  1  0  0    0  10     11   01

Z[0]  = 3D
Z[1]  = C1
Z[2]  = F0
Z[3]  = BB
Z[4]  = 58
Z[5]  = 1E
Z[6]  = 42
Z[7]  = 42
Z[8]  = 4B
Z[9]  = 8E
Z[10] = C1
Z[11] = 2A
Z[12] = 40
Z[13] = 63
Z[14] = 7A
Z[15] = 1E

================================================================================================
    Reload this pattern into the LFSRs
    Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 04B583D
LFSR2  <= 208E1EC1
LFSR3  <= 063C142F0
LFSR4  <= 0F7A2A42BB
C[t+1] <= 10

================================================================================================
    Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240 1    04B583D  208E1EC1  063C142F0  0F7A2A42BB     0  1  0  0    0  10     11   01
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 241 | 2 | 096B07A | 411C3D82 | 0C78285E1 | 1EF4548577 | 1 | 0 | 1 | 1 | 1 | 10 | 10 | 11 |
| 242 | 3 | 12D60F4 | 02387B04 | 18F050BC3 | 3DE8A90AEF | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 243 | 4 | 05AC1E9 | 0470F609 | 11E0A1786 | 7BD15215DF | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 244 | 5 | 0B583D2 | 08E1EC13 | 03C142F0C | 77A2A42BBF | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 245 | 6 | 16B07A5 | 11C3D827 | 078285E18 | 6F4548577E | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 246 | 7 | 0D60F4B | 2387B04F | 0F050BC30 | 5E8A90AEFD | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 247 | 8 | 1AC1E97 | 470F609E | 1E0A17860 | 3D15215DFA | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 248 | 9 | 1583D2E | 0E1EC13D | 1C142F0C0 | 7A2A42BBF4 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 249 | 10 | 0B07A5D | 1C3D827B | 18285E181 | 74548577E9 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 250 | 11 | 160F4BB | 387B04F7 | 1050BC302 | 68A90AEFD2 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 251 | 12 | 0C1E976 | 70F609EE | 00A178605 | 515215DFA5 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 252 | 13 | 183D2ED | 61EC13DD | 0142F0C0B | 22A42BBF4B | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 253 | 14 | 107A5DA | 43D827BA | 0285E1817 | 4548577E97 | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 254 | 15 | 00F4BB4 | 07B04F74 | 050BC302F | 0A90AEFD2E | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 255 | 16 | 01E9769 | 0F609EE8 | 0A178605E | 15215DFA5C | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 256 | 17 | 03D2ED3 | 1EC13DD0 | 142F0C0BD | 2A42BBF4B9 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 257 | 18 | 07A5DA7 | 3D827BA0 | 085E1817B | 548577E972 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 258 | 19 | 0F4BB4F | 7B04F740 | 10BC302F6 | 290AEFD2E5 | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 259 | 20 | 1E9769F | 7609EE80 | 0178605ED | 5215DFA5CA | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 260 | 21 | 1D2ED3F | 6C13DD01 | 02F0C0BDA | 242BBF4B94 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 261 | 22 | 1A5DA7E | 5827BA03 | 05E1817B4 | 48577E9729 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 262 | 23 | 14BB4FC | 304F7407 | 0BC302F69 | 10AEFD2E53 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 263 | 24 | 09769F9 | 609EE80E | 178605ED2 | 215DFA5CA7 | 1 | 1 | 0 | 0 | 0 | 10 | 00 | 01 |
| 264 | 25 | 12ED3F2 | 413DD01C | 0F0C0BDA4 | 42BBF4B94F | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 265 | 26 | 05DA7E5 | 027BA038 | 1E1817B49 | 0577E9729F | 0 | 0 | 1 | 0 | 1 | 01 | 00 | 10 |
| 266 | 27 | 0BB4FCA | 04F74071 | 1C302F693 | 0AEFD2E53F | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 267 | 28 | 1769F95 | 09EE80E3 | 18605ED27 | 15DFA5CA7F | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 268 | 29 | 0ED3F2B | 13DD01C6 | 10C0BDA4F | 2BBF4B94FE | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 269 | 30 | 1DA7E56 | 27BA038D | 01817B49F | 577E9729FD | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 270 | 31 | 1B4FCAD | 4F74071B | 0302F693E | 2EFD2E53FB | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 271 | 32 | 169F95B | 1EE80E37 | 0605ED27D | 5DFA5CA7F7 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 272 | 33 | 0D3F2B7 | 3DD01C6E | 0C0BDA4FB | 3BF4B94FEF | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 273 | 34 | 1A7E56F | 7BA038DC | 1817B49F6 | 77E9729FDE | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 274 | 35 | 14FCADF | 774071B9 | 102F693ED | 6FD2E53FBD | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 275 | 36 | 09F95BE | 6E80E373 | 005ED27DB | 5FA5CA7F7B | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 276 | 37 | 13F2B7C | 5D01C6E7 | 00BDA4FB6 | 3F4B94FEF7 | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 277 | 38 | 07E56F9 | 3A038DCE | 017B49F6C | 7E9729FDEE | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 278 | 39 | 0FCADF2 | 74071B9C | 02F693ED8 | 7D2E53FBDD | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 279 | 40 | 1F95BE5 | 680E3738 | 05ED27DB0 | 7A5CA7F7BA | 1 | 0 | 0 | 0 | 1 | 11 | 10 | 00 |
| 280 | 41 | 1F2B7CA | 501C6E71 | 0BDA4FB60 | 74B94FEF74 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 281 | 42 | 1E56F94 | 2038DCE2 | 17B49F6C0 | 69729FDEE8 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 282 | 43 | 1CADF29 | 4071B9C4 | 0F693ED80 | 52E53FBDD1 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 283 | 44 | 195BE53 | 00E37389 | 1ED27DB01 | 25CA7F7BA3 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 284 | 45 | 12B7CA6 | 01C6E713 | 1DA4FB602 | 4B94FEF747 | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 285 | 46 | 056F94C | 038DCE26 | 1B49F6C04 | 1729FDEE8E | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 286 | 47 | 0ADF299 | 071B9C4D | 1693ED808 | 2E53FBDD1C | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 287 | 48 | 15BE532 | 0E37389A | 0D27DB011 | 5CA7F7BA38 | 0 | 0 | 1 | 1 | 0 | 10 | 10 | 11 |
| 288 | 49 | 0B7CA64 | 1C6E7135 | 1A4FB6022 | 394FEF7471 | 1 | 0 | 1 | 0 | 0 | 01 | 10 | 10 |
| 289 | 50 | 16F94C9 | 38DCE26A | 149F6C044 | 729FDEE8E2 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 290 | 51 | 0DF2993 | 71B9C4D4 | 093ED8089 | 653FBDD1C4 | 1 | 1 | 1 | 0 | 0 | 00 | 01 | 01 |
| 291 | 52 | 1BE5327 | 637389A9 | 127DB0112 | 4A7F7BA388 | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 292 | 53 | 17CA64E | 46E71353 | 04FB60224 | 14FEF74710 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 293 | 54 | 0F94C9C | 0DCE26A6 | 09F6C0448 | 29FDEE8E21 | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 294 | 55 | 1F29939 | 1B9C4D4D | 13ED80890 | 53FBDD1C42 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 295 | 56 | 1E53272 | 37389A9A | 07DB01121 | 27F7BA3884 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 296 | 57 | 1CA64E5 | 6E713534 | 0FB602242 | 4FEF747108 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 297 | 58 | 194C9CB | 5CE26A69 | 1F6C04485 | 1FDEE8E210 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 298 | 59 | 1299397 | 39C4D4D3 | 1ED80890A | 3FBDD1C420 | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 299 | 60 | 053272F | 7389A9A6 | 1DB011214 | 7F7BA38840 | 0 | 1 | 1 | 0 | 0 | 11 | 00 | 11 |
| 300 | 61 | 0A64E5E | 6713534C | 1B6022428 | 7EF7471081 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 301 | 62 | 14C9CBD | 4E26A699 | 16C044850 | 7DEE8E2102 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 302 | 63 | 099397A | 1C4D4D32 | 0D80890A0 | 7BDD1C4205 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 303 | 64 | 13272F4 | 389A9A65 | 1B0112141 | 77BA38840B | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 304 | 65 | 064E5E8 | 713534CB | 160224283 | 6F74710817 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 305 | 66 | 0C9CBD1 | 626A6997 | 0C0448507 | 5EE8E2102E | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 306 | 67 | 19397A3 | 44D4D32E | 180890A0E | 3DD1C4205C | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 307 | 68 | 1272F46 | 09A9A65D | 10112141D | 7BA38840B8 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 308 | 69 | 04E5E8C | 13534CBA | 00224283A | 7747108171 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 309 | 70 | 09CBD19 | 26A69975 | 004485075 | 6E8E2102E3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 310 | 71 | 1397A32 | 4D4D32EB | 00890A0EA | 5D1C4205C7 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 311 | 72 | 072F465 | 1A9A65D7 | 0112141D5 | 3A38840B8F | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 312 | 73 | 0E5E8CA | 3534CBAF | 0224283AA | 747108171F | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 313 | 74 | 1CBD194 | 6A69975E | 044850755 | 68E2102E3E | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 314 | 75 | 197A329 | 54D32EBC | 0890A0EAB | 51C4205C7D | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 315 | 76 | 12F4653 | 29A65D79 | 112141D56 | 238840B8FA | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 316 | 77 | 05E8CA6 | 534CBAF2 | 024283AAD | 47108171F4 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 317 | 78 | 0BD194D | 269975E5 | 04850755B | 0E2102E3E9 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 318 | 79 | 17A329A | 4D32EBCB | 090A0EAB6 | 1C4205C7D2 | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 319 | 80 | 0F46535 | 1A65D797 | 12141D56D | 38840B8FA5 | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 320 | 81 | 1E8CA6A | 34CBAF2F | 04283AADA | 7108171F4B | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 321 | 82 | 1D194D5 | 69975E5F | 0850755B4 | 62102E3E97 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 322 | 83 | 1A329AA | 532EBCBF | 10A0EAB68 | 44205C7D2F | 1 | 0 | 0 | 0 | 0 | 11 | 01 | 01 |
| 323 | 84 | 1465355 | 265D797F | 0141D56D1 | 0840B8FA5E | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 324 | 85 | 08CA6AB | 4CBAF2FF | 0283AADA2 | 108171F4BC | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 325 | 86 | 1194D56 | 1975E5FF | 050755B45 | 2102E3E979 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 326 | 87 | 0329AAD | 32EBCBFF | 0A0EAB68A | 4205C7D2F3 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 327 | 88 | 065355A | 65D797FF | 141D56D14 | 040B8FA5E7 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 328 | 89 | 0CA6AB4 | 4BAF2FFF | 083AADA28 | 08171F4BCF | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 329 | 90 | 194D569 | 175E5FFF | 10755B450 | 102E3E979E | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 330 | 91 | 129AAD3 | 2EBCBFFF | 00EAB68A1 | 205C7D2F3C | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 331 | 92 | 05355A6 | 5D797FFF | 01D56D142 | 40B8FA5E78 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 332 | 93 | 0A6AB4D | 3AF2FFFE | 03AADA285 | 0171F4BCF1 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 333 | 94 | 14D569B | 75E5FFFD | 0755B450A | 02E3E979E2 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 334 | 95 | 09AAD37 | 6BCBFFFA | 0EAB68A15 | 05C7D2F3C4 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 335 | 96 | 1355A6E | 5797FFF4 | 1D56D142A | 0B8FA5E788 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 336 | 97 | 06AB4DC | 2F2FFFE8 | 1AADA2854 | 171F4BCF11 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 337 | 98 | 0D569B8 | 5E5FFFD0 | 155B450A9 | 2E3E979E23 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 338 | 99 | 1AAD370 | 3CBFFFA1 | 0AB68A153 | 5C7D2F3C46 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 339 | 100 | 155A6E0 | 797FFF43 | 156D142A7 | 38FA5E788D | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 340 | 101 | 0AB4DC0 | 72FFFE87 | 0ADA2854E | 71F4BCF11B | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 341 | 102 | 1569B81 | 65FFFD0E | 15B450A9D | 63E979E236 | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 342 | 103 | 0AD3703 | 4BFFFA1C | 0B68A153B | 47D2F3C46C | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 343 | 104 | 15A6E07 | 17FFF438 | 16D142A76 | 0FA5E788D8 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 344 | 105 | 0B4DC0F | 2FFFE870 | 0DA2854EC | 1F4BCF11B0 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 345 | 106 | 169B81F | 5FFFD0E1 | 1B450A9D8 | 3E979E2360 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 346 | 107 | 0D3703F | 3FFFA1C3 | 168A153B0 | 7D2F3C46C1 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 347 | 108 | 1A6E07E | 7FFF4386 | 0D142A761 | 7A5E788D83 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 348 | 109 | 14DC0FD | 7FFE870C | 1A2854EC2 | 74BCF11B07 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 349 | 110 | 09B81FB | 7FFD0E19 | 1450A9D84 | 6979E2360E | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 350 | 111 | 13703F6 | 7FFA1C33 | 08A153B09 | 52F3C46C1C | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 351 | 112 | 06E07EC | 7FF43867 | 1142A7612 | 25E788D838 | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 352 | 113 | 0DC0FD8 | 7FE870CF | 02854EC25 | 4BCF11B071 | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |
| 353 | 114 | 1B81FB1 | 7FD0E19E | 050A9D84B | 179E2360E3 | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 354 | 115 | 1703F62 | 7FA1C33D | 0A153B096 | 2F3C46C1C7 | 0 | 1 | 1 | 0 | 0 | 11 | 00 | 11 |
| 355 | 116 | 0E07EC4 | 7F43867B | 142A7612C | 5E788D838E | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 356 | 117 | 1C0FD88 | 7E870CF6 | 0854EC259 | 3CF11B071C | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 357 | 118 | 181FB11 | 7D0E19ED | 10A9D84B3 | 79E2360E38 | 1 | 0 | 0 | 1 | 1 | 11 | 01 | 01 |
| 358 | 119 | 103F622 | 7A1C33DA | 0153B0967 | 73C46C1C71 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 359 | 120 | 007EC45 | 743867B5 | 02A7612CE | 6788D838E3 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 360 | 121 | 00FD88B | 6870CF6B | 054EC259C | 4F11B071C6 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 361 | 122 | 01FB117 | 50E19ED7 | 0A9D84B38 | 1E2360E38C | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 362 | 123 | 03F622F | 21C33DAE | 153B09671 | 3C46C1C718 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 363 | 124 | 07EC45F | 43867B5C | 0A7612CE2 | 788D838E30 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 364 | 125 | 0FD88BF | 070CF6B9 | 14EC259C4 | 711B071C61 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |

### 1.1.3 Second set of sample data

```
Initial values for the key, BD_ADDR and clock

K_session[0]  = 00  K_session[1]  = 00  K_session[2]  = 00  K_session[3]  = 00
K_session[4]  = 00  K_session[5]  = 00  K_session[6]  = 00  K_session[7]  = 00
K_session[8]  = 00  K_session[9]  = 00  K_session[10] = 00  K_session[11] = 00
K_session[12] = 00  K_session[13] = 00  K_session[14] = 00  K_session[15] = 00

BD_ADDR_C[0] = 00  BD_ADDR_C[1] = 00  BD_ADDR_C[2] = 00
BD_ADDR_C[3] = 00  BD_ADDR_C[4] = 00  BD_ADDR_C[5] = 00

CL[0] = 00  CL[1] = 00  CL[2] = 00  CL[3] = 03

The corresponding values of CLK are 0x600_0000 and 0xE00_0000.

================================================================================================
    Fill LFSRs with initial data
================================================================================================

t  clk#  LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

0     0  0000000* 00000000* 000000000* 0000000000*    0  0  0  0    0  00     00   00
1     1  0000001* 00000001* 000000001* 0000000001*    0  0  0  0    0  00     00   00
2     2  0000002* 00000002* 000000002* 0000000003*    0  0  0  0    0  00     00   00
3     3  0000004* 00000004* 000000004* 0000000007*    0  0  0  0    0  00     00   00
4     4  0000008* 00000008* 000000008* 000000000E*    0  0  0  0    0  00     00   00
5     5  0000010* 00000010* 000000010* 000000001C*    0  0  0  0    0  00     00   00
6     6  0000020* 00000020* 000000020* 0000000038*    0  0  0  0    0  00     00   00
7     7  0000040* 00000040* 000000040* 0000000070*    0  0  0  0    0  00     00   00
8     8  0000080* 00000080* 000000080* 00000000E0*    0  0  0  0    0  00     00   00
9     9  0000100* 00000100* 000000100* 00000001C0*    0  0  0  0    0  00     00   00
10   10  0000200* 00000200* 000000200* 0000000380*    0  0  0  0    0  00     00   00
11   11  0000400* 00000400* 000000400* 0000000700*    0  0  0  0    0  00     00   00
12   12  0000800* 00000800* 000000800* 0000000E00*    0  0  0  0    0  00     00   00
13   13  0001000* 00001000* 000001000* 0000001C00*    0  0  0  0    0  00     00   00
14   14  0002000* 00002000* 000002000* 0000003800*    0  0  0  0    0  00     00   00
15   15  0004000* 00004000* 000004000* 0000007000*    0  0  0  0    0  00     00   00
16   16  0008000* 00008000* 000008000* 000000E000*    0  0  0  0    0  00     00   00
17   17  0010000* 00010000* 000010000* 000001C000*    0  0  0  0    0  00     00   00
18   18  0020000* 00020000* 000020000* 0000038000*    0  0  0  0    0  00     00   00
19   19  0040000* 00040000* 000040000* 0000070000*    0  0  0  0    0  00     00   00
20   20  0080000* 00080000* 000080000* 00000E0000*    0  0  0  0    0  00     00   00
```

*Sample Data*

```
21   21  0100000* 00100000* 000100000* 00001C0000*    0  0  0  0    0  00      00   00
22   22  0200000* 00200000* 000200000* 0000380000*    0  0  0  0    0  00      00   00
23   23  0400000* 00400000* 000400000* 0000700000*    0  0  0  0    0  00      00   00
24   24  0800000* 00800000* 000800000* 0000E00000*    1  1  0  0    0  01      00   00
25   25  1000000* 01000000* 001000000* 0001C00000*    0  0  0  0    0  00      00   00
26   26  0000001  02000000* 002000000* 0003800000*    0  0  0  0    0  00      00   00
27   27  0000002  04000000* 004000000* 0007000000*    0  0  0  0    0  00      00   00
28   28  0000004  08000000* 008000000* 000E000000*    0  0  0  0    0  00      00   00
29   29  0000008  10000000* 010000000* 001C000000*    0  0  0  0    0  00      00   00
30   30  0000010  20000000* 020000000* 0038000000*    0  0  0  0    0  00      00   00
31   31  0000020  40000000* 040000000* 0070000000*    0  0  0  0    0  00      00   00
32   32  0000040  00000001  080000000* 00E0000000*    0  0  1  1    0  01      00   00
33   33  0000080  00000002  100000000* 01C0000000*    0  0  0  1    1  00      00   00
34   34  0000101  00000004  000000001  0380000000*    0  0  0  1    1  00      00   00
35   35  0000202  00000008  000000002  0700000000*    0  0  0  0    0  00      00   00
36   36  0000404  00000010  000000004  0E00000000*    0  0  0  0    0  00      00   00
37   37  0000808  00000020  000000008  1C00000000*    0  0  0  0    0  00      00   00
38   38  0001011  00000040  000000011  3800000000*    0  0  0  0    0  00      00   00
39   39  0002022  00000080  000000022  7000000000*    0  0  0  0    0  00      00   00
================================================================================================
    Start clocking Summation Combiner
================================================================================================
40    1  0004044  00000100  000000044  6000000001     0  0  0  0    0  00      00   00
41    2  0008088  00000200  000000088  4000000003     0  0  0  0    0  00      00   00
42    3  0010111  00000400  000000111  0000000007     0  0  0  0    0  00      00   00
43    4  0020222  00000800  000000222  000000000E     0  0  0  0    0  00      00   00
44    5  0040444  00001001  000000444  000000001D     0  0  0  0    0  00      00   00
45    6  0080888  00002002  000000888  000000003B     0  0  0  0    0  00      00   00
46    7  0101111  00004004  000001111  0000000077     0  0  0  0    0  00      00   00
47    8  0202222  00008008  000002222  00000000EE     0  0  0  0    0  00      00   00
48    9  0404444  00010011  000004444  00000001DD     0  0  0  0    0  00      00   00
49   10  0808888  00020022  000008888  00000003BB     1  0  0  0    1  00      00   00
50   11  1011110  00040044  000011111  0000000777     0  0  0  0    0  00      00   00
51   12  0022221  00080088  000022222  0000000EEE     0  0  0  0    0  00      00   00
52   13  0044442  00100110  000044444  0000001DDD     0  0  0  0    0  00      00   00
53   14  0088884  00200220  000088888  0000003BBB     0  0  0  0    0  00      00   00
54   15  0111109  00400440  000111111  0000007777     0  0  0  0    0  00      00   00
55   16  0222212  00800880  000222222  000000EEEE     0  1  0  0    1  00      00   00
56   17  0444424  01001100  000444444  000001DDDD     0  0  0  0    0  00      00   00
57   18  0888848  02002200  000888888  000003BBBB     1  0  0  0    1  00      00   00
58   19  1111090  04004400  001111110  0000077777     0  0  0  0    0  00      00   00
59   20  0222120  08008800  002222220  00000EEEEE     0  0  0  0    0  00      00   00
60   21  0444240  10011000  004444440  00001DDDDD     0  0  0  0    0  00      00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 61 | 22 | 0888480 | 20022000 | 008888880 | 00003BBBBB | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 62 | 23 | 1110900 | 40044000 | 011111100 | 0000777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 63 | 24 | 0221200 | 00088001 | 022222200 | 0000EEEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 64 | 25 | 0442400 | 00110003 | 044444400 | 0001DDDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 65 | 26 | 0884800 | 00220006 | 088888800 | 0003BBBBBB | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 66 | 27 | 1109000 | 0044000C | 111111000 | 0007777777 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 67 | 28 | 0212001 | 00880018 | 022222001 | 000EEEEEEE | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 68 | 29 | 0424002 | 01100031 | 044444002 | 001DDDDDDC | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 69 | 30 | 0848004 | 02200062 | 088888004 | 003BBBBBB8 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 70 | 31 | 1090008 | 044000C4 | 111110008 | 0077777770 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 71 | 32 | 0120010 | 08800188 | 022220010 | 00EEEEEEE0 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 72 | 33 | 0240020 | 11000311 | 044440020 | 01DDDDDDC1 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 73 | 34 | 0480040 | 22000622 | 088880040 | 03BBBBBB83 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 74 | 35 | 0900081 | 44000C44 | 111100080 | 0777777707 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 75 | 36 | 1200103 | 08001888 | 022200101 | 0EEEEEEE0E | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 76 | 37 | 0400207 | 10003111 | 044400202 | 1DDDDDDC1D | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 77 | 38 | 080040E | 20006222 | 088800404 | 3BBBBBB83B | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 78 | 39 | 100081C | 4000C444 | 111000808 | 7777777077 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 79 | 40 | 0001038 | 00018888 | 022001010 | 6EEEEEE0EF | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 80 | 41 | 0002070 | 00031110 | 044002020 | 5DDDDDC1DE | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 81 | 42 | 00040E0 | 00062220 | 088004040 | 3BBBBB83BC | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 82 | 43 | 00081C1 | 000C4440 | 110008081 | 7777770779 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 83 | 44 | 0010383 | 00188880 | 020010103 | 6EEEEE0EF2 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 84 | 45 | 0020707 | 00311100 | 040020206 | 5DDDDC1DE5 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 85 | 46 | 0040E0E | 00622200 | 08004040C | 3BBBB83BCB | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 86 | 47 | 0081C1D | 00C44400 | 100080819 | 7777707797 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 87 | 48 | 010383A | 01888801 | 000101032 | 6EEEE0EF2F | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 88 | 49 | 0207075 | 03111003 | 000202064 | 5DDDC1DE5E | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 89 | 50 | 040E0EA | 06222006 | 0004040C8 | 3BBB83BCBC | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 90 | 51 | 081C1D5 | 0C44400C | 000808191 | 7777077979 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 91 | 52 | 10383AB | 18888018 | 001010323 | 6EEE0EF2F2 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 92 | 53 | 0070756 | 31110030 | 002020646 | 5DDC1DE5E5 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 93 | 54 | 00E0EAC | 62220060 | 004040C8C | 3BB83BCBCB | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 94 | 55 | 01C1D59 | 444400C1 | 008081919 | 7770779797 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 95 | 56 | 0383AB2 | 08880183 | 010103232 | 6EE0EF2F2F | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 96 | 57 | 0707565 | 11100307 | 020206464 | 5DC1DE5E5F | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 97 | 58 | 0E0EACA | 2220060E | 04040C8C8 | 3B83BCBCBF | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 98 | 59 | 1C1D594 | 44400C1C | 080819191 | 770779797E | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 99 | 60 | 183AB28 | 08801838 | 101032323 | 6E0EF2F2FC | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 100 | 61 | 1075650 | 11003070 | 002064647 | 5C1DE5E5F8 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 101 | 62 | 00EACA1 | 220060E0 | 0040C8C8E | 383BCBCBF0 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 102 | 63 | 01D5943 | 4400C1C0 | 00819191D | 70779797E0 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 103 | 64 | 03AB286 | 08018380 | 01032323A | 60EF2F2FC1 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |

*Sample Data*

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 104 | 65 | 075650C | 10030701 | 020646475 | 41DE5E5F82 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 105 | 66 | 0EACA18 | 20060E02 | 040C8C8EA | 03BCBCBF04 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 106 | 67 | 1D59430 | 400C1C05 | 0819191D4 | 0779797E09 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 107 | 68 | 1AB2861 | 0018380A | 1032323A9 | 0EF2F2FC12 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 108 | 69 | 15650C3 | 00307015 | 006464752 | 1DE5E5F825 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 109 | 70 | 0ACA186 | 0060E02A | 00C8C8EA4 | 3BCBCBF04B | 1 | 0 | 0 | 1 | 1 | 00 | 11 | 10 |
| 110 | 71 | 159430C | 00C1C055 | 019191D48 | 779797E097 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 111 | 72 | 0B28618 | 018380AA | 032323A90 | 6F2F2FC12F | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 112 | 73 | 1650C30 | 03070154 | 064647520 | 5E5E5F825E | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 113 | 74 | 0CA1860 | 060E02A8 | 0C8C8EA40 | 3CBCBF04BC | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 114 | 75 | 19430C0 | 0C1C0550 | 19191D480 | 79797E0979 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 115 | 76 | 1286180 | 18380AA0 | 12323A900 | 72F2FC12F2 | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 116 | 77 | 050C301 | 30701541 | 046475201 | 65E5F825E5 | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 117 | 78 | 0A18602 | 60E02A82 | 08C8EA402 | 4BCBF04BCB | 1 | 1 | 1 | 1 | 1 | 10 | 11 | 11 |
| 118 | 79 | 1430C04 | 41C05505 | 1191D4804 | 1797E09796 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 119 | 80 | 0861808 | 0380AA0A | 0323A9008 | 2F2FC12F2C | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 120 | 81 | 10C3011 | 07015415 | 064752011 | 5E5F825E59 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 121 | 82 | 0186022 | 0E02A82A | 0C8EA4022 | 3CBF04BCB2 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 122 | 83 | 030C045 | 1C055054 | 191D48044 | 797E097964 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 123 | 84 | 061808A | 380AA0A8 | 123A90088 | 72FC12F2C9 | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 124 | 85 | 0C30115 | 70154151 | 047520111 | 65F825E593 | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 125 | 86 | 186022A | 602A82A3 | 08EA40222 | 4BF04BCB26 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 126 | 87 | 10C0455 | 40550546 | 11D480444 | 17E097964C | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 127 | 88 | 01808AA | 00AA0A8D | 03A900888 | 2FC12F2C99 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 128 | 89 | 0301155 | 0154151A | 075201111 | 5F825E5932 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 129 | 90 | 06022AA | 02A82A34 | 0EA402222 | 3F04BCB264 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 130 | 91 | 0C04555 | 05505468 | 1D4804445 | 7E097964C9 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 131 | 92 | 1808AAA | 0AA0A8D0 | 1A900888A | 7C12F2C992 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 132 | 93 | 1011555 | 154151A1 | 152011115 | 7825E59324 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 133 | 94 | 0022AAB | 2A82A342 | 0A402222B | 704BCB2648 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 134 | 95 | 0045556 | 55054684 | 148044457 | 6097964C91 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 135 | 96 | 008AAAC | 2A0A8D09 | 0900888AE | 412F2C9923 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 136 | 97 | 0115559 | 54151A12 | 12011115D | 025E593246 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 137 | 98 | 022AAB2 | 282A3424 | 0402222BA | 04BCB2648D | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 138 | 99 | 0455564 | 50546848 | 080444575 | 097964C91A | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 139 | 100 | 08AAAC8 | 20A8D090 | 100888AEA | 12F2C99235 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 140 | 101 | 1155591 | 4151A120 | 0011115D5 | 25E593246A | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 141 | 102 | 02AAB22 | 02A34240 | 002222BAA | 4BCB2648D5 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 142 | 103 | 0555644 | 05468481 | 004445755 | 17964C91AB | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 143 | 104 | 0AAAC88 | 0A8D0903 | 00888AEAA | 2F2C992357 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 144 | 105 | 1555911 | 151A1206 | 011115D55 | 5E593246AE | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 145 | 106 | 0AAB222 | 2A34240C | 02222BAAA | 3CB2648D5C | 1 | 0 | 0 | 1 | 1 | 11 | 01 | 01 |
| 146 | 107 | 1556445 | 54684818 | 044457555 | 7964C91AB8 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 147 | 108 | 0AAC88B | 28D09030 | 0888AEAAA | 72C9923571 | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 148 | 109 | 1559117 | 51A12060 | 11115D555 | 6593246AE2 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 149 | 110 | 0AB222F | 234240C0 | 0222BAAAB | 4B2648D5C5 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 150 | 111 | 156445F | 46848180 | 044575557 | 164C91AB8A | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 151 | 112 | 0AC88BF | 0D090301 | 088AEAAAE | 2C99235714 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 152 | 113 | 159117F | 1A120602 | 1115D555D | 593246AE28 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 153 | 114 | 0B222FE | 34240C04 | 022BAAABA | 32648D5C51 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 154 | 115 | 16445FD | 68481809 | 045755574 | 64C91AB8A2 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 155 | 116 | 0C88BFA | 50903012 | 08AEAAAE8 | 4992357144 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 156 | 117 | 19117F5 | 21206024 | 115D555D1 | 13246AE288 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 157 | 118 | 1222FEA | 4240C048 | 02BAAABA2 | 2648D5C511 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 158 | 119 | 0445FD5 | 04818090 | 057555744 | 4C91AB8A23 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 159 | 120 | 088BFAA | 09030120 | 0AEAAAE88 | 1923571446 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 160 | 121 | 1117F55 | 12060240 | 15D555D11 | 3246AE288D | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 161 | 122 | 022FEAA | 240C0480 | 0BAAABA22 | 648D5C511B | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 162 | 123 | 045FD54 | 48180900 | 175557444 | 491AB8A237 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 163 | 124 | 08BFAA9 | 10301200 | 0EAAAE889 | 123571446F | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 164 | 125 | 117F553 | 20602400 | 1D555D113 | 246AE288DF | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 165 | 126 | 02FEAA7 | 40C04800 | 1AAABA227 | 48D5C511BE | 0 | 1 | 1 | 1 | 1 | 10 | 00 | 01 |
| 166 | 127 | 05FD54F | 01809001 | 15557444F | 11AB8A237D | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 167 | 128 | 0BFAA9F | 03012002 | 0AAAE889E | 23571446FA | 1 | 0 | 1 | 0 | 0 | 00 | 00 | 10 |
| 168 | 129 | 17F553F | 06024004 | 1555D113D | 46AE288DF5 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 169 | 130 | 0FEAA7E | 0C048008 | 0AABA227A | 0D5C511BEA | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 170 | 131 | 1FD54FC | 18090011 | 1557444F5 | 1AB8A237D5 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 171 | 132 | 1FAA9F9 | 30120022 | 0AAE889EB | 3571446FAA | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 172 | 133 | 1F553F2 | 60240044 | 155D113D7 | 6AE288DF55 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 173 | 134 | 1EAA7E4 | 40480089 | 0ABA227AE | 55C511BEAA | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 174 | 135 | 1D54FC9 | 00900113 | 157444F5D | 2B8A237D54 | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 175 | 136 | 1AA9F93 | 01200227 | 0AE889EBA | 571446FAA8 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 176 | 137 | 1553F26 | 0240044E | 15D113D75 | 2E288DF550 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 177 | 138 | 0AA7E4C | 0480089C | 0BA227AEA | 5C511BEAA0 | 1 | 1 | 1 | 0 | 0 | 00 | 11 | 00 |
| 178 | 139 | 154FC98 | 09001138 | 17444F5D4 | 38A237D540 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 179 | 140 | 0A9F931 | 12002270 | 0E889EBA9 | 71446FAA81 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 180 | 141 | 153F262 | 240044E0 | 1D113D753 | 6288DF5503 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 181 | 142 | 0A7E4C5 | 480089C0 | 1A227AEA7 | 4511BEAA06 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 182 | 143 | 14FC98B | 10011381 | 1444F5D4F | 0A237D540D | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 183 | 144 | 09F9316 | 20022702 | 0889EBA9E | 1446FAA81A | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 184 | 145 | 13F262D | 40044E04 | 1113D753D | 288DF55035 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 185 | 146 | 07E4C5A | 00089C08 | 0227AEA7A | 511BEAA06A | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 186 | 147 | 0FC98B4 | 00113810 | 044F5D4F5 | 2237D540D5 | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 187 | 148 | 1F93169 | 00227021 | 089EBA9EB | 446FAA81AA | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 188 | 149 | 1F262D2 | 0044E042 | 113D753D7 | 08DF550355 | 1 | 0 | 0 | 1 | 1 | 10 | 11 | 01 |
| 189 | 150 | 1E4C5A4 | 0089C085 | 027AEA7AE | 11BEAA06AA | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 190 | 151 | 1C98B48 | 0113810A | 04F5D4F5C | 237D540D54 | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 191 | 152 | 1931691 | 02270215 | 09EBA9EB8 | 46FAA81AA9 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 192 | 153 | 1262D22 | 044E042A | 13D753D71 | 0DF5503553 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 193 | 154 | 04C5A44 | 089C0854 | 07AEA7AE2 | 1BEAA06AA7 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 194 | 155 | 098B488 | 113810A8 | 0F5D4F5C4 | 37D540D54E | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 195 | 156 | 1316910 | 22702150 | 1EBA9EB89 | 6FAA81AA9D | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 196 | 157 | 062D220 | 44E042A0 | 1D753D712 | 5F5503553A | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 197 | 158 | 0C5A440 | 09C08540 | 1AEA7AE25 | 3EAA06AA75 | 1 | 1 | 1 | 1 | 1 | 10 | 11 | 11 |
| 198 | 159 | 18B4880 | 13810A80 | 15D4F5C4B | 7D540D54EA | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 199 | 160 | 1169100 | 27021500 | 0BA9EB897 | 7AA81AA9D5 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 200 | 161 | 02D2201 | 4E042A00 | 1753D712E | 75503553AB | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 201 | 162 | 05A4403 | 1C085400 | 0EA7AE25C | 6AA06AA756 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 202 | 163 | 0B48807 | 3810A800 | 1D4F5C4B8 | 5540D54EAC | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 203 | 164 | 169100F | 70215000 | 1A9EB8971 | 2A81AA9D58 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 204 | 165 | 0D2201E | 6042A001 | 153D712E3 | 5503553AB0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 205 | 166 | 1A4403C | 40854002 | 0A7AE25C6 | 2A06AA7561 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 206 | 167 | 1488079 | 010A8004 | 14F5C4B8D | 540D54EAC3 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 207 | 168 | 09100F2 | 02150009 | 09EB8971B | 281AA9D586 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 208 | 169 | 12201E5 | 042A0012 | 13D712E37 | 503553AB0C | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 209 | 170 | 04403CA | 08540024 | 07AE25C6E | 206AA75618 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 210 | 171 | 0880795 | 10A80048 | 0F5C4B8DD | 40D54EAC30 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 211 | 172 | 1100F2A | 21500091 | 1EB8971BA | 01AA9D5861 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 212 | 173 | 0201E54 | 42A00122 | 1D712E374 | 03553AB0C3 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 213 | 174 | 0403CA9 | 05400244 | 1AE25C6E9 | 06AA756186 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 214 | 175 | 0807952 | 0A800488 | 15C4B8DD3 | 0D54EAC30D | 1 | 1 | 0 | 0 | 1 | 11 | 11 | 11 |
| 215 | 176 | 100F2A5 | 15000911 | 0B8971BA6 | 1AA9D5861A | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 216 | 177 | 001E54A | 2A001223 | 1712E374C | 3553AB0C35 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 217 | 178 | 003CA94 | 54002446 | 0E25C6E98 | 6AA756186A | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 218 | 179 | 0079528 | 2800488D | 1C4B8DD31 | 554EAC30D5 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 219 | 180 | 00F2A50 | 5000911B | 18971BA62 | 2A9D5861AA | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 220 | 181 | 01E54A0 | 20012236 | 112E374C4 | 553AB0C355 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 221 | 182 | 03CA940 | 4002446C | 025C6E988 | 2A756186AA | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 222 | 183 | 0795280 | 000488D9 | 04B8DD310 | 54EAC30D54 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 223 | 184 | 0F2A500 | 000911B2 | 0971BA620 | 29D5861AA8 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 224 | 185 | 1E54A00 | 00122364 | 12E374C40 | 53AB0C3550 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 225 | 186 | 1CA9400 | 002446C8 | 05C6E9880 | 2756186AA0 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 226 | 187 | 1952800 | 00488D90 | 0B8DD3101 | 4EAC30D540 | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 227 | 188 | 12A5000 | 00911B20 | 171BA6202 | 1D5861AA81 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 228 | 189 | 054A000 | 01223640 | 0E374C404 | 3AB0C35502 | 0 | 0 | 1 | 1 | 0 | 10 | 10 | 11 |
| 229 | 190 | 0A94000 | 02446C80 | 1C6E98808 | 756186AA05 | 1 | 0 | 1 | 0 | 0 | 01 | 10 | 10 |
| 230 | 191 | 1528001 | 0488D901 | 18DD31011 | 6AC30D540B | 0 | 1 | 1 | 1 | 0 | 10 | 01 | 10 |
| 231 | 192 | 0A50003 | 0911B203 | 11BA62023 | 55861AA817 | 1 | 0 | 0 | 1 | 0 | 11 | 10 | 01 |
| 232 | 193 | 14A0006 | 12236407 | 0374C4047 | 2B0C35502F | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |

*Sample Data*

```
233 194  094000C  2446C80E  06E98808E  56186AA05F     1  0  0  0    0  11     11   11
234 195  1280018  488D901C  0DD31011D  2C30D540BF     0  1  1  0    1  11     11   11
235 196  0500030  111B2039  1BA62023A  5861AA817E     0  0  1  0    0  11     11   11
236 197  0A00060  22364072  174C40475  30C35502FD     1  0  0  1    1  11     11   11
237 198  14000C0  446C80E4  0E98808EA  6186AA05FB     0  0  1  1    1  11     11   11
238 199  0800180  08D901C8  1D31011D5  430D540BF6     1  1  1  0    0  10     11   11
239 200  1000301  11B20391  1A62023AB  061AA817EC     0  1  1  0    0  10     10   11

Z[0] = 25
Z[1] = 45
Z[2] = 6B
Z[3] = 55
Z[4] = 5F
Z[5] = C2
Z[6] = 20
Z[7] = E5
Z[8] = C4
Z[9] = F8
Z[10] = 3A
Z[11] = F1
Z[12] = FF
Z[13] = 89
Z[14] = 02
Z[15] = 35

================================================================================================
    Reload this pattern into the LFSRs
    Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1 <= 1C45F25
LFSR2 <= 7FF8C245
LFSR3 <= 1893A206B
LFSR4 <= 1A02F1E555
C[t+1] <= 10

================================================================================================
    Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240   1  1C45F25  7FF8C245  1893A206B  1A02F1E555     1  1  1  0    1  10     10   11
241   2  188BE4A  7FF1848B  1127440D7  3405E3CAAB     1  1  0  0    0  01     10   10
242   3  1117C95  7FE30917  024E881AF  680BC79557     0  1  0  0    0  01     01   10
243   4  022F92B  7FC6122F  049D1035E  50178F2AAF     0  1  0  0    0  11     01   01
244   5  045F257  7F8C245E  093A206BD  202F1E555E     0  1  1  0    1  10     11   01
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 245 | 6 | 08BE4AE | 7F1848BC | 127440D7A | 405E3CAABC | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 246 | 7 | 117C95C | 7E309178 | 04E881AF4 | 00BC795579 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 247 | 8 | 02F92B8 | 7C6122F0 | 09D1035E8 | 0178F2AAF2 | 0 | 0 | 1 | 0 | 0 | 11 | 01 | 01 |
| 248 | 9 | 05F2570 | 78C245E1 | 13A206BD0 | 02F1E555E5 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 249 | 10 | 0BE4AE1 | 71848BC2 | 07440D7A0 | 05E3CAABCA | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 250 | 11 | 17C95C3 | 63091784 | 0E881AF40 | 0BC7955795 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 251 | 12 | 0F92B87 | 46122F09 | 1D1035E80 | 178F2AAF2B | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 252 | 13 | 1F2570F | 0C245E12 | 1A206BD01 | 2F1E555E56 | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |
| 253 | 14 | 1E4AE1F | 1848BC25 | 1440D7A03 | 5E3CAABCAC | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 254 | 15 | 1C95C3E | 3091784A | 0881AF407 | 3C79557958 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 255 | 16 | 192B87D | 6122F094 | 11035E80F | 78F2AAF2B1 | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 256 | 17 | 12570FA | 4245E128 | 0206BD01E | 71E555E562 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 257 | 18 | 04AE1F4 | 048BC250 | 040D7A03D | 63CAABCAC5 | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 258 | 19 | 095C3E8 | 091784A0 | 081AF407A | 479557958A | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 259 | 20 | 12B87D1 | 122F0941 | 1035E80F4 | 0F2AAF2B14 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 260 | 21 | 0570FA3 | 245E1283 | 006BD01E9 | 1E555E5628 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 261 | 22 | 0AE1F46 | 48BC2506 | 00D7A03D2 | 3CAABCAC50 | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 262 | 23 | 15C3E8C | 11784A0C | 01AF407A5 | 79557958A0 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 263 | 24 | 0B87D18 | 22F09419 | 035E80F4A | 72AAF2B140 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 264 | 25 | 170FA30 | 45E12832 | 06BD01E94 | 6555E56280 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 265 | 26 | 0E1F460 | 0BC25065 | 0D7A03D28 | 4AABCAC501 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 266 | 27 | 1C3E8C0 | 1784A0CB | 1AF407A50 | 1557958A03 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 267 | 28 | 187D181 | 2F094196 | 15E80F4A0 | 2AAF2B1406 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 268 | 29 | 10FA302 | 5E12832C | 0BD01E941 | 555E56280C | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 269 | 30 | 01F4604 | 3C250658 | 17A03D283 | 2ABCAC5019 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 270 | 31 | 03E8C09 | 784A0CB0 | 0F407A506 | 557958A033 | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 271 | 32 | 07D1812 | 70941960 | 1E80F4A0C | 2AF2B14066 | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 272 | 33 | 0FA3024 | 612832C1 | 1D01E9419 | 55E56280CD | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 273 | 34 | 1F46049 | 42506583 | 1A03D2832 | 2BCAC5019A | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 274 | 35 | 1E8C093 | 04A0CB07 | 1407A5065 | 57958A0335 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 275 | 36 | 1D18127 | 0941960F | 080F4A0CB | 2F2B14066B | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 276 | 37 | 1A3024F | 12832C1F | 101E94196 | 5E56280CD7 | 1 | 1 | 0 | 0 | 0 | 00 | 10 | 00 |
| 277 | 38 | 146049F | 2506583E | 003D2832C | 3CAC5019AE | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 278 | 39 | 08C093E | 4A0CB07D | 007A50658 | 7958A0335D | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 279 | 40 | 118127C | 141960FA | 00F4A0CB0 | 72B14066BA | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 280 | 41 | 03024F8 | 2832C1F4 | 01E941961 | 656280CD74 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 281 | 42 | 06049F1 | 506583E9 | 03D2832C2 | 4AC5019AE9 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 282 | 43 | 0C093E2 | 20CB07D2 | 07A506585 | 158A0335D3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 283 | 44 | 18127C5 | 41960FA5 | 0F4A0CB0B | 2B14066BA7 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 284 | 45 | 1024F8A | 032C1F4B | 1E9419616 | 56280CD74F | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 285 | 46 | 0049F15 | 06583E97 | 1D2832C2C | 2C5019AE9F | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 286 | 47 | 0093E2B | 0CB07D2F | 1A5065859 | 58A0335D3E | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 287 | 48 | 0127C56 | 1960FA5E | 14A0CB0B2 | 314066BA7D | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 288 | 49 | 024F8AD | 32C1F4BC | 094196164 | 6280CD74FB | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 289 | 50 | 049F15A | 6583E978 | 12832C2C8 | 45019AE9F6 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 290 | 51 | 093E2B5 | 4B07D2F0 | 050658591 | 0A0335D3ED | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 291 | 52 | 127C56B | 160FA5E0 | 0A0CB0B22 | 14066BA7DA | 0 | 0 | 1 | 0 | 0 | 01 | 01 | 10 |
| 292 | 53 | 04F8AD7 | 2C1F4BC1 | 141961645 | 280CD74FB5 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 293 | 54 | 09F15AF | 583E9783 | 0832C2C8A | 5019AE9F6A | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |
| 294 | 55 | 13E2B5E | 307D2F06 | 106585915 | 20335D3ED5 | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 295 | 56 | 07C56BD | 60FA5E0D | 00CB0B22B | 4066BA7DAA | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 296 | 57 | 0F8AD7A | 41F4BC1B | 019616457 | 00CD74FB54 | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 297 | 58 | 1F15AF4 | 03E97836 | 032C2C8AF | 019AE9F6A9 | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 298 | 59 | 1E2B5E9 | 07D2F06C | 06585915E | 0335D3ED52 | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 299 | 60 | 1C56BD2 | 0FA5E0D8 | 0CB0B22BC | 066BA7DAA4 | 1 | 1 | 1 | 0 | 0 | 10 | 01 | 10 |
| 300 | 61 | 18AD7A5 | 1F4BC1B0 | 196164578 | 0CD74FB549 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 301 | 62 | 115AF4B | 3E978361 | 12C2C8AF0 | 19AE9F6A92 | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 302 | 63 | 02B5E96 | 7D2F06C2 | 0585915E0 | 335D3ED524 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 11 |
| 303 | 64 | 056BD2D | 7A5E0D85 | 0B0B22BC1 | 66BA7DAA49 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 304 | 65 | 0AD7A5B | 74BC1B0A | 161645783 | 4D74FB5493 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 305 | 66 | 15AF4B6 | 69783615 | 0C2C8AF07 | 1AE9F6A926 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 306 | 67 | 0B5E96D | 52F06C2B | 185915E0F | 35D3ED524C | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 307 | 68 | 16BD2DB | 25E0D857 | 10B22BC1F | 6BA7DAA499 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 308 | 69 | 0D7A5B7 | 4BC1B0AF | 01645783F | 574FB54933 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 309 | 70 | 1AF4B6F | 1783615F | 02C8AF07F | 2E9F6A9266 | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 310 | 71 | 15E96DF | 2F06C2BF | 05915E0FF | 5D3ED524CC | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 311 | 72 | 0BD2DBF | 5E0D857F | 0B22BC1FE | 3A7DAA4998 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 312 | 73 | 17A5B7F | 3C1B0AFE | 1645783FD | 74FB549331 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 313 | 74 | 0F4B6FF | 783615FD | 0C8AF07FA | 69F6A92662 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 314 | 75 | 1E96DFF | 706C2BFB | 1915E0FF5 | 53ED524CC4 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 315 | 76 | 1D2DBFE | 60D857F6 | 122BC1FEB | 27DAA49988 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 316 | 77 | 1A5B7FD | 41B0AFEC | 045783FD7 | 4FB5493310 | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 317 | 78 | 14B6FFA | 03615FD8 | 08AF07FAE | 1F6A926620 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 318 | 79 | 096DFF4 | 06C2BFB1 | 115E0FF5D | 3ED524CC40 | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 319 | 80 | 12DBFE8 | 0D857F63 | 02BC1FEBB | 7DAA499881 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 320 | 81 | 05B7FD0 | 1B0AFEC6 | 05783FD77 | 7B54933103 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 321 | 82 | 0B6FFA1 | 3615FD8C | 0AF07FAEF | 76A9266206 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 322 | 83 | 16DFF42 | 6C2BFB18 | 15E0FF5DE | 6D524CC40C | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 323 | 84 | 0DBFE85 | 5857F631 | 0BC1FEBBD | 5AA4998819 | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 324 | 85 | 1B7FD0B | 30AFEC62 | 1783FD77A | 3549331033 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 325 | 86 | 16FFA16 | 615FD8C5 | 0F07FAEF5 | 6A92662067 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 326 | 87 | 0DFF42D | 42BFB18B | 1E0FF5DEA | 5524CC40CE | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 327 | 88 | 1BFE85B | 057F6317 | 1C1FEBBD5 | 2A4998819C | 1 | 0 | 1 | 0 | 0 | 00 | 00 | 10 |
| 328 | 89 | 17FD0B7 | 0AFEC62E | 183FD77AA | 5493310339 | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 329 | 90 | 0FFA16F | 15FD8C5C | 107FAEF55 | 2926620672 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 330 | 91 | 1FF42DF | 2BFB18B9 | 00FF5DEAA | 524CC40CE5 | 1 | 1 | 0 | 0 | 0 | 10 | 00 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 331 | 92 | 1FE85BF | 57F63172 | 01FEBBD55 | 24998819CA | 1 | 1 | 0 | 1 | 1 | 00 | 10 | 00 |
| 332 | 93 | 1FD0B7F | 2FEC62E4 | 03FD77AAA | 4933103394 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 333 | 94 | 1FA16FF | 5FD8C5C9 | 07FAEF555 | 1266206728 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 334 | 95 | 1F42DFF | 3FB18B93 | 0FF5DEAAA | 24CC40CE51 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 335 | 96 | 1E85BFF | 7F631727 | 1FEBBD554 | 4998819CA3 | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 336 | 97 | 1D0B7FE | 7EC62E4F | 1FD77AAA9 | 1331033947 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 337 | 98 | 1A16FFC | 7D8C5C9F | 1FAEF5553 | 266206728E | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 338 | 99 | 142DFF9 | 7B18B93F | 1F5DEAAA7 | 4CC40CE51D | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 339 | 100 | 085BFF3 | 7631727F | 1EBBD554E | 198819CA3B | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 340 | 101 | 10B7FE6 | 6C62E4FF | 1D77AAA9C | 3310339477 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 341 | 102 | 016FFCC | 58C5C9FE | 1AEF55538 | 66206728EE | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 342 | 103 | 02DFF98 | 318B93FC | 15DEAAA70 | 4C40CE51DC | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 343 | 104 | 05BFF31 | 631727F8 | 0BBD554E1 | 18819CA3B9 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 344 | 105 | 0B7FE62 | 462E4FF1 | 177AAA9C2 | 3103394772 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 345 | 106 | 16FFCC5 | 0C5C9FE2 | 0EF555384 | 6206728EE4 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 346 | 107 | 0DFF98A | 18B93FC4 | 1DEAAA709 | 440CE51DC9 | 1 | 1 | 1 | 0 | 0 | 00 | 11 | 00 |
| 347 | 108 | 1BFF315 | 31727F88 | 1BD554E12 | 0819CA3B93 | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 348 | 109 | 17FE62A | 62E4FF11 | 17AAA9C24 | 1033947726 | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 349 | 110 | 0FFCC54 | 45C9FE22 | 0F5553849 | 206728EE4C | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 350 | 111 | 1FF98A8 | 0B93FC44 | 1EAAA7093 | 40CE51DC99 | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 351 | 112 | 1FF3150 | 1727F889 | 1D554E127 | 019CA3B933 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 352 | 113 | 1FE62A0 | 2E4FF112 | 1AAA9C24F | 0339477267 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 353 | 114 | 1FCC541 | 5C9FE225 | 15553849E | 06728EE4CF | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 354 | 115 | 1F98A82 | 393FC44B | 0AAA7093C | 0CE51DC99F | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 355 | 116 | 1F31504 | 727F8897 | 1554E1279 | 19CA3B933E | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 356 | 117 | 1E62A09 | 64FF112F | 0AA9C24F2 | 339477267D | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 357 | 118 | 1CC5412 | 49FE225E | 1553849E4 | 6728EE4CFB | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 358 | 119 | 198A824 | 13FC44BC | 0AA7093C9 | 4E51DC99F7 | 1 | 1 | 1 | 0 | 1 | 10 | 00 | 01 |
| 359 | 120 | 1315049 | 27F88979 | 154E12792 | 1CA3B933EE | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 360 | 121 | 062A093 | 4FF112F3 | 0A9C24F24 | 39477267DC | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 361 | 122 | 0C54127 | 1FE225E6 | 153849E48 | 728EE4CFB8 | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 362 | 123 | 18A824E | 3FC44BCD | 0A7093C91 | 651DC99F71 | 1 | 1 | 1 | 0 | 0 | 11 | 01 | 00 |
| 363 | 124 | 115049C | 7F88979A | 14E127922 | 4A3B933EE2 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 364 | 125 | 02A0938 | 7F112F35 | 09C24F244 | 1477267DC5 | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |

### 1.1.4 Third set of samples

```
Initial values for the key, BD_ADDR and clock

K_session[0]  = FF  K_session[1]  = FF  K_session[2]  = FF  K_session[3]  = FF
K_session[4]  = FF  K_session[5]  = FF  K_session[6]  = FF  K_session[7]  = FF
K_session[8]  = FF  K_session[9]  = FF  K_session[10] = FF  K_session[11] = FF
K_session[12] = FF  K_session[13] = FF  K_session[14] = FF  K_session[15] = FF

BD_ADDR_C[0] = FF  BD_ADDR_C[1] = FF  BD_ADDR_C[2] = FF
BD_ADDR_C[3] = FF  BD_ADDR_C[4] = FF  BD_ADDR_C[5] = FF

CL[0] = FF  CL[1] = FF  CL[2] = FF  CL[3] = 03

The corresponding values of CLK are 0x7FF_FFFE and 0xFFF_FFFE.
```

```
=============================================================================================================
    Fill LFSRs with initial data
=============================================================================================================
t  clk#  LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

0     0  0000000* 00000000* 000000000* 0000000000*    0  0  0  0    0  00     00   00
1     1  0000001* 00000001* 000000001* 0000000001*    0  0  0  0    0  00     00   00
2     2  0000003* 00000002* 000000003* 0000000003*    0  0  0  0    0  00     00   00
3     3  0000007* 00000004* 000000007* 0000000007*    0  0  0  0    0  00     00   00
4     4  000000F* 00000009* 00000000F* 000000000F*    0  0  0  0    0  00     00   00
5     5  000001F* 00000013* 00000001F* 000000001F*    0  0  0  0    0  00     00   00
6     6  000003F* 00000027* 00000003F* 000000003F*    0  0  0  0    0  00     00   00
7     7  000007F* 0000004F* 00000007F* 000000007F*    0  0  0  0    0  00     00   00
8     8  00000FF* 0000009F* 0000000FF* 00000000FF*    0  0  0  0    0  00     00   00
9     9  00001FF* 0000013F* 0000001FF* 00000001FF*    0  0  0  0    0  00     00   00
10   10  00003FF* 0000027F* 0000003FF* 00000003FF*    0  0  0  0    0  00     00   00
11   11  00007FF* 000004FF* 0000007FF* 00000007FF*    0  0  0  0    0  00     00   00
12   12  0000FFF* 000009FF* 000000FFF* 0000000FFF*    0  0  0  0    0  00     00   00
13   13  0001FFF* 000013FF* 000001FFF* 0000001FFF*    0  0  0  0    0  00     00   00
14   14  0003FFF* 000027FF* 000003FFF* 0000003FFF*    0  0  0  0    0  00     00   00
15   15  0007FFF* 00004FFF* 000007FFF* 0000007FFF*    0  0  0  0    0  00     00   00
16   16  000FFFF* 00009FFF* 00000FFFF* 000000FFFF*    0  0  0  0    0  00     00   00
17   17  001FFFF* 00013FFF* 00001FFFF* 000001FFFF*    0  0  0  0    0  00     00   00
18   18  003FFFF* 00027FFF* 00003FFFF* 000003FFFF*    0  0  0  0    0  00     00   00
19   19  007FFFF* 0004FFFF* 00007FFFF* 000007FFFF*    0  0  0  0    0  00     00   00
20   20  00FFFFF* 0009FFFF* 0000FFFFF* 00000FFFFF*    0  0  0  0    0  00     00   00
21   21  01FFFFF* 0013FFFF* 0001FFFFF* 00001FFFFF*    0  0  0  0    0  00     00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 22 | 03FFFFF* | 0027FFFF* | 0003FFFFF* | 00003FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 23 | 23 | 07FFFFF* | 004FFFFF* | 0007FFFFF* | 00007FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 24 | 24 | 0FFFFFF* | 009FFFFF* | 000FFFFFF* | 0000FFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 25 | 25 | 1FFFFFF* | 013FFFFF* | 001FFFFFF* | 0001FFFFFF* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 26 | 26 | 1FFFFFF | 027FFFFF* | 003FFFFFF* | 0003FFFFFF* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 27 | 27 | 1FFFFFF | 04FFFFFF* | 007FFFFFF* | 0007FFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 28 | 28 | 1FFFFFF | 09FFFFFF* | 00FFFFFFF* | 000FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 29 | 29 | 1FFFFFF | 13FFFFFF* | 01FFFFFFF* | 001FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 30 | 30 | 1FFFFFF | 27FFFFFF* | 03FFFFFFF* | 003FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 31 | 31 | 1FFFFFF | 4FFFFFFF* | 07FFFFFFF* | 007FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 32 | 32 | 1FFFFFF | 1FFFFFFF | 0FFFFFFFF* | 00FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 33 | 33 | 1FFFFFF | 3FFFFFFE | 1FFFFFFFF* | 01FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 34 | 34 | 1FFFFFF | 7FFFFFFC | 1FFFFFFFF | 03FFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 35 | 35 | 1FFFFFF | 7FFFFFF9 | 1FFFFFFFF | 07FFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 36 | 36 | 1FFFFFF | 7FFFFFF3 | 1FFFFFFFF | 0FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 37 | 37 | 1FFFFFF | 7FFFFFE7 | 1FFFFFFFF | 1FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 38 | 38 | 1FFFFFF | 7FFFFFCF | 1FFFFFFFF | 3FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 39 | 39 | 1FFFFFF | 7FFFFF9F | 1FFFFFFFF | 7FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 1FFFFFF | 7FFFF3F | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 41 | 2 | 1FFFFFF | 7FFFE7F | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 42 | 3 | 1FFFFFF | 7FFFCFF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 43 | 4 | 1FFFFFF | 7FFF9FF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 44 | 5 | 1FFFFFF | 7FFF3FF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 45 | 6 | 1FFFFFF | 7FFE7FE | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 46 | 7 | 1FFFFFF | 7FFCFFC | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 47 | 8 | 1FFFFFF | 7FF9FF9 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 48 | 9 | 1FFFFFF | 7FF3FF3 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 49 | 10 | 1FFFFFF | 7FE7FE6 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 50 | 11 | 1FFFFFE | 7FCFFCC | 1FFFFFFFE | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 51 | 12 | 1FFFFFC | 7FF9FF99 | 1FFFFFFFC | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 52 | 13 | 1FFFFF8 | 7FF3FF33 | 1FFFFFFF8 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 53 | 14 | 1FFFFF0 | 7FE7FE67 | 1FFFFFFF0 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 54 | 15 | 1FFFFE0 | 7FCFFCCF | 1FFFFFFE1 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 55 | 16 | 1FFFFC0 | 7F9FF99F | 1FFFFFFC3 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 56 | 17 | 1FFFF80 | 7F3FF33E | 1FFFFFF87 | 7FFFFFFFE | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 57 | 18 | 1FFFF00 | 7E7FE67C | 1FFFFFF0F | 7FFFFFFFC | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 58 | 19 | 1FFE01 | 7CFFCCF8 | 1FFFFFE1E | 7FFFFFFF8 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 59 | 20 | 1FFC03 | 79FF99F0 | 1FFFFFC3C | 7FFFFFFF0 | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 60 | 21 | 1FF807 | 73FF33E0 | 1FFFFF878 | 7FFFFFFE1 | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 61 | 22 | 1FF00F | 67FE67C0 | 1FFFFF0F0 | 7FFFFFFC3 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 62 | 23 | 1FFE01E | 4FFCCF80 | 1FFFFE1E1 | 7FFFFFFF87 | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 63 | 24 | 1FFC03C | 1FF99F00 | 1FFFFC3C3 | 7FFFFFFF0F | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 64 | 25 | 1FF8078 | 3FF33E01 | 1FFFF8787 | 7FFFFFFE1E | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 65 | 26 | 1FF00F0 | 7FE67C02 | 1FFFF0F0F | 7FFFFFFC3C | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 66 | 27 | 1FE01E1 | 7FCCF805 | 1FFFE1E1E | 7FFFFFF878 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 67 | 28 | 1FC03C3 | 7F99F00A | 1FFFC3C3C | 7FFFFFF0F0 | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 68 | 29 | 1F80787 | 7F33E015 | 1FFF87878 | 7FFFFFE1E1 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 69 | 30 | 1F00F0F | 7E67C02A | 1FFF0F0F0 | 7FFFFFC3C3 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 70 | 31 | 1E01E1E | 7CCF8054 | 1FFE1E1E1 | 7FFFFF8787 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 71 | 32 | 1C03C3C | 799F00A9 | 1FFC3C3C3 | 7FFFFF0F0F | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 72 | 33 | 1807878 | 733E0152 | 1FF878787 | 7FFFFE1E1E | 1 | 0 | 1 | 1 | 0 | 00 | 01 | 01 |
| 73 | 34 | 100F0F0 | 667C02A5 | 1FF0F0F0F | 7FFFFC3C3C | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 74 | 35 | 001E1E0 | 4CF8054B | 1FE1E1E1F | 7FFFF87878 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 75 | 36 | 003C3C1 | 19F00A96 | 1FC3C3C3F | 7FFFF0F0F0 | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 76 | 37 | 0078783 | 33E0152C | 1F878787F | 7FFFE1E1E1 | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 77 | 38 | 00F0F07 | 67C02A59 | 1F0F0F0FF | 7FFFC3C3C3 | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 78 | 39 | 01E1E0E | 4F8054B3 | 1E1E1E1FF | 7FFF878787 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 79 | 40 | 03C3C1C | 1F00A966 | 1C3C3C3FF | 7FFF0F0F0F | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 80 | 41 | 0787838 | 3E0152CC | 1878787FF | 7FFE1E1E1E | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 81 | 42 | 0F0F070 | 7C02A598 | 10F0F0FFF | 7FFC3C3C3C | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 82 | 43 | 1E1E0E0 | 78054B30 | 01E1E1FFF | 7FF8787878 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 83 | 44 | 1C3C1C0 | 700A9660 | 03C3C3FFE | 7FF0F0F0F0 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 84 | 45 | 1878380 | 60152CC0 | 078787FFC | 7FE1E1E1E0 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 85 | 46 | 10F0700 | 402A5980 | 0F0F0FFF8 | 7FC3C3C3C0 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 86 | 47 | 01E0E00 | 0054B300 | 1E1E1FFF0 | 7F87878780 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 87 | 48 | 03C1C00 | 00A96601 | 1C3C3FFE0 | 7F0F0F0F00 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 88 | 49 | 0783800 | 0152CC03 | 18787FFC0 | 7E1E1E1E01 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 89 | 50 | 0F07000 | 02A59806 | 10F0FFF80 | 7C3C3C3C03 | 1 | 1 | 0 | 0 | 1 | 11 | 11 | 11 |
| 90 | 51 | 1E0E000 | 054B300D | 01E1FFF00 | 7878787807 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 91 | 52 | 1C1C001 | 0A96601A | 03C3FFE01 | 70F0F0F00F | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 92 | 53 | 1838003 | 152CC035 | 0787FFC03 | 61E1E1E01E | 1 | 0 | 0 | 1 | 0 | 10 | 10 | 11 |
| 93 | 54 | 1070007 | 2A59806B | 0F0FFF807 | 43C3C3C03C | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 94 | 55 | 00E000F | 54B300D7 | 1E1FFF00F | 0787878078 | 0 | 1 | 1 | 1 | 0 | 10 | 01 | 10 |
| 95 | 56 | 01C001F | 296601AE | 1C3FFE01F | 0F0F0F00F1 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 96 | 57 | 038003F | 52CC035C | 187FFC03F | 1E1E1E01E2 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 97 | 58 | 070007F | 259806B8 | 10FFF807F | 3C3C3C03C4 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 98 | 59 | 0E000FE | 4B300D71 | 01FFF00FE | 7878780788 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 99 | 60 | 1C001FD | 16601AE2 | 03FFE01FD | 70F0F00F10 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 100 | 61 | 18003FA | 2CC035C5 | 07FFC03FB | 61E1E01E21 | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 101 | 62 | 10007F4 | 59806B8B | 0FFF807F7 | 43C3C03C43 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 102 | 63 | 0000FE8 | 3300D717 | 1FFF00FEE | 0787807887 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 103 | 64 | 0001FD0 | 6601AE2F | 1FFE01FDC | 0F0F00F10E | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 104 | 65 | 0003FA0 | 4C035C5F | 1FFC03FB8 | 1E1E01E21D | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 105 | 66 | 0007F40 | 1806B8BE | 1FF807F70 | 3C3C03C43B | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 106 | 67 | 000FE81 | 300D717C | 1FF00FEE1 | 7878078877 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 107 | 68 | 001FD02 | 601AE2F8 | 1FE01FDC2 | 70F00F10EF | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 108 | 69 | 003FA05 | 4035C5F0 | 1FC03FB84 | 61E01E21DE | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 109 | 70 | 007F40B | 006B8BE0 | 1F807F708 | 43C03C43BC | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 110 | 71 | 00FE816 | 00D717C0 | 1F00FEE11 | 0780788778 | 0 | 1 | 1 | 1 | 0 | 10 | 11 | 11 |
| 111 | 72 | 01FD02C | 01AE2F81 | 1E01FDC23 | 0F00F10EF1 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 112 | 73 | 03FA059 | 035C5F02 | 1C03FB847 | 1E01E21DE3 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 113 | 74 | 07F40B3 | 06B8BE05 | 1807F708F | 3C03C43BC7 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 114 | 75 | 0FE8166 | 0D717C0B | 100FEE11E | 780788778F | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 115 | 76 | 1FD02CD | 1AE2F817 | 001FDC23D | 700F10EF1F | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 116 | 77 | 1FA059B | 35C5F02F | 003FB847A | 601E21DE3F | 1 | 1 | 0 | 0 | 1 | 10 | 11 | 01 |
| 117 | 78 | 1F40B37 | 6B8BE05E | 007F708F4 | 403C43BC7F | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 118 | 79 | 1E8166E | 5717C0BD | 00FEE11E9 | 00788778FF | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 119 | 80 | 1D02CDC | 2E2F817A | 01FDC23D3 | 00F10EF1FE | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 120 | 81 | 1A059B9 | 5C5F02F5 | 03FB847A6 | 01E21DE3FD | 1 | 0 | 0 | 1 | 1 | 01 | 01 | 10 |
| 121 | 82 | 140B373 | 38BE05EB | 07F708F4C | 03C43BC7FB | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 122 | 83 | 08166E7 | 717C0BD7 | 0FEE11E98 | 0788778FF7 | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 123 | 84 | 102CDCF | 62F817AE | 1FDC23D31 | 0F10EF1FEF | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 124 | 85 | 0059B9F | 45F02F5C | 1FB847A63 | 1E21DE3FDE | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 125 | 86 | 00B373E | 0BE05EB9 | 1F708F4C7 | 3C43BC7FBC | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 126 | 87 | 0166E7D | 17C0BD72 | 1EE11E98F | 788778FF78 | 0 | 1 | 1 | 1 | 0 | 10 | 11 | 11 |
| 127 | 88 | 02CDCFB | 2F817AE5 | 1DC23D31F | 710EF1FEF1 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 128 | 89 | 059B9F7 | 5F02F5CA | 1B847A63F | 621DE3FDE2 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 129 | 90 | 0B373EF | 3E05EB94 | 1708F4C7F | 443BC7FBC4 | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 130 | 91 | 166E7DF | 7C0BD728 | 0E11E98FF | 08778FF788 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 131 | 92 | 0CDCFBE | 7817AE50 | 1C23D31FF | 10EF1FEF10 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 132 | 93 | 19B9F7D | 702F5CA1 | 1847A63FE | 21DE3FDE21 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 133 | 94 | 1373EFB | 605EB942 | 108F4C7FC | 43BC7FBC43 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 134 | 95 | 06E7DF7 | 40BD7285 | 011E98FF8 | 0778FF7886 | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 135 | 96 | 0DCFBEF | 017AE50A | 023D31FF0 | 0EF1FEF10D | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 136 | 97 | 1B9F7DF | 02F5CA15 | 047A63FE1 | 1DE3FDE21A | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 137 | 98 | 173EFBF | 05EB942B | 08F4C7FC3 | 3BC7FBC434 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 138 | 99 | 0E7DF7F | 0BD72856 | 11E98FF87 | 778FF78869 | 1 | 1 | 0 | 1 | 1 | 00 | 00 | 10 |
| 139 | 100 | 1CFBEFF | 17AE50AC | 03D31FF0F | 6F1FEF10D3 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 140 | 101 | 19F7DFE | 2F5CA159 | 07A63FE1E | 5E3FDE21A7 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 141 | 102 | 13EFBFC | 5EB942B3 | 0F4C7FC3C | 3C7FBC434F | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 142 | 103 | 07DF7F8 | 3D728566 | 1E98FF878 | 78FF78869F | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 143 | 104 | 0FBEFF0 | 7AE50ACD | 1D31FF0F0 | 71FEF10D3E | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 144 | 105 | 1F7DFE1 | 75CA159B | 1A63FE1E1 | 63FDE21A7D | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 145 | 106 | 1EFBFC3 | 6B942B36 | 14C7FC3C3 | 47FBC434FB | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |
| 146 | 107 | 1DF7F86 | 5728566D | 098FF8786 | 0FF78869F7 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 147 | 108 | 1BEFF0C | 2E50ACDB | 131FF0F0C | 1FEF10D3EF | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 148 | 109 | 17DFE19 | 5CA159B6 | 063FE1E19 | 3FDE21A7DF | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 149 | 110 | 0FBFC33 | 3942B36D | 0C7FC3C32 | 7FBC434FBF | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 150 | 111 | 1F7F866 | 728566DB | 18FF87865 | 7F78869F7E | 1 | 1 | 1 | 0 | 0 | 00 | 01 | 01 |
| 151 | 112 | 1EFF0CC | 650ACDB6 | 11FF0F0CB | 7EF10D3EFC | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 152 | 113 | 1DFE199 | 4A159B6D | 03FE1E196 | 7DE21A7DF9 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 153 | 114 | 1BFC333 | 142B36DB | 07FC3C32C | 7BC434FBF3 | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 154 | 115 | 17F8666 | 28566DB6 | 0FF878659 | 778869F7E6 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 155 | 116 | 0FF0CCC | 50ACDB6D | 1FF0F0CB3 | 6F10D3EFCC | 1 | 1 | 1 | 0 | 0 | 11 | 01 | 00 |
| 156 | 117 | 1FE1999 | 2159B6DA | 1FE1E1966 | 5E21A7DF99 | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 157 | 118 | 1FC3332 | 42B36DB5 | 1FC3C32CC | 3C434FBF33 | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 158 | 119 | 1F86664 | 0566DB6B | 1F8786599 | 78869F7E67 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 159 | 120 | 1F0CCC8 | 0ACDB6D6 | 1F0F0CB33 | 710D3EFCCE | 1 | 1 | 1 | 0 | 0 | 10 | 01 | 10 |
| 160 | 121 | 1E19991 | 159B6DAC | 1E1E19666 | 621A7DF99D | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 161 | 122 | 1C33323 | 2B36DB58 | 1C3C32CCC | 4434FBF33B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 162 | 123 | 1866647 | 566DB6B0 | 187865999 | 0869F7E676 | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 163 | 124 | 10CCC8F | 2CDB6D60 | 10F0CB333 | 10D3EFCCEC | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 164 | 125 | 019991E | 59B6DAC0 | 01E196666 | 21A7DF99D9 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 165 | 126 | 033323C | 336DB580 | 03C32CCCD | 434FBF33B3 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 166 | 127 | 0666478 | 66DB6B01 | 07865999A | 069F7E6766 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 167 | 128 | 0CCC8F0 | 4DB6D603 | 0F0CB3334 | 0D3EFCCECD | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 168 | 129 | 19991E1 | 1B6DAC07 | 1E1966669 | 1A7DF99D9B | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 169 | 130 | 13323C3 | 36DB580E | 1C32CCCD3 | 34FBF33B37 | 0 | 1 | 1 | 1 | 1 | 10 | 00 | 01 |
| 170 | 131 | 0664786 | 6DB6B01C | 1865999A7 | 69F7E6766F | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 171 | 132 | 0CC8F0D | 5B6D6039 | 10CB3334F | 53EFCCECDF | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 172 | 133 | 1991E1A | 36DAC073 | 01966669E | 27DF99D9BF | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 173 | 134 | 1323C35 | 6DB580E6 | 032CCCD3C | 4FBF33B37E | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 174 | 135 | 064786A | 5B6B01CD | 065999A78 | 1F7E6766FC | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 175 | 136 | 0C8F0D5 | 36D6039B | 0CB3334F0 | 3EFCCECDF9 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 176 | 137 | 191E1AA | 6DAC0737 | 1966669E1 | 7DF99D9BF3 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 177 | 138 | 123C354 | 5B580E6E | 12CCCD3C3 | 7BF33B37E7 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 178 | 139 | 04786A9 | 36B01CDC | 05999A787 | 77E6766FCE | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 179 | 140 | 08F0D53 | 6D6039B8 | 0B3334F0E | 6FCCECDF9C | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 180 | 141 | 11E1AA6 | 5AC07370 | 166669E1D | 5F99D9BF38 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 181 | 142 | 03C354C | 3580E6E0 | 0CCCD3C3A | 3F33B37E70 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 182 | 143 | 0786A99 | 6B01CDC0 | 1999A7875 | 7E6766FCE1 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 183 | 144 | 0F0D533 | 56039B81 | 13334F0EB | 7CCECDF9C2 | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 184 | 145 | 1E1AA66 | 2C073703 | 06669E1D6 | 799D9BF385 | 1 | 0 | 0 | 1 | 1 | 01 | 01 | 10 |
| 185 | 146 | 1C354CC | 580E6E06 | 0CCD3C3AC | 733B37E70B | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 186 | 147 | 186A998 | 301CDC0C | 199A78759 | 66766FCE17 | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 187 | 148 | 10D5331 | 6039B818 | 1334F0EB2 | 4CECDF9C2F | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 188 | 149 | 01AA662 | 40737031 | 0669E1D65 | 19D9BF385E | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 189 | 150 | 0354CC5 | 00E6E063 | 0CD3C3ACB | 33B37E70BD | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 190 | 151 | 06A998A | 01CDC0C6 | 19A787596 | 6766FCE17B | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 191 | 152 | 0D53315 | 039B818C | 134F0EB2C | 4ECDF9C2F6 | 1 | 1 | 0 | 1 | 1 | 00 | 10 | 00 |
| 192 | 153 | 1AA662A | 07370318 | 069E1D659 | 1D9BF385ED | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 193 | 154 | 154CC54 | 0E6E0630 | 0D3C3ACB3 | 3B37E70BDB | 0 | 0 | 1 | 0 | 1 | 00 | 00 | 00 |
| 194 | 155 | 0A998A8 | 1CDC0C60 | 1A7875967 | 766FCE17B6 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 195 | 156 | 1533151 | 39B818C0 | 14F0EB2CE | 6CDF9C2F6C | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 196 | 157 | 0A662A3 | 73703180 | 09E1D659D | 59BF385ED8 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 197 | 158 | 14CC547 | 66E06301 | 13C3ACB3A | 337E70BDB0 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 198 | 159 | 0998A8E | 4DC0C602 | 078759675 | 66FCE17B61 | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 199 | 160 | 133151D | 1B818C05 | 0F0EB2CEB | 4DF9C2F6C2 | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 200 | 161 | 0662A3B | 3703180B | 1E1D659D6 | 1BF385ED85 | 0 | 0 | 1 | 1 | 1 | 11 | 01 | 01 |
| 201 | 162 | 0CC5477 | 6E063017 | 1C3ACB3AC | 37E70BDB0B | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 202 | 163 | 198A8EF | 5C0C602F | 187596759 | 6FCE17B617 | 1 | 0 | 1 | 1 | 0 | 10 | 11 | 11 |
| 203 | 164 | 13151DE | 3818C05F | 10EB2CEB2 | 5F9C2F6C2F | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 204 | 165 | 062A3BC | 703180BF | 01D659D65 | 3F385ED85E | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 205 | 166 | 0C54779 | 6063017E | 03ACB3ACB | 7E70BDB0BD | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 206 | 167 | 18A8EF2 | 40C602FD | 075967597 | 7CE17B617B | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 207 | 168 | 1151DE4 | 018C05FA | 0EB2CEB2F | 79C2F6C2F7 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 208 | 169 | 02A3BC9 | 03180BF5 | 1D659D65E | 7385ED85EE | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 209 | 170 | 0547793 | 063017EB | 1ACB3ACBC | 670BDB0BDC | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 210 | 171 | 0A8EF27 | 0C602FD6 | 159675978 | 4E17B617B9 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 211 | 172 | 151DE4E | 18C05FAD | 0B2CEB2F1 | 1C2F6C2F73 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 212 | 173 | 0A3BC9C | 3180BF5A | 1659D65E3 | 385ED85EE6 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 213 | 174 | 1477938 | 63017EB5 | 0CB3ACBC6 | 70BDB0BDCC | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 214 | 175 | 08EF270 | 4602FD6A | 19675978D | 617B617B99 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 215 | 176 | 11DE4E1 | 0C05FAD5 | 12CEB2F1A | 42F6C2F733 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 216 | 177 | 03BC9C3 | 180BF5AA | 059D65E34 | 05ED85EE67 | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 217 | 178 | 0779387 | 3017EB55 | 0B3ACBC68 | 0BDB0BDCCF | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 218 | 179 | 0EF270F | 602FD6AA | 1675978D0 | 17B617B99F | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 219 | 180 | 1DE4E1F | 405FAD54 | 0CEB2F1A1 | 2F6C2F733F | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 220 | 181 | 1BC9C3F | 00BF5AA9 | 19D65E342 | 5ED85EE67F | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 221 | 182 | 179387F | 017EB552 | 13ACBC684 | 3DB0BDCCFE | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 222 | 183 | 0F270FF | 02FD6AA5 | 075978D09 | 7B617B99FC | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 223 | 184 | 1E4E1FF | 05FAD54A | 0EB2F1A12 | 76C2F733F9 | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 224 | 185 | 1C9C3FE | 0BF5AA94 | 1D65E3425 | 6D85EE67F2 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 225 | 186 | 19387FD | 17EB5529 | 1ACBC684B | 5B0BDCCFE4 | 1 | 1 | 1 | 0 | 1 | 01 | 10 | 10 |
| 226 | 187 | 1270FFA | 2FD6AA53 | 15978D096 | 3617B99FC9 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 227 | 188 | 04E1FF5 | 5FAD54A7 | 0B2F1A12C | 6C2F733F93 | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 228 | 189 | 09C3FEB | 3F5AA94E | 165E34258 | 585EE67F27 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 229 | 190 | 1387FD7 | 7EB5529C | 0CBC684B1 | 30BDCCFE4F | 0 | 1 | 1 | 1 | 1 | 10 | 10 | 11 |
| 230 | 191 | 070FFAE | 7D6AA538 | 1978D0962 | 617B99FC9E | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 231 | 192 | 0E1FF5C | 7AD54A70 | 12F1A12C4 | 42F733F93D | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 232 | 193 | 1C3FEB9 | 75AA94E1 | 05E342588 | 05EE67F27A | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 233 | 194 | 187FD73 | 6B5529C3 | 0BC684B10 | 0BDCCFE4F4 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |

*Sample Data*

```
234 195  10FFAE6  56AA5386  178D09621  17B99FC9E8     0  1  0  1    1  00     11   10
235 196  01FF5CC  2D54A70C  0F1A12C43  2F733F93D0     0  0  1  0    1  10     00   11
236 197  03FEB98  5AA94E19  1E3425887  5EE67F27A1     0  1  1  1    1  00     10   00
237 198  07FD731  35529C33  1C684B10F  3DCCFE4F42     0  0  1  1    0  00     00   10
238 199  0FFAE63  6AA53866  18D09621F  7B99FC9E84     1  1  1  1    0  10     00   00
239 200  1FF5CC6  554A70CD  11A12C43F  7733F93D09     1  0  0  0    1  11     10   00

Z[0]  = 59
Z[1]  = 3B
Z[2]  = EF
Z[3]  = 07
Z[4]  = 13
Z[5]  = 70
Z[6]  = 9B
Z[7]  = B7
Z[8]  = 52
Z[9]  = 8F
Z[10] = 3E
Z[11] = B9
Z[12] = A5
Z[13] = AC
Z[14] = EA
Z[15] = 9E

================================================================================================
    Reload this pattern into the LFSRs
    Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 1521359
LFSR2  <= 528F703B
LFSR3  <= 0AC3E9BEF
LFSR4  <= 4FEAB9B707
C[t+1] <= 00

================================================================================================
    Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240   1  1521359  528F703B  0AC3E9BEF  4FEAB9B707     0  1  1  1    1  00     10   00
241   2  0A426B3  251EE076  1587D37DE  1FD5736E0F     1  0  0  1    0  00     00   10
242   3  1484D67  4A3DC0ED  0B0FA6FBD  3FAAE6DC1E     0  0  1  1    0  01     00   00
243   4  0909ACF  147B81DA  161F4DF7A  7F55CDB83D     1  0  0  0    0  00     01   00
244   5  121359E  28F703B5  0C3E9BEF5  7EAB9B707B     0  1  1  1    1  10     00   01
245   6  0426B3C  51EE076B  187D37DEB  7D5736E0F6     0  1  1  0    0  00     10   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 246 | 7 | 084D679 | 23DC0ED6 | 10FA6FBD7 | 7AAE6DC1EC | 1 | 1 | 0 | 1 | 1 | 00 | 00 | 10 |
| 247 | 8 | 109ACF2 | 47B81DAC | 01F4DF7AF | 755CDB83D8 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 248 | 9 | 01359E4 | 0F703B59 | 03E9BEF5E | 6AB9B707B1 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 249 | 10 | 026B3C8 | 1EE076B3 | 07D37DEBD | 55736E0F63 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 250 | 11 | 04D6791 | 3DC0ED67 | 0FA6FBD7A | 2AE6DC1EC7 | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 251 | 12 | 09ACF22 | 7B81DACF | 1F4DF7AF4 | 55CDB83D8F | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 252 | 13 | 1359E44 | 7703B59E | 1E9BEF5E8 | 2B9B707B1F | 0 | 0 | 1 | 1 | 1 | 10 | 11 | 01 |
| 253 | 14 | 06B3C88 | 6E076B3C | 1D37DEBD0 | 5736E0F63F | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 254 | 15 | 0D67911 | 5C0ED678 | 1A6FBD7A1 | 2E6DC1EC7E | 1 | 0 | 1 | 0 | 1 | 01 | 01 | 10 |
| 255 | 16 | 1ACF223 | 381DACF0 | 14DF7AF42 | 5CDB83D8FD | 1 | 0 | 0 | 1 | 1 | 11 | 01 | 01 |
| 256 | 17 | 159E446 | 703B59E0 | 09BEF5E85 | 39B707B1FA | 0 | 0 | 1 | 1 | 1 | 10 | 11 | 01 |
| 257 | 18 | 0B3C88C | 6076B3C0 | 137DEBD0A | 736E0F63F4 | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 258 | 19 | 1679118 | 40ED6780 | 06FBD7A15 | 66DC1EC7E8 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 259 | 20 | 0CF2231 | 01DACF00 | 0DF7AF42A | 4DB83D8FD1 | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 260 | 21 | 19E4463 | 03B59E01 | 1BEF5E854 | 1B707B1FA3 | 1 | 1 | 1 | 0 | 1 | 10 | 00 | 01 |
| 261 | 22 | 13C88C6 | 076B3C03 | 17DEBD0A9 | 36E0F63F47 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 262 | 23 | 079118C | 0ED67807 | 0FBD7A152 | 6DC1EC7E8E | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 263 | 24 | 0F22318 | 1DACF00E | 1F7AF42A4 | 5B83D8FD1D | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 264 | 25 | 1E44630 | 3B59E01C | 1EF5E8548 | 3707B1FA3B | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 265 | 26 | 1C88C61 | 76B3C039 | 1DEBD0A91 | 6E0F63F477 | 1 | 1 | 1 | 0 | 0 | 11 | 11 | 01 |
| 266 | 27 | 19118C3 | 6D678073 | 1BD7A1523 | 5C1EC7E8EF | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 267 | 28 | 1223187 | 5ACF00E6 | 17AF42A46 | 383D8FD1DE | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 268 | 29 | 044630E | 359E01CC | 0F5E8548D | 707B1FA3BD | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 269 | 30 | 088C61C | 6B3C0399 | 1EBD0A91A | 60F63F477B | 1 | 0 | 1 | 1 | 0 | 10 | 11 | 11 |
| 270 | 31 | 1118C39 | 56780733 | 1D7A15234 | 41EC7E8EF6 | 0 | 0 | 1 | 1 | 0 | 10 | 10 | 11 |
| 271 | 32 | 0231872 | 2CF00E67 | 1AF42A468 | 03D8FD1DEC | 0 | 1 | 1 | 1 | 1 | 01 | 10 | 10 |
| 272 | 33 | 04630E5 | 59E01CCE | 15E8548D1 | 07B1FA3BD8 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 273 | 34 | 08C61CB | 33C0399D | 0BD0A91A3 | 0F63F477B1 | 1 | 1 | 1 | 0 | 0 | 00 | 01 | 01 |
| 274 | 35 | 118C396 | 6780733A | 17A152347 | 1EC7E8EF63 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 275 | 36 | 031872D | 4F00E674 | 0F42A468E | 3D8FD1DEC7 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 276 | 37 | 0630E5A | 1E01CCE8 | 1E8548D1D | 7B1FA3BD8E | 0 | 0 | 1 | 0 | 1 | 01 | 00 | 10 |
| 277 | 38 | 0C61CB5 | 3C0399D0 | 1D0A91A3B | 763F477B1C | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 278 | 39 | 18C396A | 780733A0 | 1A1523477 | 6C7E8EF639 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 279 | 40 | 11872D5 | 700E6741 | 142A468EF | 58FD1DEC72 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 280 | 41 | 030E5AB | 601CCE83 | 08548D1DF | 31FA3BD8E5 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 281 | 42 | 061CB57 | 40399D07 | 10A91A3BF | 63F477B1CB | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 282 | 43 | 0C396AF | 00733A0F | 01523477E | 47E8EF6396 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 283 | 44 | 1872D5F | 00E6741F | 02A468EFD | 0FD1DEC72C | 1 | 1 | 0 | 1 | 1 | 00 | 00 | 10 |
| 284 | 45 | 10E5ABE | 01CCE83F | 0548D1DFA | 1FA3BD8E58 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 285 | 46 | 01CB57C | 0399D07F | 0A91A3BF4 | 3F477B1CB0 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 286 | 47 | 0396AF9 | 0733A0FE | 1523477E9 | 7E8EF63961 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 287 | 48 | 072D5F3 | 0E6741FD | 0A468EFD2 | 7D1DEC72C3 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 288 | 49 | 0E5ABE7 | 1CCE83FA | 148D1DFA4 | 7A3BD8E587 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 289 | 50 | 1CB57CE | 399D07F4 | 091A3BF49 | 7477B1CB0F | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 290 | 51 | 196AF9D | 733A0FE9 | 123477E92 | 68EF63961E | 1 | 0 | 0 | 1 | 1 | 00 | 11 | 10 |
| 291 | 52 | 12D5F3B | 66741FD2 | 0468EFD25 | 51DEC72C3C | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 292 | 53 | 05ABE77 | 4CE83FA4 | 08D1DFA4B | 23BD8E5879 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 293 | 54 | 0B57CEE | 19D07F49 | 11A3BF496 | 477B1CB0F2 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 294 | 55 | 16AF9DC | 33A0FE92 | 03477E92C | 0EF63961E4 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 295 | 56 | 0D5F3B8 | 6741FD25 | 068EFD259 | 1DEC72C3C9 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 296 | 57 | 1ABE771 | 4E83FA4B | 0D1DFA4B3 | 3BD8E58793 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 297 | 58 | 157CEE2 | 1D07F496 | 1A3BF4967 | 77B1CB0F26 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 298 | 59 | 0AF9DC5 | 3A0FE92D | 1477E92CE | 6F63961E4D | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 299 | 60 | 15F3B8B | 741FD25A | 08EFD259C | 5EC72C3C9B | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 300 | 61 | 0BE7716 | 683FA4B4 | 11DFA4B39 | 3D8E587937 | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 301 | 62 | 17CEE2D | 507F4968 | 03BF49672 | 7B1CB0F26E | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 302 | 63 | 0F9DC5B | 20FE92D0 | 077E92CE4 | 763961E4DC | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 303 | 64 | 1F3B8B6 | 41FD25A0 | 0EFD259C9 | 6C72C3C9B9 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 304 | 65 | 1E7716D | 03FA4B40 | 1DFA4B393 | 58E5879373 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 305 | 66 | 1CEE2DB | 07F49680 | 1BF496727 | 31CB0F26E6 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 306 | 67 | 19DC5B7 | 0FE92D00 | 17E92CE4E | 63961E4DCD | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 307 | 68 | 13B8B6F | 1FD25A00 | 0FD259C9C | 472C3C9B9A | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 308 | 69 | 07716DF | 3FA4B400 | 1FA4B3938 | 0E58793735 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 309 | 70 | 0EE2DBF | 7F496800 | 1F4967271 | 1CB0F26E6A | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 310 | 71 | 1DC5B7F | 7E92D000 | 1E92CE4E2 | 3961E4DCD4 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 311 | 72 | 1B8B6FF | 7D25A001 | 1D259C9C4 | 72C3C9B9A9 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 312 | 73 | 1716DFF | 7A4B4002 | 1A4B39389 | 6587937352 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 313 | 74 | 0E2DBFF | 74968005 | 149672713 | 4B0F26E6A5 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 314 | 75 | 1C5B7FE | 692D000B | 092CE4E26 | 161E4DCD4B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 315 | 76 | 18B6FFC | 525A0017 | 1259C9C4D | 2C3C9B9A96 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 316 | 77 | 116DFF8 | 24B4002F | 04B39389B | 587937352C | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 317 | 78 | 02DBFF1 | 4968005F | 096727136 | 30F26E6A58 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 318 | 79 | 05B7FE3 | 12D000BF | 12CE4E26C | 61E4DCD4B1 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 319 | 80 | 0B6FFC7 | 25A0017F | 059C9C4D8 | 43C9B9A963 | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 320 | 81 | 16DFF8E | 4B4002FF | 0B39389B1 | 07937352C6 | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 321 | 82 | 0DBFF1C | 168005FF | 167271363 | 0F26E6A58C | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 322 | 83 | 1B7FE38 | 2D000BFF | 0CE4E26C7 | 1E4DCD4B18 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 323 | 84 | 16FFC70 | 5A0017FF | 19C9C4D8F | 3C9B9A9631 | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 324 | 85 | 0DFF8E1 | 34002FFF | 139389B1E | 7937352C62 | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 325 | 86 | 1BFF1C3 | 68005FFF | 07271363D | 726E6A58C4 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 326 | 87 | 17FE387 | 5000BFFE | 0E4E26C7B | 64DCD4B188 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 327 | 88 | 0FFC70F | 20017FFD | 1C9C4D8F6 | 49B9A96311 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 328 | 89 | 1FF8E1F | 4002FFFB | 19389B1ED | 137352C623 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 329 | 90 | 1FF1C3F | 0005FFF7 | 1271363DB | 26E6A58C46 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 330 | 91 | 1FE387F | 000BFFEE | 04E26C7B6 | 4DCD4B188C | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 331 | 92 | 1FC70FF | 0017FFDC | 09C4D8F6D | 1B9A963118 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 332 | 93 | 1F8E1FF | 002FFFB8 | 1389B1EDA | 37352C6231 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 333 | 94 | 1F1C3FF | 005FFF70 | 071363DB4 | 6E6A58C462 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 334 | 95 | 1E387FE | 00BFFEE0 | 0E26C7B68 | 5CD4B188C5 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 335 | 96 | 1C70FFC | 017FFDC1 | 1C4D8F6D1 | 39A963118A | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 336 | 97 | 18E1FF9 | 02FFFB82 | 189B1EDA2 | 7352C62315 | 1 | 1 | 1 | 0 | 0 | 11 | 11 | 01 |
| 337 | 98 | 11C3FF2 | 05FFF705 | 11363DB45 | 66A58C462B | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 338 | 99 | 0387FE4 | 0BFFEE0A | 026C7B68B | 4D4B188C56 | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 339 | 100 | 070FFC9 | 17FFDC15 | 04D8F6D16 | 1A963118AD | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 340 | 101 | 0E1FF92 | 2FFFB82B | 09B1EDA2C | 352C62315A | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 341 | 102 | 1C3FF24 | 5FFF7057 | 1363DB458 | 6A58C462B4 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 342 | 103 | 187FE48 | 3FFEE0AE | 06C7B68B0 | 54B188C569 | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 343 | 104 | 10FFC90 | 7FFDC15C | 0D8F6D161 | 2963118AD2 | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 344 | 105 | 01FF920 | 7FFB82B9 | 1B1EDA2C2 | 52C62315A5 | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 345 | 106 | 03FF240 | 7FF70573 | 163DB4584 | 258C462B4B | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 346 | 107 | 07FE481 | 7FEE0AE6 | 0C7B68B08 | 4B188C5696 | 0 | 1 | 1 | 0 | 0 | 00 | 10 | 00 |
| 347 | 108 | 0FFC902 | 7FDC15CD | 18F6D1610 | 163118AD2D | 1 | 1 | 1 | 0 | 1 | 00 | 00 | 10 |
| 348 | 109 | 1FF9204 | 7FB82B9A | 11EDA2C20 | 2C62315A5B | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 349 | 110 | 1FF2408 | 7F705735 | 03DB45841 | 58C462B4B6 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 350 | 111 | 1FE4810 | 7EE0AE6B | 07B68B082 | 3188C5696C | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 351 | 112 | 1FC9021 | 7DC15CD6 | 0F6D16105 | 63118AD2D8 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 352 | 113 | 1F92042 | 7B82B9AD | 1EDA2C20B | 462315A5B0 | 1 | 1 | 1 | 0 | 1 | 00 | 00 | 10 |
| 353 | 114 | 1F24084 | 7705735A | 1DB458416 | 0C462B4B61 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 354 | 115 | 1E48108 | 6E0AE6B5 | 1B68B082C | 188C5696C3 | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 355 | 116 | 1C90211 | 5C15CD6A | 16D161059 | 3118AD2D86 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 356 | 117 | 1920422 | 382B9AD5 | 0DA2C20B3 | 62315A5B0D | 1 | 0 | 1 | 0 | 0 | 10 | 10 | 11 |
| 357 | 118 | 1240845 | 705735AA | 1B4584167 | 4462B4B61A | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 358 | 119 | 048108A | 60AE6B55 | 168B082CF | 08C5696C34 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 359 | 120 | 0902114 | 415CD6AB | 0D161059E | 118AD2D869 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 360 | 121 | 1204228 | 02B9AD56 | 1A2C20B3D | 2315A5B0D2 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 361 | 122 | 0408451 | 05735AAD | 14584167B | 462B4B61A4 | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 362 | 123 | 08108A2 | 0AE6B55B | 08B082CF7 | 0C5696C348 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 363 | 124 | 1021144 | 15CD6AB6 | 1161059EF | 18AD2D8690 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 364 | 125 | 0042289 | 2B9AD56C | 02C20B3DE | 315A5B0D20 | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |

*Sample Data*

### 1.1.5 Fourth set of samples

```
Initial values for the key, BD_ADDR and clock

K_session[0]  = 21  K_session[1]  = 87  K_session[2]  = F0  K_session[3]  = 4A
K_session[4]  = BA  K_session[5]  = 90  K_session[6]  = 31  K_session[7]  = D0
K_session[8]  = 78  K_session[9]  = 0D  K_session[10] = 4C  K_session[11] = 53
K_session[12] = E0  K_session[13] = 15  K_session[14] = 3A  K_session[15] = 63

BD_ADDR_C[0] = 2C  BD_ADDR_C[1] = 7F  BD_ADDR_C[2] = 94
BD_ADDR_C[3] = 56  BD_ADDR_C[4] = 0F  BD_ADDR_C[5] = 1B

CL[0] = 5F  CL[1] = 1A  CL[2] = 00  CL[3] = 02

The corresponding values of CLK are 0x400_34BE and 0xC00_34BE.

=============================================================================================
    Fill LFSRs with initial data
=============================================================================================

t  clk#  LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

0     0  0000000* 00000000* 000000000* 0000000000*    0  0  0  0    0  00     00   00
1     1  0000000* 00000001* 000000001* 0000000001*    0  0  0  0    0  00     00   00
2     2  0000001* 00000002* 000000002* 0000000003*    0  0  0  0    0  00     00   00
3     3  0000002* 00000004* 000000004* 0000000007*    0  0  0  0    0  00     00   00
4     4  0000004* 00000009* 000000008* 000000000F*    0  0  0  0    0  00     00   00
5     5  0000008* 00000013* 000000010* 000000001E*    0  0  0  0    0  00     00   00
6     6  0000010* 00000027* 000000021* 000000003D*    0  0  0  0    0  00     00   00
7     7  0000021* 0000004F* 000000043* 000000007A*    0  0  0  0    0  00     00   00
8     8  0000042* 0000009F* 000000087* 00000000F4*    0  0  0  0    0  00     00   00
9     9  0000084* 0000013F* 00000010F* 00000001E9*    0  0  0  0    0  00     00   00
10   10  0000108* 0000027F* 00000021F* 00000003D2*    0  0  0  0    0  00     00   00
11   11  0000211* 000004FE* 00000043E* 00000007A5*    0  0  0  0    0  00     00   00
12   12  0000422* 000009FC* 00000087C* 0000000F4A*    0  0  0  0    0  00     00   00
13   13  0000845* 000013F8* 0000010F8* 0000001E94*    0  0  0  0    0  00     00   00
14   14  000108B* 000027F0* 0000021F1* 0000003D29*    0  0  0  0    0  00     00   00
15   15  0002117* 00004FE1* 0000043E3* 0000007A52*    0  0  0  0    0  00     00   00
16   16  000422E* 00009FC2* 0000087C6* 000000F4A4*    0  0  0  0    0  00     00   00
17   17  000845D* 00013F84* 000010F8C* 000001E948*    0  0  0  0    0  00     00   00
18   18  00108BA* 00027F08* 000021F18* 000003D290*    0  0  0  0    0  00     00   00
19   19  0021174* 0004FE10* 000043E30* 000007A520*    0  0  0  0    0  00     00   00
20   20  00422E8* 0009FC21* 000087C61* 00000F4A41*    0  0  0  0    0  00     00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 21 | 21 | 00845D1* | 0013F842* | 00010F8C3* | 00001E9482* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 22 | 22 | 0108BA3* | 0027F084* | 00021F186* | 00003D2905* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 23 | 23 | 0211747* | 004FE109* | 00043E30C* | 00007A520B* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 24 | 24 | 0422E8F* | 009FC213* | 00087C619* | 0000F4A417* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 25 | 25 | 0845D1E* | 013F8426* | 0010F8C32* | 0001E9482F* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 26 | 26 | 108BA3D | 027F084D* | 0021F1864* | 0003D2905E* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 27 | 27 | 011747B | 04FE109B* | 0043E30C9* | 0007A520BC* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 28 | 28 | 022E8F6 | 09FC2136* | 0087C6192* | 000F4A4179* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 29 | 29 | 045D1EC | 13F8426C* | 010F8C325* | 001E9482F2* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 30 | 30 | 08BA3D9 | 27F084D8* | 021F1864B* | 003D2905E5* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 31 | 31 | 11747B3 | 4FE109B0* | 043E30C97* | 007A520BCA* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 32 | 32 | 02E8F67 | 1FC21360 | 087C6192E* | 00F4A41795* | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 33 | 33 | 05D1ECF | 3F8426C1 | 10F8C325C* | 01E9482F2B* | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 34 | 34 | 0BA3D9F | 7F084D82 | 01F1864B8 | 03D2905E56* | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 35 | 35 | 1747B3E | 7E109B04 | 03E30C970 | 07A520BCAC* | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 36 | 36 | 0E8F67C | 7C213608 | 07C6192E1 | 0F4A417958* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 37 | 37 | 1D1ECF8 | 78426C11 | 0F8C325C3 | 1E9482F2B1* | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 38 | 38 | 1A3D9F0 | 7084D822 | 1F1864B86 | 3D2905E563* | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 39 | 39 | 147B3E1 | 6109B044 | 1E30C970C | 7A520BCAC6* | 0 | 0 | 1 | 0 | 1 | 00 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 08F67C2 | 42136088 | 1C6192E18 | 74A417958D | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 41 | 2 | 11ECF84 | 0426C111 | 18C325C30 | 69482F2B1B | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 42 | 3 | 03D9F08 | 084D8222 | 11864B861 | 52905E5637 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 43 | 4 | 07B3E10 | 109B0444 | 030C970C3 | 2520BCAC6E | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 44 | 5 | 0F67C21 | 21360889 | 06192E186 | 4A417958DC | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 45 | 6 | 1ECF843 | 426C1112 | 0C325C30C | 1482F2B1B8 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 46 | 7 | 1D9F086 | 04D82225 | 1864B8619 | 2905E56370 | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 47 | 8 | 1B3E10D | 09B0444B | 10C970C32 | 520BCAC6E1 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 48 | 9 | 167C21B | 13608897 | 0192E1865 | 2417958DC3 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 49 | 10 | 0CF8436 | 26C1112F | 0325C30CB | 482F2B1B87 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 50 | 11 | 19F086D | 4D82225E | 064B86197 | 105E56370F | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 51 | 12 | 13E10DB | 1B0444BC | 0C970C32F | 20BCAC6E1F | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 52 | 13 | 07C21B7 | 36088979 | 192E1865E | 417958DC3F | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 53 | 14 | 0F8436E | 6C1112F2 | 125C30CBD | 02F2B1B87F | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 54 | 15 | 1F086DD | 582225E4 | 04B86197B | 05E56370FF | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 55 | 16 | 1E10DBA | 30444BC9 | 0970C32F7 | 0BCAC6E1FF | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 56 | 17 | 1C21B75 | 60889793 | 12E1865EE | 17958DC3FF | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 57 | 18 | 18436EA | 41112F27 | 05C30CBDD | 2F2B1B87FF | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 58 | 19 | 1086DD4 | 02225E4E | 0B86197BA | 5E56370FFF | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 59 | 20 | 010DBA8 | 0444BC9D | 170C32F74 | 3CAC6E1FFF | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 60 | 21 | 021B750 | 0889793A | 0E1865EE8 | 7958DC3FFF | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 61 | 22 | 0436EA0 | 1112F274 | 1C30CBDD0 | 72B1B87FFE | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 62 | 23 | 086DD40 | 2225E4E9 | 186197BA1 | 656370FFFC | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 63 | 24 | 10DBA81 | 444BC9D3 | 10C32F743 | 4AC6E1FFF8 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 64 | 25 | 01B7502 | 089793A7 | 01865EE86 | 158DC3FFF1 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 65 | 26 | 036EA05 | 112F274E | 030CBDD0D | 2B1B87FFE3 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 66 | 27 | 06DD40B | 225E4E9C | 06197BA1A | 56370FFFC6 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 67 | 28 | 0DBA817 | 44BC9D39 | 0C32F7434 | 2C6E1FFF8D | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 68 | 29 | 1B7502E | 09793A72 | 1865EE868 | 58DC3FFF1B | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 69 | 30 | 16EA05D | 12F274E5 | 10CBDD0D0 | 31B87FFE36 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 70 | 31 | 0DD40BA | 25E4E9CB | 0197BA1A1 | 6370FFFC6D | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 71 | 32 | 1BA8174 | 4BC9D397 | 032F74343 | 46E1FFF8DA | 1 | 1 | 0 | 1 | 0 | 11 | 11 | 01 |
| 72 | 33 | 17502E8 | 1793A72F | 065EE8687 | 0DC3FFF1B4 | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 73 | 34 | 0EA05D0 | 2F274E5E | 0CBDD0D0F | 1B87FFE369 | 1 | 0 | 1 | 1 | 0 | 10 | 11 | 11 |
| 74 | 35 | 1D40BA0 | 5E4E9CBD | 197BA1A1F | 370FFFC6D2 | 1 | 0 | 1 | 0 | 0 | 10 | 10 | 11 |
| 75 | 36 | 1A81741 | 3C9D397B | 12F74343F | 6E1FFF8DA5 | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 76 | 37 | 1502E82 | 793A72F6 | 05EE8687F | 5C3FFF1B4B | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 77 | 38 | 0A05D05 | 7274E5ED | 0BDD0D0FF | 387FFE3696 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 78 | 39 | 140BA0B | 64E9CBDA | 17BA1A1FF | 70FFFC6D2C | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 79 | 40 | 0817416 | 49D397B4 | 0F74343FE | 61FFF8DA59 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 80 | 41 | 102E82C | 13A72F69 | 1EE8687FD | 43FFF1B4B3 | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 81 | 42 | 005D058 | 274E5ED2 | 1DD0D0FFA | 07FFE36966 | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 82 | 43 | 00BA0B0 | 4E9CBDA5 | 1BA1A1FF5 | 0FFFC6D2CD | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 83 | 44 | 0174160 | 1D397B4A | 174343FEA | 1FFF8DA59B | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 84 | 45 | 02E82C0 | 3A72F695 | 0E8687FD4 | 3FFF1B4B37 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 85 | 46 | 05D0580 | 74E5ED2B | 1D0D0FFA9 | 7FFE36966E | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 86 | 47 | 0BA0B00 | 69CBDA56 | 1A1A1FF53 | 7FFC6D2CDC | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 87 | 48 | 1741600 | 5397B4AC | 14343FEA6 | 7FF8DA59B8 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 88 | 49 | 0E82C01 | 272F6959 | 08687FD4D | 7FF1B4B370 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 89 | 50 | 1D05802 | 4E5ED2B3 | 10D0FFA9A | 7FE36966E0 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 90 | 51 | 1A0B004 | 1CBDA566 | 01A1FF535 | 7FC6D2CDC0 | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 91 | 52 | 1416009 | 397B4ACC | 0343FEA6B | 7F8DA59B80 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 92 | 53 | 082C013 | 72F69599 | 0687FD4D7 | 7F1B4B3701 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 93 | 54 | 1058026 | 65ED2B33 | 0D0FFA9AF | 7E36966E03 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 94 | 55 | 00B004D | 4BDA5667 | 1A1FF535E | 7C6D2CDC06 | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 95 | 56 | 016009B | 17B4ACCE | 143FEA6BD | 78DA59B80D | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 96 | 57 | 02C0137 | 2F69599D | 087FD4D7B | 71B4B3701A | 0 | 0 | 1 | 1 | 1 | 10 | 11 | 01 |
| 97 | 58 | 058026F | 5ED2B33B | 10FFA9AF6 | 636966E034 | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 98 | 59 | 0B004DF | 3DA56677 | 01FF535ED | 46D2CDC068 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 99 | 60 | 16009BF | 7B4ACCEF | 03FEA6BDB | 0DA59B80D0 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 100 | 61 | 0C0137F | 769599DF | 07FD4D7B7 | 1B4B3701A1 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 101 | 62 | 18026FE | 6D2B33BE | 0FFA9AF6E | 36966E0342 | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 102 | 63 | 1004DFC | 5A56677D | 1FF535EDD | 6D2CDC0684 | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 103 | 64 | 0009BF9 | 34ACCEFB | 1FEA6BDBB | 5A59B80D09 | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 104 | 65 | 00137F2 | 69599DF7 | 1FD4D7B76 | 34B3701A12 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 105 | 66 | 0026FE5 | 52B33BEF | 1FA9AF6EC | 6966E03424 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 106 | 67 | 004DFCA | 256677DF | 1F535EDD8 | 52CDC06848 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 107 | 68 | 009BF94 | 4ACCEFBE | 1EA6BDBB0 | 259B80D091 | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 108 | 69 | 0137F29 | 1599DF7C | 1D4D7B760 | 4B3701A123 | 0 | 1 | 1 | 0 | 1 | 10 | 11 | 01 |
| 109 | 70 | 026FE53 | 2B33BEF9 | 1A9AF6EC0 | 166E034246 | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 110 | 71 | 04DFCA7 | 56677DF2 | 1535EDD81 | 2CDC06848D | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 111 | 72 | 09BF94F | 2CCEFBE4 | 0A6BDBB03 | 59B80D091B | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 112 | 73 | 137F29E | 599DF7C9 | 14D7B7607 | 33701A1236 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 113 | 74 | 06FE53C | 333BEF93 | 09AF6EC0E | 66E034246C | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 114 | 75 | 0DFCA79 | 6677DF26 | 135EDD81D | 4DC06848D8 | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 115 | 76 | 1BF94F2 | 4CEFBE4D | 06BDBB03B | 1B80D091B1 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 116 | 77 | 17F29E5 | 19DF7C9A | 0D7B76077 | 3701A12363 | 0 | 1 | 1 | 0 | 1 | 00 | 11 | 10 |
| 117 | 78 | 0FE53CA | 33BEF934 | 1AF6EC0EF | 6E034246C6 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 118 | 79 | 1FCA794 | 677DF269 | 15EDD81DF | 5C06848D8C | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 119 | 80 | 1F94F29 | 4EFBE4D2 | 0BDBB03BE | 380D091B19 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 120 | 81 | 1F29E53 | 1DF7C9A5 | 17B76077D | 701A123633 | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 121 | 82 | 1E53CA6 | 3BEF934B | 0F6EC0EFB | 6034246C66 | 1 | 1 | 1 | 0 | 0 | 11 | 11 | 01 |
| 122 | 83 | 1CA794D | 77DF2696 | 1EDD81DF6 | 406848D8CD | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 123 | 84 | 194F29B | 6FBE4D2C | 1DBB03BED | 00D091B19B | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 124 | 85 | 129E536 | 5F7C9A59 | 1B76077DA | 01A1236337 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 125 | 86 | 053CA6C | 3EF934B3 | 16EC0EFB4 | 034246C66E | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 126 | 87 | 0A794D9 | 7DF26967 | 0DD81DF69 | 06848D8CDD | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 127 | 88 | 14F29B3 | 7BE4D2CF | 1BB03BED3 | 0D091B19BB | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 128 | 89 | 09E5366 | 77C9A59F | 176077DA6 | 1A12363377 | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 129 | 90 | 13CA6CD | 6F934B3F | 0EC0EFB4D | 34246C66EF | 0 | 1 | 1 | 0 | 1 | 10 | 11 | 01 |
| 130 | 91 | 0794D9B | 5F26967F | 1D81DF69A | 6848D8CDDF | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 131 | 92 | 0F29B37 | 3E4D2CFE | 1B03BED35 | 5091B19BBE | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 132 | 93 | 1E5366F | 7C9A59FD | 16077DA6B | 212363377C | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 133 | 94 | 1CA6CDF | 7934B3FB | 0C0EFB4D6 | 4246C66EF9 | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 134 | 95 | 194D9BE | 726967F6 | 181DF69AD | 048D8CDDF2 | 1 | 0 | 1 | 1 | 1 | 11 | 00 | 11 |
| 135 | 96 | 129B37D | 64D2CFED | 103BED35B | 091B19BBE5 | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 136 | 97 | 05366FA | 49A59FDA | 0077DA6B7 | 12363377CA | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 137 | 98 | 0A6CDF5 | 134B3FB4 | 00EFB4D6E | 246C66EF95 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 138 | 99 | 14D9BEA | 26967F69 | 01DF69ADD | 48D8CDDF2B | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 139 | 100 | 09B37D4 | 4D2CFED2 | 03BED35BB | 11B19BBE56 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 140 | 101 | 1366FA8 | 1A59FDA5 | 077DA6B77 | 2363377CAC | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 141 | 102 | 06CDF51 | 34B3FB4A | 0EFB4D6EF | 46C66EF959 | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 142 | 103 | 0D9BEA2 | 6967F695 | 1DF69ADDF | 0D8CDDF2B2 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 143 | 104 | 1B37D45 | 52CFED2A | 1BED35BBF | 1B19BBE564 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 144 | 105 | 166FA8A | 259FDA54 | 17DA6B77E | 363377CAC8 | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 145 | 106 | 0CDF515 | 4B3FB4A9 | 0FB4D6EFC | 6C66EF9591 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 146 | 107 | 19BEA2B | 167F6952 | 1F69ADDF8 | 58CDDF2B22 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 147 | 108 | 137D457 | 2CFED2A5 | 1ED35BBF1 | 319BBE5645 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 148 | 109 | 06FA8AF | 59FDA54A | 1DA6B77E2 | 63377CAC8B | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 149 | 110 | 0DF515F | 33FB4A95 | 1B4D6EFC4 | 466EF95916 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 150 | 111 | 1BEA2BF | 67F6952A | 169ADDF88 | 0CDDF2B22C | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 151 | 112 | 17D457F | 4FED2A55 | 0D35BBF10 | 19BBE56459 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 152 | 113 | 0FA8AFE | 1FDA54AB | 1A6B77E20 | 3377CAC8B3 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 153 | 114 | 1F515FD | 3FB4A957 | 14D6EFC40 | 66EF959166 | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 154 | 115 | 1EA2BFA | 7F6952AF | 09ADDF880 | 4DDF2B22CC | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 155 | 116 | 1D457F4 | 7ED2A55F | 135BBF100 | 1BBE564598 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 156 | 117 | 1A8AFE8 | 7DA54ABF | 06B77E200 | 377CAC8B31 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 157 | 118 | 1515FD0 | 7B4A957F | 0D6EFC401 | 6EF9591663 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 158 | 119 | 0A2BFA1 | 76952AFE | 1ADDF8803 | 5DF2B22CC7 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 159 | 120 | 1457F42 | 6D2A55FD | 15BBF1007 | 3BE564598E | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 160 | 121 | 08AFE84 | 5A54ABFB | 0B77E200F | 77CAC8B31C | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 161 | 122 | 115FD09 | 34A957F7 | 16EFC401F | 6F95916639 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 162 | 123 | 02BFA12 | 6952AFEF | 0DDF8803E | 5F2B22CC73 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 163 | 124 | 057F424 | 52A55FDF | 1BBF1007D | 3E564598E7 | 0 | 1 | 1 | 0 | 1 | 01 | 11 | 00 |
| 164 | 125 | 0AFE848 | 254ABFBF | 177E200FA | 7CAC8B31CF | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 165 | 126 | 15FD090 | 4A957F7E | 0EFC401F5 | 795916639E | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 166 | 127 | 0BFA121 | 152AFEFD | 1DF8803EA | 72B22CC73C | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 167 | 128 | 17F4243 | 2A55FDFA | 1BF1007D4 | 6564598E78 | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 168 | 129 | 0FE8486 | 54ABFBF4 | 17E200FA8 | 4AC8B31CF0 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 169 | 130 | 1FD090C | 2957F7E8 | 0FC401F51 | 15916639E1 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 170 | 131 | 1FA1219 | 52AFEFD1 | 1F8803EA3 | 2B22CC73C2 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 171 | 132 | 1F42432 | 255FDFA2 | 1F1007D47 | 564598E785 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 172 | 133 | 1E84865 | 4ABFBF44 | 1E200FA8F | 2C8B31CF0B | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 173 | 134 | 1D090CB | 157F7E88 | 1C401F51E | 5916639E17 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 174 | 135 | 1A12196 | 2AFEFD11 | 18803EA3C | 322CC73C2E | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 175 | 136 | 142432C | 55FDFA23 | 11007D479 | 64598E785C | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 176 | 137 | 0848659 | 2BFBF446 | 0200FA8F2 | 48B31CF0B9 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 177 | 138 | 1090CB2 | 57F7E88C | 0401F51E4 | 116639E173 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 178 | 139 | 0121964 | 2FEFD118 | 0803EA3C8 | 22CC73C2E6 | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 179 | 140 | 02432C9 | 5FDFA230 | 1007D4791 | 4598E785CD | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 180 | 141 | 0486593 | 3FBF4461 | 000FA8F23 | 0B31CF0B9B | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 181 | 142 | 090CB26 | 7F7E88C3 | 001F51E47 | 16639E1736 | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 182 | 143 | 121964D | 7EFD1187 | 003EA3C8F | 2CC73C2E6C | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 183 | 144 | 0432C9B | 7DFA230E | 007D4791E | 598E785CD8 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 184 | 145 | 0865936 | 7BF4461C | 00FA8F23C | 331CF0B9B0 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 185 | 146 | 10CB26D | 77E88C38 | 01F51E479 | 6639E17361 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 186 | 147 | 01964DA | 6FD11870 | 03EA3C8F2 | 4C73C2E6C2 | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 187 | 148 | 032C9B4 | 5FA230E1 | 07D4791E4 | 18E785CD84 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 188 | 149 | 0659368 | 3F4461C2 | 0FA8F23C9 | 31CF0B9B09 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 189 | 150 | 0CB26D0 | 7E88C384 | 1F51E4793 | 639E173612 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 190 | 151 | 1964DA0 | 7D118709 | 1EA3C8F27 | 473C2E6C24 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 191 | 152 | 12C9B41 | 7A230E12 | 1D4791E4E | 0E785CD848 | 0 | 0 | 1 | 0 | 1 | 01 | 00 | 10 |
| 192 | 153 | 0593683 | 74461C24 | 1A8F23C9C | 1CF0B9B091 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 193 | 154 | 0B26D06 | 688C3848 | 151E47938 | 39E1736123 | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 194 | 155 | 164DA0D | 51187091 | 0A3C8F271 | 73C2E6C247 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 195 | 156 | 0C9B41A | 2230E123 | 14791E4E3 | 6785CD848F | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 196 | 157 | 1936835 | 4461C247 | 08F23C9C6 | 4F0B9B091E | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 197 | 158 | 126D06A | 08C3848E | 11E47938D | 1E1736123C | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 198 | 159 | 04DA0D5 | 1187091C | 03C8F271B | 3C2E6C2478 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 199 | 160 | 09B41AA | 230E1238 | 0791E4E37 | 785CD848F1 | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 200 | 161 | 1368354 | 461C2470 | 0F23C9C6F | 70B9B091E3 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 201 | 162 | 06D06A9 | 0C3848E1 | 1E47938DF | 61736123C6 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 202 | 163 | 0DA0D52 | 187091C3 | 1C8F271BE | 42E6C2478D | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 203 | 164 | 1B41AA4 | 30E12387 | 191E4E37C | 05CD848F1A | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 204 | 165 | 1683549 | 61C2470F | 123C9C6F9 | 0B9B091E34 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 205 | 166 | 0D06A92 | 43848E1E | 047938DF3 | 1736123C68 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 206 | 167 | 1A0D524 | 07091C3C | 08F271BE7 | 2E6C2478D1 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 207 | 168 | 141AA49 | 0E123879 | 11E4E37CF | 5CD848F1A2 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 208 | 169 | 0835492 | 1C2470F3 | 03C9C6F9F | 39B091E345 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 209 | 170 | 106A925 | 3848E1E6 | 07938DF3F | 736123C68B | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 210 | 171 | 00D524A | 7091C3CD | 0F271BE7E | 66C2478D16 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 211 | 172 | 01AA495 | 6123879B | 1E4E37CFD | 4D848F1A2D | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 212 | 173 | 035492A | 42470F36 | 1C9C6F9FB | 1B091E345B | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 213 | 174 | 06A9255 | 048E1E6C | 1938DF3F6 | 36123C68B7 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 214 | 175 | 0D524AB | 091C3CD8 | 1271BE7EC | 6C2478D16E | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 215 | 176 | 1AA4957 | 123879B1 | 04E37CFD8 | 5848F1A2DD | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 216 | 177 | 15492AF | 2470F363 | 09C6F9FB0 | 3091E345BA | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 217 | 178 | 0A9255E | 48E1E6C7 | 138DF3F61 | 6123C68B75 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 218 | 179 | 1524ABD | 11C3CD8F | 071BE7EC3 | 42478D16EB | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 219 | 180 | 0A4957B | 23879B1F | 0E37CFD87 | 048F1A2DD6 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 220 | 181 | 1492AF6 | 470F363F | 1C6F9FB0E | 091E345BAD | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 221 | 182 | 09255EC | 0E1E6C7F | 18DF3F61D | 123C68B75B | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 222 | 183 | 124ABD9 | 1C3CD8FF | 11BE7EC3A | 2478D16EB6 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 223 | 184 | 04957B3 | 3879B1FE | 037CFD874 | 48F1A2DD6D | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 224 | 185 | 092AF66 | 70F363FD | 06F9FB0E9 | 11E345BADB | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 225 | 186 | 1255ECD | 61E6C7FA | 0DF3F61D3 | 23C68B75B7 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 226 | 187 | 04ABD9B | 43CD8FF5 | 1BE7EC3A7 | 478D16EB6E | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 227 | 188 | 0957B37 | 079B1FEA | 17CFD874E | 0F1A2DD6DD | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 228 | 189 | 12AF66F | 0F363FD4 | 0F9FB0E9C | 1E345BADBB | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 229 | 190 | 055ECDE | 1E6C7FA9 | 1F3F61D39 | 3C68B75B76 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 230 | 191 | 0ABD9BC | 3CD8FF53 | 1E7EC3A73 | 78D16EB6EC | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 231 | 192 | 157B379 | 79B1FEA7 | 1CFD874E6 | 71A2DD6DD9 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 232 | 193 | 0AF66F3 | 7363FD4E | 19FB0E9CD | 6345BADBB2 | 1 | 0 | 1 | 0 | 1 | 01 | 11 | 00 |

*Sample Data*

```
233 194  15ECDE6  66C7FA9D  13F61D39A  468B75B765     0  1  0  1    1  10     01   11
234 195  0BD9BCC  4D8FF53A  07EC3A735  0D16EB6ECA     1  1  0  0    0  11     10   01
235 196  17B3799  1B1FEA75  0FD874E6A  1A2DD6DD94     0  0  1  0    0  00     11   10
236 197  0F66F33  363FD4EA  1FB0E9CD5  345BADBB28     1  0  1  0    0  11     00   11
237 198  1ECDE67  6C7FA9D5  1F61D39AA  68B75B7650     1  0  1  1    0  00     11   00
238 199  1D9BCCF  58FF53AB  1EC3A7354  516EB6ECA0     1  1  1  0    1  11     00   11
239 200  1B3799E  31FEA756  1D874E6A8  22DD6DD940     1  1  1  1    1  00     11   00

Z[0]  = 3F
Z[1]  = B1
Z[2]  = 67
Z[3]  = D2
Z[4]  = 2F
Z[5]  = A6
Z[6]  = 1F
Z[7]  = B9
Z[8]  = E6
Z[9]  = 84
Z[10] = 43
Z[11] = 07
Z[12] = D8
Z[13] = 1E
Z[14] = E7
Z[15] = C3

================================================================================================
    Reload this pattern into the LFSRs
    Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 0E62F3F
LFSR2  <= 6C84A6B1
LFSR3  <= 11E431F67
LFSR4  <= 61E707B9D2
C[t+1] <= 00

================================================================================================
    Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240   1  0E62F3F  6C84A6B1  11E431F67  61E707B9D2     1  1  0  1    0  00     11   00
241   2  1CC5E7F  59094D63  03C863ECE  43CE0F73A5     1  0  0  1    0  11     00   11
242   3  198BCFF  32129AC6  0790C7D9D  079C1EE74A     1  0  0  1    1  01     11   00
243   4  13179FE  6425358C  0F218FB3A  0F383DCE94     0  0  1  0    0  10     01   11
244   5  062F3FD  484A6B19  1E431F675  1E707B9D28     0  0  1  0    1  00     10   01
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 245 | 6 | 0C5E7FB | 1094D632 | 1C863ECEB | 3CE0F73A50 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 246 | 7 | 18BCFF7 | 2129AC64 | 190C7D9D7 | 79C1EE74A1 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 247 | 8 | 1179FEE | 425358C8 | 1218FB3AE | 7383DCE942 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 248 | 9 | 02F3FDD | 04A6B190 | 0431F675D | 6707B9D285 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 249 | 10 | 05E7FBB | 094D6320 | 0863ECEBB | 4E0F73A50B | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 250 | 11 | 0BCFF77 | 129AC640 | 10C7D9D77 | 1C1EE74A16 | 1 | 1 | 0 | 0 | 0 | 11 | 00 | 11 |
| 251 | 12 | 179FEEE | 25358C80 | 018FB3AEE | 383DCE942C | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 252 | 13 | 0F3FDDC | 4A6B1900 | 031F675DD | 707B9D2859 | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 253 | 14 | 1E7FBB8 | 14D63200 | 063ECEBBA | 60F73A50B3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 254 | 15 | 1CFF771 | 29AC6401 | 0C7D9D774 | 41EE74A167 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 255 | 16 | 19FEEE2 | 5358C803 | 18FB3AEE9 | 03DCE942CE | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 256 | 17 | 13FDDC4 | 26B19007 | 11F675DD2 | 07B9D2859C | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 257 | 18 | 07FBB88 | 4D63200E | 03ECEBBA4 | 0F73A50B38 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 258 | 19 | 0FF7711 | 1AC6401D | 07D9D7748 | 1EE74A1670 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 259 | 20 | 1FEEE23 | 358C803B | 0FB3AEE91 | 3DCE942CE1 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 260 | 21 | 1FDDC47 | 6B190076 | 1F675DD23 | 7B9D2859C2 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 261 | 22 | 1FBB88F | 563200ED | 1ECEBBA47 | 773A50B385 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 262 | 23 | 1F7711E | 2C6401DB | 1D9D7748F | 6E74A1670A | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 263 | 24 | 1EEE23D | 58C803B6 | 1B3AEE91E | 5CE942CE15 | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 264 | 25 | 1DDC47A | 3190076C | 1675DD23D | 39D2859C2B | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 265 | 26 | 1BB88F4 | 63200ED9 | 0CEBBA47A | 73A50B3856 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 266 | 27 | 17711E8 | 46401DB2 | 19D7748F5 | 674A1670AD | 0 | 0 | 1 | 0 | 0 | 11 | 01 | 01 |
| 267 | 28 | 0EE23D0 | 0C803B64 | 13AEE91EA | 4E942CE15B | 1 | 1 | 0 | 1 | 0 | 11 | 11 | 01 |
| 268 | 29 | 1DC47A0 | 190076C8 | 075DD23D4 | 1D2859C2B7 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 269 | 30 | 1B88F41 | 3200ED90 | 0EBBA47A9 | 3A50B3856E | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 270 | 31 | 1711E83 | 6401DB20 | 1D7748F53 | 74A1670ADC | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 271 | 32 | 0E23D07 | 4803B641 | 1AEE91EA7 | 6942CE15B8 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 272 | 33 | 1C47A0F | 10076C82 | 15DD23D4F | 52859C2B71 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 273 | 34 | 188F41E | 200ED905 | 0BBA47A9E | 250B3856E3 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 274 | 35 | 111E83C | 401DB20A | 17748F53D | 4A1670ADC7 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 275 | 36 | 023D078 | 003B6414 | 0EE91EA7A | 142CE15B8E | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 276 | 37 | 047A0F0 | 0076C828 | 1DD23D4F5 | 2859C2B71C | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 277 | 38 | 08F41E1 | 00ED9050 | 1BA47A9EA | 50B3856E39 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 278 | 39 | 11E83C2 | 01DB20A0 | 1748F53D5 | 21670ADC72 | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 279 | 40 | 03D0785 | 03B64141 | 0E91EA7AA | 42CE15B8E4 | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 280 | 41 | 07A0F0A | 076C8283 | 1D23D4F54 | 059C2B71C8 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 281 | 42 | 0F41E14 | 0ED90507 | 1A47A9EA9 | 0B3856E390 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 282 | 43 | 1E83C29 | 1DB20A0F | 148F53D52 | 1670ADC720 | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 283 | 44 | 1D07853 | 3B64141E | 091EA7AA5 | 2CE15B8E40 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 284 | 45 | 1A0F0A6 | 76C8283C | 123D4F54B | 59C2B71C81 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 285 | 46 | 141E14C | 6D905079 | 047A9EA97 | 33856E3902 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 286 | 47 | 083C299 | 5B20A0F2 | 08F53D52F | 670ADC7204 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 287 | 48 | 1078533 | 364141E4 | 11EA7AA5E | 4E15B8E408 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 288 | 49 | 00F0A67 | 6C8283C8 | 03D4F54BC | 1C2B71C811 | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 289 | 50 | 01E14CE | 59050791 | 07A9EA978 | 3856E39022 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 290 | 51 | 03C299C | 320A0F23 | 0F53D52F1 | 70ADC72045 | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 291 | 52 | 0785339 | 64141E47 | 1EA7AA5E2 | 615B8E408A | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 292 | 53 | 0F0A673 | 48283C8E | 1D4F54BC4 | 42B71C8115 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 293 | 54 | 1E14CE6 | 1050791C | 1A9EA9788 | 056E39022B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 294 | 55 | 1C299CD | 20A0F239 | 153D52F10 | 0ADC720456 | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |
| 295 | 56 | 185339B | 4141E472 | 0A7AA5E20 | 15B8E408AC | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 296 | 57 | 10A6736 | 0283C8E4 | 14F54BC41 | 2B71C81158 | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 297 | 58 | 014CE6C | 050791C9 | 09EA97882 | 56E39022B0 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 298 | 59 | 0299CD9 | 0A0F2393 | 13D52F104 | 2DC7204561 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 299 | 60 | 05339B3 | 141E4726 | 07AA5E208 | 5B8E408AC3 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 300 | 61 | 0A67366 | 283C8E4C | 0F54BC411 | 371C811587 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 301 | 62 | 14CE6CC | 50791C98 | 1EA978822 | 6E39022B0F | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 302 | 63 | 099CD99 | 20F23930 | 1D52F1045 | 5C7204561E | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 303 | 64 | 1339B33 | 41E47260 | 1AA5E208B | 38E408AC3D | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 304 | 65 | 0673666 | 03C8E4C0 | 154BC4117 | 71C811587A | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 305 | 66 | 0CE6CCC | 0791C980 | 0A978822E | 639022B0F5 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 306 | 67 | 19CD999 | 0F239301 | 152F1045C | 47204561EB | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 307 | 68 | 139B332 | 1E472603 | 0A5E208B9 | 0E408AC3D6 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 308 | 69 | 0736664 | 3C8E4C06 | 14BC41172 | 1C811587AD | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 309 | 70 | 0E6CCC8 | 791C980C | 0978822E5 | 39022B0F5A | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 310 | 71 | 1CD9990 | 72393019 | 12F1045CB | 7204561EB4 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 311 | 72 | 19B3320 | 64726033 | 05E208B97 | 6408AC3D69 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 312 | 73 | 1366640 | 48E4C067 | 0BC41172F | 4811587AD3 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 313 | 74 | 06CCC81 | 11C980CF | 178822E5E | 1022B0F5A6 | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 314 | 75 | 0D99903 | 2393019E | 0F1045CBC | 204561EB4C | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 315 | 76 | 1B33206 | 4726033D | 1E208B979 | 408AC3D699 | 1 | 0 | 1 | 1 | 1 | 10 | 10 | 11 |
| 316 | 77 | 166640D | 0E4C067B | 1C41172F2 | 011587AD33 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 317 | 78 | 0CCC81B | 1C980CF6 | 18822E5E5 | 022B0F5A66 | 1 | 1 | 1 | 0 | 1 | 01 | 10 | 10 |
| 318 | 79 | 1999036 | 393019EC | 11045CBCA | 04561EB4CD | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 319 | 80 | 133206C | 726033D9 | 0208B9794 | 08AC3D699B | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 320 | 81 | 06640D9 | 64C067B3 | 041172F29 | 11587AD337 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 321 | 82 | 0CC81B3 | 4980CF66 | 0822E5E53 | 22B0F5A66F | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 322 | 83 | 1990366 | 13019ECC | 1045CBCA6 | 4561EB4CDF | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 323 | 84 | 13206CC | 26033D98 | 008B9794D | 0AC3D699BE | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 324 | 85 | 0640D98 | 4C067B31 | 01172F29B | 1587AD337C | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 325 | 86 | 0C81B30 | 180CF662 | 022E5E537 | 2B0F5A66F9 | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 326 | 87 | 1903660 | 3019ECC5 | 045CBCA6F | 561EB4CDF3 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 327 | 88 | 1206CC1 | 6033D98A | 08B9794DE | 2C3D699BE6 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 328 | 89 | 040D983 | 4067B315 | 1172F29BD | 587AD337CC | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 329 | 90 | 081B306 | 00CF662A | 02E5E537A | 30F5A66F98 | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 330 | 91 | 103660C | 019ECC55 | 05CBCA6F4 | 61EB4CDF31 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |

*Sample Data*

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 331 | 92 | 006CC19 | 033D98AB | 0B9794DE8 | 43D699BE62 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 332 | 93 | 00D9833 | 067B3156 | 172F29BD0 | 07AD337CC5 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 333 | 94 | 01B3066 | 0CF662AC | 0E5E537A0 | 0F5A66F98B | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 334 | 95 | 03660CD | 19ECC559 | 1CBCA6F41 | 1EB4CDF317 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 335 | 96 | 06CC19B | 33D98AB2 | 19794DE83 | 3D699BE62F | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 336 | 97 | 0D98336 | 67B31565 | 12F29BD06 | 7AD337CC5F | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 337 | 98 | 1B3066D | 4F662ACA | 05E537A0C | 75A66F98BF | 1 | 0 | 0 | 1 | 0 | 10 | 10 | 11 |
| 338 | 99 | 1660CDB | 1ECC5594 | 0BCA6F418 | 6B4CDF317E | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 339 | 100 | 0CC19B7 | 3D98AB29 | 1794DE831 | 5699BE62FC | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 340 | 101 | 198336F | 7B315653 | 0F29BD062 | 2D337CC5F9 | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |
| 341 | 102 | 13066DE | 7662ACA7 | 1E537A0C5 | 5A66F98BF2 | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 342 | 103 | 060CDBC | 6CC5594F | 1CA6F418B | 34CDF317E4 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 343 | 104 | 0C19B78 | 598AB29F | 194DE8317 | 699BE62FC9 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 344 | 105 | 18336F1 | 3315653F | 129BD062E | 5337CC5F92 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 345 | 106 | 1066DE2 | 662ACA7E | 0537A0C5C | 266F98BF25 | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 346 | 107 | 00CDBC5 | 4C5594FD | 0A6F418B9 | 4CDF317E4B | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 347 | 108 | 019B78B | 18AB29FA | 14DE83172 | 19BE62FC96 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 348 | 109 | 0336F16 | 315653F4 | 09BD062E5 | 337CC5F92C | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 349 | 110 | 066DE2D | 62ACA7E8 | 137A0C5CA | 66F98BF258 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 350 | 111 | 0CDBC5B | 45594FD1 | 06F418B95 | 4DF317E4B1 | 1 | 0 | 0 | 1 | 0 | 11 | 10 | 01 |
| 351 | 112 | 19B78B6 | 0AB29FA2 | 0DE83172B | 1BE62FC962 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 352 | 113 | 136F16C | 15653F45 | 1BD062E57 | 37CC5F92C5 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 353 | 114 | 06DE2D9 | 2ACA7E8B | 17A0C5CAE | 6F98BF258B | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 354 | 115 | 0DBC5B2 | 5594FD16 | 0F418B95D | 5F317E4B16 | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 355 | 116 | 1B78B64 | 2B29FA2C | 1E83172BB | 3E62FC962C | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 356 | 117 | 16F16C8 | 5653F458 | 1D062E577 | 7CC5F92C58 | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 357 | 118 | 0DE2D91 | 2CA7E8B0 | 1A0C5CAEF | 798BF258B1 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 358 | 119 | 1BC5B23 | 594FD161 | 1418B95DF | 7317E4B163 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 359 | 120 | 178B647 | 329FA2C2 | 083172BBF | 662FC962C7 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 360 | 121 | 0F16C8E | 653F4584 | 1062E577F | 4C5F92C58E | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 361 | 122 | 1E2D91C | 4A7E8B09 | 00C5CAEFE | 18BF258B1C | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 362 | 123 | 1C5B238 | 14FD1613 | 018B95DFC | 317E4B1639 | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 363 | 124 | 18B6471 | 29FA2C27 | 03172BBF9 | 62FC962C72 | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 364 | 125 | 116C8E2 | 53F4584E | 062E577F3 | 45F92C58E4 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |

*Sample Data*

## 1.2 AES-CCM encryption sample data

All values below are hexadecimal and follow notation of AES-CCM: MSbyte to LSbyte & msbit to lsbit.

### 1.2.1 Sample data 1 (DM1, Central → Peripheral)

```
Payload byte length: 00
K: 89678967 89678967 45234523 45234523
Payload counter: 0000bc614e
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 66778899 aabbccdd
LT_ADDR: 1
Packet Type: 3
LLID: 2
Payload:

B0: 494e61bc 0000ddcc bbaa9988 77660000
B1: 00190200 00000000 00000000 00000000

Y0: bb01f0c5 16dfd7b5 0d0cccb8 eaebb347
Y1: a9adf6e6 7876cf95 118a09d5 ac3f216e

T: a9adf6e6

CTR0: 014e61bc 0000ddcc bbaa9988 77660000

S0: b90f2b23 f63717d3 38e0559d 1e7e785e

MIC: 10a2ddc5
Encrypted payload:
```

### 1.2.2 Sample data 2 (DM1, Central → Peripheral)

```
Payload byte length: 08
K: 89678967 89678967 45234523 45234523
Payload counter: 0000bc614e
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 66778899 aabbccdd
LT_ADDR: 1
Packet Type: 3
LLID: 2
Payload: 68696a6b 6c6d6e6f

B0: 494e61bc 0000ddcc bbaa9988 77660008
B1: 00190200 00000000 00000000 00000000
B2: 68696a6b 6c6d6e6f 00000000 00000000

Y0: 95ddc3d4 2c9a70f1 61a28ee2 c08271ab
Y1: 418635ff 54615443 8aceca41 fe274779
Y2: 08d78b32 9d78ed33 b285fc42 e178d781

T: 08d78b32

CTR0: 014e61bc 0000ddcc bbaa9988 77660000
CTR1: 014e61bc 0000ddcc bbaa9988 77660001

S0: b90f2b23 f63717d3 38e0559d 1e7e785e
S1: d8c7e3e1 02050abb 025d0895 17cbe5fb

MIC: b1d8a011
Encrypted payload: b0ae898a 6e6864d4
```

*Sample Data*

### 1.2.3 Sample data 3 (DM1, Peripheral → Central)

```
Payload byte length: 08
K: 89678967 89678967 45234523 45234523
Payload counter: 0000bc614e
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 66778899 aabbccdd
LT_ADDR: 1
Packet Type: 3
LLID: 2
Payload: 68696a6b 6c6d6e6f

B0: 494e61bc 0020ddcc bbaa9988 77660008
B1: 00190200 00000000 00000000 00000000
B2: 68696a6b 6c6d6e6f 00000000 00000000

Y0: 31081122 b1cca37a 5f04d238 897b9bc8
Y1: 02ee3065 95c5d55a d0a030a3 3bee507b
Y2: 7382a2ba aa874418 14eafbef 41f57180

T: 7382a2ba

CTR0: 014e61bc 0020ddcc bbaa9988 77660000
CTR1: 014e61bc 0020ddcc bbaa9988 77660001

S0: 2a4d408d 2035b058 cc2fbf3b 8de15c73
S1: 9c89f68f b31bf4b5 7fbc7e83 123bd8a8

MIC: 59cfe237
Encrypted payload: f4e09ce4 df769ada
```

*Sample Data*

### 1.2.4 Sample data 4 (DM1, Central → Peripheral)

```
Payload byte length: 11
K: ce2ad11b a11456bd bd9d8b1f 848322fc
Payload counter: 00bdb3be95
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 82b8b727 5bf92769
LT_ADDR: 1
Packet Type: 3
LLID: 2
Payload: 86126da5 dbb39164 9ba1cac4 60917233
         05

B0: 4995beb3 bd006927 f95b27b7 b8820011
B1: 00190200 00000000 00000000 00000000
B2: 86126da5 dbb39164 9ba1cac4 60917233
B3: 05000000 00000000 00000000 00000000

Y0: ab182b6f e8bca0a9 7cc306e0 eab19e84
Y1: c198f821 49061035 977a5aae 60c51726
Y2: 45dd4181 40facb43 0f73f71b 49ea36ae
Y3: 7b112114 38d06bc2 98cb22db c5218041

T: 7b112114

CTR0: 0195beb3 bd006927 f95b27b7 b8820000
CTR1: 0195beb3 bd006927 f95b27b7 b8820001
CTR2: 0195beb3 bd006927 f95b27b7 b8820002

S0: bd3d1368 1478c30c 62b734ac e8e00c6e
S1: bfaa326d 8d84d8f6 e4518d12 20babe4f
S2: fc4a6327 776a3136 604a1ab8 20836505

MIC: c62c327c
Encrypted payload: 39b85fc8 56374992 7ff047d6 402bcc7c
                   f9
```

### 1.2.5 Sample data 5 (DM1, Peripheral → Central)

```
Payload byte length: 11
K: ce2ad11b a11456bd bd9d8b1f 848322fc
Payload counter: 00bdb3be95
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 82b8b727 5bf92769
LT_ADDR: 1
Packet Type: 3
LLID: 2
Payload: 86126da5 dbb39164 9ba1cac4 60917233
         05

B0: 4995beb3 bd206927 f95b27b7 b8820011
B1: 00190200 00000000 00000000 00000000
B2: 86126da5 dbb39164 9ba1cac4 60917233
B3: 05000000 00000000 00000000 00000000

Y0: 2c317af0 b12026df 8400f84e aa8e53e7
Y1: 1ec2c0c5 74e2cad3 3e143b2b 9d63095d
Y2: e7f08f4b d7c24c04 651434d8 a84f8ae9
Y3: d6f08416 0d556004 6c9b850b 1b579614

T: d6f08416

CTR0: 0195beb3 bd206927 f95b27b7 b8820000
CTR1: 0195beb3 bd206927 f95b27b7 b8820001
CTR2: 0195beb3 bd206927 f95b27b7 b8820002

S0: d8bc791d b48ea182 ef438e70 ee0f50e1
S1: c5e90ff5 6e1e06b3 4d6b699c 9fb72e3d
S2: b68f4956 19bea370 0a1f118e a5dd6aff

MIC: 0e4cfd0b
Encrypted payload: 43fb6250 b5ad97d7 d6caa358 ff265c0e
                   b3
```

### 1.2.6 Sample data 6 (DH1, Central → Peripheral)

```
Payload byte length: 14
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: 4
LLID: 2
Payload: 9bb3a2bd dd043b3a 904cc247 0a1d545f
         b2095e3d

B0: 498bf767 62006f98 36b1e858 ca740014
B1: 00210200 00000000 00000000 00000000
B2: 9bb3a2bd dd043b3a 904cc247 0a1d545f
B3: b2095e3d 00000000 00000000 00000000

Y0: ae691727 39e614a3 e0be3227 ac9afd99
Y1: 47b13424 8ff2f5f7 eaea4fdd 0fab9d92
Y2: 36154960 3c1fc026 4509902e de57dfc3
Y3: 1d45d8f7 950a39c3 9779bb7c d1b3fe17

T: 1d45d8f7

CTR0: 018bf767 62006f98 36b1e858 ca740000
CTR1: 018bf767 62006f98 36b1e858 ca740001
CTR2: 018bf767 62006f98 36b1e858 ca740002

S0: 254593c4 cd12c6a7 d9dec572 95524b75
S1: af492e65 ca391b26 e8ce9653 498ed0de
S2: 869271ed ac79c1bc 3cf0f959 c2711f3b

MIC: 38004b33
Encrypted payload: 34fa8cd8 173d201c 78825414 43938481
                   349b2fd0
```

### 1.2.7 Sample data 7 (DH1, Peripheral → Central)

```
Payload byte length: 14
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: 4
LLID: 2
Payload: 9bb3a2bd dd043b3a 904cc247 0a1d545f
         b2095e3d

B0: 498bf767 62206f98 36b1e858 ca740014
B1: 00210200 00000000 00000000 00000000
B2: 9bb3a2bd dd043b3a 904cc247 0a1d545f
B3: b2095e3d 00000000 00000000 00000000

Y0: 55410490 6f3a5827 e5a04a60 7cf19ad5
Y1: 000cc95c c0d099ca b15b244b d3440c24
Y2: bd1d9815 96438c28 eebfd508 6cf80d34
Y3: 8c227888 d0725a21 ffba99b2 38043d5e

T: 8c227888

CTR0: 018bf767 62206f98 36b1e858 ca740000
CTR1: 018bf767 62206f98 36b1e858 ca740001
CTR2: 018bf767 62206f98 36b1e858 ca740002

S0: 1404487a 919e16c8 b3245d80 2b364231
S1: 82082cbc 57038db7 4823be9a 34e0a8d7
S2: 1b5f7526 d26fe763 7669dfee 63743d3a

MIC: 982630f2
Encrypted payload: 19bb8e01 8a07b68d d86f7cdd 3efdfc88
                   a9562b1b
```

### 1.2.8 Sample data 8 (DH1, Central → Peripheral)

```
Payload byte length: 1b
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: 4
LLID: 2
Payload: 8f11d05e e0e749b5 eeda42f9 2b184502
         95388ce5 872916b4 bf7260

B0: 498bf767 62006f98 36b1e858 ca74001b
B1: 00210200 00000000 00000000 00000000
B2: 8f11d05e e0e749b5 eeda42f9 2b184502
B3: 95388ce5 872916b4 bf726000 00000000

Y0: 28015834 f3117a84 904800f7 ebd4b0d4
Y1: 15c4a61e b7ae954e 2c9d1b19 3ba2a9e5
Y2: 832c185d 1effc0ee 94d3a26e 23aca8e6
Y3: dcef7067 fc38a84e 893670a6 fb9e069b

T: dcef7067

CTR0: 018bf767 62006f98 36b1e858 ca740000
CTR1: 018bf767 62006f98 36b1e858 ca740001
CTR2: 018bf767 62006f98 36b1e858 ca740002

S0: 254593c4 cd12c6a7 d9dec572 95524b75
S1: af492e65 ca391b26 e8ce9653 498ed0de
S2: 869271ed ac79c1bc 3cf0f959 c2711f3b

MIC: f9aae3a3
Encrypted payload: 2058fe3b 2ade5293 0614d4aa 629695dc
                   13aafd08 2b50d708 838299
```

### 1.2.9 Sample data 9 (DH1, Peripheral → Central)

```
Payload byte length: 1b
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: 4
LLID: 2
Payload: 8f11d05e e0e749b5 eeda42f9 2b184502
         95388ce5 872916b4 bf7260

B0: 498bf767 62206f98 36b1e858 ca74001b
B1: 00210200 00000000 00000000 00000000
B2: 8f11d05e e0e749b5 eeda42f9 2b184502
B3: 95388ce5 872916b4 bf726000 00000000

Y0: feb39b06 54b486da bf1cec46 b5c5ec2a
Y1: eeda0fc3 4057d94e 3572448d 67b640f4
Y2: 48461729 ec1c7060 3b0f88ce becef21a
Y3: b1ff4755 658c2aa2 862952a5 1ca041a1

T: b1ff4755

CTR0: 018bf767 62206f98 36b1e858 ca740000
CTR1: 018bf767 62206f98 36b1e858 ca740001
CTR2: 018bf767 62206f98 36b1e858 ca740002

S0: 1404487a 919e16c8 b3245d80 2b364231
S1: 82082cbc 57038db7 4823be9a 34e0a8d7
S2: 1b5f7526 d26fe763 7669dfee 63743d3a

MIC: a5fb0f2f
Encrypted payload: 0d19fce2 b7e4c402 a6f9fc63 1ff8edd5
                   8e67f9c3 5546f1d7 c91bbf
```

### 1.2.10 Sample data 10 (2-DH3, Central → Peripheral)

```
Payload byte length: 16f
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: a
LLID: 2
Payload: 969b0972 549738ea 89120710 55797f19
         631dd8e7 219308a0 836e8d6b a55ec08f
         42604406 543c2f96 60c261c3 1c3d8826
         73aab82e bd5a8278 93625aa8 b9a4c5b3
         bc310174 e4d6436e 2e44aa08 1d64e751
         b5501222 dcc34270 6aefd398 1e10b2e2
         56e20d95 1e4e68cc 3fdd4b5c 8e93809a
         ff008232 3b6a864e b8556219 e94fbdd2
         500550e9 939e6108 43a375ab a75d1f6d
         a0304656 b45f488c 0ba40259 4e1ee6a1
         c59301e8 f1507906 40dc0c24 330120c0
         ac7f6707 e7f00d4a ea6c0577 a31abbb6
         4f9b6bab 47bfa387 c89bbbe1 6d8cbd49
         4a9c452f 9d46ab05 dcf0f434 f4c27bce
         2e0e177d 1aba438d 64a8cd72 ca0c170c
         9fa6e227 992fe354 98c94581 f1d869ee
         b07ffcf2 c19b35c8 5e22939e b54c772c
         2c4c0963 f51a653d 777879f2 d1ab67fc
         ba300c9e fa3ba62e 9f70e4b9 1a996f81
         7a9dff0b 56fd15c2 e9858db3 9b33e8c2
         254df11b 64b9ac36 409f2406 5c9e478a
         fc3b8161 b32d1b56 9236e631 23ed2a53
         89d4c4e0 8a799f0a 370e7310 734c9f

B0: 498bf767 62006f98 36b1e858 ca74016f
B1: 00510200 00000000 00000000 00000000
B2: 969b0972 549738ea 89120710 55797f19
B3: 631dd8e7 219308a0 836e8d6b a55ec08f
B4: 42604406 543c2f96 60c261c3 1c3d8826
B5: 73aab82e bd5a8278 93625aa8 b9a4c5b3
```

*Sample Data*

```
B6: bc310174 e4d6436e 2e44aa08 1d64e751
B7: b5501222 dcc34270 6aefd398 1e10b2e2
B8: 56e20d95 1e4e68cc 3fdd4b5c 8e93809a
B9: ff008232 3b6a864e b8556219 e94fbdd2
B10: 500550e9 939e6108 43a375ab a75d1f6d
B11: a0304656 b45f488c 0ba40259 4e1ee6a1
B12: c59301e8 f1507906 40dc0c24 330120c0
B13: ac7f6707 e7f00d4a ea6c0577 a31abbb6
B14: 4f9b6bab 47bfa387 c89bbbe1 6d8cbd49
B15: 4a9c452f 9d46ab05 dcf0f434 f4c27bce
B16: 2e0e177d 1aba438d 64a8cd72 ca0c170c
B17: 9fa6e227 992fe354 98c94581 f1d869ee
B18: b07ffcf2 c19b35c8 5e22939e b54c772c
B19: 2c4c0963 f51a653d 777879f2 d1ab67fc
B20: ba300c9e fa3ba62e 9f70e4b9 1a996f81
B21: 7a9dff0b 56fd15c2 e9858db3 9b33e8c2
B22: 254df11b 64b9ac36 409f2406 5c9e478a
B23: fc3b8161 b32d1b56 9236e631 23ed2a53
B24: 89d4c4e0 8a799f0a 370e7310 734c9f00

Y0: 29d89cd3 f2a1cf11 de30fb32 7e036fd8
Y1: 969453f9 b7ed6ed5 0bb166ac ad84b447
Y2: 29c1dbe6 569d3bcd 517acede fdf9b2a3
Y3: 87ae6d80 ceb1d9ae e7e009f3 0564b2b4
Y4: f7b4d9bf a1fa7bba 484cba56 f72c649c
Y5: 7cff7307 318bed9a 94c81c3f 7fa87554
Y6: ccf442c7 832413c6 1387b50f 1d6af991
Y7: fda7fb71 83c5a785 a798814c d0a4f4ef
Y8: ca85d795 514e510a e715b603 ad7fd821
Y9: 65527777 4efa23ca d5124e05 0597d5ff
Y10: b17f66c4 8a523b41 950f09c0 fc3a2a15
Y11: 886ec25e ed7cd1af 44de48bf fde11c40
Y12: cd685d0d 10a34a2f 2fa75fee c8a36979
Y13: d7486efa 056ebad9 0f6fe2ac f9871d36
Y14: c28103ad 2fe40cb6 42337daa fee66a03
Y15: c4a313af aad33282 8db4432c 73550d1c
Y16: e29b7baa b3c83223 72fc11d5 b15c01bb
Y17: 22737a00 4e7f26b9 c9f368e1 a90843ae
Y18: 11f9a9f7 997119fb ebad9814 230e9f11
Y19: 026c4112 d016c255 9595bdc9 a8b14e95
Y20: dfe4c01a b101dfca 4bd2ca7e 4342d595
```

*Sample Data*

```
Y21: 5a128313 b4180f77 80a029cc 23a0c1fc
Y22: b4df1ace 2e0a3a2a 968d45e4 7a11a3cf
Y23: d2fb23c5 4849439d 80e8fed4 7ee0e5cb
Y24: da1c4547 7eab6f64 b771cf1e 8ca278ad

T: da1c4547

CTR0: 018bf767 62006f98 36b1e858 ca740000
CTR1: 018bf767 62006f98 36b1e858 ca740001
CTR2: 018bf767 62006f98 36b1e858 ca740002
CTR3: 018bf767 62006f98 36b1e858 ca740003
CTR4: 018bf767 62006f98 36b1e858 ca740004
CTR5: 018bf767 62006f98 36b1e858 ca740005
CTR6: 018bf767 62006f98 36b1e858 ca740006
CTR7: 018bf767 62006f98 36b1e858 ca740007
CTR8: 018bf767 62006f98 36b1e858 ca740008
CTR9: 018bf767 62006f98 36b1e858 ca740009
CTR10: 018bf767 62006f98 36b1e858 ca74000a
CTR11: 018bf767 62006f98 36b1e858 ca74000b
CTR12: 018bf767 62006f98 36b1e858 ca74000c
CTR13: 018bf767 62006f98 36b1e858 ca74000d
CTR14: 018bf767 62006f98 36b1e858 ca74000e
CTR15: 018bf767 62006f98 36b1e858 ca74000f
CTR16: 018bf767 62006f98 36b1e858 ca740010
CTR17: 018bf767 62006f98 36b1e858 ca740011
CTR18: 018bf767 62006f98 36b1e858 ca740012
CTR19: 018bf767 62006f98 36b1e858 ca740013
CTR20: 018bf767 62006f98 36b1e858 ca740014
CTR21: 018bf767 62006f98 36b1e858 ca740015
CTR22: 018bf767 62006f98 36b1e858 ca740016
CTR23: 018bf767 62006f98 36b1e858 ca740017

S0: 254593c4 cd12c6a7 d9dec572 95524b75
S1: af492e65 ca391b26 e8ce9653 498ed0de
S2: 869271ed ac79c1bc 3cf0f959 c2711f3b
S3: 31554cb7 aa9b1dfb 24ba8b26 eab59ad8
S4: 7cb39415 9dce80e6 0ac0e5e0 9667747e
S5: 9727b319 ccbcc251 d539fd08 497942c8
S6: bd972408 212a147c 3eb66687 1488e2d9
S7: aef3e149 6b4e615b 309a7f93 53ca2981
S8: 6b276914 1d957bec 4c87e76d ee681e76
```

*Sample Data*

```
S9: a2df383f eece9b0d 4553f2e8 6f8b6035
S10: 11dac6bc c4177830 c7038ee1 e6cb0579
S11: 30bfd1cf 7fa95155 7c0757bf f14840a0
S12: 2315682b 1f4cb1f6 10fd4365 4d9b6155
S13: 7f4aa2fd 211bbc18 9ff3dcd7 c8102220
S14: a072aa38 70e32a62 f7214fb1 97dbfbfd
S15: f7c2b622 96a1f9e1 d3e7837f 293f6e86
S16: e63dc9fd 830944d5 b9fa2257 b0e19402
S17: eceef697 e76f671f 5e7e8c6e ae43f63b
S18: a9bccd3a 4ae6b498 40a6ead6 cd6200a8
S19: 2b624b6b f923eb0f 110ff341 4b1ab902
S20: a47db290 f068ca62 c9236eec ddb431f4
S21: a4ad3dc5 7e324250 1938950e 7b3827fd
S22: 364c32d8 35f63c05 5c8f32dd e560cc8f
S23: f2fd81dc cd12d0cc 2e4f7834 43a74630

MIC: ff59d683
Encrypted payload: 39d22717 9eae23cc 61dc9143 1cf7afc7
                   e58fa90a 8deac91c bf9e7432 672fdfb4
                   733508b1 fea7326d 4478eae5 f68812fe
                   0f192c3b 2094029e 99a2bf48 2fc3b1cd
                   2b16b26d 286a813f fb7d5700 541da599
                   08c7362a fde9560c 5459b51f 0a98503b
                   f811ecdc 75000997 0f4734cf dd59a91b
                   9427eb26 26fffda2 f4d28574 0727a3a4
                   f2da68d6 7d50fa05 06f08743 c8d67f58
                   b1ea80ea 704830bc cca78cb8 a8d5e3d8
                   f52cd027 8ef92853 3cdb5b9b c2496060
                   8f6a0f2c f8bcbcbc fa914612 ee81dae3
                   30d1c956 66a41f9f 57686736 a59c9f69
                   eaeeef17 eda58167 2bd1bb85 63198033
                   d9cca15f 8c1bba6c b74f4e0d e333798a
                   799b2bda 1a26a781 213367d6 4139fdec
                   5c910a65 26f452d7 005c1ff0 1b0f8117
                   85f0c459 bffcd1a5 37de9324 1cc96754
                   915247f5 03184d21 8e7f17f8 5183d683
                   dee04d9b a695dfa0 20a6e35f 4687d936
                   81e0ccde 1a8bee66 59a7b108 27a66077
                   ca77b3b9 86db2753 ceb9d4ec c68de6dc
                   7b29453c 476b4fc6 19410b24 30ebd9
```

### 1.2.11 Sample data 11 (2-DH3, Peripheral → Central)

```
Payload byte length: 16f
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: a
LLID: 2
Payload:  969b0972 549738ea 89120710 55797f19
          631dd8e7 219308a0 836e8d6b a55ec08f
          42604406 543c2f96 60c261c3 1c3d8826
          73aab82e bd5a8278 93625aa8 b9a4c5b3
          bc310174 e4d6436e 2e44aa08 1d64e751
          b5501222 dcc34270 6aefd398 1e10b2e2
          56e20d95 1e4e68cc 3fdd4b5c 8e93809a
          ff008232 3b6a864e b8556219 e94fbdd2
          500550e9 939e6108 43a375ab a75d1f6d
          a0304656 b45f488c 0ba40259 4e1ee6a1
          c59301e8 f1507906 40dc0c24 330120c0
          ac7f6707 e7f00d4a ea6c0577 a31abbb6
          4f9b6bab 47bfa387 c89bbbe1 6d8cbd49
          4a9c452f 9d46ab05 dcf0f434 f4c27bce
          2e0e177d 1aba438d 64a8cd72 ca0c170c
          9fa6e227 992fe354 98c94581 f1d869ee
          b07ffcf2 c19b35c8 5e22939e b54c772c
          2c4c0963 f51a653d 777879f2 d1ab67fc
          ba300c9e fa3ba62e 9f70e4b9 1a996f81
          7a9dff0b 56fd15c2 e9858db3 9b33e8c2
          254df11b 64b9ac36 409f2406 5c9e478a
          fc3b8161 b32d1b56 9236e631 23ed2a53
          89d4c4e0 8a799f0a 370e7310 734c9f

B0: 498bf767 62206f98 36b1e858 ca74016f
B1: 00510200 00000000 00000000 00000000
B2: 969b0972 549738ea 89120710 55797f19
B3: 631dd8e7 219308a0 836e8d6b a55ec08f
B4: 42604406 543c2f96 60c261c3 1c3d8826
B5: 73aab82e bd5a8278 93625aa8 b9a4c5b3
```

*Sample Data*

```
B6: bc310174 e4d6436e 2e44aa08 1d64e751
B7: b5501222 dcc34270 6aefd398 1e10b2e2
B8: 56e20d95 1e4e68cc 3fdd4b5c 8e93809a
B9: ff008232 3b6a864e b8556219 e94fbdd2
B10: 500550e9 939e6108 43a375ab a75d1f6d
B11: a0304656 b45f488c 0ba40259 4e1ee6a1
B12: c59301e8 f1507906 40dc0c24 330120c0
B13: ac7f6707 e7f00d4a ea6c0577 a31abbb6
B14: 4f9b6bab 47bfa387 c89bbbe1 6d8cbd49
B15: 4a9c452f 9d46ab05 dcf0f434 f4c27bce
B16: 2e0e177d 1aba438d 64a8cd72 ca0c170c
B17: 9fa6e227 992fe354 98c94581 f1d869ee
B18: b07ffcf2 c19b35c8 5e22939e b54c772c
B19: 2c4c0963 f51a653d 777879f2 d1ab67fc
B20: ba300c9e fa3ba62e 9f70e4b9 1a996f81
B21: 7a9dff0b 56fd15c2 e9858db3 9b33e8c2
B22: 254df11b 64b9ac36 409f2406 5c9e478a
B23: fc3b8161 b32d1b56 9236e631 23ed2a53
B24: 89d4c4e0 8a799f0a 370e7310 734c9f00

Y0: 34969012 2648722b fb7771dd b1b38b1b
Y1: 8c511978 793004c6 975e5f19 93c19d99
Y2: 482014e5 39ddbbd4 0833c079 5bab45ef
Y3: b994181f a79795ce a6968237 28ad1659
Y4: 7b8fea33 1c4c8c50 d8ae584b bfc65033
Y5: e185c7b7 dd9831f5 2e0d2140 95368c09
Y6: d598bb23 fb4d4e82 a8b74b2b c95b5b71
Y7: 6c5c7fc1 8cc70897 3bb594a8 39541943
Y8: c7f822e8 7546115f 80d57035 7960abb8
Y9: a430ddc8 58f529f1 97bebcb7 e9160550
Y10: c9bf7627 33253b87 1490fbd7 353a3175
Y11: 1d93c403 de2c3b76 62e6ea6e cbf757c0
Y12: 0d607c00 af5502d9 2d56d483 745f6855
Y13: e27bdc66 de323cff a3a1620d 1637310e
Y14: 840d6d95 6785fead 710de246 e944bf2e
Y15: 806b9eb6 6517c4f8 1d55f260 e08f455b
Y16: 36883866 2f7275ab 6ba45111 2431882d
Y17: e5b4f1f6 cbbcc363 33ef1b05 94b5385e
Y18: e8d47695 67be9d89 04445e73 8a45e019
Y19: 77639b94 f0f9907c db541ef0 c8d45e4b
Y20: a2bd914a 281e1d5f 0c9b8de4 3fcb6565
```

*Sample Data*

```
Y21: 6c1072e6 5581e658 7d7a0561 e8a85ddf
Y22: 8256c105 5c932ba8 bb71936b c727d10f
Y23: 2f4cc66a d568be7e 500a7448 6c2278ad
Y24: 2ca06cb0 0009e3c3 9e123a8a 2c2869dc

T: 2ca06cb0

CTR0: 018bf767 62206f98 36b1e858 ca740000
CTR1: 018bf767 62206f98 36b1e858 ca740001
CTR2: 018bf767 62206f98 36b1e858 ca740002
CTR3: 018bf767 62206f98 36b1e858 ca740003
CTR4: 018bf767 62206f98 36b1e858 ca740004
CTR5: 018bf767 62206f98 36b1e858 ca740005
CTR6: 018bf767 62206f98 36b1e858 ca740006
CTR7: 018bf767 62206f98 36b1e858 ca740007
CTR8: 018bf767 62206f98 36b1e858 ca740008
CTR9: 018bf767 62206f98 36b1e858 ca740009
CTR10: 018bf767 62206f98 36b1e858 ca74000a
CTR11: 018bf767 62206f98 36b1e858 ca74000b
CTR12: 018bf767 62206f98 36b1e858 ca74000c
CTR13: 018bf767 62206f98 36b1e858 ca74000d
CTR14: 018bf767 62206f98 36b1e858 ca74000e
CTR15: 018bf767 62206f98 36b1e858 ca74000f
CTR16: 018bf767 62206f98 36b1e858 ca740010
CTR17: 018bf767 62206f98 36b1e858 ca740011
CTR18: 018bf767 62206f98 36b1e858 ca740012
CTR19: 018bf767 62206f98 36b1e858 ca740013
CTR20: 018bf767 62206f98 36b1e858 ca740014
CTR21: 018bf767 62206f98 36b1e858 ca740015
CTR22: 018bf767 62206f98 36b1e858 ca740016
CTR23: 018bf767 62206f98 36b1e858 ca740017

S0: 1404487a 919e16c8 b3245d80 2b364231
S1: 82082cbc 57038db7 4823be9a 34e0a8d7
S2: 1b5f7526 d26fe763 7669dfee 63743d3a
S3: a3673258 5020c6cd d72bf517 accb632d
S4: 8a60ebb6 59c59ac1 a45a1ffd f54f7cd7
S5: 036f35c7 130c8a30 25e9da14 706df5bc
S6: e328cbb0 09c91d56 f010b40e e0fc5c3f
S7: 626d3d67 b53eb139 e155c9a2 df407b17
S8: d94e430b 829c7caf 289d2898 3c68aa5a
```

*Sample Data*

```
S9: d6343452 d92cb1ac b3fde90b e2f0789f
S10: 8ab8413c 0df18adf ae07a74f 82b3c91f
S11: 61bfac52 125b8fc4 eaac30f7 5a543821
S12: 68899f8f 2892931d 88b08926 7fa3cfc9
S13: 86b668c5 4ef24455 22a45f42 61aea816
S14: f57b4941 8332f0b3 7749ea30 53a711ee
S15: 593562a8 45c75a7c 70030f08 6fedd965
S16: f85e742f a743f353 6a22b384 be1dabdc
S17: ed7e317b 635deed3 e8b619f2 9bc65eb0
S18: 2035622d 94718ad6 c1631fa2 755883f4
S19: 40f50638 ad826c4a 3ca4fb88 467445ae
S20: e27eb632 0c2aee5c 9210b1c7 736fb897
S21: 711a9be3 c7054fa5 82ec70b6 028489e7
S22: 7b543081 c52e2cba 26400e6b 46642d0d
S23: d9564733 01fa7fae 7cfac2a7 239639e2

MIC: 38a424ca
Encrypted payload: 149325ce 0394b55d c131b98a 6199d7ce
                   7842adc1 f3fcefc3 f5075285 c62afdb5
                   e107765e 041ce95b b7e994d4 b0f6eb0b
                   f9ca5398 e49f18b9 37384555 4cebb964
                   bf5e34b3 f7dac95e 0bad701c 6d0912ed
                   5678d992 d50a5f26 9aff6796 feeceedd
                   348f30f2 ab70d9f5 de8882fe 51d3fb8d
                   264ec139 b9f6fae1 90c84a81 d5271788
                   863164bb 4ab2d0a4 f05e9ca0 45ad67f2
                   2a88076a b9aec253 a5a3a516 ccad2fbe
                   a42cadba e30bf6c2 aa703cd3 695518e1
                   c4f6f888 cf629e57 62dc8c51 dcb9747f
                   c92d036e 094de7d2 ea3fe4a3 0c22155f
                   bfe70c6e 1e745bb6 abb91e04 a7656a20
                   773b75d5 5f7d19f1 14abc27a a5e1ce69
                   67f89608 3e6c1007 f2ebf605 4fc5c232
                   5d01cd89 a2c6db1b b6948a6c 2e8a299c
                   0c796b4e 616befeb b61b6650 a4f3e408
                   fac50aa6 57b9ca64 a3d41f31 5ced2a2f
                   98e34939 5ad7fb9e 7b953c74 e85c5055
                   54576af8 a3bce393 c27354b0 5e1ace6d
                   876fb1e0 760337ec b476e85a 6589075e
                   508283d3 8b83e0a4 4bf4b1b7 50daa6
```

### 1.2.12 Sample data 12 (3-DH5, Central → Peripheral)

```
Payload byte length: 3fd
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 0
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: f
LLID: 2
Payload: 6d42614c 50694940 209c8bd5 5be57733
         3ec1066a 76cb602e a58ced25 f98ceb94
         58bc3db4 e7c46bb9 e23f9220 68bfca00
         c1da8f08 1c10e526 0ea37fab 3d91be9f
         3a4e68e9 006cca11 fdc76c59 1e20769d
         e1e34385 af105dab 4d44eda7 eacd1974
         5414d5a9 568d67af c05aedd9 6726a130
         7ebe31fd 81881237 c953d2a5 42c57c3b
         019691ef 911953fb 39264712 c61e3e5e
         21286421 85891af5 bf8ca291 59c30596
         11bbe5cd 8f88a7bb b8afd34a 4211eed5
         850ca781 cc9cf5b9 06d5fced 79d35981
         39a1a239 2965b0d5 c6c03a9f e22433ba
         08e7aac7 7d207392 b3486ead dd5c81c6
         5454d575 edd91892 0a2f0fe9 f6d5c037
         fec1272e f22c9aa0 d02b3412 81f60847
         887cd303 a82937cf ded4be2d 139342ce
         bd09041a f5eaa675 4307eb2d 20a60f7b
         1b944afe e3ae1a6f 476021c7 d30d300e
         44f9eaa2 42e8cb7a 6d74d5b5 0f2d6c1b
         d436f44f 1ddf8579 70821a65 117e1200
         e0270f00 7cbe6bb2 020ec332 bc464299
         20131eb1 e8864206 3b4a8324 522cfe0a
         a5209fd7 3f11a1e0 da00c945 835b6b5f
         ec9eaea9 9d177dc0 cbd2efe8 21b388c3
         78b2e137 c84f37a4 5599ffc0 a9106204
         5ed1439f 7e67ea1d 6ab024f7 247e85df
         bf15d19c b0f488d2 cb06bed9 644ec34c
         2e69f752 4af38319 81c7556e 359bfaf5
         22a00878 4ce3e7e2 362698ea 6c00001d
```

*Sample Data*

```
          fdf0936d 2cf7a318 ed4f0447 ad506cdd
          c2fcf8b7 328bb527 063859b5 f60819d3
          eebdd291 0a12c6af 1c670a30 38fd9e2c
          4a6a3ad7 a51982bd 8d4fabf5 a8c16517
          831661b0 09405052 9fba337b 2b3544cf
          6811a761 093afd66 8b154a21 a5941b88
          a482c5ce f04b18c1 c2e67d7d f90c3a4d
          d4ee12fe 4b734174 d3ee8b0c 1ade74eb
          237710da 4694764b 7cce26c4 7a2570bf
          30bb18c7 6571ab05 26892de7 b5d62840
          7f300971 14d6014b 2ca566b3 d6ad1ef5
          96e552e6 defc287e 6a5a5c16 be31d26a
          392e1570 a9f9e0b3 32d223e5 b15407f3
          41cc55f8 3296f3f5 175ebece 580a3f24
          49494406 fa75e051 c829441b ce7cba98
          cb7ea85d 8031787d 9495b971 c6925f64
          2726ef05 932d3f1a 14a9bd1a d88a9b31
          6f54d80e 9fe31dcf c94c1f6a 92cc2c82
          60bd9296 e075b884 2976b667 041800df
          520e7a28 e5b9314a 0bb93966 1aba4643
          829544e2 d69f255c ce5bfa89 9a4704f3
          be803081 fbc36f5c 65fb13c8 69fc770a
          07cbc8ff 50b8dbe3 b171e9f9 ebc8cf22
          49127607 32973806 95979b3c d75a3f8f
          f933a408 d4945f63 96755d6e 493b554b
          58e78f5b a21d31b3 bbcddf62 83e94233
          15a3bea8 3a6ff73f f02809da 3c6d8208
          acfd6f05 b62bb5a5 91f1e4d5 4b84d357
          2f00672a 3e3c434c 1736072b 45f6370c
          bb46706b 56a57e86 3ed2217a 816e5c09
          1e2895f7 89b57c5e c0a67011 d5f2f69d
          6dac3941 3bc897dc cb42d3bc ffda31e0
          e961f3fb e4f40041 69e86cc0 530e891c
          7665902a 1ce0804c 921780ae e2

B0: 498bf767 62006f98 36b1e858 ca7403fd
B1: 00790200 00000000 00000000 00000000
B2: 6d42614c 50694940 209c8bd5 5be57733
B3: 3ec1066a 76cb602e a58ced25 f98ceb94
B4: 58bc3db4 e7c46bb9 e23f9220 68bfca00
B5: c1da8f08 1c10e526 0ea37fab 3d91be9f
```

*Sample Data*

```
B6: 3a4e68e9 006cca11 fdc76c59 1e20769d
B7: e1e34385 af105dab 4d44eda7 eacd1974
B8: 5414d5a9 568d67af c05aedd9 6726a130
B9: 7ebe31fd 81881237 c953d2a5 42c57c3b
B10: 019691ef 911953fb 39264712 c61e3e5e
B11: 21286421 85891af5 bf8ca291 59c30596
B12: 11bbe5cd 8f88a7bb b8afd34a 4211eed5
B13: 850ca781 cc9cf5b9 06d5fced 79d35981
B14: 39a1a239 2965b0d5 c6c03a9f e22433ba
B15: 08e7aac7 7d207392 b3486ead dd5c81c6
B16: 5454d575 edd91892 0a2f0fe9 f6d5c037
B17: fec1272e f22c9aa0 d02b3412 81f60847
B18: 887cd303 a82937cf ded4be2d 139342ce
B19: bd09041a f5eaa675 4307eb2d 20a60f7b
B20: 1b944afe e3ae1a6f 476021c7 d30d300e
B21: 44f9eaa2 42e8cb7a 6d74d5b5 0f2d6c1b
B22: d436f44f 1ddf8579 70821a65 117e1200
B23: e0270f00 7cbe6bb2 020ec332 bc464299
B24: 20131eb1 e8864206 3b4a8324 522cfe0a
B25: a5209fd7 3f11a1e0 da00c945 835b6b5f
B26: ec9eaea9 9d177dc0 cbd2efe8 21b388c3
B27: 78b2e137 c84f37a4 5599ffc0 a9106204
B28: 5ed1439f 7e67ea1d 6ab024f7 247e85df
B29: bf15d19c b0f488d2 cb06bed9 644ec34c
B30: 2e69f752 4af38319 81c7556e 359bfaf5
B31: 22a00878 4ce3e7e2 362698ea 6c00001d
B32: fdf0936d 2cf7a318 ed4f0447 ad506cdd
B33: c2fcf8b7 328bb527 063859b5 f60819d3
B34: eebdd291 0a12c6af 1c670a30 38fd9e2c
B35: 4a6a3ad7 a51982bd 8d4fabf5 a8c16517
B36: 831661b0 09405052 9fba337b 2b3544cf
B37: 6811a761 093afd66 8b154a21 a5941b88
B38: a482c5ce f04b18c1 c2e67d7d f90c3a4d
B39: d4ee12fe 4b734174 d3ee8b0c 1ade74eb
B40: 237710da 4694764b 7cce26c4 7a2570bf
B41: 30bb18c7 6571ab05 26892de7 b5d62840
B42: 7f300971 14d6014b 2ca566b3 d6ad1ef5
B43: 96e552e6 defc287e 6a5a5c16 be31d26a
B44: 392e1570 a9f9e0b3 32d223e5 b15407f3
B45: 41cc55f8 3296f3f5 175ebece 580a3f24
B46: 49494406 fa75e051 c829441b ce7cba98
```

*Sample Data*

```
B47: cb7ea85d 8031787d 9495b971 c6925f64
B48: 2726ef05 932d3f1a 14a9bd1a d88a9b31
B49: 6f54d80e 9fe31dcf c94c1f6a 92cc2c82
B50: 60bd9296 e075b884 2976b667 041800df
B51: 520e7a28 e5b9314a 0bb93966 1aba4643
B52: 829544e2 d69f255c ce5bfa89 9a4704f3
B53: be803081 fbc36f5c 65fb13c8 69fc770a
B54: 07cbc8ff 50b8dbe3 b171e9f9 ebc8cf22
B55: 49127607 32973806 95979b3c d75a3f8f
B56: f933a408 d4945f63 96755d6e 493b554b
B57: 58e78f5b a21d31b3 bbcddf62 83e94233
B58: 15a3bea8 3a6ff73f f02809da 3c6d8208
B59: acfd6f05 b62bb5a5 91f1e4d5 4b84d357
B60: 2f00672a 3e3c434c 1736072b 45f6370c
B61: bb46706b 56a57e86 3ed2217a 816e5c09
B62: 1e2895f7 89b57c5e c0a67011 d5f2f69d
B63: 6dac3941 3bc897dc cb42d3bc ffda31e0
B64: e961f3fb e4f40041 69e86cc0 530e891c
B65: 7665902a 1ce0804c 921780ae e2000000

Y0: ebe10d25 4ab27e31 1ea87d16 867d7904
Y1: 99b84ef8 a25d519e 700f76f1 85a74583
Y2: 23fd3478 f96bddd9 dd2e7ded 25f2515e
Y3: 5659d15b 1b569b1f 298f4430 7459cbe0
Y4: d0e6d2f8 939e7c9d 3774cf46 642295ce
Y5: 7e6662bc 99ec7ecc d985ddd4 d2365187
Y6: 421bb569 a5f4d07a 5157fed4 db03f630
Y7: 9fa62969 f43263b1 ac3e269f 15b844ff
Y8: a36e0d50 b75b2feb 45d8c9ee 052f33e4
Y9: 6245ece7 0ad0e314 2ce6ad6b f406b745
Y10: 41ff1de6 e1b3ccec eb6cfb07 fd150751
Y11: 43cae883 dd43266d 2f717cd4 b7c777ba
Y12: 92e3f4b7 bc4b8613 38c0043b 6885fbfb
Y13: b1595644 ec0a8e4e 803994e6 f3d284c5
Y14: e9f7ec01 c4f11349 a5f0a3aa efe88c98
Y15: 8c48cadc 7483c350 c11b6cb4 b8f32b25
Y16: b76f72f5 d54da8e0 b70bcaae 0727f4ed
Y17: b700d8e8 585256f8 38530196 ef3a4a6e
Y18: e0a0f7f0 8f252298 48f8e404 ff4d9f93
Y19: aea86952 8f028974 d890f4e5 52e6da13
Y20: 36566635 1c0a0078 9ee4499b d30ac682
```

*Sample Data*

```
Y21: 886e6fcb 9e9fe3e8 dbfa13ae 7d1e44a7
Y22: 43c5285e de2846da 266a4720 cbd3713e
Y23: 52c47b3e d9a6666e 566eaba7 9275a90f
Y24: ac04ee00 56922a78 5f48347a 3a2360f7
Y25: 33b5496f 546e71f0 80272f6a d189898a
Y26: 5519cbaf ff8fe542 4934f09a 39584456
Y27: 913a3746 5e6ad4c1 7fce844c f72dc1d8
Y28: 46cef034 a103a8f4 ac13040c 6c466b4b
Y29: 6cbffa5e 2dff5b95 87ba863b 26eab593
Y30: f3dbf258 03f21f71 4d03c4a2 7be84598
Y31: b3c7f849 ab6e9612 e9ece9d5 57c4c813
Y32: 39043423 c60f8fd8 2c108055 6f387c76
Y33: f9dd8872 1fa6f4f4 e5345613 fd5495fd
Y34: cf3d7135 63d2ccfd 41507bd0 fe759830
Y35: 8215cff7 93ae5d92 446fb102 632b4fb8
Y36: 8b02ecb6 fcc063f6 fb41e34a 19afdb3a
Y37: 982e82d3 171d9fb0 ab5b8fbf 17dc00d2
Y38: 2dc18652 f4818748 e3c5fe0e 36b47716
Y39: e6eaef36 46e56552 8ed8c309 ea46ba7f
Y40: 40842220 827424b0 b2f7d5cf b201b5e4
Y41: 5153e29d bc85bafb 36dea69c ec1f1a43
Y42: 08614ff3 44d96b3a f5b9e763 5600b69c
Y43: 2550a964 b17af58b 1843199b 1394de57
Y44: 11cda830 619505fa 49d971c9 c8051abb
Y45: d73824df 17b8ccc2 b52ba9ea ff6f6097
Y46: 100d3219 3486065d 9e7e7ebd 235c34c1
Y47: 440e8fbf e2af1797 2fa75056 2e1941d4
Y48: 91eb5850 4d92d2b8 a64b0e8e 34cf38ef
Y49: 715b4924 0a081cee 2509e363 c746ba6b
Y50: 1cc42047 e5fa2dfb 1c0901f0 a01676c5
Y51: 2634d6db 62d0d5b7 328c5278 6d44b7b1
Y52: 79a0548c 8589f663 323c1604 6af753a8
Y53: e0ba54a0 412e68c1 13c5daf1 8a6e275a
Y54: a1a5e608 3359873b cfb66476 052fcb79
Y55: ce70f15c f2971ebe 7c2d1e3c a288f591
Y56: d9a64946 58968fda 4756668d a5b82b89
Y57: 0c2d5786 6ae786a6 03c6db95 6d26186a
Y58: 00732c24 192905a4 3edaeb0d cc3d4a95
Y59: ae9a2e1c 042112df 4578395b cba685b1
Y60: 50fdc48d 0af8812c b3e64e85 3316b083
Y61: d2c14e67 888a7401 85bd5b91 37d6977a
```

*Sample Data*

```
Y62: 4b882e82 ca8338bb 2278c4d3 8d07f474
Y63: 4bf051e5 19bc961e 38c52cae 50e61d87
Y64: 657025a1 064ec7d6 e1d16bc0 049931bf
Y65: 6de9dbe8 c5a2ed24 66701ea3 4caee13e

T: 6de9dbe8

CTR0: 018bf767 62006f98 36b1e858 ca740000
CTR1: 018bf767 62006f98 36b1e858 ca740001
CTR2: 018bf767 62006f98 36b1e858 ca740002
CTR3: 018bf767 62006f98 36b1e858 ca740003
CTR4: 018bf767 62006f98 36b1e858 ca740004
CTR5: 018bf767 62006f98 36b1e858 ca740005
CTR6: 018bf767 62006f98 36b1e858 ca740006
CTR7: 018bf767 62006f98 36b1e858 ca740007
CTR8: 018bf767 62006f98 36b1e858 ca740008
CTR9: 018bf767 62006f98 36b1e858 ca740009
CTR10: 018bf767 62006f98 36b1e858 ca74000a
CTR11: 018bf767 62006f98 36b1e858 ca74000b
CTR12: 018bf767 62006f98 36b1e858 ca74000c
CTR13: 018bf767 62006f98 36b1e858 ca74000d
CTR14: 018bf767 62006f98 36b1e858 ca74000e
CTR15: 018bf767 62006f98 36b1e858 ca74000f
CTR16: 018bf767 62006f98 36b1e858 ca740010
CTR17: 018bf767 62006f98 36b1e858 ca740011
CTR18: 018bf767 62006f98 36b1e858 ca740012
CTR19: 018bf767 62006f98 36b1e858 ca740013
CTR20: 018bf767 62006f98 36b1e858 ca740014
CTR21: 018bf767 62006f98 36b1e858 ca740015
CTR22: 018bf767 62006f98 36b1e858 ca740016
CTR23: 018bf767 62006f98 36b1e858 ca740017
CTR24: 018bf767 62006f98 36b1e858 ca740018
CTR25: 018bf767 62006f98 36b1e858 ca740019
CTR26: 018bf767 62006f98 36b1e858 ca74001a
CTR27: 018bf767 62006f98 36b1e858 ca74001b
CTR28: 018bf767 62006f98 36b1e858 ca74001c
CTR29: 018bf767 62006f98 36b1e858 ca74001d
CTR30: 018bf767 62006f98 36b1e858 ca74001e
CTR31: 018bf767 62006f98 36b1e858 ca74001f
CTR32: 018bf767 62006f98 36b1e858 ca740020
CTR33: 018bf767 62006f98 36b1e858 ca740021
```

*Sample Data*

```
CTR34: 018bf767 62006f98 36b1e858 ca740022
CTR35: 018bf767 62006f98 36b1e858 ca740023
CTR36: 018bf767 62006f98 36b1e858 ca740024
CTR37: 018bf767 62006f98 36b1e858 ca740025
CTR38: 018bf767 62006f98 36b1e858 ca740026
CTR39: 018bf767 62006f98 36b1e858 ca740027
CTR40: 018bf767 62006f98 36b1e858 ca740028
CTR41: 018bf767 62006f98 36b1e858 ca740029
CTR42: 018bf767 62006f98 36b1e858 ca74002a
CTR43: 018bf767 62006f98 36b1e858 ca74002b
CTR44: 018bf767 62006f98 36b1e858 ca74002c
CTR45: 018bf767 62006f98 36b1e858 ca74002d
CTR46: 018bf767 62006f98 36b1e858 ca74002e
CTR47: 018bf767 62006f98 36b1e858 ca74002f
CTR48: 018bf767 62006f98 36b1e858 ca740030
CTR49: 018bf767 62006f98 36b1e858 ca740031
CTR50: 018bf767 62006f98 36b1e858 ca740032
CTR51: 018bf767 62006f98 36b1e858 ca740033
CTR52: 018bf767 62006f98 36b1e858 ca740034
CTR53: 018bf767 62006f98 36b1e858 ca740035
CTR54: 018bf767 62006f98 36b1e858 ca740036
CTR55: 018bf767 62006f98 36b1e858 ca740037
CTR56: 018bf767 62006f98 36b1e858 ca740038
CTR57: 018bf767 62006f98 36b1e858 ca740039
CTR58: 018bf767 62006f98 36b1e858 ca74003a
CTR59: 018bf767 62006f98 36b1e858 ca74003b
CTR60: 018bf767 62006f98 36b1e858 ca74003c
CTR61: 018bf767 62006f98 36b1e858 ca74003d
CTR62: 018bf767 62006f98 36b1e858 ca74003e
CTR63: 018bf767 62006f98 36b1e858 ca74003f
CTR64: 018bf767 62006f98 36b1e858 ca740040

S0: 254593c4 cd12c6a7 d9dec572 95524b75
S1: af492e65 ca391b26 e8ce9653 498ed0de
S2: 869271ed ac79c1bc 3cf0f959 c2711f3b
S3: 31554cb7 aa9b1dfb 24ba8b26 eab59ad8
S4: 7cb39415 9dce80e6 0ac0e5e0 9667747e
S5: 9727b319 ccbcc251 d539fd08 497942c8
S6: bd972408 212a147c 3eb66687 1488e2d9
S7: aef3e149 6b4e615b 309a7f93 53ca2981
S8: 6b276914 1d957bec 4c87e76d ee681e76
```

*Sample Data*

```
S9: a2df383f eece9b0d 4553f2e8 6f8b6035
S10: 11dac6bc c4177830 c7038ee1 e6cb0579
S11: 30bfd1cf 7fa95155 7c0757bf f14840a0
S12: 2315682b 1f4cb1f6 10fd4365 4d9b6155
S13: 7f4aa2fd 211bbc18 9ff3dcd7 c8102220
S14: a072aa38 70e32a62 f7214fb1 97dbfbfd
S15: f7c2b622 96a1f9e1 d3e7837f 293f6e86
S16: e63dc9fd 830944d5 b9fa2257 b0e19402
S17: eceef697 e76f671f 5e7e8c6e ae43f63b
S18: a9bccd3a 4ae6b498 40a6ead6 cd6200a8
S19: 2b624b6b f923eb0f 110ff341 4b1ab902
S20: a47db290 f068ca62 c9236eec ddb431f4
S21: a4ad3dc5 7e324250 1938950e 7b3827fd
S22: 364c32d8 35f63c05 5c8f32dd e560cc8f
S23: f2fd81dc cd12d0cc 2e4f7834 43a74630
S24: 7875bf58 8a162375 b25069d6 b82f4f36
S25: a66bb2d0 43870301 47dbe7f1 92f04b34
S26: 52b87f3e 796b208e 5a0f57c1 6de88c53
S27: 312e84cd 8f627142 ffa9b6cd 56ce3c59
S28: 6695a4dd 1e85cb05 c2070e7c fa16dd90
S29: 10df7734 8a106a97 3b37c508 e54fc157
S30: 54abd840 17756b69 0b7e187b ef5c8ea5
S31: 0bf5f442 c3fecabd 64b313ca 9787f134
S32: aaed31c5 98d1b73a ceaf242d 37b08fd4
S33: 37946570 fcf94220 77cf63da 4781771b
S34: ab5dd397 a7c0d893 7b17c1d2 eb9bc233
S35: 7367b29c 0ecc6a76 1a3c5602 40ca2ca2
S36: 6b593c6d f7485995 f36f1169 089c0be9
S37: ffca503f cfb8848c 73e37041 508128a3
S38: 444e1461 b95ef61a 90c93236 5aaa9fa8
S39: 4099c123 717f86df 23e16eaf 7ed015df
S40: 730217d1 28fa1af8 720864ad 99430469
S41: 43e6b844 95068645 5aca15ed 1bc668a6
S42: 9ee17b6a 65f56326 42e4b829 b5348945
S43: acf20ec7 151b75f0 dd4023ba d099b71c
S44: 41880777 e033c519 79eee51e 14a9246d
S45: 02bb2b01 4981e028 ddb66c67 a7077a1b
S46: db90a526 db75c631 d31af69d ef7b9082
S47: fddfda45 35a30553 78f655cb eae4d1d3
S48: 3472e976 3205e39b e8295a5d 9be10244
S49: fbe961bd 23f25c70 c87cac28 157b55d1
```

*Sample Data*

```
S50: 20a9174a 8da17e48 167086fd 5201ccbb
S51: 390a62b8 90a669c1 b89476ac a46ee788
S52: e354ab7f bd3afd7d 02361116 91afbf21
S53: cbe14419 ab1a4841 3155aaab e1d41a64
S54: cf8fd9f8 8d63ce6a c4c81f7b 852a081f
S55: c6abaae1 44875175 f239963a 194565d1
S56: 8f420880 95982bda d1317139 e9ca663b
S57: 5c7e5c2e 9a457be9 325620ed e0465cf7
S58: f9eb32f4 e1804702 eaff74d1 dea8a295
S59: eb66a965 5551f198 b13495a8 2200509c
S60: a0ee623d 5562b3a6 2cea2c8d f60eb0d8
S61: e7562630 9f7d5b7b 4da8effc e4a1e015
S62: fa1a5da9 7cdfed2b efd7b409 e065c33e
S63: 238ac0b6 f00d1b5b 77b091cc f4a4ffc4
S64: 533e37d7 b179b943 53511492 e4f58248

MIC: 48ac482c
Encrypted payload: c20b4f29 9a505266 c8521d86 126ba7ed
                   b8537787 dab2a192 997c147c 3bfdf4af
                   69e97103 4d5f7642 c6851906 820a50d8
                   bd691b1d 81de65c0 04639a4b abf6cae1
                   ad69dbf0 ccd00840 28fe9151 57593455
                   5c74678d 8e3a49d7 73f28b20 fe45fbad
                   fae734e0 3dc306f4 f0c0924a 34ec88b1
                   159958e9 9c1d69db 85d435c8 acad624d
                   a349a9d0 7fd7c8f6 7c75b5fa a9955e6b
                   30f2a29d 419e62c5 788f2c70 bf0800ef
                   21043402 f021f6ee c4a884f5 b359ae75
                   a619cfaa d3d0444f 1628bf88 344838d4
                   46eb00c4 087e0ccd 5933e648 2a34119a
                   a89500ff 0dc359f0 4469211c 4a877a3b
                   a3966357 7b78e173 d9c88c96 dfeaaeb1
                   18fceed3 7125de75 69d11645 31179c45
                   64922594 4f4650d0 80aa3243 bdd0b4f5
                   14b5c920 bf0c12ed 03a101fb edc40fd3
                   30f60195 1a8df160 566fd286 9817890c
                   e0845832 b2800118 a457bb59 d2995def
                   709bc98a 63edc729 69ba8f6b 6a4635fd
                   d66b3dd8 494857b7 5e81f1ef 59268e16
                   d2ee9f6d 259492ca 1505fb10 118bb83a
                   dd55208f b5078295 6850a093 3b742469
```

*Sample Data*

```
4af51c79 de907ec1 8c090819 b343c3f7
2a0a9e09 b124172a 0f96a801 c4f8ee57
6fffc752 f1059b5f 9519923a 72b0b986
d9807541 ae7143d7 0901b0a5 9e581edc
3eb68066 c0e3e98e baf09066 d0d43ba2
760bd038 5b968c8b 3d588091 835c8eb8
f605672f ef0969a5 89fc178d 3ad79de9
6811c972 aa5a021d c8977d98 c1b89607
d929b7e1 f6eb848f 6ba869ea 7f7ce937
e137e940 02d95a2e f6586a27 435aa724
f071d32c 078c3a24 85866579 6bff686d
03489b0c fe72a4f3 787a5b48 ad081061
5b4895f1 3ff39c4d b1050d3c a98d12ee
90a0069f f22db76e 4327b93a 4074eb43
63eed1f9 37ebf094 5f2f486b 04f56560
43b90f16 4d8bb1fd 5481494a 2c952c29
3cd6b135 81d0870e 766f735e cd6b7653
0804298c bb094b58 28bee43f 0b055b2f
95dc1bb7 bce29543 ef92005f 61cdb0ef
0044528f d2a536ec 6eb05bd0 4ca31b49
4bf26f07 b3f40079 159f287c 697bc083
10ee0d7b 5b44be4c 478f4fec 29e9cfe6
daf93540 a68e3a49 6c5fe8d1 326e4ae2
5b263178 ade6fe54 21654537 092d2ec6
9b54f32b c387e4f4 e10a1a4f 1163550e
72a76d62 68184f02 1dc9bf9b 48bb8af8
bb9f265a 46394c9d 76cf8c25 3e29e37b
5dd49bfe 46f99221 67cd02de f853c82b
cc2a8ce6 fba293a2 80244352 0a1cd546
869dafff bff4f66c 515f8447 52703790
3f980ee9 90130e16 644ccb54 507e309a
d7a587db 37851a69 6afcae5b 6a232408
49dde286 a02a8cd6 c27e2937 dc2bdeff
55165df1 57abf2a7 7b0e9004 952c71c2
c466ce4f 6b6db2d4 a6029283 67f66790
1ba81256 03c7cd20 12380df7 7760ecd1
f97eb3c7 16c82725 8d0e9fed 31531688
97b664e8 47177af7 249567b5 1fbff2de
caeb334d 14f91b1a 1e58fd0c a7aa76d8
255ba7fd ad99390f c146943c 06
```

### 1.2.13 Sample data 13 (3-DH5, Peripheral → Central)

```
Payload byte length: 3fd
K: 7b04934f d9d25294 ef1a014d a094f0b5
Payload counter: 006267f78b
Zero-length ACL-U Continuation: 0
Direction: 1
Initialization vector: 74ca58e8 b136986f
LT_ADDR: 1
Packet Type: f
LLID: 2
Payload: 6d42614c 50694940 209c8bd5 5be57733
         3ec1066a 76cb602e a58ced25 f98ceb94
         58bc3db4 e7c46bb9 e23f9220 68bfca00
         c1da8f08 1c10e526 0ea37fab 3d91be9f
         3a4e68e9 006cca11 fdc76c59 1e20769d
         e1e34385 af105dab 4d44eda7 eacd1974
         5414d5a9 568d67af c05aedd9 6726a130
         7ebe31fd 81881237 c953d2a5 42c57c3b
         019691ef 911953fb 39264712 c61e3e5e
         21286421 85891af5 bf8ca291 59c30596
         11bbe5cd 8f88a7bb b8afd34a 4211eed5
         850ca781 cc9cf5b9 06d5fced 79d35981
         39a1a239 2965b0d5 c6c03a9f e22433ba
         08e7aac7 7d207392 b3486ead dd5c81c6
         5454d575 edd91892 0a2f0fe9 f6d5c037
         fec1272e f22c9aa0 d02b3412 81f60847
         887cd303 a82937cf ded4be2d 139342ce
         bd09041a f5eaa675 4307eb2d 20a60f7b
         1b944afe e3ae1a6f 476021c7 d30d300e
         44f9eaa2 42e8cb7a 6d74d5b5 0f2d6c1b
         d436f44f 1ddf8579 70821a65 117e1200
         e0270f00 7cbe6bb2 020ec332 bc464299
         20131eb1 e8864206 3b4a8324 522cfe0a
         a5209fd7 3f11a1e0 da00c945 835b6b5f
         ec9eaea9 9d177dc0 cbd2efe8 21b388c3
         78b2e137 c84f37a4 5599ffc0 a9106204
         5ed1439f 7e67ea1d 6ab024f7 247e85df
         bf15d19c b0f488d2 cb06bed9 644ec34c
         2e69f752 4af38319 81c7556e 359bfaf5
         22a00878 4ce3e7e2 362698ea 6c00001d
```

*Sample Data*

```
          fdf0936d 2cf7a318 ed4f0447 ad506cdd
          c2fcf8b7 328bb527 063859b5 f60819d3
          eebdd291 0a12c6af 1c670a30 38fd9e2c
          4a6a3ad7 a51982bd 8d4fabf5 a8c16517
          831661b0 09405052 9fba337b 2b3544cf
          6811a761 093afd66 8b154a21 a5941b88
          a482c5ce f04b18c1 c2e67d7d f90c3a4d
          d4ee12fe 4b734174 d3ee8b0c 1ade74eb
          237710da 4694764b 7cce26c4 7a2570bf
          30bb18c7 6571ab05 26892de7 b5d62840
          7f300971 14d6014b 2ca566b3 d6ad1ef5
          96e552e6 defc287e 6a5a5c16 be31d26a
          392e1570 a9f9e0b3 32d223e5 b15407f3
          41cc55f8 3296f3f5 175ebece 580a3f24
          49494406 fa75e051 c829441b ce7cba98
          cb7ea85d 8031787d 9495b971 c6925f64
          2726ef05 932d3f1a 14a9bd1a d88a9b31
          6f54d80e 9fe31dcf c94c1f6a 92cc2c82
          60bd9296 e075b884 2976b667 041800df
          520e7a28 e5b9314a 0bb93966 1aba4643
          829544e2 d69f255c ce5bfa89 9a4704f3
          be803081 fbc36f5c 65fb13c8 69fc770a
          07cbc8ff 50b8dbe3 b171e9f9 ebc8cf22
          49127607 32973806 95979b3c d75a3f8f
          f933a408 d4945f63 96755d6e 493b554b
          58e78f5b a21d31b3 bbcddf62 83e94233
          15a3bea8 3a6ff73f f02809da 3c6d8208
          acfd6f05 b62bb5a5 91f1e4d5 4b84d357
          2f00672a 3e3c434c 1736072b 45f6370c
          bb46706b 56a57e86 3ed2217a 816e5c09
          1e2895f7 89b57c5e c0a67011 d5f2f69d
          6dac3941 3bc897dc cb42d3bc ffda31e0
          e961f3fb e4f40041 69e86cc0 530e891c
          7665902a 1ce0804c 921780ae e2

B0: 498bf767 62206f98 36b1e858 ca7403fd
B1: 00790200 00000000 00000000 00000000
B2: 6d42614c 50694940 209c8bd5 5be57733
B3: 3ec1066a 76cb602e a58ced25 f98ceb94
B4: 58bc3db4 e7c46bb9 e23f9220 68bfca00
B5: c1da8f08 1c10e526 0ea37fab 3d91be9f
```

*Sample Data*

```
B6: 3a4e68e9 006cca11 fdc76c59 1e20769d
B7: e1e34385 af105dab 4d44eda7 eacd1974
B8: 5414d5a9 568d67af c05aedd9 6726a130
B9: 7ebe31fd 81881237 c953d2a5 42c57c3b
B10: 019691ef 911953fb 39264712 c61e3e5e
B11: 21286421 85891af5 bf8ca291 59c30596
B12: 11bbe5cd 8f88a7bb b8afd34a 4211eed5
B13: 850ca781 cc9cf5b9 06d5fced 79d35981
B14: 39a1a239 2965b0d5 c6c03a9f e22433ba
B15: 08e7aac7 7d207392 b3486ead dd5c81c6
B16: 5454d575 edd91892 0a2f0fe9 f6d5c037
B17: fec1272e f22c9aa0 d02b3412 81f60847
B18: 887cd303 a82937cf ded4be2d 139342ce
B19: bd09041a f5eaa675 4307eb2d 20a60f7b
B20: 1b944afe e3ae1a6f 476021c7 d30d300e
B21: 44f9eaa2 42e8cb7a 6d74d5b5 0f2d6c1b
B22: d436f44f 1ddf8579 70821a65 117e1200
B23: e0270f00 7cbe6bb2 020ec332 bc464299
B24: 20131eb1 e8864206 3b4a8324 522cfe0a
B25: a5209fd7 3f11a1e0 da00c945 835b6b5f
B26: ec9eaea9 9d177dc0 cbd2efe8 21b388c3
B27: 78b2e137 c84f37a4 5599ffc0 a9106204
B28: 5ed1439f 7e67ea1d 6ab024f7 247e85df
B29: bf15d19c b0f488d2 cb06bed9 644ec34c
B30: 2e69f752 4af38319 81c7556e 359bfaf5
B31: 22a00878 4ce3e7e2 362698ea 6c00001d
B32: fdf0936d 2cf7a318 ed4f0447 ad506cdd
B33: c2fcf8b7 328bb527 063859b5 f60819d3
B34: eebdd291 0a12c6af 1c670a30 38fd9e2c
B35: 4a6a3ad7 a51982bd 8d4fabf5 a8c16517
B36: 831661b0 09405052 9fba337b 2b3544cf
B37: 6811a761 093afd66 8b154a21 a5941b88
B38: a482c5ce f04b18c1 c2e67d7d f90c3a4d
B39: d4ee12fe 4b734174 d3ee8b0c 1ade74eb
B40: 237710da 4694764b 7cce26c4 7a2570bf
B41: 30bb18c7 6571ab05 26892de7 b5d62840
B42: 7f300971 14d6014b 2ca566b3 d6ad1ef5
B43: 96e552e6 defc287e 6a5a5c16 be31d26a
B44: 392e1570 a9f9e0b3 32d223e5 b15407f3
B45: 41cc55f8 3296f3f5 175ebece 580a3f24
B46: 49494406 fa75e051 c829441b ce7cba98
```

*Sample Data*

```
B47: cb7ea85d 8031787d 9495b971 c6925f64
B48: 2726ef05 932d3f1a 14a9bd1a d88a9b31
B49: 6f54d80e 9fe31dcf c94c1f6a 92cc2c82
B50: 60bd9296 e075b884 2976b667 041800df
B51: 520e7a28 e5b9314a 0bb93966 1aba4643
B52: 829544e2 d69f255c ce5bfa89 9a4704f3
B53: be803081 fbc36f5c 65fb13c8 69fc770a
B54: 07cbc8ff 50b8dbe3 b171e9f9 ebc8cf22
B55: 49127607 32973806 95979b3c d75a3f8f
B56: f933a408 d4945f63 96755d6e 493b554b
B57: 58e78f5b a21d31b3 bbcddf62 83e94233
B58: 15a3bea8 3a6ff73f f02809da 3c6d8208
B59: acfd6f05 b62bb5a5 91f1e4d5 4b84d357
B60: 2f00672a 3e3c434c 1736072b 45f6370c
B61: bb46706b 56a57e86 3ed2217a 816e5c09
B62: 1e2895f7 89b57c5e c0a67011 d5f2f69d
B63: 6dac3941 3bc897dc cb42d3bc ffda31e0
B64: e961f3fb e4f40041 69e86cc0 530e891c
B65: 7665902a 1ce0804c 921780ae e2000000

Y0: b7b63a11 d2ff710c 5d821353 120fe064
Y1: 31635446 16989968 dbd392df 77ab44b2
Y2: fe667b06 7ef88846 a74b5119 476c5ca9
Y3: 03db41e6 902e40ff fa6d86e6 92a2c630
Y4: 0a1c2366 e55a2cd0 701fe43d 34becee0
Y5: 8e8c6fd4 dfb9071b 3feb7938 defa835e
Y6: 61b2fe53 6f876519 43698e8d f7ca327e
Y7: 3705a0c3 39989f81 377afb1b d9f01569
Y8: d96211f1 959c7a64 3ee0a727 b4ad2ae2
Y9: 7cdc85ed d204d4ce 33665710 bcd4f7f1
Y10: 9a6d0fed cfde263e 62eddf28 fab38c98
Y11: 98915f60 f70cc4df 74ac9931 1a4f2990
Y12: 4a88113a 754243af c00324ef 117678f7
Y13: a0f620c0 b3a903fa 648210ef 63eef46f
Y14: f526db99 20291004 225a762f 129e58ff
Y15: 8ee96b03 494a8b18 1e1194b7 babb8a27
Y16: 59b69b74 0455f2ac 65f9c651 34ceae70
Y17: 9c0a2ee9 64e57ffb 57bcd10d f3c8a567
Y18: ee339202 c09ff61a 72fc2a2f 183b370e
Y19: 57b1bac2 63769a7b db0eac84 6b49aca4
Y20: 20e3786a e9526a50 cdc9dcc5 c579a3bf
```

*Sample Data*

```
Y21: 6bc61121 214dd538 aa3cc0a3 9abab72a
Y22: 5a42d758 fb854450 55199e78 ba42ca62
Y23: fc9cf1f6 58691ce8 01ca2501 25223403
Y24: cfa5ab4f 90431bc8 f832f148 db633e8e
Y25: 7254ae54 60a7dcbd 3470df60 7af3c1ca
Y26: 436205c4 763e68ab 390907e2 da236e8d
Y27: 6738b5db e1b8b739 d5eccc0d 47ab511c
Y28: 65a1dd1e f661d376 1c917ea7 f171dc7e
Y29: 55a82897 a7d19703 87e783cd 52ddecd8
Y30: 7e0ae507 5c9b8dc1 e4981e02 377472b5
Y31: 035d4dd3 493a84b5 0c2bd3ae 2f9f728b
Y32: 34441d1c cd9400d9 164c74b3 5021a97e
Y33: 4c5c3051 0c1d8571 782993c0 3a7a9e13
Y34: 63453300 3bc6cc3d 53a36db3 79f2b047
Y35: e3630826 7ac4040c b087d657 953a4560
Y36: 967ef5bd bf821572 466bf00f aefd751b
Y37: 4f19c371 c5da0b9f 4bc55452 05dab223
Y38: 6fa54807 d40a41ca 3c41bf6a 890d56c9
Y39: fb2d42e7 013c1e0d 17aa1e01 090fe190
Y40: f6104499 02efdc76 6cae97b7 f27d8765
Y41: 5d9ebc6d 22b48c7c 276b9670 636433cb
Y42: d85fb261 60c02b6d 348e6616 8d79f268
Y43: c66ab5e0 7fc159dd 07f41e8d 511989d0
Y44: e18f58b0 c831550b 2aa0f002 51899793
Y45: c36620e2 7a396916 74cc358f 36d4ae68
Y46: 6b1cb137 b69d021f 764150fd 5b29744f
Y47: cc3b7f27 a69a9666 6919d755 6893a18f
Y48: b1e1c8f8 936c88d1 b2540bf8 e82032d9
Y49: 045ad0f2 638df0e6 79357e37 0240dd8a
Y50: 455f563e 5cf94329 a4efbede ac30d09f
Y51: f6e8f947 ec78897c 96fe5879 a1519ede
Y52: f2a0a601 d7765bb7 0a41a0b6 fae10c36
Y53: 86bcc098 bcd1e930 7b008ec5 6c027ccb
Y54: 8f0f380b c973be6a c88d8a4d 45f86b0c
Y55: 2b5ca348 5bb4015a e9b907f6 1cc77400
Y56: a2755ca8 2c52702b ed08736d ef68938a
Y57: d92292d5 0a0cf7cd 79d95799 6c6dff88
Y58: 469d1262 83afb58c 9810c732 a4a36a79
Y59: bb563499 7ca1c960 8357c393 60148fff
Y60: 83e8dbc7 67e501de 080aac2d 0501b9ec
Y61: 956f13a2 c14f9476 d268896c f24ed47a
```

*Sample Data*

```
Y62: 1dfcb90c 4b088e40 830784fe f9dbe7c6
Y63: e7c8898d 9c033db9 117191d0 57d0d5cb
Y64: 6a187312 e7bc05cc bd8e8199 e9e7348d
Y65: 682ec5d4 8fec944f 0e02eeb0 bcf11786

T: 682ec5d4

CTR0: 018bf767 62206f98 36b1e858 ca740000
CTR1: 018bf767 62206f98 36b1e858 ca740001
CTR2: 018bf767 62206f98 36b1e858 ca740002
CTR3: 018bf767 62206f98 36b1e858 ca740003
CTR4: 018bf767 62206f98 36b1e858 ca740004
CTR5: 018bf767 62206f98 36b1e858 ca740005
CTR6: 018bf767 62206f98 36b1e858 ca740006
CTR7: 018bf767 62206f98 36b1e858 ca740007
CTR8: 018bf767 62206f98 36b1e858 ca740008
CTR9: 018bf767 62206f98 36b1e858 ca740009
CTR10: 018bf767 62206f98 36b1e858 ca74000a
CTR11: 018bf767 62206f98 36b1e858 ca74000b
CTR12: 018bf767 62206f98 36b1e858 ca74000c
CTR13: 018bf767 62206f98 36b1e858 ca74000d
CTR14: 018bf767 62206f98 36b1e858 ca74000e
CTR15: 018bf767 62206f98 36b1e858 ca74000f
CTR16: 018bf767 62206f98 36b1e858 ca740010
CTR17: 018bf767 62206f98 36b1e858 ca740011
CTR18: 018bf767 62206f98 36b1e858 ca740012
CTR19: 018bf767 62206f98 36b1e858 ca740013
CTR20: 018bf767 62206f98 36b1e858 ca740014
CTR21: 018bf767 62206f98 36b1e858 ca740015
CTR22: 018bf767 62206f98 36b1e858 ca740016
CTR23: 018bf767 62206f98 36b1e858 ca740017
CTR24: 018bf767 62206f98 36b1e858 ca740018
CTR25: 018bf767 62206f98 36b1e858 ca740019
CTR26: 018bf767 62206f98 36b1e858 ca74001a
CTR27: 018bf767 62206f98 36b1e858 ca74001b
CTR28: 018bf767 62206f98 36b1e858 ca74001c
CTR29: 018bf767 62206f98 36b1e858 ca74001d
CTR30: 018bf767 62206f98 36b1e858 ca74001e
CTR31: 018bf767 62206f98 36b1e858 ca74001f
CTR32: 018bf767 62206f98 36b1e858 ca740020
CTR33: 018bf767 62206f98 36b1e858 ca740021
```

*Sample Data*

```
CTR34: 018bf767 62206f98 36b1e858 ca740022
CTR35: 018bf767 62206f98 36b1e858 ca740023
CTR36: 018bf767 62206f98 36b1e858 ca740024
CTR37: 018bf767 62206f98 36b1e858 ca740025
CTR38: 018bf767 62206f98 36b1e858 ca740026
CTR39: 018bf767 62206f98 36b1e858 ca740027
CTR40: 018bf767 62206f98 36b1e858 ca740028
CTR41: 018bf767 62206f98 36b1e858 ca740029
CTR42: 018bf767 62206f98 36b1e858 ca74002a
CTR43: 018bf767 62206f98 36b1e858 ca74002b
CTR44: 018bf767 62206f98 36b1e858 ca74002c
CTR45: 018bf767 62206f98 36b1e858 ca74002d
CTR46: 018bf767 62206f98 36b1e858 ca74002e
CTR47: 018bf767 62206f98 36b1e858 ca74002f
CTR48: 018bf767 62206f98 36b1e858 ca740030
CTR49: 018bf767 62206f98 36b1e858 ca740031
CTR50: 018bf767 62206f98 36b1e858 ca740032
CTR51: 018bf767 62206f98 36b1e858 ca740033
CTR52: 018bf767 62206f98 36b1e858 ca740034
CTR53: 018bf767 62206f98 36b1e858 ca740035
CTR54: 018bf767 62206f98 36b1e858 ca740036
CTR55: 018bf767 62206f98 36b1e858 ca740037
CTR56: 018bf767 62206f98 36b1e858 ca740038
CTR57: 018bf767 62206f98 36b1e858 ca740039
CTR58: 018bf767 62206f98 36b1e858 ca74003a
CTR59: 018bf767 62206f98 36b1e858 ca74003b
CTR60: 018bf767 62206f98 36b1e858 ca74003c
CTR61: 018bf767 62206f98 36b1e858 ca74003d
CTR62: 018bf767 62206f98 36b1e858 ca74003e
CTR63: 018bf767 62206f98 36b1e858 ca74003f
CTR64: 018bf767 62206f98 36b1e858 ca740040

S0: 1404487a 919e16c8 b3245d80 2b364231
S1: 82082cbc 57038db7 4823be9a 34e0a8d7
S2: 1b5f7526 d26fe763 7669dfee 63743d3a
S3: a3673258 5020c6cd d72bf517 accb632d
S4: 8a60ebb6 59c59ac1 a45a1ffd f54f7cd7
S5: 036f35c7 130c8a30 25e9da14 706df5bc
S6: e328cbb0 09c91d56 f010b40e e0fc5c3f
S7: 626d3d67 b53eb139 e155c9a2 df407b17
S8: d94e430b 829c7caf 289d2898 3c68aa5a
```

*Sample Data*

```
S9: d6343452 d92cb1ac b3fde90b e2f0789f
S10: 8ab8413c 0df18adf ae07a74f 82b3c91f
S11: 61bfac52 125b8fc4 eaac30f7 5a543821
S12: 68899f8f 2892931d 88b08926 7fa3cfc9
S13: 86b668c5 4ef24455 22a45f42 61aea816
S14: f57b4941 8332f0b3 7749ea30 53a711ee
S15: 593562a8 45c75a7c 70030f08 6fedd965
S16: f85e742f a743f353 6a22b384 be1dabdc
S17: ed7e317b 635deed3 e8b619f2 9bc65eb0
S18: 2035622d 94718ad6 c1631fa2 755883f4
S19: 40f50638 ad826c4a 3ca4fb88 467445ae
S20: e27eb632 0c2aee5c 9210b1c7 736fb897
S21: 711a9be3 c7054fa5 82ec70b6 028489e7
S22: 7b543081 c52e2cba 26400e6b 46642d0d
S23: d9564733 01fa7fae 7cfac2a7 239639e2
S24: 202ed696 6fbc2ecf f0982979 25bef3ea
S25: e2148e71 5935e2a5 bd175ee9 a799a549
S26: 77310821 6671e6dc 3a121369 a557aa4f
S27: 6d6c2659 06036f0c 1cec57ed 4a36158d
S28: b216aab3 be06fde0 262ae2ea e81cd2e5
S29: 0b56812e 52b653f9 0e27362e dd7dcbd4
S30: df590fb5 d38ea1cd f8eb2d5b 3d474253
S31: 6493924c 073da4bf d5658181 48b52d76
S32: e8491efc f7d18b0a 027f656e c6886a7d
S33: eb26ca54 5d221211 73a037ec e43053e9
S34: fa2f9edf 4e9d9c1c aa6e786c e743c1c0
S35: cf833d58 0d508e0e da69560a 66d3c03b
S36: f5922c3b 4382f00f 9b64ebb1 b2accbb0
S37: 9a3faeeb 69fdb9e1 3a025f2d 299c177e
S38: 376d0236 76cfe2b1 1c0e342a afe6418b
S39: 1d2a5ab1 7dc816ee 6c37e855 07f11bd7
S40: c33e7e9f 7669ad90 44625f47 1155c411
S41: bae0615e 9d48b215 0b9e7065 04b09ef7
S42: 606e0db2 17e5251d ba08c3db f20915fd
S43: f28282de b65b5dcc 0b88f0d0 20fa4810
S44: 701e67ab aa5af6c2 1bf15c6f ea339a58
S45: bd5176ce 91b3298f fe989ff0 c6ff991f
S46: 9eb857c4 bfec9af6 b1c78acd f3aabd69
S47: 7acfb2d7 217154f8 09cd5f4c 9728c6cc
S48: 66b07ebe 3665809d 7b855491 4bf90528
S49: 91a82a7d f3744bd0 7fc8850c 08554c8d
```

*Sample Data*

```
S50: 0fa45385 3cca72db eaa031b2 65c9d590
S51: 2bf70aa1 dab678d7 f8d8a2b2 88dc22da
S52: 1633856d aacda2bb 94f1d66b 0f4ad876
S53: b09b906a ff48f724 b9524495 409a0a62
S54: 7cda29ed 1e6f903d b7a9f4c5 26c28530
S55: f5338a50 74e0c5ed 214c4380 70886009
S56: 9cd2fb3c f8e8d55f 93a10ccd f288eead
S57: 9f1b72ba 2e506062 9dcc02a2 a3536e0e
S58: ea50be8b e9ffe171 c6f64b44 1357e68e
S59: 5d3c94b8 ef4051ff 79cd4200 3d9a3208
S60: 14267f84 ee8d7e71 df5d46ec 361f1648
S61: 7871fd55 dbcbcf6c 20c2902d 58935861
S62: 33b892d6 63807bf4 ddc0cc2b 3e037547
S63: 611a7ca9 85d3571a 1b6a0917 40795e7c
S64: ef6adb06 6b6cb075 5ac5a39a 08ae7e7b

MIC: 7c2a8dae
Encrypted payload: ef4a4df0 076ac4f7 68bf354f 6f05dfe4
                   259e734c a4a4874d d3e532cb 9af8d6ae
                   fbdb0fec b7e4ad74 35146737 c474a92d
                   4bba64be 45d57fe7 aaf96056 c8dec248
                   39215d2e 13604021 d82eb64d 6e4d8321
                   02cb8835 a6d940fd bd5459a9 0a31454b
                   3679e8ce e3b3d696 210f247b b866da27
                   a7f072f6 03146e98 e1cefa3d 7eadd661
                   d7a2a5bd 4835e257 8adbae19 24ee46c1
                   ab90251d 8878902a 118b05de db70cc89
                   7004499f 9dd3287f 5203e3bd 1845d6f4
                   ed85380e e40e66a4 8e6575cb 06709648
                   bf17cafc 6797f480 e46465dd 838a9bac
                   fd9ce386 fe128321 c401849d 8efb9028
                   0d61b7dd a81e42ee 7a2c00e1 99381952
                   069f5301 556f69f3 ba098796 3feba39b
                   6502e278 cb74d91c 3662a7df 88551c7e
                   9d3c6637 619b2ca3 8264f48f 55fe8c8f
                   5b614cc6 4e2c7625 7bc4da4f 957975a0
                   a6875c90 4ec22526 ff646472 7c42d48c
                   a52c6fac dadacadc f26e6ad3 13fa9be7
                   9b733f81 b9904708 244ecd59 fa226f94
                   f9455982 e97c3da8 47b04183 71bac7e8
                   850e4941 50ad8f2f 2a98e03c a6e598b5
```

*Sample Data*

```
0e8a20d8 c4229f65 76c5b101 862a2d8a
0f83e916 ae3ed178 6f8beca9 0c47c84b
33bd65c6 78648511 765c731a 6e489052
0d037b2f 0ef27532 ed2c5c33 8c5211a9
253f767c 1845d0e0 8fe06340 e8e63121
fdf907cd 9f6d462f cecdb5b1 5147424e
99630121 2bca07a7 382a85c6 e5e541ab
2ab5e64b c55a3e2d 04473cdb 308073ae
059b18c5 5730d4be 6fc73ddc dccdcdc5
b045a408 eb841ea1 2721d399 4f82a4d7
4c955ce8 0410de5c 45d36571 4de684f4
9d838b5a 4ab80d69 1071a190 1738d038
3ebd6b25 99b6a120 f8e42250 d0902d33
e38310c8 3dbca3c5 cfe0bf26 b5383560
3e5d4a6b 3b5c60a5 10f9ce91 7dd46b68
f3856658 13180695 62eb72a0 a483ec51
c5d0682f 899eb35e 273b16d6 d21d8002
f68b5f54 c9190d63 d0529fcd 4c38c797
cbac97ae 1fa2bd7f 395ad335 91ae4fe3
31d23253 98cc0537 0cafe2a1 b239a57c
f41832c8 6bc6c9de 36b1dbeb 08832387
55c6ff99 3fdde28b 255233bc 3538e20d
5de95dd2 b25c6be2 1d64e256 4fa25dfd
09e4a6b0 a9869d52 b2c94bfb d93529aa
f115b8eb 1301f354 56be336b 0c4d4c52
5daa29ad d9734391 e11908d4 7f7393d3
a9624e43 0c295d8b 3683583b 129b2629
a8b3b5ec 510ecde7 f10ac5a3 66b6af7c
b7505895 aff02cc7 0823ad6c ab52c540
35c85fea 2cf8a83b 223e6ff9 f198babf
0c002e58 a0749a8e b7391eee 39b33542
c4357467 5af5e4ec 286cd3af 7161ac9e
8ab8cc12 143f975d 6de40b78 9f3eec06
46add18e 5fd454d4 5707af91 58d335d9
723cf392 d17c12b3 6efb452b 786c0504
af600fef b82800f7 e18f6796 b7714a41
665968a2 527eb332 e064e03c 8d61aefc
5e14ab97 5848ec28 16821f97 c1d944a7
887b8f52 6127575b 728265d7 1377d760
990f4b2c 778c3039 c8d22334 ea
```

### 1.2.14 Sample data 14 (EV3)

```
Payload byte length: 1e
K: 972b7dcd be8942b7 d8fdc356 a5590aca
CLK[27:1]: 030b5a8f
dayCounter: 061a
Initialization vector: 70b690bd dfe9630f
LT_ADDR: 1
Packet Type: 7
Payload: 4625f778 578290f9 0ee83576 b3f8a898
         faff3fe4 5db812b2 e189297a fd89

CTR0: 018f5a0b d3700f63 e9dfbd90 b6700000
CTR1: 018f5a0b d3700f63 e9dfbd90 b6700001
CTR2: 018f5a0b d3700f63 e9dfbd90 b6700002

Encrypted payload: 76637937 69b2ba4c 41704100 d6c4602a
                   dbae2a87 f77a9fdb a0c56d2c b532
```

# 2 FREQUENCY HOPPING SAMPLE DATA

The section contains three sets of sample data showing the basic and adapted hopping schemes for different combinations of addresses and initial clock values.

## 2.1 First set

```
Hop sequence {k} for Page Scan and Inquiry Scan substates:
CLKN start:   0x0000000
UAP / LAP:    0x00000000
#ticks:       0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
              -------------------------------------------------------
0x0000000:       0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0008000:      16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0010000:      32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0018000:      48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |
0x0020000:       0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0028000:      16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0030000:      32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0038000:      48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |

Hop sequence {k} for Page and Inquiry substates:
CLKE start:   0x0000000
UAP / LAP:    0x00000000
#ticks:       00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
              -----------------------------------------------------------
0x0000000:    48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000010:    00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0000020:    48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000030:    00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0001000:    48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001010:    32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0001020:    48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001030:    32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
...
0x0002000:    16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002010:    00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0002020:    16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002030:    00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0003000:    48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
```

*Sample Data*

```
0x0003010:    32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0003020:    48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
0x0003030:    32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |

Hop sequence {k} for Peripheral Response substate:
CLKN* =       0x0000010
UAP / LAP:    0x00000000
#ticks:       00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
              ------------------------------------------------------------------
0x0000012:    64 | 02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16
0x0000032:    01 | 18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32
0x0000052:    72 | 34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48
0x0000072:    09 | 50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00

Hop sequence {k} for Central Response substate:
Offset value: 24
CLKE* =       0x0000012
UAP / LAP:    0x00000000
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000014:    02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16 01 |
0x0000034:    18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32 72 |
0x0000054:    34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48 09 |
0x0000074:    50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00 64 |

Hop sequence {k} for Connection state (Basic channel hopping sequence;
ie, non-AFH):
CLK start:    0x0000010
UAP/LAP:      0x00000000
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010:    08 66 | 10 70 | 12 19 | 14 23 | 16 01 | 18 05 | 20 33 | 22 37 |
0x0000030:    24 03 | 26 07 | 28 35 | 30 39 | 32 72 | 34 76 | 36 25 | 38 29 |
0x0000050:    40 74 | 42 78 | 44 27 | 46 31 | 48 09 | 50 13 | 52 41 | 54 45 |
0x0000070:    56 11 | 58 15 | 60 43 | 62 47 | 32 17 | 36 19 | 34 49 | 38 51 |
0x0000090:    40 21 | 44 23 | 42 53 | 46 55 | 48 33 | 52 35 | 50 65 | 54 67 |
0x00000b0:    56 37 | 60 39 | 58 69 | 62 71 | 64 25 | 68 27 | 66 57 | 70 59 |
0x00000d0:    72 29 | 76 31 | 74 61 | 78 63 | 01 41 | 05 43 | 03 73 | 07 75 |
0x00000f0:    09 45 | 13 47 | 11 77 | 15 00 | 64 49 | 66 53 | 68 02 | 70 06 |
0x0000110:    01 51 | 03 55 | 05 04 | 07 08 | 72 57 | 74 61 | 76 10 | 78 14 |
0x0000130:    09 59 | 11 63 | 13 12 | 15 16 | 17 65 | 19 69 | 21 18 | 23 22 |
0x0000150:    33 67 | 35 71 | 37 20 | 39 24 | 25 73 | 27 77 | 29 26 | 31 30 |
0x0000170:    41 75 | 43 00 | 45 28 | 47 32 | 17 02 | 21 04 | 19 34 | 23 36 |
0x0000190:    33 06 | 37 08 | 35 38 | 39 40 | 25 10 | 29 12 | 27 42 | 31 44 |
```

*Sample Data*

```
0x00001b0:    41 14 | 45 16 | 43 46 | 47 48 | 49 18 | 53 20 | 51 50 | 55 52 |
0x00001d0:    65 22 | 69 24 | 67 54 | 71 56 | 57 26 | 61 28 | 59 58 | 63 60 |
0x00001f0:    73 30 | 77 32 | 75 62 | 00 64 | 49 34 | 51 42 | 57 66 | 59 74 |
0x0000210:    53 36 | 55 44 | 61 68 | 63 76 | 65 50 | 67 58 | 73 03 | 75 11 |
0x0000230:    69 52 | 71 60 | 77 05 | 00 13 | 02 38 | 04 46 | 10 70 | 12 78 |
0x0000250:    06 40 | 08 48 | 14 72 | 16 01 | 18 54 | 20 62 | 26 07 | 28 15 |
0x0000270:    22 56 | 24 64 | 30 09 | 32 17 | 02 66 | 06 74 | 10 19 | 14 27 |
0x0000290:    04 70 | 08 78 | 12 23 | 16 31 | 18 03 | 22 11 | 26 35 | 30 43 |
0x00002b0:    20 07 | 24 15 | 28 39 | 32 47 | 34 68 | 38 76 | 42 21 | 46 29 |
0x00002d0:    36 72 | 40 01 | 44 25 | 48 33 | 50 05 | 54 13 | 58 37 | 62 45 |
0x00002f0:    52 09 | 56 17 | 60 41 | 64 49 | 34 19 | 36 35 | 50 51 | 52 67 |
0x0000310:    38 21 | 40 37 | 54 53 | 56 69 | 42 27 | 44 43 | 58 59 | 60 75 |
0x0000330:    46 29 | 48 45 | 62 61 | 64 77 | 66 23 | 68 39 | 03 55 | 05 71 |
0x0000350:    70 25 | 72 41 | 07 57 | 09 73 | 74 31 | 76 47 | 11 63 | 13 00 |
0x0000370:    78 33 | 01 49 | 15 65 | 17 02 | 66 51 | 70 67 | 03 04 | 07 20 |
0x0000390:    68 55 | 72 71 | 05 08 | 09 24 | 74 59 | 78 75 | 11 12 | 15 28 |
0x00003b0:    76 63 | 01 00 | 13 16 | 17 32 | 19 53 | 23 69 | 35 06 | 39 22 |
0x00003d0:    21 57 | 25 73 | 37 10 | 41 26 | 27 61 | 31 77 | 43 14 | 47 30 |
0x00003f0:    29 65 | 33 02 | 45 18 | 49 34 | 19 04 | 21 08 | 23 20 | 25 24 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence
with all channels used; ie, AFH(79)):
CLK start:    0x0000010
ULAP:         0x00000000
Used Channels:0x7fffffffffffffffffff
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
0x0000030     24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
0x0000050     40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
0x0000070     56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
0x0000090     40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
0x00000b0     56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
0x00000d0     72 72 | 76 76 | 74 74 | 78 78 | 01 01 | 05 05 | 03 03 | 07 07 |
0x00000f0     09 09 | 13 13 | 11 11 | 15 15 | 64 64 | 66 66 | 68 68 | 70 70 |
0x0000110     01 01 | 03 03 | 05 05 | 07 07 | 72 72 | 74 74 | 76 76 | 78 78 |
0x0000130     09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
0x0000150     33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
0x0000170     41 41 | 43 43 | 45 45 | 47 47 | 17 17 | 21 21 | 19 19 | 23 23 |
0x0000190     33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
0x00001b0     41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
0x00001d0     65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
0x00001f0     73 73 | 77 77 | 75 75 | 00 00 | 49 49 | 51 51 | 57 57 | 59 59 |
```

*Sample Data*

```
0x0000210     53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
0x0000230     69 69 | 71 71 | 77 77 | 00 00 | 02 02 | 04 04 | 10 10 | 12 12 |
0x0000250     06 06 | 08 08 | 14 14 | 16 16 | 18 18 | 20 20 | 26 26 | 28 28 |
0x0000270     22 22 | 24 24 | 30 30 | 32 32 | 02 02 | 06 06 | 10 10 | 14 14 |
0x0000290     04 04 | 08 08 | 12 12 | 16 16 | 18 18 | 22 22 | 26 26 | 30 30 |
0x00002b0     20 20 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
0x00002d0     36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
0x00002f0     52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
0x0000310     38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
0x0000330     46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 03 03 | 05 05 |
0x0000350     70 70 | 72 72 | 07 07 | 09 09 | 74 74 | 76 76 | 11 11 | 13 13 |
0x0000370     78 78 | 01 01 | 15 15 | 17 17 | 66 66 | 70 70 | 03 03 | 07 07 |
0x0000390     68 68 | 72 72 | 05 05 | 09 09 | 74 74 | 78 78 | 11 11 | 15 15 |
0x00003b0     76 76 | 01 01 | 13 13 | 17 17 | 19 19 | 23 23 | 35 35 | 39 39 |
0x00003d0     21 21 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
0x00003f0     29 29 | 33 33 | 45 45 | 49 49 | 19 19 | 21 21 | 23 23 | 25 25 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence
with channels 0 to 21 unused):
CLK start:    0x0000010
ULAP:         0x00000000
Used Channels:0x7fffffffffffc00000
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     30 30 | 32 32 | 34 34 | 36 36 | 38 38 | 40 40 | 42 42 | 22 22 |
0x0000030     24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
0x0000050     40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
0x0000070     56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
0x0000090     40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
0x00000b0     56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
0x00000d0     72 72 | 76 76 | 74 74 | 78 78 | 45 45 | 49 49 | 47 47 | 51 51 |
0x00000f0     53 53 | 57 57 | 55 55 | 59 59 | 64 64 | 66 66 | 68 68 | 70 70 |
0x0000110     45 45 | 47 47 | 49 49 | 51 51 | 72 72 | 74 74 | 76 76 | 78 78 |
0x0000130     53 53 | 55 55 | 57 57 | 59 59 | 61 61 | 63 63 | 65 65 | 23 23 |
0x0000150     33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
0x0000170     41 41 | 43 43 | 45 45 | 47 47 | 61 61 | 65 65 | 63 63 | 23 23 |
0x0000190     33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
0x00001b0     41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
0x00001d0     65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
0x00001f0     73 73 | 77 77 | 75 75 | 66 66 | 49 49 | 51 51 | 57 57 | 59 59 |
0x0000210     53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
0x0000230     69 69 | 71 71 | 77 77 | 66 66 | 68 68 | 70 70 | 76 76 | 78 78 |
0x0000250     72 72 | 74 74 | 23 23 | 25 25 | 27 27 | 29 29 | 26 26 | 28 28 |
```

*Sample Data*

```
0x0000270     22 22 | 24 24 | 30 30 | 32 32 | 68 68 | 72 72 | 76 76 | 23 23 |
0x0000290     70 70 | 74 74 | 78 78 | 25 25 | 27 27 | 22 22 | 26 26 | 30 30 |
0x00002b0     29 29 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
0x00002d0     36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
0x00002f0     52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
0x0000310     38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
0x0000330     46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 34 34 | 36 36 |
0x0000350     70 70 | 72 72 | 38 38 | 40 40 | 74 74 | 76 76 | 42 42 | 44 44 |
0x0000370     78 78 | 32 32 | 46 46 | 48 48 | 66 66 | 70 70 | 34 34 | 38 38 |
0x0000390     68 68 | 72 72 | 36 36 | 40 40 | 74 74 | 78 78 | 42 42 | 46 46 |
0x00003b0     76 76 | 32 32 | 44 44 | 48 48 | 50 50 | 23 23 | 35 35 | 39 39 |
0x00003d0     52 52 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
0x00003f0     29 29 | 33 33 | 45 45 | 49 49 | 50 50 | 52 52 | 23 23 | 25 25 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence
with even channels used):
CLK start:    0x0000010
ULAP:         0x00000000
Used Channels:0x55555555555555555
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ---------------------------------------------------------------
0x0000010     08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
0x0000030     24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
0x0000050     40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
0x0000070     56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
0x0000090     40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
0x00000b0     56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
0x00000d0     72 72 | 76 76 | 74 74 | 78 78 | 00 00 | 04 04 | 02 02 | 06 06 |
0x00000f0     08 08 | 12 12 | 10 10 | 14 14 | 64 64 | 66 66 | 68 68 | 70 70 |
0x0000110     00 00 | 02 02 | 04 04 | 06 06 | 72 72 | 74 74 | 76 76 | 78 78 |
0x0000130     08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
0x0000150     32 32 | 34 34 | 36 36 | 38 38 | 24 24 | 26 26 | 28 28 | 30 30 |
0x0000170     40 40 | 42 42 | 44 44 | 46 46 | 16 16 | 20 20 | 18 18 | 22 22 |
0x0000190     32 32 | 36 36 | 34 34 | 38 38 | 24 24 | 28 28 | 26 26 | 30 30 |
0x00001b0     40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
0x00001d0     64 64 | 68 68 | 66 66 | 70 70 | 56 56 | 60 60 | 58 58 | 62 62 |
0x00001f0     72 72 | 76 76 | 74 74 | 00 00 | 48 48 | 50 50 | 56 56 | 58 58 |
0x0000210     52 52 | 54 54 | 60 60 | 62 62 | 64 64 | 66 66 | 72 72 | 74 74 |
0x0000230     68 68 | 70 70 | 76 76 | 00 00 | 02 02 | 04 04 | 10 10 | 12 12 |
0x0000250     06 06 | 08 08 | 14 14 | 16 16 | 18 18 | 20 20 | 26 26 | 28 28 |
0x0000270     22 22 | 24 24 | 30 30 | 32 32 | 02 02 | 06 06 | 10 10 | 14 14 |
0x0000290     04 04 | 08 08 | 12 12 | 16 16 | 18 18 | 22 22 | 26 26 | 30 30 |
0x00002b0     20 20 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
```

*Sample Data*

```
0x00002d0     36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
0x00002f0     52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
0x0000310     38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
0x0000330     46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 00 00 | 02 02 |
0x0000350     70 70 | 72 72 | 04 04 | 06 06 | 74 74 | 76 76 | 08 08 | 10 10 |
0x0000370     78 78 | 78 78 | 12 12 | 14 14 | 66 66 | 70 70 | 00 00 | 04 04 |
0x0000390     68 68 | 72 72 | 02 02 | 06 06 | 74 74 | 78 78 | 08 08 | 12 12 |
0x00003b0     76 76 | 78 78 | 10 10 | 14 14 | 16 16 | 20 20 | 32 32 | 36 36 |
0x00003d0     18 18 | 22 22 | 34 34 | 38 38 | 24 24 | 28 28 | 40 40 | 44 44 |
0x00003f0     26 26 | 30 30 | 42 42 | 46 46 | 16 16 | 18 18 | 20 20 | 22 22 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence
with odd channels used):
CLK start:    0x0000010
ULAP:         0x00000000
Used Channels:0x2aaaaaaaaaaaaaaaaaa
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              -----------------------------------------------------------------
0x0000010     09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
0x0000030     25 25 | 27 27 | 29 29 | 31 31 | 33 33 | 35 35 | 37 37 | 39 39 |
0x0000050     41 41 | 43 43 | 45 45 | 47 47 | 49 49 | 51 51 | 53 53 | 55 55 |
0x0000070     57 57 | 59 59 | 61 61 | 63 63 | 33 33 | 37 37 | 35 35 | 39 39 |
0x0000090     41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
0x00000b0     57 57 | 61 61 | 59 59 | 63 63 | 65 65 | 69 69 | 67 67 | 71 71 |
0x00000d0     73 73 | 77 77 | 75 75 | 01 01 | 01 01 | 05 05 | 03 03 | 07 07 |
0x00000f0     09 09 | 13 13 | 11 11 | 15 15 | 65 65 | 67 67 | 69 69 | 71 71 |
0x0000110     01 01 | 03 03 | 05 05 | 07 07 | 73 73 | 75 75 | 77 77 | 01 01 |
0x0000130     09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
0x0000150     33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
0x0000170     41 41 | 43 43 | 45 45 | 47 47 | 17 17 | 21 21 | 19 19 | 23 23 |
0x0000190     33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
0x00001b0     41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
0x00001d0     65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
0x00001f0     73 73 | 77 77 | 75 75 | 03 03 | 49 49 | 51 51 | 57 57 | 59 59 |
0x0000210     53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
0x0000230     69 69 | 71 71 | 77 77 | 03 03 | 05 05 | 07 07 | 13 13 | 15 15 |
0x0000250     09 09 | 11 11 | 17 17 | 19 19 | 21 21 | 23 23 | 29 29 | 31 31 |
0x0000270     25 25 | 27 27 | 33 33 | 35 35 | 05 05 | 09 09 | 13 13 | 17 17 |
0x0000290     07 07 | 11 11 | 15 15 | 19 19 | 21 21 | 25 25 | 29 29 | 33 33 |
0x00002b0     23 23 | 27 27 | 31 31 | 35 35 | 37 37 | 41 41 | 45 45 | 49 49 |
0x00002d0     39 39 | 43 43 | 47 47 | 51 51 | 53 53 | 57 57 | 61 61 | 65 65 |
0x00002f0     55 55 | 59 59 | 63 63 | 67 67 | 37 37 | 39 39 | 53 53 | 55 55 |
0x0000310     41 41 | 43 43 | 57 57 | 59 59 | 45 45 | 47 47 | 61 61 | 63 63 |
```

*Sample Data*

```
0x0000330     49 49 | 51 51 | 65 65 | 67 67 | 69 69 | 71 71 | 03 03 | 05 05 |
0x0000350     73 73 | 75 75 | 07 07 | 09 09 | 77 77 | 01 01 | 11 11 | 13 13 |
0x0000370     03 03 | 01 01 | 15 15 | 17 17 | 69 69 | 73 73 | 03 03 | 07 07 |
0x0000390     71 71 | 75 75 | 05 05 | 09 09 | 77 77 | 03 03 | 11 11 | 15 15 |
0x00003b0     01 01 | 01 01 | 13 13 | 17 17 | 19 19 | 23 23 | 35 35 | 39 39 |
0x00003d0     21 21 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
0x00003f0     29 29 | 33 33 | 45 45 | 49 49 | 19 19 | 21 21 | 23 23 | 25 25 |
```

## 2.2 Second set

```
Hop sequence {k} for Page Scan and Inquiry Scan substates:
CLKN start:   0x0000000
ULAP:         0x2a96ef25
#ticks:       0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
              -------------------------------------------------------
0x0000000:      49 |   13 |   17 |   51 |   55 |   19 |   23 |   53 |
0x0008000:      57 |   21 |   25 |   27 |   31 |   74 |   78 |   29 |
0x0010000:      33 |   76 |    1 |   35 |   39 |    3 |    7 |   37 |
0x0018000:      41 |    5 |    9 |   43 |   47 |   11 |   15 |   45 |
0x0020000:      49 |   13 |   17 |   51 |   55 |   19 |   23 |   53 |
0x0028000:      57 |   21 |   25 |   27 |   31 |   74 |   78 |   29 |
0x0030000:      33 |   76 |    1 |   35 |   39 |    3 |    7 |   37 |
0x0038000:      41 |    5 |    9 |   43 |   47 |   11 |   15 |   45 |

Hop sequence {k} for Page and Inquiry substates:
CLKE start:   0x0000000
ULAP:         0x2a96ef25
#ticks:       00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
              -----------------------------------------------------
0x0000000:    41 05 10 04 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0000010:    49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
0x0000020:    41 05 10 04 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0000030:    49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
...
0x0001000:    41 21 10 36 | 25 27 38 63 | 31 74 65 59 | 78 29 61 00 |
0x0001010:    33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
0x0001020:    41 21 10 36 | 25 27 38 63 | 31 74 65 59 | 78 29 61 00 |
0x0001030:    33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
...
0x0002000:    57 21 42 36 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0002010:    49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
0x0002020:    57 21 42 36 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0002030:    49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
```

*Sample Data*

```
...
0x0003000:    41 05 10 04 | 09 27 06 63 | 31 74 65 59 | 78 29 61 00 |
0x0003010:    33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
0x0003020:    41 05 10 04 | 09 27 06 63 | 31 74 65 59 | 78 29 61 00 |
0x0003030:    33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |

Hop sequence {k} for Peripheral Response substate:
CLKN* =       0x0000010
ULAP:         0x2a96ef25
#ticks:       00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a
              ------------------------------------------------------
0x0000012:    34 | 13 28 | 17 30 | 51 24 | 55 26 | 19 20 | 23 22 | 53
0x0000032:    42 | 21 36 | 25 38 | 27 63 | 31 65 | 74 59 | 78 61 | 29
0x0000052:    02 | 76 75 | 01 77 | 35 71 | 39 73 | 03 67 | 07 69 | 37
0x0000072:    10 | 05 04 | 09 06 | 43 16 | 47 18 | 11 12 | 15 14 | 45

Hop sequence {k} for Central Response substate:
Offset value: 24
CLKE* =       0x0000012
ULAP:         0x2a96ef25
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a |
              ------------------------------------------------------
0x0000014:    13 28 | 17 30 | 51 24 | 55 26 | 19 20 | 23 22 | 53 40 |
0x0000034:    21 36 | 25 38 | 27 63 | 31 65 | 74 59 | 78 61 | 29 00 |
0x0000054:    76 75 | 01 77 | 35 71 | 39 73 | 03 67 | 07 69 | 37 08 |
0x0000074:    05 04 | 09 06 | 43 16 | 47 18 | 11 12 | 15 14 | 45 32 |

Hop sequence {k} for Connection state (Basic channel hopping sequence; ie,
non-AFH):
CLK start:    0x0000010
ULAP:         0x2a96ef25
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ---------------------------------------------------------------
0x0000010:    55 26 | 19 20 | 23 22 | 53 40 | 57 42 | 21 36 | 25 38 | 27 63 |
0x0000030:    31 65 | 74 59 | 78 61 | 29 00 | 33 02 | 76 75 | 01 77 | 35 71 |
0x0000050:    39 73 | 03 67 | 07 69 | 37 08 | 41 10 | 05 04 | 09 06 | 43 16 |
0x0000070:    47 18 | 11 12 | 15 14 | 45 32 | 02 66 | 47 60 | 49 64 | 04 54 |
0x0000090:    06 58 | 51 52 | 53 56 | 08 70 | 10 74 | 55 68 | 57 72 | 59 14 |
0x00000b0:    61 18 | 27 12 | 29 16 | 63 30 | 65 34 | 31 28 | 33 32 | 67 22 |
0x00000d0:    69 26 | 35 20 | 37 24 | 71 38 | 73 42 | 39 36 | 41 40 | 75 46 |
0x00000f0:    77 50 | 43 44 | 45 48 | 00 62 | 26 11 | 69 05 | 73 07 | 36 17 |
0x0000110:    40 19 | 04 13 | 08 15 | 38 25 | 42 27 | 06 21 | 10 23 | 12 48 |
0x0000130:    16 50 | 59 44 | 63 46 | 14 56 | 18 58 | 61 52 | 65 54 | 28 64 |
0x0000150:    32 66 | 75 60 | 00 62 | 30 72 | 34 74 | 77 68 | 02 70 | 20 01 |
```

*Sample Data*

```
0x0000170:    24 03 | 67 76 | 71 78 | 22 09 | 58 43 | 24 37 | 26 41 | 68 47 |
0x0000190:    70 51 | 36 45 | 38 49 | 72 55 | 74 59 | 40 53 | 42 57 | 44 78 |
0x00001b0:    46 03 | 12 76 | 14 01 | 48 07 | 50 11 | 16 05 | 18 09 | 60 15 |
0x00001d0:    62 19 | 28 13 | 30 17 | 64 23 | 66 27 | 32 21 | 34 25 | 52 31 |
0x00001f0:    54 35 | 20 29 | 22 33 | 56 39 | 19 04 | 62 63 | 66 00 | 07 73 |
0x0000210:    11 10 | 54 69 | 58 06 | 23 75 | 27 12 | 70 71 | 74 08 | 76 33 |
0x0000230:    01 49 | 44 29 | 48 45 | 13 35 | 17 51 | 60 31 | 64 47 | 05 41 |
0x0000250:    09 57 | 52 37 | 56 53 | 21 43 | 25 59 | 68 39 | 72 55 | 78 65 |
0x0000270:    03 02 | 46 61 | 50 77 | 15 67 | 51 36 | 17 18 | 19 34 | 41 24 |
0x0000290:    43 40 | 09 22 | 11 38 | 57 28 | 59 44 | 25 26 | 27 42 | 29 63 |
0x00002b0:    31 00 | 76 61 | 78 77 | 45 67 | 47 04 | 13 65 | 15 02 | 37 71 |
0x00002d0:    39 08 | 05 69 | 07 06 | 53 75 | 55 12 | 21 73 | 23 10 | 33 16 |
0x00002f0:    35 32 | 01 14 | 03 30 | 49 20 | 75 60 | 39 48 | 43 56 | 00 66 |
0x0000310:    04 74 | 47 62 | 51 70 | 08 68 | 12 76 | 55 64 | 59 72 | 61 18 |
0x0000330:    65 26 | 29 14 | 33 22 | 69 20 | 73 28 | 37 16 | 41 24 | 77 34 |
0x0000350:    02 42 | 45 30 | 49 38 | 06 36 | 10 44 | 53 32 | 57 40 | 63 50 |
0x0000370:    67 58 | 31 46 | 35 54 | 71 52 | 28 13 | 73 03 | 75 11 | 34 17 |
0x0000390:    36 25 | 02 15 | 04 23 | 42 21 | 44 29 | 10 19 | 12 27 | 14 48 |
0x00003b0:    16 56 | 61 46 | 63 54 | 22 52 | 24 60 | 69 50 | 71 58 | 30 64 |
0x00003d0:    32 72 | 77 62 | 00 70 | 38 68 | 40 76 | 06 66 | 08 74 | 18 01 |
0x00003f0:    20 09 | 65 78 | 67 07 | 26 05 | 44 29 | 32 23 | 36 25 | 70 43 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
all channels used; ie, AFH(79)):
CLK start:    0x0000010
ULAP:         0x2a96ef25
Used Channels:0x7fffffffffffffffffff
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     55 55 | 19 19 | 23 23 | 53 53 | 57 57 | 21 21 | 25 25 | 27 27 |
0x0000030     31 31 | 74 74 | 78 78 | 29 29 | 33 33 | 76 76 | 01 01 | 35 35 |
0x0000050     39 39 | 03 03 | 07 07 | 37 37 | 41 41 | 05 05 | 09 09 | 43 43 |
0x0000070     47 47 | 11 11 | 15 15 | 45 45 | 02 02 | 47 47 | 49 49 | 04 04 |
0x0000090     06 06 | 51 51 | 53 53 | 08 08 | 10 10 | 55 55 | 57 57 | 59 59 |
0x00000b0     61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
0x00000d0     69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
0x00000f0     77 77 | 43 43 | 45 45 | 00 00 | 26 26 | 69 69 | 73 73 | 36 36 |
0x0000110     40 40 | 04 04 | 08 08 | 38 38 | 42 42 | 06 06 | 10 10 | 12 12 |
0x0000130     16 16 | 59 59 | 63 63 | 14 14 | 18 18 | 61 61 | 65 65 | 28 28 |
0x0000150     32 32 | 75 75 | 00 00 | 30 30 | 34 34 | 77 77 | 02 02 | 20 20 |
0x0000170     24 24 | 67 67 | 71 71 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
0x0000190     70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
0x00001b0     46 46 | 12 12 | 14 14 | 48 48 | 50 50 | 16 16 | 18 18 | 60 60 |
```

*Sample Data*

```
0x00001d0     62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
0x00001f0     54 54 | 20 20 | 22 22 | 56 56 | 19 19 | 62 62 | 66 66 | 07 07 |
0x0000210     11 11 | 54 54 | 58 58 | 23 23 | 27 27 | 70 70 | 74 74 | 76 76 |
0x0000230     01 01 | 44 44 | 48 48 | 13 13 | 17 17 | 60 60 | 64 64 | 05 05 |
0x0000250     09 09 | 52 52 | 56 56 | 21 21 | 25 25 | 68 68 | 72 72 | 78 78 |
0x0000270     03 03 | 46 46 | 50 50 | 15 15 | 51 51 | 17 17 | 19 19 | 41 41 |
0x0000290     43 43 | 09 09 | 11 11 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
0x00002b0     31 31 | 76 76 | 78 78 | 45 45 | 47 47 | 13 13 | 15 15 | 37 37 |
0x00002d0     39 39 | 05 05 | 07 07 | 53 53 | 55 55 | 21 21 | 23 23 | 33 33 |
0x00002f0     35 35 | 01 01 | 03 03 | 49 49 | 75 75 | 39 39 | 43 43 | 00 00 |
0x0000310     04 04 | 47 47 | 51 51 | 08 08 | 12 12 | 55 55 | 59 59 | 61 61 |
0x0000330     65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
0x0000350     02 02 | 45 45 | 49 49 | 06 06 | 10 10 | 53 53 | 57 57 | 63 63 |
0x0000370     67 67 | 31 31 | 35 35 | 71 71 | 28 28 | 73 73 | 75 75 | 34 34 |
0x0000390     36 36 | 02 02 | 04 04 | 42 42 | 44 44 | 10 10 | 12 12 | 14 14 |
0x00003b0     16 16 | 61 61 | 63 63 | 22 22 | 24 24 | 69 69 | 71 71 | 30 30 |
0x00003d0     32 32 | 77 77 | 00 00 | 38 38 | 40 40 | 06 06 | 08 08 | 18 18 |
0x00003f0     20 20 | 65 65 | 67 67 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
channels 0 to 21 unused):
CLK start:    0x0000010
ULAP:         0x2a96ef25
Used Channels:0x7fffffffffffc00000
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     55 55 | 50 50 | 23 23 | 53 53 | 57 57 | 52 52 | 25 25 | 27 27 |
0x0000030     31 31 | 74 74 | 78 78 | 29 29 | 33 33 | 76 76 | 32 32 | 35 35 |
0x0000050     39 39 | 34 34 | 38 38 | 37 37 | 41 41 | 36 36 | 40 40 | 43 43 |
0x0000070     47 47 | 42 42 | 46 46 | 45 45 | 55 55 | 47 47 | 49 49 | 57 57 |
0x0000090     59 59 | 51 51 | 53 53 | 61 61 | 63 63 | 55 55 | 57 57 | 59 59 |
0x00000b0     61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
0x00000d0     69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
0x00000f0     77 77 | 43 43 | 45 45 | 53 53 | 26 26 | 69 69 | 73 73 | 36 36 |
0x0000110     40 40 | 57 57 | 61 61 | 38 38 | 42 42 | 59 59 | 63 63 | 65 65 |
0x0000130     69 69 | 59 59 | 63 63 | 67 67 | 71 71 | 61 61 | 65 65 | 28 28 |
0x0000150     32 32 | 75 75 | 53 53 | 30 30 | 34 34 | 77 77 | 55 55 | 73 73 |
0x0000170     24 24 | 67 67 | 71 71 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
0x0000190     70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
0x00001b0     46 46 | 65 65 | 67 67 | 48 48 | 50 50 | 69 69 | 71 71 | 60 60 |
0x00001d0     62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
0x00001f0     54 54 | 73 73 | 22 22 | 56 56 | 37 37 | 62 62 | 66 66 | 25 25 |
0x0000210     29 29 | 54 54 | 58 58 | 23 23 | 27 27 | 70 70 | 74 74 | 76 76 |
```

*Sample Data*

```
0x0000230     76 76 | 44 44 | 48 48 | 31 31 | 35 35 | 60 60 | 64 64 | 23 23 |
0x0000250     27 27 | 52 52 | 56 56 | 39 39 | 25 25 | 68 68 | 72 72 | 78 78 |
0x0000270     78 78 | 46 46 | 50 50 | 33 33 | 51 51 | 35 35 | 37 37 | 41 41 |
0x0000290     43 43 | 27 27 | 29 29 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
0x00002b0     31 31 | 76 76 | 78 78 | 45 45 | 47 47 | 31 31 | 33 33 | 37 37 |
0x00002d0     39 39 | 23 23 | 25 25 | 53 53 | 55 55 | 39 39 | 23 23 | 33 33 |
0x00002f0     35 35 | 76 76 | 78 78 | 49 49 | 75 75 | 39 39 | 43 43 | 40 40 |
0x0000310     44 44 | 47 47 | 51 51 | 48 48 | 52 52 | 55 55 | 59 59 | 61 61 |
0x0000330     65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
0x0000350     42 42 | 45 45 | 49 49 | 46 46 | 50 50 | 53 53 | 57 57 | 63 63 |
0x0000370     67 67 | 31 31 | 35 35 | 71 71 | 28 28 | 73 73 | 75 75 | 34 34 |
0x0000390     36 36 | 42 42 | 44 44 | 42 42 | 44 44 | 50 50 | 52 52 | 54 54 |
0x00003b0     56 56 | 61 61 | 63 63 | 22 22 | 24 24 | 69 69 | 71 71 | 30 30 |
0x00003d0     32 32 | 77 77 | 40 40 | 38 38 | 40 40 | 46 46 | 48 48 | 58 58 |
0x00003f0     60 60 | 65 65 | 67 67 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
even channels used):
CLK start:    0x0000010
ULAP:         0x2a96ef25
Used Channels:0x55555555555555555
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     52 52 | 16 16 | 20 20 | 50 50 | 54 54 | 18 18 | 22 22 | 24 24 |
0x0000030     28 28 | 74 74 | 78 78 | 26 26 | 30 30 | 76 76 | 78 78 | 32 32 |
0x0000050     36 36 | 00 00 | 04 04 | 34 34 | 38 38 | 02 02 | 06 06 | 40 40 |
0x0000070     44 44 | 08 08 | 12 12 | 42 42 | 02 02 | 44 44 | 46 46 | 04 04 |
0x0000090     06 06 | 48 48 | 50 50 | 08 08 | 10 10 | 52 52 | 54 54 | 56 56 |
0x00000b0     58 58 | 24 24 | 26 26 | 60 60 | 62 62 | 28 28 | 30 30 | 64 64 |
0x00000d0     66 66 | 32 32 | 34 34 | 68 68 | 70 70 | 36 36 | 38 38 | 72 72 |
0x00000f0     74 74 | 40 40 | 42 42 | 00 00 | 26 26 | 66 66 | 70 70 | 36 36 |
0x0000110     40 40 | 04 04 | 08 08 | 38 38 | 42 42 | 06 06 | 10 10 | 12 12 |
0x0000130     16 16 | 56 56 | 60 60 | 14 14 | 18 18 | 58 58 | 62 62 | 28 28 |
0x0000150     32 32 | 72 72 | 00 00 | 30 30 | 34 34 | 74 74 | 02 02 | 20 20 |
0x0000170     24 24 | 64 64 | 68 68 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
0x0000190     70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
0x00001b0     46 46 | 12 12 | 14 14 | 48 48 | 50 50 | 16 16 | 18 18 | 60 60 |
0x00001d0     62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
0x00001f0     54 54 | 20 20 | 22 22 | 56 56 | 14 14 | 62 62 | 66 66 | 02 02 |
0x0000210     06 06 | 54 54 | 58 58 | 18 18 | 22 22 | 70 70 | 74 74 | 76 76 |
0x0000230     76 76 | 44 44 | 48 48 | 08 08 | 12 12 | 60 60 | 64 64 | 00 00 |
0x0000250     04 04 | 52 52 | 56 56 | 16 16 | 20 20 | 68 68 | 72 72 | 78 78 |
0x0000270     78 78 | 46 46 | 50 50 | 10 10 | 46 46 | 12 12 | 14 14 | 36 36 |
```

*Sample Data*

```
0x0000290     38 38 | 04 04 | 06 06 | 52 52 | 54 54 | 20 20 | 22 22 | 24 24 |
0x00002b0     26 26 | 76 76 | 78 78 | 40 40 | 42 42 | 08 08 | 10 10 | 32 32 |
0x00002d0     34 34 | 00 00 | 02 02 | 48 48 | 50 50 | 16 16 | 18 18 | 28 28 |
0x00002f0     30 30 | 76 76 | 78 78 | 44 44 | 70 70 | 34 34 | 38 38 | 00 00 |
0x0000310     04 04 | 42 42 | 46 46 | 08 08 | 12 12 | 50 50 | 54 54 | 56 56 |
0x0000330     60 60 | 24 24 | 28 28 | 64 64 | 68 68 | 32 32 | 36 36 | 72 72 |
0x0000350     02 02 | 40 40 | 44 44 | 06 06 | 10 10 | 48 48 | 52 52 | 58 58 |
0x0000370     62 62 | 26 26 | 30 30 | 66 66 | 28 28 | 68 68 | 70 70 | 34 34 |
0x0000390     36 36 | 02 02 | 04 04 | 42 42 | 44 44 | 10 10 | 12 12 | 14 14 |
0x00003b0     16 16 | 56 56 | 58 58 | 22 22 | 24 24 | 64 64 | 66 66 | 30 30 |
0x00003d0     32 32 | 72 72 | 00 00 | 38 38 | 40 40 | 06 06 | 08 08 | 18 18 |
0x00003f0     20 20 | 60 60 | 62 62 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
odd channels used):
CLK start:    0x0000010
ULAP:         0x2a96ef25
Used Channels:0x2aaaaaaaaaaaaaaaaaaa
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------
0x0000010     55 55 | 19 19 | 23 23 | 53 53 | 57 57 | 21 21 | 25 25 | 27 27 |
0x0000030     31 31 | 77 77 | 03 03 | 29 29 | 33 33 | 01 01 | 01 01 | 35 35 |
0x0000050     39 39 | 03 03 | 07 07 | 37 37 | 41 41 | 05 05 | 09 09 | 43 43 |
0x0000070     47 47 | 11 11 | 15 15 | 45 45 | 07 07 | 47 47 | 49 49 | 09 09 |
0x0000090     11 11 | 51 51 | 53 53 | 13 13 | 15 15 | 55 55 | 57 57 | 59 59 |
0x00000b0     61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
0x00000d0     69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
0x00000f0     77 77 | 43 43 | 45 45 | 05 05 | 31 31 | 69 69 | 73 73 | 41 41 |
0x0000110     45 45 | 09 09 | 13 13 | 43 43 | 47 47 | 11 11 | 15 15 | 17 17 |
0x0000130     21 21 | 59 59 | 63 63 | 19 19 | 23 23 | 61 61 | 65 65 | 33 33 |
0x0000150     37 37 | 75 75 | 05 05 | 35 35 | 39 39 | 77 77 | 07 07 | 25 25 |
0x0000170     29 29 | 67 67 | 71 71 | 27 27 | 63 63 | 29 29 | 31 31 | 73 73 |
0x0000190     75 75 | 41 41 | 43 43 | 77 77 | 01 01 | 45 45 | 47 47 | 49 49 |
0x00001b0     51 51 | 17 17 | 19 19 | 53 53 | 55 55 | 21 21 | 23 23 | 65 65 |
0x00001d0     67 67 | 33 33 | 35 35 | 69 69 | 71 71 | 37 37 | 39 39 | 57 57 |
0x00001f0     59 59 | 25 25 | 27 27 | 61 61 | 19 19 | 67 67 | 71 71 | 07 07 |
0x0000210     11 11 | 59 59 | 63 63 | 23 23 | 27 27 | 75 75 | 01 01 | 03 03 |
0x0000230     01 01 | 49 49 | 53 53 | 13 13 | 17 17 | 65 65 | 69 69 | 05 05 |
0x0000250     09 09 | 57 57 | 61 61 | 21 21 | 25 25 | 73 73 | 77 77 | 05 05 |
0x0000270     03 03 | 51 51 | 55 55 | 15 15 | 51 51 | 17 17 | 19 19 | 41 41 |
0x0000290     43 43 | 09 09 | 11 11 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
0x00002b0     31 31 | 03 03 | 05 05 | 45 45 | 47 47 | 13 13 | 15 15 | 37 37 |
0x00002d0     39 39 | 05 05 | 07 07 | 53 53 | 55 55 | 21 21 | 23 23 | 33 33 |
```

*Sample Data*

```
0x00002f0     35 35 | 01 01 | 03 03 | 49 49 | 75 75 | 39 39 | 43 43 | 07 07 |
0x0000310     11 11 | 47 47 | 51 51 | 15 15 | 19 19 | 55 55 | 59 59 | 61 61 |
0x0000330     65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
0x0000350     09 09 | 45 45 | 49 49 | 13 13 | 17 17 | 53 53 | 57 57 | 63 63 |
0x0000370     67 67 | 31 31 | 35 35 | 71 71 | 35 35 | 73 73 | 75 75 | 41 41 |
0x0000390     43 43 | 09 09 | 11 11 | 49 49 | 51 51 | 17 17 | 19 19 | 21 21 |
0x00003b0     23 23 | 61 61 | 63 63 | 29 29 | 31 31 | 69 69 | 71 71 | 37 37 |
0x00003d0     39 39 | 77 77 | 07 07 | 45 45 | 47 47 | 13 13 | 15 15 | 25 25 |
0x00003f0     27 27 | 65 65 | 67 67 | 33 33 | 51 51 | 39 39 | 43 43 | 77 77 |
```

## 2.3 Third set

```
Hop sequence {k} for Page Scan and Inquiry Scan substates:
CLKN start:   0x0000000
ULAP:         0x6587cba9
#ticks:       0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
              -------------------------------------------------------
0x0000000:      16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0008000:       8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0010000:      24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0018000:       0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |
0x0020000:      16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0028000:       8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0030000:      24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0038000:       0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |

Hop sequence {k} for Page and Inquiry substates:
CLKE start:   0x0000000
ULAP:         0x6587cba9
#ticks:       00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
              -------------------------------------------------------
0x0000000:    00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000010:    16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
0x0000020:    00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000030:    16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0001000:    00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001010:    24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0001020:    00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001030:    24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
...
0x0002000:    08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002010:    16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
```

*Sample Data*

```
0x0002020:    08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002030:    16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0003000:    00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003010:    24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0003020:    00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003030:    24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |

Hop sequence {k} for Peripheral Response substate:
CLKN* =       0x0000010
ULAP:         0x6587cba9
#ticks:       00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
              ------------------------------------------------------------
0x0000012:    52 | 65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08
0x0000032:    68 | 57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24
0x0000052:    05 | 73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00
0x0000072:    36 | 49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16

Hop sequence {k} for Central Response substate:
Offset value: 24
CLKE* =       0x0000012
ULAP:         0x6587cba9
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------
0x0000014:    65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08 68 |
0x0000034:    57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24 05 |
0x0000054:    73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00 36 |
0x0000074:    49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16 52 |

Hop sequence {k} for Connection state (Basic channel hopping sequence; ie,
non-AFH):
CLK start:    0x0000010
ULAP:         0x6587cba9
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------
0x0000010:    20 60 | 53 62 | 55 66 | 06 64 | 08 68 | 57 70 | 59 74 | 10 72 |
0x0000030:    12 76 | 69 78 | 71 03 | 22 01 | 24 05 | 73 07 | 75 11 | 26 09 |
0x0000050:    28 13 | 45 30 | 47 34 | 77 32 | 00 36 | 49 38 | 51 42 | 02 40 |
0x0000070:    04 44 | 61 46 | 63 50 | 14 48 | 50 05 | 16 07 | 20 09 | 48 11 |
0x0000090:    52 13 | 06 15 | 10 17 | 38 19 | 42 21 | 08 23 | 12 25 | 40 27 |
0x00000b0:    44 29 | 22 31 | 26 33 | 54 35 | 58 37 | 24 39 | 28 41 | 56 43 |
0x00000d0:    60 45 | 77 62 | 02 64 | 30 66 | 34 68 | 00 70 | 04 72 | 32 74 |
0x00000f0:    36 76 | 14 78 | 18 01 | 46 03 | 72 29 | 42 39 | 44 43 | 74 41 |
0x0000110:    76 45 | 46 47 | 48 51 | 78 49 | 01 53 | 50 63 | 52 67 | 03 65 |
```

*Sample Data*

```
0x0000130:    05 69 | 54 55 | 56 59 | 07 57 | 09 61 | 58 71 | 60 75 | 11 73 |
0x0000150:    13 77 | 30 15 | 32 19 | 62 17 | 64 21 | 34 31 | 36 35 | 66 33 |
0x0000170:    68 37 | 38 23 | 40 27 | 70 25 | 27 61 | 72 71 | 76 73 | 25 75 |
0x0000190:    29 77 | 78 00 | 03 02 | 31 04 | 35 06 | 01 16 | 05 18 | 33 20 |
0x00001b0:    37 22 | 07 08 | 11 10 | 39 12 | 43 14 | 09 24 | 13 26 | 41 28 |
0x00001d0:    45 30 | 62 47 | 66 49 | 15 51 | 19 53 | 64 63 | 68 65 | 17 67 |
0x00001f0:    21 69 | 70 55 | 74 57 | 23 59 | 53 22 | 35 12 | 37 28 | 67 14 |
0x0000210:    69 30 | 23 32 | 25 48 | 55 34 | 57 50 | 39 40 | 41 56 | 71 42 |
0x0000230:    73 58 | 27 36 | 29 52 | 59 38 | 61 54 | 43 44 | 45 60 | 75 46 |
0x0000250:    77 62 | 15 00 | 17 16 | 47 02 | 49 18 | 31 08 | 33 24 | 63 10 |
0x0000270:    65 26 | 19 04 | 21 20 | 51 06 | 06 54 | 65 42 | 69 58 | 18 46 |
0x0000290:    22 62 | 55 64 | 59 01 | 08 68 | 12 05 | 71 72 | 75 09 | 24 76 |
0x00002b0:    28 13 | 57 66 | 61 03 | 10 70 | 14 07 | 73 74 | 77 11 | 26 78 |
0x00002d0:    30 15 | 47 32 | 51 48 | 00 36 | 04 52 | 63 40 | 67 56 | 16 44 |
0x00002f0:    20 60 | 49 34 | 53 50 | 02 38 | 38 78 | 12 05 | 14 13 | 44 07 |
0x0000310:    46 15 | 16 17 | 18 25 | 48 19 | 50 27 | 24 33 | 26 41 | 56 35 |
0x0000330:    58 43 | 20 21 | 22 29 | 52 23 | 54 31 | 28 37 | 30 45 | 60 39 |
0x0000350:    62 47 | 00 64 | 02 72 | 32 66 | 34 74 | 08 01 | 10 09 | 40 03 |
0x0000370:    42 11 | 04 68 | 06 76 | 36 70 | 70 31 | 42 35 | 46 43 | 74 39 |
0x0000390:    78 47 | 48 49 | 52 57 | 01 53 | 05 61 | 56 65 | 60 73 | 09 69 |
0x00003b0:    13 77 | 50 51 | 54 59 | 03 55 | 07 63 | 58 67 | 62 75 | 11 71 |
0x00003d0:    15 00 | 32 17 | 36 25 | 64 21 | 68 29 | 40 33 | 44 41 | 72 37 |
0x00003f0:    76 45 | 34 19 | 38 27 | 66 23 | 11 71 | 05 18 | 07 22 | 13 20 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
all channels used; ie, AFH(79)):
CLK start:    0x0000010
ULAP:         0x6587cba9
Used Channels:0x7fffffffffffffffff
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     20 20 | 53 53 | 55 55 | 06 06 | 08 08 | 57 57 | 59 59 | 10 10 |
0x0000030     12 12 | 69 69 | 71 71 | 22 22 | 24 24 | 73 73 | 75 75 | 26 26 |
0x0000050     28 28 | 45 45 | 47 47 | 77 77 | 00 00 | 49 49 | 51 51 | 02 02 |
0x0000070     04 04 | 61 61 | 63 63 | 14 14 | 50 50 | 16 16 | 20 20 | 48 48 |
0x0000090     52 52 | 06 06 | 10 10 | 38 38 | 42 42 | 08 08 | 12 12 | 40 40 |
0x00000b0     44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
0x00000d0     60 60 | 77 77 | 02 02 | 30 30 | 34 34 | 00 00 | 04 04 | 32 32 |
0x00000f0     36 36 | 14 14 | 18 18 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
0x0000110     76 76 | 46 46 | 48 48 | 78 78 | 01 01 | 50 50 | 52 52 | 03 03 |
0x0000130     05 05 | 54 54 | 56 56 | 07 07 | 09 09 | 58 58 | 60 60 | 11 11 |
0x0000150     13 13 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
0x0000170     68 68 | 38 38 | 40 40 | 70 70 | 27 27 | 72 72 | 76 76 | 25 25 |
```

*Sample Data*

```
0x0000190     29 29 | 78 78 | 03 03 | 31 31 | 35 35 | 01 01 | 05 05 | 33 33 |
0x00001b0     37 37 | 07 07 | 11 11 | 39 39 | 43 43 | 09 09 | 13 13 | 41 41 |
0x00001d0     45 45 | 62 62 | 66 66 | 15 15 | 19 19 | 64 64 | 68 68 | 17 17 |
0x00001f0     21 21 | 70 70 | 74 74 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
0x0000210     69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
0x0000230     73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
0x0000250     77 77 | 15 15 | 17 17 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
0x0000270     65 65 | 19 19 | 21 21 | 51 51 | 06 06 | 65 65 | 69 69 | 18 18 |
0x0000290     22 22 | 55 55 | 59 59 | 08 08 | 12 12 | 71 71 | 75 75 | 24 24 |
0x00002b0     28 28 | 57 57 | 61 61 | 10 10 | 14 14 | 73 73 | 77 77 | 26 26 |
0x00002d0     30 30 | 47 47 | 51 51 | 00 00 | 04 04 | 63 63 | 67 67 | 16 16 |
0x00002f0     20 20 | 49 49 | 53 53 | 02 02 | 38 38 | 12 12 | 14 14 | 44 44 |
0x0000310     46 46 | 16 16 | 18 18 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
0x0000330     58 58 | 20 20 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
0x0000350     62 62 | 00 00 | 02 02 | 32 32 | 34 34 | 08 08 | 10 10 | 40 40 |
0x0000370     42 42 | 04 04 | 06 06 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
0x0000390     78 78 | 48 48 | 52 52 | 01 01 | 05 05 | 56 56 | 60 60 | 09 09 |
0x00003b0     13 13 | 50 50 | 54 54 | 03 03 | 07 07 | 58 58 | 62 62 | 11 11 |
0x00003d0     15 15 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
0x00003f0     76 76 | 34 34 | 38 38 | 66 66 | 11 11 | 05 05 | 07 07 | 13 13 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
channels 0 to 21 unused):
CLK start:    0x0000010
ULAP:         0x6587cba9
Used Channels:0x7ffffffffffffc00000
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              ------------------------------------------------------------------
0x0000010     29 29 | 53 53 | 55 55 | 72 72 | 74 74 | 57 57 | 59 59 | 76 76 |
0x0000030     78 78 | 69 69 | 71 71 | 22 22 | 24 24 | 73 73 | 75 75 | 26 26 |
0x0000050     28 28 | 45 45 | 47 47 | 77 77 | 66 66 | 49 49 | 51 51 | 68 68 |
0x0000070     70 70 | 61 61 | 63 63 | 23 23 | 50 50 | 25 25 | 29 29 | 48 48 |
0x0000090     52 52 | 72 72 | 76 76 | 38 38 | 42 42 | 74 74 | 78 78 | 40 40 |
0x00000b0     44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
0x00000d0     60 60 | 77 77 | 68 68 | 30 30 | 34 34 | 66 66 | 70 70 | 32 32 |
0x00000f0     36 36 | 23 23 | 27 27 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
0x0000110     76 76 | 46 46 | 48 48 | 78 78 | 32 32 | 50 50 | 52 52 | 34 34 |
0x0000130     36 36 | 54 54 | 56 56 | 38 38 | 40 40 | 58 58 | 60 60 | 42 42 |
0x0000150     44 44 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
0x0000170     68 68 | 38 38 | 40 40 | 70 70 | 27 27 | 72 72 | 76 76 | 25 25 |
0x0000190     29 29 | 78 78 | 34 34 | 31 31 | 35 35 | 32 32 | 36 36 | 33 33 |
0x00001b0     37 37 | 38 38 | 42 42 | 39 39 | 43 43 | 40 40 | 44 44 | 41 41 |
0x00001d0     45 45 | 62 62 | 66 66 | 46 46 | 50 50 | 64 64 | 68 68 | 48 48 |
```

*Sample Data*

```
0x00001f0     52 52 | 70 70 | 74 74 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
0x0000210     69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
0x0000230     73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
0x0000250     77 77 | 46 46 | 48 48 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
0x0000270     65 65 | 50 50 | 52 52 | 51 51 | 59 59 | 65 65 | 69 69 | 71 71 |
0x0000290     22 22 | 55 55 | 59 59 | 61 61 | 65 65 | 71 71 | 75 75 | 24 24 |
0x00002b0     28 28 | 57 57 | 61 61 | 63 63 | 67 67 | 73 73 | 77 77 | 26 26 |
0x00002d0     30 30 | 47 47 | 51 51 | 53 53 | 57 57 | 63 63 | 67 67 | 69 69 |
0x00002f0     73 73 | 49 49 | 53 53 | 55 55 | 38 38 | 65 65 | 67 67 | 44 44 |
0x0000310     46 46 | 69 69 | 71 71 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
0x0000330     58 58 | 73 73 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
0x0000350     62 62 | 53 53 | 55 55 | 32 32 | 34 34 | 61 61 | 63 63 | 40 40 |
0x0000370     42 42 | 57 57 | 59 59 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
0x0000390     78 78 | 48 48 | 52 52 | 76 76 | 23 23 | 56 56 | 60 60 | 27 27 |
0x00003b0     31 31 | 50 50 | 54 54 | 78 78 | 25 25 | 58 58 | 62 62 | 29 29 |
0x00003d0     33 33 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
0x00003f0     76 76 | 34 34 | 38 38 | 66 66 | 29 29 | 23 23 | 25 25 | 31 31 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
even channels used):
CLK start:    0x0000010
ULAP:         0x6587cba9
Used Channels:0x55555555555555555
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
              -----------------------------------------------------------------
0x0000010     20 20 | 52 52 | 54 54 | 06 06 | 08 08 | 56 56 | 58 58 | 10 10 |
0x0000030     12 12 | 68 68 | 70 70 | 22 22 | 24 24 | 72 72 | 74 74 | 26 26 |
0x0000050     28 28 | 44 44 | 46 46 | 76 76 | 00 00 | 48 48 | 50 50 | 02 02 |
0x0000070     04 04 | 60 60 | 62 62 | 14 14 | 50 50 | 16 16 | 20 20 | 48 48 |
0x0000090     52 52 | 06 06 | 10 10 | 38 38 | 42 42 | 08 08 | 12 12 | 40 40 |
0x00000b0     44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
0x00000d0     60 60 | 76 76 | 02 02 | 30 30 | 34 34 | 00 00 | 04 04 | 32 32 |
0x00000f0     36 36 | 14 14 | 18 18 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
0x0000110     76 76 | 46 46 | 48 48 | 78 78 | 78 78 | 50 50 | 52 52 | 00 00 |
0x0000130     02 02 | 54 54 | 56 56 | 04 04 | 06 06 | 58 58 | 60 60 | 08 08 |
0x0000150     10 10 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
0x0000170     68 68 | 38 38 | 40 40 | 70 70 | 24 24 | 72 72 | 76 76 | 22 22 |
0x0000190     26 26 | 78 78 | 00 00 | 28 28 | 32 32 | 78 78 | 02 02 | 30 30 |
0x00001b0     34 34 | 04 04 | 08 08 | 36 36 | 40 40 | 06 06 | 10 10 | 38 38 |
0x00001d0     42 42 | 62 62 | 66 66 | 12 12 | 16 16 | 64 64 | 68 68 | 14 14 |
0x00001f0     18 18 | 70 70 | 74 74 | 20 20 | 50 50 | 32 32 | 34 34 | 64 64 |
0x0000210     66 66 | 20 20 | 22 22 | 52 52 | 54 54 | 36 36 | 38 38 | 68 68 |
0x0000230     70 70 | 24 24 | 26 26 | 56 56 | 58 58 | 40 40 | 42 42 | 72 72 |
```

*Sample Data*

```
0x0000250     74 74 | 12 12 | 14 14 | 44 44 | 46 46 | 28 28 | 30 30 | 60 60 |
0x0000270     62 62 | 16 16 | 18 18 | 48 48 | 06 06 | 62 62 | 66 66 | 18 18 |
0x0000290     22 22 | 52 52 | 56 56 | 08 08 | 12 12 | 68 68 | 72 72 | 24 24 |
0x00002b0     28 28 | 54 54 | 58 58 | 10 10 | 14 14 | 70 70 | 74 74 | 26 26 |
0x00002d0     30 30 | 44 44 | 48 48 | 00 00 | 04 04 | 60 60 | 64 64 | 16 16 |
0x00002f0     20 20 | 46 46 | 50 50 | 02 02 | 38 38 | 12 12 | 14 14 | 44 44 |
0x0000310     46 46 | 16 16 | 18 18 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
0x0000330     58 58 | 20 20 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
0x0000350     62 62 | 00 00 | 02 02 | 32 32 | 34 34 | 08 08 | 10 10 | 40 40 |
0x0000370     42 42 | 04 04 | 06 06 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
0x0000390     78 78 | 48 48 | 52 52 | 76 76 | 00 00 | 56 56 | 60 60 | 04 04 |
0x00003b0     08 08 | 50 50 | 54 54 | 78 78 | 02 02 | 58 58 | 62 62 | 06 06 |
0x00003d0     10 10 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
0x00003f0     76 76 | 34 34 | 38 38 | 66 66 | 06 06 | 00 00 | 02 02 | 08 08 |

Hop Sequence {k} for Connection state (Adapted channel hopping sequence with
odd channels used):
CLK start:    0x0000010
ULAP:         0x6587cba9
Used Channels:0x2aaaaaaaaaaaaaaaaaaa
#ticks:       00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
             ------------------------------------------------------------------
0x0000010     23 23 | 53 53 | 55 55 | 09 09 | 11 11 | 57 57 | 59 59 | 13 13 |
0x0000030     15 15 | 69 69 | 71 71 | 25 25 | 27 27 | 73 73 | 75 75 | 29 29 |
0x0000050     31 31 | 45 45 | 47 47 | 77 77 | 03 03 | 49 49 | 51 51 | 05 05 |
0x0000070     07 07 | 61 61 | 63 63 | 17 17 | 53 53 | 19 19 | 23 23 | 51 51 |
0x0000090     55 55 | 09 09 | 13 13 | 41 41 | 45 45 | 11 11 | 15 15 | 43 43 |
0x00000b0     47 47 | 25 25 | 29 29 | 57 57 | 61 61 | 27 27 | 31 31 | 59 59 |
0x00000d0     63 63 | 77 77 | 05 05 | 33 33 | 37 37 | 03 03 | 07 07 | 35 35 |
0x00000f0     39 39 | 17 17 | 21 21 | 49 49 | 75 75 | 45 45 | 47 47 | 77 77 |
0x0000110     01 01 | 49 49 | 51 51 | 03 03 | 01 01 | 53 53 | 55 55 | 03 03 |
0x0000130     05 05 | 57 57 | 59 59 | 07 07 | 09 09 | 61 61 | 63 63 | 11 11 |
0x0000150     13 13 | 33 33 | 35 35 | 65 65 | 67 67 | 37 37 | 39 39 | 69 69 |
0x0000170     71 71 | 41 41 | 43 43 | 73 73 | 27 27 | 75 75 | 01 01 | 25 25 |
0x0000190     29 29 | 03 03 | 03 03 | 31 31 | 35 35 | 01 01 | 05 05 | 33 33 |
0x00001b0     37 37 | 07 07 | 11 11 | 39 39 | 43 43 | 09 09 | 13 13 | 41 41 |
0x00001d0     45 45 | 65 65 | 69 69 | 15 15 | 19 19 | 67 67 | 71 71 | 17 17 |
0x00001f0     21 21 | 73 73 | 77 77 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
0x0000210     69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
0x0000230     73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
0x0000250     77 77 | 15 15 | 17 17 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
0x0000270     65 65 | 19 19 | 21 21 | 51 51 | 11 11 | 65 65 | 69 69 | 23 23 |
0x0000290     27 27 | 55 55 | 59 59 | 13 13 | 17 17 | 71 71 | 75 75 | 29 29 |
```

*Sample Data*

```
0x00002b0     33 33 | 57 57 | 61 61 | 15 15 | 19 19 | 73 73 | 77 77 | 31 31 |
0x00002d0     35 35 | 47 47 | 51 51 | 05 05 | 09 09 | 63 63 | 67 67 | 21 21 |
0x00002f0     25 25 | 49 49 | 53 53 | 07 07 | 43 43 | 17 17 | 19 19 | 49 49 |
0x0000310     51 51 | 21 21 | 23 23 | 53 53 | 55 55 | 29 29 | 31 31 | 61 61 |
0x0000330     63 63 | 25 25 | 27 27 | 57 57 | 59 59 | 33 33 | 35 35 | 65 65 |
0x0000350     67 67 | 05 05 | 07 07 | 37 37 | 39 39 | 13 13 | 15 15 | 45 45 |
0x0000370     47 47 | 09 09 | 11 11 | 41 41 | 75 75 | 47 47 | 51 51 | 01 01 |
0x0000390     05 05 | 53 53 | 57 57 | 01 01 | 05 05 | 61 61 | 65 65 | 09 09 |
0x00003b0     13 13 | 55 55 | 59 59 | 03 03 | 07 07 | 63 63 | 67 67 | 11 11 |
0x00003d0     15 15 | 37 37 | 41 41 | 69 69 | 73 73 | 45 45 | 49 49 | 77 77 |
0x00003f0     03 03 | 39 39 | 43 43 | 71 71 | 11 11 | 05 05 | 07 07 | 13 13 |
```

# 3 ACCESS CODE SAMPLE DATA

Different access codes (GIAC, DIACs, others...)

LAP with LSB as rightmost bit.

| | Bit transmit order on air | | |
|---|---|---|---|
| LAP: | Preamble: | Sync word: | Trailer: |
| 000000 | 5 | 7e7041e3 4000000d | 5 |
| ffffff | a | e758b522 7ffffff2 | a |
| 9e8b33 | 5 | 475c58cc 73345e72 | a |
| 9e8b34 | 5 | 28ed3c34 cb345e72 | a |
| 9e8b36 | 5 | 62337b64 1b345e72 | a |
| 9e8b39 | a | c05747b9 e7345e72 | a |
| 9e8b3d | 5 | 7084eab0 2f345e72 | a |
| 9e8b42 | 5 | 64c86d2b 90b45e72 | a |
| 9e8b48 | a | e3c3725e 04b45e72 | a |
| 9e8b4f | a | 8c7216a6 bcb45e72 | a |
| 9e8b57 | a | b2f16c30 fab45e72 | a |
| 9e8b60 | 5 | 57bd3b22 c1b45e72 | a |
| 9e8b6a | a | d0b62457 55b45e72 | a |
| 9e8b75 | a | 81843a39 abb45e72 | a |
| 9e8b81 | 5 | 0ca96681 e0745e72 | a |
| 9e8b8e | a | aecd5a5c 1c745e72 | a |
| 9e8b9c | 5 | 17453fbf ce745e72 | a |
| 9e8bab | a | f20968ad f5745e72 | a |
| 9e8bbb | 5 | 015f4a1e f7745e72 | a |
| 9e8bcc | a | d8c695a0 0cf45e72 | a |
| 9e8bde | 5 | 614ef043 def45e72 | a |
| 9e8bf1 | a | ba81ddc7 a3f45e72 | a |
| 9e8c05 | 5 | 64a7dc4f 680c5e72 | a |
| 9e8c1a | 5 | 3595c221 960c5e72 | a |
| 9e8c30 | a | cb35cc0d 830c5e72 | a |
| 9e8c47 | 5 | 12ac13b3 788c5e72 | a |
| 9e8c5f | 5 | 2c2f6925 3e8c5e72 | a |
| 9e8c78 | 5 | 3a351c84 078c5e72 | a |
| 9e8c92 | 5 | 7396d0f3 124c5e72 | a |
| 9e8cad | 5 | 5b0fdfc4 6d4c5e72 | a |
| 9e8cc9 | a | aea2eb38 e4cc5e72 | a |
| 9e8ce6 | 5 | 756dc6bc 99cc5e72 | a |
| 9e8d04 | 5 | 214cf934 882c5e72 | a |
| 9e8d23 | 5 | 37568c95 b12c5e72 | a |
| 9e8d43 | 5 | 72281560 f0ac5e72 | a |
| 9e8d64 | 5 | 643260c1 c9ac5e72 | a |
| 9e8d86 | a | e044f493 986c5e72 | a |
| 9e8da9 | 5 | 3b8bd917 e56c5e72 | a |
| 9e8dcd | a | ce26edeb 6cec5e72 | a |

*Sample Data*

```
9e8df2 |     a     | e6bfe2dc 13ec5e72 |     a     |
9e8e18 |     a     | 82dcde3d c61c5e72 |     a     |
9e8e3f |     a     | 94c6ab9c ff1c5e72 |     a     |
9e8e67 |     a     | 969059a6 799c5e72 |     a     |
9e8e90 |     a     | c4dfccef 425c5e72 |     a     |
9e8eba |     5     | 3a7fc2c3 575c5e72 |     a     |
9e8ee5 |     5     | 57985401 69dc5e72 |     a     |
9e8f11 |     5     | 0ae2a363 623c5e72 |     a     |
9e8f3e |     a     | d12d8ee7 1f3c5e72 |     a     |
9e8f6c |     5     | 547063a8 0dbc5e72 |     a     |
9e8f9b |     5     | 063ff6e1 367c5e72 |     a     |
9e8fcb |     a     | c9bc5cfe f4fc5e72 |     a     |
9e8ffc |     5     | 2cf00bec cffc5e72 |     a     |
9e902e |     a     | 8ec5052f 5d025e72 |     a     |
9e9061 |     5     | 1074b15e 61825e72 |     a     |
9e9095 |     a     | 9d59ede6 2a425e72 |     a     |
9e90ca |     a     | f0be7b24 14c25e72 |     a     |
9e9100 |     5     | 10e10dd0 c0225e72 |     a     |
9e9137 |     a     | f5ad5ac2 fb225e72 |     a     |
9e916f |     a     | f7fba8f8 7da25e72 |     a     |
9e91a8 |     5     | 2f490e5b c5625e72 |     a     |
9e91e2 |     a     | 94979982 91e25e72 |     a     |
9e921d |     5     | 26cda478 2e125e72 |     a     |
9e9259 |     a     | aacb81dd 26925e72 |     a     |
9e9296 |     a     | bfac7f5b da525e72 |     a     |
9e92d4 |     a     | c9a7b0a7 cad25e72 |     a     |
9e9313 |     a     | c142bdde 32325e72 |     a     |
616cec |     5     | 586a491f 0dcda18d |     5     |
616ceb |     5     | 37db2de7 b5cda18d |     5     |
616ce9 |     5     | 7d056ab7 65cda18d |     5     |
616ce6 |     a     | df61566a 99cda18d |     5     |
616ce2 |     5     | 6fb2fb63 51cda18d |     5     |
616cdd |     5     | 472bf454 2ecda18d |     5     |
616cd7 |     a     | c020eb21 bacda18d |     5     |
616cd0 |     a     | af918fd9 02cda18d |     5     |
616cc8 |     a     | 9112f54f 44cda18d |     5     |
616cbf |     5     | 488b2af1 bf4da18d |     5     |
616cb5 |     a     | cf803584 2b4da18d |     5     |
616caa |     a     | 9eb22bea d54da18d |     5     |
616c9e |     a     | a49cb509 9e4da18d |     5     |
616c91 |     5     | 06f889d4 624da18d |     5     |
616c83 |     a     | bf70ec37 b04da18d |     5     |
616c74 |     a     | ed3f797e 8b8da18d |     5     |
616c64 |     5     | 1e695bcd 898da18d |     5     |
616c53 |     a     | fb250cdf b28da18d |     5     |
616c41 |     5     | 42ad693c 608da18d |     5     |
616c2e |     a     | a5b7cc14 dd0da18d |     5     |
616c1a |     a     | 9f9952f7 960da18d |     5     |
616c05 |     a     | ceab4c99 680da18d |     5     |
616bef |     a     | d403ddde fdf5a18d |     5     |
616bd8 |     5     | 314f8acc c6f5a18d |     5     |
```

*Sample Data*

```
616bc0 |     5     | 0fccf05a 80f5a18d |     5     |
616ba7 |     5     | 25030d57 7975a18d |     5     |
616b8d |     a     | dba3037b 6c75a18d |     5     |
616b72 |     5     | 4439ce17 13b5a18d |     5     |
616b56 |     a     | 8d417247 5ab5a18d |     5     |
616b39 |     5     | 6a5bd76f e735a18d |     5     |
616b1b |     5     | 592e8166 b635a18d |     5     |
616afc |     5     | 28609d46 cfd5a18d |     5     |
616adc |     5     | 51cb8c1f 4ed5a18d |     5     |
616abb |     5     | 7b047112 b755a18d |     5     |
616a99 |     5     | 4871271b e655a18d |     5     |
616a76 |     5     | 24bdc8c4 9b95a18d |     5     |
616a52 |     a     | edc57494 d295a18d |     5     |
616a2d |     a     | f989f30f 6d15a18d |     5     |
616a07 |     5     | 0729fd23 7815a18d |     5     |
6169e0 |     a     | 8bf0ba4f 81e5a18d |     5     |
6169b8 |     a     | 89a64875 0765a18d |     5     |
61698f |     5     | 6cea1f67 3c65a18d |     5     |
616965 |     5     | 2549d310 29a5a18d |     5     |
61693a |     5     | 48ae45d2 1725a18d |     5     |
61690e |     5     | 7280db31 5c25a18d |     5     |
6168e1 |     a     | ce1b9f34 61c5a18d |     5     |
6168b3 |     5     | 4b46727b 7345a18d |     5     |
616884 |     a     | ae0a2569 4845a18d |     5     |
616854 |     a     | ea5fc581 4a85a18d |     5     |
616823 |     5     | 33c61a3f b105a18d |     5     |
6167f1 |     a     | c49fb8c5 63f9a18d |     5     |
6167be |     5     | 5a2e0cb4 5f79a18d |     5     |
61678a |     5     | 60009257 1479a18d |     5     |
616755 |     a     | 86314e62 eab9a18d |     5     |
61671f |     5     | 3defd9bb be39a18d |     5     |
6166e8 |     a     | bff7e728 c5d9a18d |     5     |
6166b0 |     a     | bda11512 4359a18d |     5     |
616677 |     5     | 6513b3b1 fb99a18d |     5     |
61663d |     a     | decd2468 af19a18d |     5     |
616602 |     a     | f6542b5f d019a18d |     5     |
6165c6 |     a     | dc44b49b d8e9a18d |     5     |
616589 |     5     | 42f500ea e469a18d |     5     |
61654b |     a     | bf2885e1 34a9a18d |     5     |
61650c |     a     | ec4c69b5 4c29a18d |     5     |
```

# 4 HEC AND PACKET HEADER SAMPLE DATA

This section contains examples of HECs computed for sample UAP and packet header contents (Data). The resulting 54 bit packet headers are shown in the rightmost column. The UAP, Data and HEC values are in hexadecimal notation, while the header is in octal notation. The header is transmitted from left to right over the air.

| UAP | Data | HEC | Header (octal) |
|---|---|---|---|
| 00 | 123 | e1 | 770007 007070 000777 |
| 47 | 123 | 06 | 770007 007007 700000 |
| 00 | 124 | 32 | 007007 007007 007700 |
| 47 | 124 | d5 | 007007 007070 707077 |
| 00 | 125 | 5a | 707007 007007 077070 |
| 47 | 125 | bd | 707007 007070 777707 |
| 00 | 126 | e2 | 077007 007007 000777 |
| 47 | 126 | 05 | 077007 007070 700000 |
| 00 | 127 | 8a | 777007 007007 070007 |
| 47 | 127 | 6d | 777007 007070 770770 |
| 00 | 11b | 9e | 770770 007007 777007 |
| 47 | 11b | 79 | 770770 007070 077770 |
| 00 | 11c | 4d | 007770 007070 770070 |
| 47 | 11c | aa | 007770 007007 070707 |
| 00 | 11d | 25 | 707770 007070 700700 |
| 47 | 11d | c2 | 707770 007007 000077 |
| 00 | 11e | 9d | 077770 007070 777007 |
| 47 | 11e | 7a | 077770 007007 077770 |
| 00 | 11f | f5 | 777770 007070 707777 |
| 47 | 11f | 12 | 777770 007007 007000 |

# 5 CRC SAMPLE DATA

This section shows the CRC computed for a sample 10 byte payload and a UAP of 0x47.

```
Data:
-----
data[0] = 0x4e
data[1] = 0x01
data[2] = 0x02
data[3] = 0x03
data[4] = 0x04
data[5] = 0x05
data[6] = 0x06
data[7] = 0x07
data[8] = 0x08
data[9] = 0x09

UAP = 0x47

==> CRC = 6d d2

Codeword (hexadecimal notation):
---------------------------------
4e 01 02 03 04 05 06 07 08 09 6d d2


NB: Over the air each byte in the codeword
    is sent with the LSB first.
```

# 6 COMPLETE SAMPLE PACKETS

## 6.1 Example of DH1 packet

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0100 (DH1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED

AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
000000111000
000111000
000111111000000000000000
```

*Sample Data*

```
Payload (incl payload header and CRC):
------------------------------------------------------------
01110100
10000000
01000000
11000000
00100000
10100000
1110110000110110
```

## 6.2 Example of DM1 packet

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0011 (DM1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED

AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
111111000000
000111000
111000000111111111111000
```

*Sample Data*

```
Payload (incl payload header, FEC23, CRC and 6 padded zeros):
-------------------------------------------------------------
0111010010 11001
0000000100 01011
0000110000 11110
0000100000 00111
1010000011 01100
1011000011 00010
0110000000 10001
```

# 7 SECURE SIMPLE PAIRING SAMPLE DATA

This section provides sample data for the Secure Simple Pairing cryptographic functions (f1, f2, f3, g and the ECDH calculations).

## 7.1 Elliptic curve sample data

In each data set, the bytes are ordered from least significant on the right to most significant on the left.

### 7.1.1 P-192 sample data

#### 7.1.1.1 P-192 data set 1

```
Private A:   07915f86918ddc27005df1d6cf0c142b625ed2eff4a518ff
Private B:   1e636ca790b50f68f15d8dbe86244e309211d635de00e16d
Public A(x): 15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
Public A(y): b09d42b81bc5bd009f79e4b59dbbaa857fca856fb9f7ea25
DHKey:       fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
```

#### 7.1.1.2 P-192 data set 2

```
Private A:   52ec1ca6e0ec973c29065c3ca10be80057243002f09bb43e
Private B:   57231203533e9efe18cc622fd0e34c6a29c6e0fa3ab3bc53
Public A(x): 45571f027e0d690795d61560804da5de789a48f94ab4b07e
Public A(y): 0220016e8a6bce74b45ffec1e664aaa0273b7cbd907a8e2b
DHKey:       a20a34b5497332aa7a76ab135cc0c168333be309d463c0c0
```

#### 7.1.1.3 P-192 data set 3

```
Private A:   00a0df08eaf51e6e7be519d67c6749ea3f4517cdd2e9e821
Private B:   2bf5e0d1699d50ca5025e8e2d9b13244b4d322a328be1821
Public A(x): 2ed35b430fa45f9d329186d754eeeb0495f0f653127f613d
Public A(y): 27e08db74e424395052ddae7e3d5a8fecb52a8039b735b73
DHKey:       3b3986ba70790762f282a12a6d3bcae7a2ca01e25b87724e
```

#### 7.1.1.4 P-192 data set 4

```
Private A:   030a4af66e1a4d590a83e0284fca5cdf83292b84f4c71168
Private B:   12448b5c69ecd10c0471060f2bf86345c5e83c03d16bae2c
Public A(x): f24a6899218fa912e7e4a8ba9357cb8182958f9fa42c968c
Public A(y): 7c0b8a9ebe6ea92e968c3a65f9f1a9716fe826ad88c97032
DHKey:       4a78f83fba757c35f94abea43e92effdd2bc700723c61939
```

#### 7.1.1.5 P-192 data set 5

```
Private A:   604df406c649cb460be16244589a40895c0db7367dc11a2f
Private B:   526c2327303cd505b9cf0c012471902bb9e842ce32b0addc
```

*Sample Data*

```
Public A(x): cbe3c629aceb41b73d475a79fbfe8c08cdc80ceec00ee7c9
Public A(y): f9f70f7ae42abda4f33af56f7f6aa383354e453fa1a2bd18
DHKey:       64d4fe35567e6ea0ca31f947e1533a635436d4870ce88c45
```

#### 7.1.1.6 P-192 data set 6

```
Private A:   1a2c582a09852979eb2cee18fb0befb9a55a6d06f6a8fad3
Private B:   243778916920d68df535955bc1a3cccd5811133a8205ae41
Public A(x): eca2d8d30bbef3ba8b7d591fdb98064a6c7b870cdcebe67c
Public A(y): 2e4163a44f3ae26e70dae86f1bf786e1a5db5562a8ed9fee
DHKey:       6433b36a7e9341940e78a63e31b3cf023282f7f1e3bf83bd
```

#### 7.1.1.7 P-192 data set 7

```
Private A:   0f494dd08b493edb07228058a9f30797ff147a5a2adef9b3
Private B:   2da4cd46d9e06e81b1542503f2da89372e927877bcec1be
Public A(x): 9f56a8aa27346d66652a546abacc7d69c17fd66e0853989f
Public A(y): d7234c1464882250df7bbe67e0fa22aae475dc58af0c4210
DHKey:       c67beda9baf3c96a30616bf87a7d0ae704bc969e5cad354b
```

#### 7.1.1.8 P-192 data set 8

```
Private A:   7381d2bc6ddecb65126564cb1af6ca1985d19fb57f0fff16
Private B:   18e276beff75adc3d520badb3806822e1c820f1064447848
Public A(x): 61c7f3c6f9e09f41423dce889de1973d346f2505a5a3b19b
Public A(y): 919972ff4cd6aed8a4821e3adc358b41f7be07ede20137df
DHKey:       6931496eef2fcfb03e0b1eef515dd4e1b0115b8b241b0b84
```

#### 7.1.1.9 P-192 data set 9

```
Private A:   41c7b484ddc37ef6b7952c379f87593789dac6e4f3d8d8e6
Private B:   33e4eaa77f78216e0e99a9b200f81d2ca20dc74ad62d9b78
Public A(x): 9f09c773adb8e7b66b5d986cd15b143341a66d824113c15f
Public A(y): d2000a91738217ab8070a76c5f96c03de317dfab774f4837
DHKey:       a518f3826bb5fa3d5bc37da4217296d5b6af51e5445c6625
```

#### 7.1.1.10 P-192 data set 10

```
Private A:   703cf5ee9c075f7726d0bb36d131c664f5534a6e6305d631
Private B:   757291c620a0e7e9dd13ce09ceb729c0ce1980e64d569b5f
Public A(x): fa2b96d382cf894aeeb0bd985f3891e655a6315cd5060d03
Public A(y): f7e8206d05c7255300cc56c88448158c497f2df596add7a2
DHKey:       12a3343bb453bb5408da42d20c2d0fcc18ff078f56d9c68c
```

### 7.1.2 P-256 sample data

#### 7.1.2.1 P-256 data set 1

```
Private A:   3f49f6d4 a3c55f38 74c9b3e3 d2103f50 4aff607b eb40b799 5899b8a6 cd3c1abd
Private B:   55188b3d 32f6bb9a 900afcfb eed4e72a 59cb9ac2 f19d7cfb 6b4fdd49 f47fc5fd
```

*Sample Data*

```
Public A(x): 20b003d2 f297be2c 5e2c83a7 e9f9a5b9 eff49111 acf4fddb cc030148 0e359de6
Public A(y): dc809c49 652aeb6d 63329abf 5a52155c 766345c2 8fed3024 741c8ed0 1589d28b
Public B(x): 1ea1f0f0 1faf1d96 09592284 f19e4c00 47b58afd 8615a69f 559077b2 2faaa190
Public B(y): 4c55f33e 429dad37 7356703a 9ab85160 472d1130 e28e3676 5f89aff9 15b1214a
DHKey:       ec0234a3 57c8ad05 341010a6 0a397d9b 99796b13 b4f866f1 868d34f3 73bfa698
```

#### 7.1.2.2 P-256 data set 2

```
Private A:   06a51669 3c9aa31a 6084545d 0c5db641 b48572b9 7203ddff b7ac73f7 d0457663
Private B:   529aa067 0d72cd64 97502ed4 73502b03 7e8803b5 c60829a5 a3caa219 505530ba
Public A(x): 2c31a47b 5779809e f44cb5ea af5c3e43 d5f8faad 4a8794cb 987e9b03 745c78dd
Public A(y): 91951218 3898dfbe cd52e240 8e43871f d0211091 17bd3ed4 eaf84377 43715d4f
Public B(x): f465e43f f23d3f1b 9dc7dfc0 4da87581 84dbc966 204796ec cf0d6cf5 e16500cc
Public B(y): 0201d048 bcbbd899 eeefc424 164e33c2 01c2b010 ca6b4d43 a8a155ca d8ecb279
DHKey:       ab85843a 2f6d883f 62e5684b 38e30733 5fe6e194 5ecd1960 4105c6f2 3221eb69
```

## 7.2 Hash functions sample data

In each data set, the bytes are ordered from least significant on the right to most significant on the left.

### 7.2.1 f1()

#### 7.2.1.1 f1() with P-192 inputs

```
Set 1a
U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:      d5cb8454d177733effffb2ec712baeab
Z:      00
output: 1bdc955a9d542ffc9f9e670cdf665010

Set 1b
U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:      d5cb8454d177733effffb2ec712baeab
Z:      80
output: 611325ebcb6e5269b868113306095fa6

Set 1c
U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:      d5cb8454d177733effffb2ec712baeab
Z:      81
output: b68df39fd8a406b06a6c517d3666cf91

Set 2a (swapped U and V inputs compared with set 1)
U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
```

*Sample Data*

```
V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
X:      d5cb8454d177733effffb2ec712baeab
Z:      00
output: f4e1ec4b88f305e81477627b1643a927

Set 2b (swapped U and V inputs compared with set 1)
U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
X:      d5cb8454d177733effffb2ec712baeab
Z:      80
output: ac6aa7cfa96ae99dd3a74225adb068ae

Set 2c (swapped U and V inputs compared with set 1)
U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
X:      d5cb8454d177733effffb2ec712baeab
Z:      81
output: 5ad4721258aa1fa06082edad980d0cc5

Set 3a (U and V set to same value as U in set 1)
U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
X:      d5cb8454d177733effffb2ec712baeab
Z:      00
output: 49125fc1e8cdc615826c15e5d23ede41

Set 3b (U and V set to same value as V in set 1)
U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:      d5cb8454d177733effffb2ec712baeab
Z:      80
output: 159f204c520565175c2b9c523acad2eb

Set 3c (U and V set to same value as V in set 1)
U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:      d5cb8454d177733effffb2ec712baeab
Z:      81
output: 9a162ff9a8235e5b12539ba0ff9179da
```

#### 7.2.1.2 f1() with P-256 inputs

```
Set 1a
U:      20b003d2f297be2c5e2c83a7e9f9a5b9eff49111acf4fddbcc0301480e359de6
V:      55188b3d32f6bb9a900afcfbeed4e72a59cb9ac2f19d7cfb6b4fdd49f47fc5fd
X:      d5cb8454d177733effffb2ec712baeab
Z:      00
output: D301CE92CC7B9E3F51D2924B8B33FACA

Set 1b
U:      20b003d2f297be2c5e2c83a7e9f9a5b9eff49111acf4fddbcc0301480e359de6
```

*Sample Data*

```
V:      55188b3d32f6bb9a900afcfbeed4e72a59cb9ac2f19d7cfb6b4fdd49f47fc5fd
X:      d5cb8454d177733effffb2ec712baeab
Z:      80
output: 7E431112C10DE8A3984C8AC8149FF6EC
```

### 7.2.2 g()

#### 7.2.2.1 g() with P-192 inputs

```
Set 1
U:                 15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
V:                 356b31938421fbbf2fb331c89fd588a69367e9a833f56812
X:                 d5cb8454d177733effffb2ec712baeab
Y:                 a6e8e7cc25a75f6e216583f7ff3dc4cf
output:            52146a1e
output (decimal):  1377069598
6 digits (decimal):069598
```

#### 7.2.2.2 g() with P-256 inputs

```
Set 1
U: 20b003d2f297be2c5e2c83a7e9f9a5b9eff49111acf4fddbcc0301480e359de6
V: 55188b3d32f6bb9a900afcfbeed4e72a59cb9ac2f19d7cfb6b4fdd49f47fc5fd
X: d5cb8454d177733effffb2ec712baeab
Y: a6e8e7cc25a75f6e216583f7ff3dc4cf
output: 000240E0
output (decimal): 147680
6 digits (decimal): 147680
```

### 7.2.3 f2()

#### 7.2.3.1 f2() with P-192 inputs

```
Set 1
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
keyID:  62746c6b
A1:     56123737bfce
A2:     a713702dcfc1
output: c234c1198f3b520186ab92a2f874934e
```

#### 7.2.3.2 f2() with P-256 inputs

```
Set 1
W: ec0234a357c8ad05341010a60a397d9b99796b13b4f866f1868d34f373bfa698
N1: d5cb8454d177733effffb2ec712baeab
N2: a6e8e7cc25a75f6e216583f7ff3dc4cf
keyID: 62746c6b
A1: 56123737bfce
```

*Sample Data*

```
A2: a713702dcfc1
output: 47300bb95c7404129450674b1741104d
```

### 7.2.4 f3()

#### 7.2.4.1 f3() with P-192 inputs

```
Set 1
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000000
A1:     56123737bfce
A2:     a713702dcfc1
output: 5e6a346b8add7ee80e7ec0c2461b1509


Set 2
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000001
A1:     56123737bfce
A2:     a713702dcfc1
output: 7840e5445a13e3ce6e48a2decbe51482


Set 3
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000002
A1:     56123737bfce
A2:     a713702dcfc1
output: da9afb5c6c9dbe0af4722b532520c4b3


Set 4
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000003
A1:     56123737bfce
A2:     a713702dcfc1
output: 2c0f220c50075285852e01bcee4b5f90

Set 5
```

*Sample Data*

```
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000100
A1:     56123737bfce
A2:     a713702dcfc1
output: 0a096af0fa61dce0933987febe95fc7d

Set 6
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000101
A1:     56123737bfce
A2:     a713702dcfc1
output: 49b8d74007888e770e1a49d6810069b9

Set 7
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000102
A1:     56123737bfce
A2:     a713702dcfc1
output: 309cd0327dec2514894a0c88b101a436

Set 8
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000103
A1:     56123737bfce
A2:     a713702dcfc1
output: 4512274ba875b156c2187e2061b90434

Set 9 (same as set 1 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000000
A1:     a713702dcfc1
A2:     56123737bfce
output: 8d56dc59e70855f563b5e85e42d5964e

Set 10 (same as set 2 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
```

*Sample Data*

```
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000001
A1:     a713702dcfc1
A2:     56123737bfce
output: c92fdacbf0ce931e9c4087a9dfb7bc0b

Set 11 (same as set 3 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000002
A1:     a713702dcfc1
A2:     56123737bfce
output: 52ac910200dc34285bbbf2144883c498

Set 12 (same as set 4 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000003
A1:     a713702dcfc1
A2:     56123737bfce
output: c419d677e0d426e6bb36de5fa54c5041

Set 13 (same as set 5 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000100
A1:     a713702dcfc1
A2:     56123737bfce
output: fb0e1f9f7c623c1bf2675fcff1551137

Set 14 (same as set 6 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000101
A1:     a713702dcfc1
A2:     56123737bfce
output: 16c7be68184f1170fbbb2bef5a9c515d

Set 15 (same as set 7 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
```

*Sample Data*

```
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000102
A1:     a713702dcfc1
A2:     56123737bfce
output: 24849f33d3ac05fef9034c18d9adb310

Set 16 (same as set 8 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000103
A1:     a713702dcfc1
A2:     56123737bfce
output: e0f484bb0b071483285903e85094046b

Set 17
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010000
A1:     56123737bfce
A2:     a713702dcfc1
output: 4bf22677415ed90aceb21873c71c1884

Set 18
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010001
A1:     56123737bfce
A2:     a713702dcfc1
output: 0d4b97992eb570efb369cfe45e1681b5

Set 19
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010002
A1:     56123737bfce
A2:     a713702dcfc1
output: 0f0906bbfa75e95c471e97c4211b2741

Set 20
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
```

*Sample Data*

```
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010003
A1:     56123737bfce
A2:     a713702dcfc1
output: 88f1f60ce1ff4bf8aa08a170dd061d4e

Set 21
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010100
A1:     56123737bfce
A2:     a713702dcfc1
output: 940f88f25317c358d9bd2f146778e36b

Set 22
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010101
A1:     56123737bfce
A2:     a713702dcfc1
output: 591396355ac4dc72be05a15e718ec945

Set 23
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010102
A1:     56123737bfce
A2:     a713702dcfc1
output: a3dc055f656abb1d6e11d3f37340189a

Set 24
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010103
A1:     56123737bfce
A2:     a713702dcfc1
output: fd5412a22ba5dd893852608f8ab0c934

Set 25 (same as set 1 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
```

*Sample Data*

```
IOcap:  010000
A1:     a713702dcfc1
A2:     56123737bfce
output: 2a742039c5fd6c6faafce17b619ac56f

Set 26 (same as set 2 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010001
A1:     a713702dcfc1
A2:     56123737bfce
output: a60d89efb52db7905179a6c63b8f212a

Set 27 (same as set 3 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010002
A1:     a713702dcfc1
A2:     56123737bfce
output: cb02f803d755fd936f0a832f4ee9fd4a

Set 28 (same as set 4 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010003
A1:     a713702dcfc1
A2:     56123737bfce
output: 00786c04a24561485aaf22808871b874

Set 29 (same as set 5 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010100
A1:     a713702dcfc1
A2:     56123737bfce
output: 2a58ef2d99281d472a88027423f8215f

Set 30 (same as set 6 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010101
```

*Sample Data*

```
A1:     a713702dcfc1
A2:     56123737bfce
output: ff7ab3752a144232f2cbbcbf979f5517

Set 31 (same as set 7 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010102
A1:     a713702dcfc1
A2:     56123737bfce
output: 9d7fccef23625b1cc684fbf3f8b0e182

Set 32 (same as set 8 with N1 and N2 swapped and A1 and A2 swapped)
W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
N2:     d5cb8454d177733effffb2ec712baeab
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  010103
A1:     a713702dcfc1
A2:     56123737bfce
output: 864f87e26dece4dfdde30ade1e463d4f
```

#### 7.2.4.2 f3() with P-256 inputs

```
Set 1
W:      ec0234a357c8ad05341010a60a397d9b99796b13b4f866f1868d34f373bfa698
N1:     d5cb8454d177733effffb2ec712baeab
N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
R:      12a3343bb453bb5408da42d20c2d0fc8
IOcap:  000000
A1:     56123737bfce
A2:     a713702dcfc1
output: 5634c83c9996a86b53473fe25979ec90
```

### 7.2.5 [This section is no longer used]

### 7.2.6 h4()

```
Set 1a
keyID:  6274646b
A1:     56123737bfce
A2:     a713702dcfc1
input:  6274646b56123737bfcea713702dcfc1
W:      c234c1198f3b520186ab92a2f874934e
output: b089c4e39d7c192c3aba3c2109d24c0dc039e700adf3a263008e65a8b00fb1fa
Device Authentication Key: b089c4e39d7c192c3aba3c2109d24c0d
```

*Sample Data*

### 7.2.7 h5()

```
Set 1a
R1:      d5cb8454d177733effffb2ec712baeab
R2:      a6e8e7cc25a75f6e216583f7ff3dc4cf
input:   d5cb8454d177733effffb2ec712baeaba6e8e7cc25a75f6e216583f7ff3dc4cf
W:       b089c4e39d7c192c3aba3c2109d24c0d
output: 746af87e1eeb1137c683b97d9d421f911f3ddf100403871b362958c458976d65
SRES_C: 746af87e
SRES_P: 1eeb1137
ACO:         c683b97d9d421f91
```

### 7.2.8 h3()

```
Set 1a
keyID:   6274616b
A1:      56123737bfce
A2:      a713702dcfc1
ACO:     c683b97d9d421f91
input:   6274616b56123737bfcea713702dcfc1c683b97d9d421f91
W:       c234c1198f3b520186ab92a2f874934e
output: 677b377f74a5d501121c46492d4cb489e27fe151026ab47f271a47399c8969ff
AES Encryption key: 677b377f74a5d501121c46492d4cb489
```

# 8 WHITENING SEQUENCE SAMPLE DATA

This section shows the output of the whitening sequence generator.

| Whitening Sequence (=D7) | Whitening LFSR D7.....D0 |
|---|---|
| 1 | 1111111 |
| 1 | 1101111 |
| 1 | 1001111 |
| 0 | 0001111 |
| 0 | 0011110 |
| 0 | 0111100 |
| 1 | 1111000 |
| 1 | 1100001 |
| 1 | 1010011 |
| 0 | 0110111 |
| 1 | 1101110 |
| 1 | 1001101 |
| 0 | 0001011 |
| 0 | 0010110 |
| 0 | 0101100 |
| 1 | 1011000 |
| 0 | 0100001 |
| 1 | 1000010 |
| 0 | 0010101 |
| 0 | 0101010 |
| 1 | 1010100 |
| 0 | 0111001 |
| 1 | 1110010 |
| 1 | 1110101 |
| 1 | 1111011 |
| 1 | 1100111 |
| 1 | 1011111 |
| 0 | 0101111 |
| 1 | 1011110 |
| 0 | 0101101 |
| 1 | 1011010 |
| 0 | 0100101 |
| 1 | 1001010 |
| 0 | 0000101 |
| 0 | 0001010 |
| 0 | 0010100 |
| 0 | 0101000 |
| 1 | 1010000 |
| 0 | 0110001 |
| 1 | 1100010 |
| 1 | 1010101 |

*Sample Data*

| | |
|---|---|
| 0 | 0111011 |
| 1 | 1110110 |
| 1 | 1111101 |
| 1 | 1101011 |
| 1 | 1000111 |
| 0 | 0011111 |
| 0 | 0111110 |
| 1 | 1111100 |
| 1 | 1101001 |
| 1 | 1000011 |
| 0 | 0010111 |
| 0 | 0101110 |
| 1 | 1011100 |
| 0 | 0101001 |
| 1 | 1010010 |
| 0 | 0110101 |
| 1 | 1101010 |
| 1 | 1000101 |
| 0 | 0011011 |
| 0 | 0110110 |
| 1 | 1101100 |
| 1 | 1001001 |
| 0 | 0000011 |
| 0 | 0000110 |
| 0 | 0001100 |
| 0 | 0011000 |
| 0 | 0110000 |
| 1 | 1100000 |
| 1 | 1010001 |
| 0 | 0110011 |
| 1 | 1100110 |
| 1 | 1011101 |
| 0 | 0101011 |
| 1 | 1010110 |
| 0 | 0111101 |
| 1 | 1111010 |
| 1 | 1100101 |
| 1 | 1011011 |
| 0 | 0100111 |
| 1 | 1001110 |
| 0 | 0001101 |
| 0 | 0011010 |
| 0 | 0110100 |
| 1 | 1101000 |
| 1 | 1000001 |
| 0 | 0010011 |
| 0 | 0100110 |
| 1 | 1001100 |
| 0 | 0001001 |
| 0 | 0010010 |
| 0 | 0100100 |

*Sample Data*

| | |
|---|---|
| 1 | 1001000 |
| 0 | 0000001 |
| 0 | 0000010 |
| 0 | 0000100 |
| 0 | 0001000 |
| 0 | 0010000 |
| 0 | 0100000 |
| 1 | 1000000 |
| 0 | 0010001 |
| 0 | 0100010 |
| 1 | 1000100 |
| 0 | 0011001 |
| 0 | 0110010 |
| 1 | 1100100 |
| 1 | 1011001 |
| 0 | 0100011 |
| 1 | 1000110 |
| 0 | 0011101 |
| 0 | 0111010 |
| 1 | 1110100 |
| 1 | 1111001 |
| 1 | 1100011 |
| 1 | 1010111 |
| 0 | 0111111 |
| 1 | 1111110 |
| 1 | 1101101 |
| 1 | 1001011 |
| 0 | 0000111 |
| 0 | 0001110 |
| 0 | 0011100 |
| 0 | 0111000 |
| 1 | 1110000 |
| 1 | 1110001 |
| 1 | 1110011 |
| 1 | 1110111 |
| 1 | 1111111 |

# 9 FEC SAMPLE DATA

```
================================================
Rate 2/3 FEC -- (15,10) Shortened Hamming Code
================================================

Data is in hexadecimal notation, the codewords are in binary notation.
The codeword bits are sent from left to right over the air interface.
The space in the codeword indicates the start of parity bits.

Data:   Codeword:

0x001   1000000000 11010
0x002   0100000000 01101
0x004   0010000000 11100
0x008   0001000000 01110
0x010   0000100000 00111
0x020   0000010000 11001
0x040   0000001000 10110
0x080   0000000100 01011
0x100   0000000010 11111
0x200   0000000001 10101
```

# 10 ENCRYPTION KEY SAMPLE DATA

For Section 10.1 to Section 10.5, the hexadecimal sample data is written with the least significant byte at the leftmost position and the most significant byte at the rightmost position. Within each byte, the *least significant bit* (LSB) is at the rightmost position and the *most significant bit* (MSB) is at the leftmost position Thus, a line reading:

aco: 48afcdd4bd40fef76693b113

means aco[0]=0x48, ac[1]=0xAF, ..., aco[11]=0x13. The LSB of aco[11] is 1 and the MSB of aco[11] is 0.

Key [ i]: denotes the ith sub-key in Ar or A'r;

round r: denotes the input to the rth round;

added ->: denotes the input to round 3 in A'r after adding original input (of round 1).

## 10.1 Four tests of E1

```
rand    :00000000000000000000000000000000
address :000000000000
key     :00000000000000000000000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000000
Key [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round  2:78d19f9307d2476a523ec7a8a026042a
Key [ 3]:ecabaac66795580df89af66e66dc053d
Key [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round  3:600265247668dda0e81c07bbb30ed503
Key [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key [ 6]:2a61b8343219fdfb1740e6511d41448f
round  4:d7552ef7cc9dbde568d80c2215bc4277
Key [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round  5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key [ 9]:7dadb2efc287ce75061302904f2e7233
Key [10]:c08dcfa981e2c4272f6c7a9f52e11538
round  6:b46b711ebb3cf69e847a75f0ab884bdd
Key [11]:fc2042c708e409555e8c147660ffdfd7
Key [12]:fa0b21001af9a6b9e89e624cd99150d2
round  7:c585f308ff19404294f06b292e978994
Key [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key [14]:454d54e5253c0c4a8b3fcca7db6baef4
round  8:2665fadb13acf952bf74b4ab12264b9f
Key [15]:2df37c6d9db52674f29353b0f011ed83
```

*Sample Data*

```
Key [16]:b60316733b1e8e70bd861b477e2456f1
Key [17]:884697b1baa3b7100ac537b3c95a28ac
round  1:158ffe43352085e8a5ec7a88e1ff2ba8
Key [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key [ 2]:7595bf57e0632c59f435c16697d4c864
round  2:0b5cc75febcdf7827ca29ec0901b6b5b
Key [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round  3:e4278526c8429211f7f2f0016220aef4
added ->:f1b68365fd6217f952de6a89831fd95c
Key [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key [ 6]:fd2b79282408ddd4ea0aa7511133336f
round  4:d0304ad18337f86040145d27aa5c8153
Key [ 7]:331227756638a41d57b0f7e071ee2a98
Key [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round  5:84db909d213bb0172b8b6aaf71bf1472
Key [ 9]:669291b0752e63f806fce76f10e119c8
Key [10]:ef8bdd46be8ee0277e9b78adef1ec154
round  6:f835f52921e903dfa762f1df5abd7f95
Key [11]:f3902eb06dc409cfd78384624964bf51
Key [12]:7d72702b21f97984a721c99b0498239d
round  7:ae6c0b4bb09f25c6a5d9788a31b605d1
Key [13]:532e60bceaf902c52a06c2c283ecfa32
Key [14]:181715e5192efb2a64129668cf5d9dd4
round  8:744a6235b86cc0b853cc9f74f6b65311
Key [15]:83017c1434342d4290e961578790f451
Key [16]:2603532f365604646ff65803795ccce5
Key [17]:882f7c907b565ea58dae1c928a0dcf41
sres    :056c0fe6
aco     :48afcdd4bd40fef76693b113
----------------------------------------
rand    :bc3f30689647c8d7c5a03ca80a91eceb
address :7ca89b233c2d
key     :159dd9f43fc3d328efba0cd8a861fa57
round  1:bc3f30689647c8d7c5a03ca80a91eceb
Key [ 1]:159dd9f43fc3d328efba0cd8a861fa57
Key [ 2]:326558b3c15551899a97790e65ff669e
round  2:3e950edf197615638cc19c09f8fedc9b
Key [ 3]:62e879b65b9f53bbfbd020c624b1d682
Key [ 4]:73415f30bac8ab61f410adc9442992db
round  3:6a7640791cb536678936c5ecd4ae5a73
Key [ 5]:5093cfa1d31c1c48acd76df030ea3c31
Key [ 6]:0b4acc2b8f1f694fc7bd91f4a70f3009
round  4:fca2c022a577e2ffb2aa007589693ec7
Key [ 7]:2ca43fc817947804ecff148d50d6f6c6
Key [ 8]:3fcd73524b533e00b7f7825bea2040a4
round  5:e97f8ea4ed1a6f4a36ffc179dc6bb563
Key [ 9]:6c67bec76ae8c8cc4d289f69436d3506
Key [10]:95ed95ee8cb97e61d75848464bffb379
round  6:38b07261d7340d028749de1773a415c7
Key [11]:ff566c1fc6b9da9ac502514550f3e9d2
```

*Sample Data*

```
Key [12]:ab5ce3f5c887d0f49b87e0d380e12f47
round  7:58241f1aed7c1c3e047d724331a0b774
Key [13]:a2cab6f95eac7d655dbe84a6cd4c47f5
Key [14]:f5caff88af0af8c42a20b5bbd2c8b460
round  8:3d1aaeff53c0910de63b9788b13c490f
Key [15]:185099c1131cf97001e2f36fda415025
Key [16]:a0ebb82676bc75e8378b189eff3f6b1d
Key [17]:cf5b348aaee27ae332b4f1bfa10289a6
round  1:2e4b417b9a2a9cfd7d8417d9a6a556eb
Key [ 1]:fe78b835f26468ab069fd3991b086fda
Key [ 2]:095c5a51c6fa6d3ac1d57fa19aa382bd
round  2:b8bca81d6bb45af9d92beadd9300f5ed
Key [ 3]:1af866df817fd9f4ec00bc704192cffc
Key [ 4]:f4a8a059c1f575f076f5fbb24bf16590
round  3:351aa16dec2c3a4787080249ed323eae
added ->:1b65e2167656d6bafa8c19904bd79445
Key [ 5]:8c9d18d9356a9954d341b4286e88ea1f
Key [ 6]:5c958d370102c9881bf753e69c7da029
round  4:2ce8fef47dda6a5bee74372e33e478a2
Key [ 7]:7eb2985c3697429fbe0da334bb51f795
Key [ 8]:af900f4b63a1138e2874bfb7c628b7b8
round  5:572787f563e1643c1c862b7555637fb4
Key [ 9]:834c8588dd8f3d4f31117a488420d69b
Key [10]:bc2b9b81c15d9a80262f3f48e9045895
round  6:16b4968c5d02853c3a43aa4cdb5f26ac
Key [11]:f08608c9e39ad3147cba61327919c958
Key [12]:2d4131decf4fa3a959084714a9e85c11
round  7:10e4120c7cccef9dd4ba4e6da8571b01
Key [13]:c934fd319c4a2b5361fa8eef05ae9572
Key [14]:4904c17aa47868e40471007cde3a97c0
round  8:f9081772498fed41b6ffd72b71fcf6c6
Key [15]:ea5e28687e97fa3f833401c86e6053ef
Key [16]:1168f58252c4ecfccafbdb3af857b9f2
Key [17]:b3440f69ef951b78b5cbd6866275301b
sres    :8d5205c5
aco     :3ed75df4abd9af638d144e94
----------------------------------------
rand    :0891caee063f5da1809577ff94ccdcfb
address :c62f19f6ce98
key     :45298d06e46bac21421ddfbed94c032b
round  1:0891caee063f5da1809577ff94ccdcfb
Key [ 1]:45298d06e46bac21421ddfbed94c032b
Key [ 2]:8f03e1e1fe1c191cad35a897bc400597
round  2:1c6ca013480a685c1b28e0317f7167e1
Key [ 3]:4f2ce3a092dde854ef496c8126a69e8e
Key [ 4]:968caee2ac6d7008c07283daec67f2f2
round  3:06b4915f5fcc1fc551a52048f0af8a26
Key [ 5]:ab0d5c31f94259a6bf85ee2d22edf56c
Key [ 6]:dfb74855c0085ce73dc17b84bfd50a92
round  4:077a92b040acc86e6e0a877db197a167
Key [ 7]:8f888952662b3db00d4e904e7ea53b5d
```

*Sample Data*

```
Key [ 8]:5e18bfcc07799b0132db88cd6042f599
round  5:7204881fb300914825fdc863e8ceadf3
Key [ 9]:bfca91ad9bd3d1a06c582b1d5512dddf
Key [10]:a88bc477e3fa1d5a59b5e6cf793c7a41
round  6:27031131d86cea2d747deb4f756143aa
Key [11]:f3cfb8dac8aea2a6a8ef95af3a2a2767
Key [12]:77beb90670c5300b03aa2b2232d3d40c
round  7:fc8c13d49149b1ce8d86f96e44a00065
Key [13]:b578373650af36a06e19fe335d726d32
Key [14]:6bcee918c7d0d24dfdf42237fcf99d53
round  8:04ef5f5a7ddf846cda0a07782fc23866
Key [15]:399f158241eb3e079f45d7b96490e7ea
Key [16]:1bcfbe98ecde2add52aa63ea79fb917a
Key [17]:ee8bc03ec08722bc2b075492873374af
round  1:d989d7a40cde7032d17b52f8117b69d5
Key [ 1]:2ecc6cc797cc41a2ab02007f6af396ae
Key [ 2]:acfaef7609c12567d537ae1cf9dc2198
round  2:8e76eb9a29b2ad5eea790db97aee37c1
Key [ 3]:079c8ff9b73d428df879906a0b87a6c8
Key [ 4]:19f2710baf403a494193d201f3a8c439
round  3:346bb7c35b2539676375aafe3af69089
added ->:edf48e675703a955b2f0fc062b71f95c
Key [ 5]:d623a6498f915cb2c8002765247b2f5a
Key [ 6]:900109093319bc30108b3d9434a77a72
round  4:fafb6c1f3ebbd2477be2da49dd923f69
Key [ 7]:e28e2ee6e72e7f4e5b5c11f10d204228
Key [ 8]:8e455cd11f8b9073a2dfa5413c7a4bc5
round  5:7c72230df588060a3cf920f9b0a08f06
Key [ 9]:28afb26e2c7a64238c41cefc16c53e74
Key [10]:d08dcafc2096395ba0d2dddd0e471f4d
round  6:55991df991db26ff00073a12baa3031d
Key [11]:fcffdcc3ad8faae091a7055b934f87c1
Key [12]:f8df082d77060252c02d91e55bd6a7d6
round  7:70ec682ff864375f63701fa4f6be5377
Key [13]:bef3706e523d708e8a44147d7508bc35
Key [14]:3e98ab283ca2422d56a56cf8b06caeb3
round  8:172f12ec933da85504b4ea5c90f8f0ea
Key [15]:87ad9625d06645d22598dd5ef811ea2c
Key [16]:8bd3db0cc8168009e5da90877e13a36f
Key [17]:0e74631d813a8351ac7039b348c41b42
sres    :00507e5f
aco     :2a5f19fbf60907e69f39ca9f
----------------------------------------
rand    :0ecd61782b4128480c05dc45542b1b8c
address :f428f0e624b3
key     :35949a914225fabad91995d226de1d92
round  1:0ecd61782b4128480c05dc45542b1b8c
Key [ 1]:35949a914225fabad91995d226de1d92
Key [ 2]:ea6b3dcccc8ee5d88de349fa5010404f
round  2:8935e2e263fbc4b9302cabdfc06bce3e
Key [ 3]:920f3a0f2543ce535d4e7f25ad80648a
```

*Sample Data*

```
Key [ 4]:ad47227edf9c6874e80ba80ebb95d2c9
round  3:b4c8b878675f184a0c72f3dab51f8f05
Key [ 5]:81a941ca7202b5e884ae8fa493ecac3d
Key [ 6]:bcde1520bee3660e86ce2f0fb78b9157
round  4:77ced9f2fc42bdd5c6312b87fc2377c5
Key [ 7]:c8eee7423d7c6efa75ecec0d2cd969d3
Key [ 8]:910b3f838a02ed441fbe863a02b4a1d0
round  5:fe28e8056f3004d60bb207e628b39cf2
Key [ 9]:56c647c1e865eb078348962ae070972d
Key [10]:883965da77ca5812d8104e2b640aec0d
round  6:1f2ba92259d9e88101518f145a33840f
Key [11]:61d4cb7e4f8868a283327806a9bd8d4d
Key [12]:9f57de3a3ff310e21dc1e696ce060304
round  7:cc9b5d0218d29037e88475152ebebb2f
Key [13]:7aa1d8adc1aeed7127ef9a18f6eb2d8e
Key [14]:b4db9da3bf865912acd14904c7f7785d
round  8:b04d352bedc02682e4a7f59d7cda1dba
Key [15]:a13d7141ef1f6c7d867e3d175467381b
Key [16]:08b2bc058e50d6141cdd566a307e1acc
Key [17]:057b2b4b4be5dc0ac49e50489b8006c9
round  1:5cfacc773bae995cd7f1b81e7c9ec7df
Key [ 1]:1e717950f5828f3930fe4a9395858815
Key [ 2]:d1623369b733d98bbc894f75866c544c
round  2:d571ffa21d9daa797b1a0a3c962fc64c
Key [ 3]:4abf27664ae364cc8a7e5bcf88214cc4
Key [ 4]:2aaedda8dc4933dd6aeaf6e5c0d5a482
round  3:e17c8e498a00f125bf654c938c23f36d
added ->:bd765a3eb1ae8a796856048df0c1bab2
Key [ 5]:bc7f8ab2d86000f47b1946cc8d7a7a2b
Key [ 6]:6b28544cb13ec6c5d98470df2cf900b7
round  4:a9727c26f2f06bd9920e83c8605dcd76
Key [ 7]:1be840d9107f2c9523f66bb19f5464a1
Key [ 8]:61d6fb1aa2f0c2b26fb2a3d6de8c177c
round  5:aeff751f146eab7e4626b2e2c9e2fb39
Key [ 9]:adabfc82570c568a233173099f23f4c2
Key [10]:b7df6b55ad266c0f1ff7452101f59101
round  6:cf412b95f454d5185e67ca671892e5bd
Key [11]:8e04a7282a2950dcbaea28f300e22de3
Key [12]:21362c114433e29bda3e4d51f803b0cf
round  7:16165722fe4e07ef88f8056b17d89567
Key [13]:710c8fd5bb3cbb5f132a7061de518bd9
Key [14]:0791de7334f4c87285809343f3ead3bd
round  8:28854cd6ad4a3c572b15490d4b81bc3f
Key [15]:4f47f0e5629a674bfcd13770eb3a3bd9
Key [16]:58a6d9a16a284cc0aead2126c79608a1
Key [17]:a564082a0a98399f43f535fd5cefad34
sres    :80e5629c
aco     :a6fe4dcde3924611d3cc6ba1
```

*Sample Data*

## 10.2 Four tests of E21

```
rand    :00000000000000000000000000000000
address :000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000006
Key [ 2]:4697b1baa3b7100ac537b3c95a28dc94
round  2:98611307ab76bbde9a86af1ce8cad412
Key [ 3]:ecabaac66795580df89af66e665d863d
Key [ 4]:8ac3d8896ae9364943bfebd4a2a768a0
round  3:820999ad2e6618f4b578974beeedf9e7
added ->:820999ad2e6618f4b578974beeedf9e7
Key [ 5]:5d57921fd5715cbb22c1bedb1c996394
Key [ 6]:2a61b8343219fdfb1740e9541d41448f
round  4:acd6edec87581ac22dbdc64ea4ced3a2
Key [ 7]:dd0480dee731d67f01ba0f39da6f23ca
Key [ 8]:3ad01cd1303e12a18dcfe0a8af82592c
round  5:1c7798732f09fbfe25795a4a2fbc93c2
Key [ 9]:7dadb2efc287ce7b0c1302904f2e7233
Key [10]:c08dcfa981e2f4572f6c7a9f52e11538
round  6:c05b88b56aa70e9c40c79bb81cd911bd
Key [11]:fc2042c708658a555e8c147660ffdfd7
Key [12]:fa0b21002605a6b9e89e624cd99150d2
round  7:abacc71b481c84c798d1bdf3d62f7e20
Key [13]:18b407e44a5ba4c80ecb48694b4e9c35
Key [14]:454d57e8253c0c4a8b3fcca7db6baef4
round  8:e8204e1183ae85cf19edb2c86215b700
Key [15]:2d0b946d9db52674f29353b0f011ed83
Key [16]:76c316733b1e8e70bd861b477e2456f1
Key [17]:8e4697b1baa3b7100ac537b3c95a28ac
Ka      :d14ca028545ec262cee700e39b5c39ee
----------------------------------------
rand    :2dd9a550343191304013b2d7e1189d09
address :cac4364303b6
round  1:cac4364303b6cac4364303b6cac43643
Key [ 1]:2dd9a550343191304013b2d7e1189d0f
Key [ 2]:14c4335b2c43910c5dcc71d81a14242b
round  2:e169f788aad45a9011f11db5270b1277
Key [ 3]:55bfb712cba168d1a48f6e74cd9f4388
Key [ 4]:2a2b3aacca695caef2821b0fb48cc253
round  3:540f9c76652e92c44987c617035037bf
added ->:9ed3d23566e45c007fcac9a1c9146dfc
Key [ 5]:a06aab22d9a2873840429 76b4b6b00ee
Key [ 6]:c229d054bb72e8eb230e6dcdb32d16b7
round  4:83659a41675f7171ea57909dc5a79ab4
Key [ 7]:23c4812ab1905ddf77dedaed4105649a
Key [ 8]:40d87e272a7a1554ae2e85e3638cdf52
round  5:0b9382d0ed4f2fccdbb69d0db7b130a4
Key [ 9]:bdc064c6a39f6b84fe40db359f62a3c4
Key [10]:58228db841ce3cee983aa721f36aa1b9
round  6:c6ebda0f8f489792f09c189568226c1f
```

*Sample Data*

```
Key [11]:a815bacd6fa747a0d4f52883ac63ebe7
Key [12]:a9ce513b38ea006c333ecaaefcf1d0f8
round  7:75a8aba07e69c9065bcd831c40115116
Key [13]:3635e074792d4122130e5b824e52cd60
Key [14]:511bdb61bb28de72a5d794bffbf407df
round  8:57a6e279dcb764cf7dd6a749dd60c735
Key [15]:a32f5f21044b6744b6d913b13cdb4c0a
Key [16]:9722bbaeef281496ef8c23a9d41e92f4
Key [17]:807370560ad7e8a13a054a65a03b4049
Ka      :e62f8bac609139b3999aedbc9d228042
----------------------------------------
rand    :dab3cffe9d5739d1b7bf4a667ae5ee24
address :02f8fd4cd661
round  1:02f8fd4cd66102f8fd4cd66102f8fd4c
Key [ 1]:dab3cffe9d5739d1b7bf4a667ae5ee22
Key [ 2]:e315a8a65d809ec7c289e69c899fbdcc
round  2:ef85ff081b8709405e19f3e275cec7dc
Key [ 3]:df6a119bb50945fc8a3394e7216448f3
Key [ 4]:87fe86fb0d58b5dd0fb3b6b1dab51d07
round  3:aa25c21bf577d92dd97381e3e9edcc54
added ->:a81dbf5723d8dbd524bf5782ebe5c918
Key [ 5]:36cc253c506c0021c91fac9d8c469e90
Key [ 6]:d5fda00f113e303809b7f7d78a1a2b0e
round  4:9e69ce9b53caec3990894d2baed41e0d
Key [ 7]:c14b5edc10cabf16bc2a2ba4a8ae1e40
Key [ 8]:74c6131afc8dce7e11b03b1ea8610c16
round  5:a5460fa8cedca48a14fd02209e01f02e
Key [ 9]:346cfc553c6cbc9713edb55f4dcbc96c
Key [10]:bddf027cb059d58f0509f8963e9bdec6
round  6:92b33f11eadcacc5a43dd05f13d334dd
Key [11]:8eb9e040c36c4c0b4a7fd3dd354d53c4
Key [12]:c6ffecdd5e135b20879b9dfa4b34bf51
round  7:fb0541aa5e5df1a61c51aef606eb5a41
Key [13]:bf12f5a6ba08dfc4fda4bdfc68c997d9
Key [14]:37c4656b9215f3c959ea688fb64ad327
round  8:f0bbd2b94ae374346730581fc77a9c98
Key [15]:e87bb0d86bf421ea4f779a8eee3a866c
Key [16]:faa471e934fd415ae4c0113ec7f0a5ad
Key [17]:95204a80b8400e49db7cf6fd2fd40d9a
Ka      :b0376d0a9b338c2e133c32b69cb816b3
----------------------------------------
rand    :13ecad08ad63c37f8a54dc56e82f4dc1
address :9846c5ead4d9
round  1:9846c5ead4d99846c5ead4d99846c5ea
Key [ 1]:13ecad08ad63c37f8a54dc56e82f4dc7
Key [ 2]:ad04f127bed50b5e671d6510d392eaed
round  2:97374e18cdd0a6f7a5aa49d1ac875c84
Key [ 3]:57ad159e5774fa222f2f3039b9cd5101
Key [ 4]:9a1e9e1068fede02ef90496e25fd8e79
round  3:9dd3260373edd9d5f4e774826b88fd2d
added ->:0519ebe9a7c6719331d1485bf3cec2c7
```

*Sample Data*

```
Key [ 5]:378dce167db62920b0b392f7cfca316e
Key [ 6]:db4277795c87286faee6c9e9a6b71a93
round  4:40ec6563450299ac4e120d88672504d6
Key [ 7]:ec01aa2f5a8a793b36c1bb858d254380
Key [ 8]:2921a66cfa5bf74ac535424564830e98
round  5:57287bbb041bd6a56c2bd931ed410cd4
Key [ 9]:07018e45aab61b3c3726ee3d57dbd5f6
Key [10]:627381f0fa4c02b0c7d3e7dfbffc3333
round  6:66affa66a8dcd36e36bf6c3f1c6a276e
Key [11]:33b57c925bd5551999f716e138efbe79
Key [12]:a6dc7f9aa95bcc9243aebd12608f657a
round  7:450e65184fd8c72c578d5cdecb286743
Key [13]:a6a6db00fd8c72a28ea57ea542f6e102
Key [14]:dcf3377daeb2e24e61f0ad6620951c1f
round  8:e5eb180b519a4e673f21b7c4f4573f3d
Key [15]:621240b9506b462a7fa250da41844626
Key [16]:ae297810f01f43dc35756cd119ee73d6
Key [17]:b959835ec2501ad3894f8b8f1f4257f9
Ka      :5b61e83ad04d23e9d1c698851fa30447
```

*Sample Data*

## 10.3 Three tests of E22

(for K_temp and overlay generation)

```
rand    :001de169248850245a5f7cc7f0d6d633
PIN     :d5a51083a04a1971f18649ea8b79311a
round  1:001de169248850245a5f7cc7f0d6d623
Key [ 1]:d5a51083a04a1971f18649ea8b79311a
Key [ 2]:7317cdbff57f9b99f9810a2525b17cc7
round  2:5f05c143347b59acae3cb002db23830f
Key [ 3]:f08bd258adf1d4ae4a54d8ccb26220b2
Key [ 4]:91046cbb4ccc43db18d6dd36ca7313eb
round  3:c8f3e3300541a25b6ac5a80c3105f3c4
added ->:c810c45921c9f27f302424cbc1dbc9e7
Key [ 5]:67fb2336f4d9f069da58d11c82f6bd95
Key [ 6]:4fed702c75bd72c0d3d8f38707134c50
round  4:bd5e0c3a97fa55b91a3bbbf306ebb978
Key [ 7]:41c947f80cdc0464c50aa89070af314c
Key [ 8]:680eecfa8daf41c7109c9a5cb1f26d75
round  5:21c1a762c3cc33e75ce8976a73983087
Key [ 9]:6e33fbd94d00ff8f72e8a7a0d2cebc4c
Key [10]:f4d726054c6b948add99fabb5733ddc3
round  6:56d0df484345582f6b574a449ba155eb
Key [11]:4eda2425546a24cac790f49ef2453b53
Key [12]:cf2213624ed1510408a5a3e00b7333df
round  7:120cf9963fe9ff22993f7fdf9600d9b8
Key [13]:d04b1a25b0b8fec946d5ecfa626d04c9
Key [14]:01e5611b0f0e140bdb64585fd3ae5269
round  8:a6337400ad8cb47fefb91332f5cb2713
Key [15]:f15b2dc433f534f61bf718770a3698b1
Key [16]:f990d0273d8ea2b9e0b45917a781c720
Key [17]:f41b3cc13d4301297bb6bdfcb3e5a1dd
Ka      :539e4f2732e5ae2de1e0401f0813bd0d
----------------------------------------
rand    :67ed56bfcf99825f0c6b349369da30ab
PIN     :7885b515e84b1f082cc499976f1725ce
round  1:67ed56bfcf99825f0c6b349369da30bb
Key [ 1]:7885b515e84b1f082cc499976f1725ce
Key [ 2]:72445901fdaf506beb036f4412512248
round  2:6b160b66a1f6c26c1f3432f463ef5aa1
Key [ 3]:59f0e4982e97633e5e7fd133af8f2c5b
Key [ 4]:b4946ec77a41bf7c729d191e33d458ab
round  3:3f22046c964c3e5ca2a26ec9a76a9f67
added ->:580f5ad359e5c003ae0da25ace44cfdc
Key [ 5]:eb0b839f97bdf534183210678520bbef
Key [ 6]:cff0bc4a94e5c8b2a2d24d9f59031e19
round  4:87aa61fc0ff88e744c195249b9a33632
Key [ 7]:592430f14d8f93db95dd691af045776d
Key [ 8]:3b55b404222bf445a6a2ef5865247695
round  5:83dcf592a854226c4dcd94e1ecf1bc75
Key [ 9]:a9714b86319ef343a28b87456416bd52
```

*Sample Data*

```
Key [10]:e6598b24390b3a0bf2982747993b0d78
round  6:dee0d13a52e96bcf7c72045a21609fc6
Key [11]:62051d8c51973073bff959b032c6e1e2
Key [12]:29e94f4ab73296c453c833e217a1a85b
round  7:08488005761e6c7c4dbb203ae453fe3a
Key [13]:0e255970b3e2fc235f59fc5acb10e8ce
Key [14]:d0dfbb3361fee6d4ffe45babf1cd7abf
round  8:0d81e89bddde7a7065316c47574feb8f
Key [15]:c12eee4eb38b7a171f0f736003774b40
Key [16]:8f962523f1c0abd9a087a0dfb11643d3
Key [17]:24be1c66cf8b022f12f1fb4c60c93fd1
Ka      :04435771e03a9daceb8bb1a493ee9bd8
----------------------------------------
rand    :40a94509238664f244ff8e3d13b119d3
PIN     :1ce44839badde30396d03c4c36f23006
round  1:40a94509238664f244ff8e3d13b119c3
Key [ 1]:1ce44839badde30396d03c4c36f23006
Key [ 2]:6dd97a8f91d628be4b18157af1a9dcba
round  2:0eac5288057d9947a24eabc1744c4582
Key [ 3]:fef9583d5f55fd4107ad832a725db744
Key [ 4]:fc3893507016d7c1db2bd034a230a069
round  3:60b424f1082b0cc3bd61be7b4c0155f0
added ->:205d69f82bb17031f9604c465fb26e33
Key [ 5]:0834d04f3e7e1f7f85f0c1db685ab118
Key [ 6]:1852397f9a3723169058e9b62bb3682b
round  4:2c6b65a49d66af6566675afdd6fa7d7d
Key [ 7]:6c10da21d762ae4ac1ba22a96d9007b4
Key [ 8]:9aa23658b90470a78d686344b8a9b0e7
round  5:a2c537899665113a42f1ac24773bdc31
Key [ 9]:137dee3bf879fe7bd02fe6d888e84f16
Key [10]:466e315a1863f47d0f93bc6827cf3450
round  6:e26982980d79b21ed3e20f8c3e71ba96
Key [11]:0b33cf831465bb5c979e6224d7f79f7c
Key [12]:92770660268ede827810d707a0977d73
round  7:e7b063c4e2e3110b89b7e1631c762dd5
Key [13]:7be30ae4961cf24ca17625a77bb7a9f8
Key [14]:be65574a33ae30e6e82dbd2826d3cc1a
round  8:7a963e37b2c2e76b489cfe40a2cf00e5
Key [15]:ed0ba7dd30d60a5e69225f0a33011e5b
Key [16]:765c990f4445e52b39e6ed6105ad1c4f
Key [17]:52627bf9f35d94f30d5b07ef15901adc
Ka      :9cde4b60f9b5861ed9df80858bac6f7f
```

*Sample Data*

## 10.4 Tests of E22 with Pin augmenting

for PIN lengths 1,...,16 bytes

```
rand    :24b101fd56117d42c0545a4247357048
PIN length =16 octets
PIN     :fd397c7f5c1f937cdf82d8816cc377e2
round  1:24b101fd56117d42c0545a4247357058
Key [ 1]:fd397c7f5c1f937cdf82d8816cc377e2
Key [ 2]:0f7aac9c9b53f308d9fdbf2c78e3c30e
round  2:838edfe1226266953ccba8379d873107
Key [ 3]:0b8ac18d4bb44fad2efa115e43945abc
Key [ 4]:887b16b062a83bfa469772c25b456312
round  3:8cd0c9283120aba89a7f9d635dd4fe3f
added ->:a881cad5673128ea5ad3f7211a096e67
Key [ 5]:2248cbe6d299e9d3e8fd35a91178f65b
Key [ 6]:b92af6237385bd31f8fb57fb1bdd824e
round  4:2648d9c618a622b10ef80c4dbaf68b99
Key [ 7]:2bf5ffe84a37878ede2d4c30be60203b
Key [ 8]:c9cb6cec60cb8a8f29b99fcf3e71e40f
round  5:b5a7d9e96f68b14ccebf361de3914d0f
Key [ 9]:5c2f8a702e4a45575b103b0cce8a91c6
Key [10]:d453db0c9f9ddbd11e355d9a34d9b11b
round  6:632a091e7eefe1336857ddafd1ff3265
Key [11]:32805db7e59c5ed4acabf38d27e3fece
Key [12]:fde3a8eedfa3a12be09c1a8a00890fd7
round  7:048531e9fd3efa95910540150f8b137b
Key [13]:def07eb23f3a378f059039a2124bc4c2
Key [14]:2608c58f23d84a09b9ce95e5caac1ab4
round  8:461814ec7439d412d0732f0a6f799a6a
Key [15]:0a7ed16481a623e56ee1442ffa74f334
Key [16]:12add59aca0d19532f1516979954e369
Key [17]:dd43d02d39ffd6a386a4b98b4ac6eb23
Ka      :a5f2adf328e4e6a2b42f19c8b74ba884
----------------------------------------
rand    :321964061ac49a436f9fb9824ac63f8b
PIN length =15 octets
PIN     :ad955d58b6b8857820ac1262d617a6
address :0314c0642543
round  1:321964061ac49a436f9fb9824ac63f9b
Key [ 1]:ad955d58b6b8857820ac1262d617a603
Key [ 2]:f281736f68e3d30b2ac7c67f125dc416
round  2:7c4a4ece1398681f4bafd309328b7770
Key [ 3]:43c157f4c8b360387c32ab330f9c9aa8
Key [ 4]:3a3049945a298f6d076c19219c47c3cb
round  3:9672b00738bdfaf9bd92a855bc6f3afb
added ->:a48b1401228194bad23161d7f6357960
Key [ 5]:c8e2eaa6d73b7de18f3228ab2173bc69
Key [ 6]:8623f44488222e66a293677cf30bf2bb
round  4:9b30247aad3bf133712d034b46d21c68
Key [ 7]:f3e500902fba31db9bae50ef30e484a4
```

*Sample Data*

```
Key [ 8]:49d4b1137c18f4752dd9955a5a8d2f43
round  5:4492c25fda08083a768b4b5588966b23
Key [ 9]:9d59c451989e74785cc097eda7e42ab8
Key [10]:251de25f3917dcd99c18646107a641fb
round  6:21ae346635714d2623041f269978c0ee
Key [11]:80b8f78cb1a49ec0c3e32a238e60fddf
Key [12]:beb84f4d20a501e4a24ecfbde481902b
round  7:9b56a3d0f8932f20c6a77a229514fb00
Key [13]:852571b44f35fd9d9336d3c1d2506656
Key [14]:d0a0d510fb06ba76e69b8ee3ebc1b725
round  8:6cd8492b2fd31a86978bcdf644eb08a8
Key [15]:c7ffd523f32a874ed4a93430a25976de
Key [16]:16cdcb25e62964876d951fdcc07030d3
Key [17]:def32c0e12596f9582e5e3c52b303f52
Ka      :c0ec1a5694e2b48d54297911e6c98b8f
----------------------------------------
rand    :d4ae20c80094547d7051931b5cc2a8d6
PIN length =14 octets
PIN     :e1232e2c5f3b833b3309088a87b6
address :fabecc58e609
round  1:d4ae20c80094547d7051931b5cc2a8c6
Key [ 1]:e1232e2c5f3b833b3309088a87b6fabe
Key [ 2]:5f0812b47cd3e9a30d7707050fffa1f2
round  2:1f45f16be89794bef33e4547c9c0916a
Key [ 3]:77b681944763244ffa3cd71b248b79b5
Key [ 4]:e2814e90e04f485958ce58c9133e2be6
round  3:b10d2f4ac941035263cee3552d774d2f
added ->:65bb4f82c9d5572f131f764e7139f5e9
Key [ 5]:520acad20801dc639a2c6d66d9b79576
Key [ 6]:c72255cdb61d42be72bd45390dd25ba5
round  4:ead4dc34207b6ea721c62166e155aaad
Key [ 7]:ebf04c02075bf459ec9c3ec06627d347
Key [ 8]:a1363dd2812ee800a4491c0c74074493
round  5:f507944f3018e20586d81d7f326aae9d
Key [ 9]:b0b6ba79493dc833d7f425be7b8dadb6
Key [10]:08cd23e536b9b9b53e85eb004cba3111
round  6:fff450f4302a2b3571e8405e148346da
Key [11]:fec22374c6937dcd26171f4d2edfada3
Key [12]:0f1a8ef5979c69ff44f620c2e007b6e4
round  7:de558779589897f3402a90ee78c3f921
Key [13]:901fb66f0779d6aad0c0fba1fe812cb5
Key [14]:a0cab3cd15cd23603adc8d4474efb239
round  8:b2df0aa0c9f07fbbaa02f510a29cf540
Key [15]:18edc3f4296dd6f1dea13f7c143117a1
Key [16]:8d3d52d700a379d72ded81687f7546c7
Key [17]:5927badfe602f29345f840bb53e1dea6
Ka      :d7b39be13e3692c65b4a9e17a9c55e17
----------------------------------------
rand    :272b73a2e40db52a6a61c6520549794a
PIN length =13 octets
PIN     :549f2694f353f5145772d8ae1e
```

*Sample Data*

```
address :20487681eb9f
round  1:272b73a2e40db52a6a61c6520549795a
Key [ 1]:549f2694f353f5145772d8ae1e204876
Key [ 2]:42c855593d66b0c458fd28b95b6a5fbf
round  2:d7276dc8073f7677c31f855bde9501e2
Key [ 3]:75d0a69ae49a2da92e457d767879df52
Key [ 4]:b3aa7e7492971afaa0fb2b64827110df
round  3:71aae503831133d19bc452da4d0e409b
added ->:56d558a1671ee8fbf12518884857b9c1
Key [ 5]:9c8cf1604a98e9a503c342e272de5cf6
Key [ 6]:d35bc2df6b85540a27642106471057d9
round  4:f41a709c89ea80481aa3d2b9b2a9f8ca
Key [ 7]:b454dda74aeb4eff227ba48a58077599
Key [ 8]:bcba6aec050116aa9b7c6a9b7314d796
round  5:20fdda20f4a26b1bd38eb7f355a7be87
Key [ 9]:d41f8a9de0a716eb7167a1b6e321c528
Key [10]:53534499822477782d168ab43f17bc4d8
round  6:a70e316997eeed49a5a9ef9ba5e913b5
Key [11]:32cbc9cf1a81e36a45153972347ce4ac
Key [12]:5747619006cf4ef834c749f2c4b9feb6
round  7:e66f2317a825f589f76b47b6aa6e73fb
Key [13]:f9b68beba0a09d2a570a7dc88cc3c3c2
Key [14]:55718f9a4f0b1f9484e8c6b186a41a4b
round  8:5f68f940440a9798e074776019804ada
Key [15]:4ecc29be1b4d78433f6aa30db974a7fb
Key [16]:8470a066ffb00cda7b08059599f919f5
Key [17]:f39a36d74e960a051e1ca98b777848f4
Ka      :9ac64309a37c25c3b4a584fc002a1618
----------------------------------------
rand    :7edb65f01a2f45a2bc9b24fb3390667e
PIN length =12 octets
PIN     :2e5a42797958557b23447ca8
address :04f0d2737f02
round  1:7edb65f01a2f45a2bc9b24fb3390666e
Key [ 1]:2e5a42797958557b23447ca804f0d273
Key [ 2]:18a97c856561eb23e71af8e9e1be4799
round  2:3436e12db8ffdc1265cb5a86da2fed0b
Key [ 3]:7c0908dcbc73201e17c4f7aa1ab8aec8
Key [ 4]:7cb58833602fbe4194c7cc797ce8c454
round  3:caed6af4226f67e4ad1914620803ef2a
added ->:b4c8cf04389eac4611b438993b935544
Key [ 5]:f4dce7d607b5234562d0ebb2267b08b8
Key [ 6]:560b75c5545751fd8fa99fa4346e654b
round  4:ee67c87d6f74bb75db98f68bff0192c1
Key [ 7]:32f10cefd8d3e6424c6f91f1437808af
Key [ 8]:a934a46045be30fb3be3a5f3f7b18837
round  5:792398dcbeb8d10bdb07ae3c819e943c
Key [ 9]:a0f12e97c677a0e8ac415cd2c8a7ca88
Key [10]:e27014c908785f5ca03e8c6a1da3bf13
round  6:e778b6e0c3e8e7edf90861c7916d97a8
Key [11]:1b4a4303bcc0b2e0f41c72d47654bd9f
```

*Sample Data*

```
Key [12]:4b1302a50046026d6c9054fc8387965a
round  7:1fafddc7efa5f04c1dec1869d3f2d9bb
Key [13]:58c334bb543d49eca562cdbe0280e0fc
Key [14]:bdb60d383c692d06476b76646c8dec48
round  8:3d7c326d074bd6aa222ea050f04a3c7f
Key [15]:78c0162506be0b5953e8403c01028f93
Key [16]:24d7dbbe834dbd7b67f57fcf0d39d60f
Key [17]:2e74f1f3331c0f6585e87b2f715e187e
Ka      :d3af4c81e3f482f062999dee7882a73b
----------------------------------------
rand    :26a92358294dce97b1d79ec32a67e81a
PIN length =11 octets
PIN     :05fbad03f52fa9324f7732
address :b9ac071f9d70
round  1:26a92358294dce97b1d79ec32a67e80a
Key [ 1]:05fbad03f52fa9324f7732b9ac071f9d
Key [ 2]:2504c9691c04a18480c8802e922098c0
round  2:0be20e3d76888e57b6bf77f97a8714fb
Key [ 3]:576b2791d1212bea8408212f2d43e77e
Key [ 4]:90ae36dcce8724adb618f912d1b27297
round  3:1969667060764453257d906b7e58bd5b
added ->:3f12892849c312c494542ea854bfa551
Key [ 5]:bc492c42c9e87f56ec31af5474e9226e
Key [ 6]:c135d1dbed32d9519acfb4169f3e1a10
round  4:ac404205118fe771e54aa6f392da1153
Key [ 7]:83ccbdbbaf17889b7d18254dc9252fa1
Key [ 8]:80b90a1767d3f2848080802764e21711
round  5:41795e89ae9a0cf776ffece76f47fd7a
Key [ 9]:cc24e4a86e8eed129118fd3d5223a1dc
Key [10]:7b1e9c0eb9dab083574be7b7015a62c9
round  6:29ca9e2f87ca00370ef1633505bfba4b
Key [11]:888e6d88cf4beb965cf7d4f32b696baa
Key [12]:6d642f3e5510b0b043a44daa2cf5eec0
round  7:81fc891c3c6fd99acc00028a387e2366
Key [13]:e224f85da2ab63a23e2a3a036e421358
Key [14]:c8dc22aaa739e2cb85d6a0c08226c7d0
round  8:e30b537e7a000e3d2424a9c0f04c4042
Key [15]:a969aa818c6b324bae391bedcdd9d335
Key [16]:6974b6f2f07e4c55f2cc0435c45bebd1
Key [17]:134b925ebd98e6b93c14aee582062fcb
Ka      :be87b44d079d45a08a71d15208c5cb50
----------------------------------------
rand    :0edef05327eab5262430f21fc91ce682
PIN length =10 octets
PIN     :8210e47390f3f48c32b3
address :7a3cdfe377d1
round  1:0edef05327eab5262430f21fc91ce692
Key [ 1]:8210e47390f3f48c32b37a3cdfe377d1
Key [ 2]:c6be4c3e425e749b620a94c779e33a7e
round  2:07ca3c7a7a6bcbc31d79a856d9cffc0e
Key [ 3]:2587cec2a4b8e4f996a9ed664350d5dd
```

*Sample Data*

```
Key [ 4]:70e4bf72834d9d3dbb7eb2c239216dc0
round  3:792ad2ac4e4559d1463714d2f161b6f4
added ->:7708c2ff692f0ef7626706cd387d9c66
Key [ 5]:6696e1e7f8ac037e1fff3598f0c164e2
Key [ 6]:23dbfe4d0b561bea08fbcef25e49b648
round  4:7d8c71a9d7fbdcbd851bdf074550b100
Key [ 7]:b03648acd021550edee904431a02f00c
Key [ 8]:cb169220b7398e8f077730aa4bf06baa
round  5:b6fcaa45064ffd557e4b7b30cfbb83e0
Key [ 9]:af602c2ba16a454649951274c2be6527
Key [10]:5d60b0a7a09d524143eca13ad680bc9c
round  6:b3416d391a0c26c558843debd0601e9e
Key [11]:9a2f39bfe558d9f562c5f09a5c3c0263
Key [12]:72cae8eebd7fabd9b1848333c2aab439
round  7:abe4b498d9c36ea97b8fd27d7f813913
Key [13]:15f27ea11e83a51645d487b81371d7dc
Key [14]:36083c8666447e03d33846edf444eb12
round  8:8032104338a945ba044d102eabda3b22
Key [15]:0a3a8977dd48f3b6c1668578befadd02
Key [16]:f06b6675d78ca0ee5b1761bdcdab516d
Key [17]:cbc8a7952d33aa0496f7ea2d05390b23
Ka      :bf0706d76ec3b11cce724b311bf71ff5
----------------------------------------
rand    :86290e2892f278ff6c3fb917b020576a
PIN length = 9 octets
PIN     :3dcdffcfd086802107
address :791a6a2c5cc3
round  1:86290e2892f278ff6c3fb917b0205765
Key [ 1]:3dcdffcfd086802107791a6a2c5cc33d
Key [ 2]:b4962f40d7bb19429007062a3c469521
round  2:1ec59ffd3065f19991872a7863b0ef02
Key [ 3]:eb9ede6787dd196b7e340185562bf28c
Key [ 4]:2964e58aacf7287d1717a35b100ae23b
round  3:f817406f1423fc2fe33e25152679eaaf
added ->:7e404e47861574d08f7dde02969941ca
Key [ 5]:6abf9a314508fd61e486fa4e376c3f93
Key [ 6]:6da148b7ee2632114521842cbb274376
round  4:e9c2a8fac22b8c7cf0c619e2b3f890ed
Key [ 7]:df889cc34fda86f01096d52d116e620d
Key [ 8]:5eb04b147dc39d1974058761ae7b73fc
round  5:444a8aac0efee1c02f8d38f8274b7b28
Key [ 9]:8426cc59eee391b2bd50cf8f1efef8b3
Key [10]:8b5d220a6300ade418da791dd8151941
round  6:9185f983db150b1bccab1e5c12eb63a1
Key [11]:82ba4ddef833f6a4d18b07aa011f2798
Key [12]:ce63d98794682054e73d0359dad35ec4
round  7:5eded2668f5916dfd036c09e87902886
Key [13]:da794357652e80c70ad8b0715dbe33d6
Key [14]:732ef2c0c3220b31f3820c375e27bb29
round  8:88a5291b4acbba009a85b7dd6a834b3b
Key [15]:3ce75a61d4b465b70c95d7ccd5799633
```

*Sample Data*

```
Key [16]:5df9bd2c3a17a840cdaafb76c171db7c
Key [17]:3f8364b089733d902bccb0cd3386846f
Ka :    cdb0cc68f6f6fbd70b46652de3ef3ffb
----------------------------------------
rand    :3ab52a65bb3b24a08eb6cd284b4b9d4b
PIN length = 8 octets
PIN     :d0fb9b6838d464d8
address :25a868db91ab
round  1:3ab52a65bb3b24a08eb6cd284b4b9d45
Key [ 1]:d0fb9b6838d464d825a868db91abd0fb
Key [ 2]:2573f47b49dad6330a7a9155b7ae8ba1
round  2:ad2ffdff408fcfab44941016a9199251
Key [ 3]:d2c5b8fb80cba13712905a589adaee71
Key [ 4]:5a3381511b338719fae242758dea0997
round  3:2ddc17e570d7931a2b1d13f6ace928a5
added ->:17914180cb12b7baa5d3e0dee734c5e0
Key [ 5]:e0a4d8ac27fbe2783b7bcb3a36a6224d
Key [ 6]:949324c6864deac3eca8e324853e11c3
round  4:62c1db5cf31590d331ec40ad692e8df5
Key [ 7]:6e67148088a01c2d4491957cc9ddc4aa
Key [ 8]:557431deab7087bb4c03fa27228f60c6
round  5:9c8933bc361f4bde4d1bda2b5f8bb235
Key [ 9]:a2551aca53329e70ade3fd2bb7664697
Key [10]:05d0ad35de68a364b54b56e2138738fe
round  6:9156db34136aa06655bf28a05be0596a
Key [11]:1616a6b13ce2f2895c722e8495181520
Key [12]:b12e78a1114847b01f6ed2f5a1429a23
round  7:84dcc292ed836c1c2d523f2a899a2ad5
Key [13]:316e144364686381944e95afd8a026bb
Key [14]:1ab551b88d39d97ea7a9fe136dbfe2e1
round  8:87bdcac878d777877f4eccf042cfee5e
Key [15]:70e21ab08c23c7544524b64492b25cc9
Key [16]:35f730f2ae2b950a49a1bf5c8b9f8866
Key [17]:2f16924c22db8b74e2eadf1ba4ebd37c
Ka      :983218718ca9aa97892e312d86dd9516
----------------------------------------
rand    :a6dc447ff08d4b366ff96e6cf207e179
PIN length = 7 octets
PIN     :9c57e10b4766cc
address :54ebd9328cb6
round  1:a6dc447ff08d4b366ff96e6cf207e174
Key [ 1]:9c57e10b4766cc54ebd9328cb69c57e1
Key [ 2]:00a609f4d61db26993c8177e3ee2bba8
round  2:1ed26b96a306d7014f4e5c9ee523b73d
Key [ 3]:646d7b5f9aaa528384bda3953b542764
Key [ 4]:a051a42212c0e9ad5c2c248259aca14e
round  3:a53f526db18e3d7d53edbfc9711041ed
added ->:031b9612411b884b3ce62da583172299
Key [ 5]:d1bd5e64930e7f838d8a33994462d8b2
Key [ 6]:5dc7e2291e32435665ebd6956bec3414
round  4:9438be308ec83f35c560e2796f4e0559
```

*Sample Data*

```
Key [ 7]:10552f45af63b0f15e2919ab37f64fe7
Key [ 8]:c44d5717c114a58b09207392ebe341f8
round  5:b79a7b14386066d339f799c40479cb3d
Key [ 9]:6886e47b782325568eaf59715a75d8ff
Key [10]:8e1e335e659cd36b132689f78c147bda
round  6:ef232462228aa166438d10c34e17424b
Key [11]:8843efeedd5c2b7c3304d647f932f4d1
Key [12]:13785aaedd0adf67abb4f01872392785
round  7:02d133fe40d15f1073673b36bba35abd
Key [13]:837d7ca2722419e6be3fae35900c3958
Key [14]:93f8442973e7fccf2e7232d1d057c73a
round  8:275506a3d08c84e94cc58ed60054505e
Key [15]:8a7a9edffa3c52918bc6a45f57d91f5d
Key [16]:f214a95d777f763c56109882c4b52c84
Key [17]:10e2ee92c5ea1ddc5eb010e55510c403
Ka      :9cd6650ead86323e87cafb1ff516d1e0
----------------------------------------
rand    :3348470a7ea6cc6eb81b40472133262c
PIN length = 6 octets
PIN     :fcad169d7295
address :430d572f8842
round  1:3348470a7ea6cc6eb81b404721332620
Key [ 1]:fcad169d7295430d572f8842fcad169d
Key [ 2]:b3479d4d4fd178c43e7bc5b0c7d8983c
round  2:af976da9225066d563e10ab955e6fc32
Key [ 3]:7112462b37d82dd81a2a35d9eb43cb7c
Key [ 4]:c5a7030f8497945ac7b84600d1d161fb
round  3:d08f826ebd55a0bd7591c19a89ed9bde
added ->:e3d7c964c3fb6cd3cdac01dda820c1fe
Key [ 5]:84b0c6ef4a63e4dff19b1f546d683df5
Key [ 6]:f4023edfc95d1e79ed4bb4de9b174f5d
round  4:6cd952785630dfc7cf81eea625e42c5c
Key [ 7]:ea38dd9a093ac9355918632c90c79993
Key [ 8]:dbba01e278ddc76380727f5d7135a7de
round  5:93573b2971515495978264b88f330f7f
Key [ 9]:d4dc3a31be34e412210fafa6eca00776
Key [10]:39d1e190ee92b0ff16d92a8be58d2fa0
round  6:b3f01d5e7fe1ce6da7b46d8c389baf47
Key [11]:1eb081328d4bcf94c9117b12c5cf22ac
Key [12]:7e047c2c552f9f1414d946775fabfe30
round  7:0b833bff6106d5bae033b4ce5af5a924
Key [13]:e78e685d9b2a7e29e7f2a19d1bc38ebd
Key [14]:1b582272a3121718c4096d2d8602f215
round  8:23de0bbdc70850a7803f4f10c63b2c0f
Key [15]:8569e860530d9c3d48a0870dac33f676
Key [16]:6966b528fdd1dc222527052c8f6cf5a6
Key [17]:a34244c757154c53171c663b0b56d5c2
Ka      :98f1543ab4d87bd5ef5296fb5e3d3a21
----------------------------------------
rand    :0f5bb150b4371ae4e5785293d22b7b0c
PIN length = 5 octets
```

*Sample Data*

```
PIN     :b10d068bca
address :b44775199f29
round  1:0f5bb150b4371ae4e5785293d22b7b07
Key [ 1]:b10d068bcab44775199f29b10d068bca
Key [ 2]:aec70d1048f1bbd2c18040318a8402ad
round  2:342d2b79d7fb7cd110379742b9842c79
Key [ 3]:6d8d5cf338f29ef4420639ef488e4fa9
Key [ 4]:a1584117541b759ba6d9f7eb2bedcbba
round  3:9407e8e3e810603921bf81cfda62770a
added ->:9b6299b35c477addc437d35c088df20d
Key [ 5]:09a20676666aeed6f22176274eb433f4
Key [ 6]:840472c001add5811a054be5f5c74754
round  4:9a3ba953225a7862c0a842ed3d0b2679
Key [ 7]:fad9e45c8bf70a972fcd9bff0e8751f5
Key [ 8]:e8f30ff666dfd212263416496ff3b2c2
round  5:2c573b6480852e875df34b28a5c44509
Key [ 9]:964cdba0cf8d593f2fc40f96daf8267a
Key [10]:bcd65c11b13e1a70bcd4aafba8864fe3
round  6:21b0cc49e880c5811d24dee0194e6e9e
Key [11]:468c8548ea9653c1a10df6288dd03c1d
Key [12]:5d252d17af4b09d3f4b5f7b5677b8211
round  7:e6d6bdcd63e1d37d9883543ba86392fd
Key [13]:e814bf307c767428c67793dda2df95c7
Key [14]:4812b979fdc20f0ff0996f61673a42cc
round  8:e3dde7ce6bd7d8a34599aa04d6a760ab
Key [15]:5b1e2033d1cd549fc4b028146eb5b3b7
Key [16]:0f284c14fb8fe706a5343e3aa35af7b1
Key [17]:b1f7a4b7456d6b577fded6dc7a672e37
Ka      :c55070b72bc982adb972ed05d1a74ddb
----------------------------------------
rand    :148662a4baa73cfadb55489159e476e1
PIN length = 4 octets
PIN     :fb20f177
address :a683bd0b1896
round  1:148662a4baa73cfadb55489159e476eb
Key [ 1]:fb20f177a683bd0b1896fb20f177a683
Key [ 2]:47266cefbfa468ca7916b458155dc825
round  2:3a942eb6271c3f4e433838a5d3fcbd27
Key [ 3]:688853a6d6575eb2f6a2724b0fbc133b
Key [ 4]:7810df048019634083a2d9219d0b5fe0
round  3:9c835b98a063701c0887943596780769
added ->:8809bd3c1a0aace6d3dcdca4cf5c7d82
Key [ 5]:c78f6dcf56da1bbd413828b33f5865b3
Key [ 6]:eb3f3d407d160df3d293a76d1a513c4a
round  4:7e68c4bafa020a4a59b5a1968105bab5
Key [ 7]:d330e038d6b19d5c9bb0d7285a360064
Key [ 8]:9bd3ee50347c00753d165faced702d9c
round  5:227bad0cf0838bdb15b3b3872c24f592
Key [ 9]:9543ad0fb3fe74f83e0e2281c6d4f5f0
Key [10]:746cd0383c17e0e80e6d095a87fd0290
round  6:e026e98c71121a0cb739ef6f59e14d26
```

*Sample Data*

```
Key [11]:fa28bea4b1c417536608f11f406ea1dd
Key [12]:3aee0f4d21699df9cb8caf5354a780ff
round  7:cd6a6d8137d55140046f8991da1fa40a
Key [13]:372b71bc6d1aa6e785358044fbcf05f4
Key [14]:00a01501224c0405de00aa2ce7b6ab04
round  8:52cd7257fe8d0c782c259bcb6c9f5942
Key [15]:c7015c5c1d7c030e00897f104a006d4a
Key [16]:260a9577790c62e074e71e19fd2894df
Key [17]:c041b7a231493acd15ddcdaee94b9f52
Ka      :7ec864df2f1637c7e81f2319ae8f4671
----------------------------------------
rand    :193a1b84376c88882c8d3b4ee93ba8d5
PIN length = 3 octets
PIN     :a123b9
address :4459a44610f6
round  1:193a1b84376c88882c8d3b4ee93ba8dc
Key [ 1]:a123b94459a44610f6a123b94459a446
Key [ 2]:5f64d384c8e990c1d25080eb244dde9b
round  2:3badbd58f100831d781ddd3ccedefd3f
Key [ 3]:5abc00eff8991575c00807c48f6dbea5
Key [ 4]:127521158ad6798fb6479d1d2268abe6
round  3:0b53075a49c6bf2df2421c655fdedf68
added ->:128d22de7e3247a5decf572bb61987b4
Key [ 5]:f2a1f620448b8e56665608df2ab3952f
Key [ 6]:7c84c0af02aad91dc39209c4edd220b1
round  4:793f4484fb592e7a78756fd4662f990d
Key [ 7]:f6445b647317e7e493bb92bf6655342f
Key [ 8]:3cae503567c63d3595eb140ce60a84c0
round  5:9e46a8df925916a342f299a8306220a0
Key [ 9]:734ed5a806e072bbecb4254993871679
Key [10]:cda69ccb4b07f65e3c8547c11c0647b8
round  6:6bf9cd82c9e1be13fc58eae0b936c75a
Key [11]:c48e531d3175c2bd26fa25cc8990e394
Key [12]:6d93d349a6c6e9ff5b26149565b13d15
round  7:e96a9871471240f198811d4b8311e9a6
Key [13]:5c4951e85875d663526092cd4cbdb667
Key [14]:f19f7758f5cde86c3791efaf563b3fd0
round  8:e94ca67d3721d5fb08ec069191801a46
Key [15]:bf0c17f3299b37d984ac938b769dd394
Key [16]:7edf4ad772a6b9048588f97be25bde1c
Key [17]:6ee7ba6afefc5b561abbd8d6829e8150
Ka      :ac0daabf17732f632e34ef193658bf5d
----------------------------------------
rand    :1453db4d057654e8eb62d7d62ec3608c
PIN length = 2 octets
PIN     :3eaf
address :411fbbb51d1e
round  1:1453db4d057654e8eb62d7d62ec36084
Key [ 1]:3eaf411fbbb51d1e3eaf411fbbb51d1e
Key [ 2]:c3a1a997509f00fb4241aba607109c64
round  2:0b78276c1ebc65707d38c9c5fa1372bd
```

*Sample Data*

```
Key [ 3]:3c729833ae1ce7f84861e4dbad6305cc
Key [ 4]:c83a43c3a66595cb8136560ed29be4ff
round  3:23f3f0f6441563d4c202cee0e5cb2335
added ->:3746cbbb418bb73c2964a536cb8e83b1
Key [ 5]:18b26300b86b70acdd1c8f5cbc7c5da8
Key [ 6]:04efc75309b98cd8f1cef5513c18e41e
round  4:c61afa90d3c14bdf588320e857afdc00
Key [ 7]:517c789cecadc455751af73198749fb8
Key [ 8]:fd9711f913b5c844900fa79dd765d0e2
round  5:a8a0e02ceb556af8bfa321789801183a
Key [ 9]:bb5cf30e7d3ceb930651b1d16ee92750
Key [10]:3d97c7862ecab42720e984972f8efd28
round  6:0b58e922438d224db34b68fca9a5ea12
Key [11]:4ce730344f6b09e449dcdb64cd466666
Key [12]:38828c3a56f922186adcd9b713cdcc31
round  7:b90664c4ac29a8b4bb26debec9ffc5f2
Key [13]:d30fd865ea3e9edcff86a33a2c319649
Key [14]:1fdb63e54413acd968195ab6fa424e83
round  8:6934de3067817cefd811abc5736c163b
Key [15]:a16b7c655bbaa262c807cba8ae166971
Key [16]:7903dd68630105266049e23ca607cda7
Key [17]:888446f2d95e6c2d2803e6f4e815ddc9
Ka      :1674f9dc2063cc2b83d3ef8ba692ebef
----------------------------------------
rand    :1313f7115a9db842fcedc4b10088b48d
PIN length = 1 octets
PIN     :6d
address :008aa9be62d5
round  1:1313f7115a9db842fcedc4b10088b48a
Key [ 1]:6d008aa9be62d56d008aa9be62d56d00
Key [ 2]:46ebfeafb6657b0a1984a8dc0893accf
round  2:839b23b83b5701ab095bafd162ec0ac7
Key [ 3]:8e15595edcf058af62498ee3c1dc6098
Key [ 4]:dd409c3444e94b9cc08396ae967542a0
round  3:c0a2010cc44f2139427f093f4f97ae68
added ->:d3b5f81d9eecd97bbe6ccd8e4f1f62e2
Key [ 5]:487deff5d519f6a6481e947b926f633c
Key [ 6]:5b4b6e3477ed5c2c01f6e607d3418963
round  4:1a5517a0efad3575931d8ea3bee8bd07
Key [ 7]:34b980088d2b5fd6b6a2aceeda99c9c4
Key [ 8]:e7d06d06078acc4ecdbc8da800b73078
round  5:d3ce1fdfe716d72c1075ff37a8a2093f
Key [ 9]:7d375bad245c3b757380021af8ecd408
Key [10]:14dac4bc2f4dc4929a6cceec47f4c3a3
round  6:47e90cb55be6e8dd0f583623c2f2257b
Key [11]:66cfda3c63e464b05e2e7e25f8743ad7
Key [12]:77cfccda1ad380b9fdf1df10846b50e7
round  7:f866ae6624f7abd4a4f5bd24b04b6d43
Key [13]:3e11dd84c031a470a8b66ec6214e44cf
Key [14]:2f03549bdb3c511eea70b65ddbb08253
round  8:02e8e17cf8be4837c9c40706b613dfa8
```

*Sample Data*

```
Key [15]:e2f331229ddfcc6e7bea08b01ab7e70c
Key [16]:b6b0c3738c5365bc77331b98b3fba2ab
Key [17]:f5b3973b636119e577c5c15c87bcfd19
Ka      :38ec0258134ec3f08461ae5c328968a1
```

*Sample Data*

## 10.5 Four tests of E3

```
rand     :00000000000000000000000000000000
aco      :48afcdd4bd40fef76693b113
key      :00000000000000000000000000000000
round   1:00000000000000000000000000000000
Key  [ 1]:00000000000000000000000000000000
Key  [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round   2:78d19f9307d2476a523ec7a8a026042a
Key  [ 3]:ecabaac66795580df89af66e66dc053d
Key  [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round   3:600265247668dda0e81c07bbb30ed503
Key  [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key  [ 6]:2a61b8343219fdfb1740e6511d41448f
round   4:d7552ef7cc9dbde568d80c2215bc4277
Key  [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key  [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round   5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key  [ 9]:7dadb2efc287ce75061302904f2e7233
Key  [10]:c08dcfa981e2c4272f6c7a9f52e11538
round   6:b46b711ebb3cf69e847a75f0ab884bdd
Key  [11]:fc2042c708e409555e8c147660ffdfd7
Key  [12]:fa0b21001af9a6b9e89e624cd99150d2
round   7:c585f308ff19404294f06b292e978994
Key  [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key  [14]:454d54e5253c0c4a8b3fcca7db6baef4
round   8:2665fadb13acf952bf74b4ab12264b9f
Key  [15]:2df37c6d9db52674f29353b0f011ed83
Key  [16]:b60316733b1e8e70bd861b477e2456f1
Key  [17]:884697b1baa3b7100ac537b3c95a28ac
round   1:5d3ecb17f26083df0b7f2b9b29aef87c
Key  [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key  [ 2]:7595bf57e0632c59f435c16697d4c864
round   2:de6fe85c5827233fe22514a16f321bd8
Key  [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key  [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round   3:7cd335b50d09d139ea6702623af85edb
added -> :211100a2ff6954e6e1e62df913a656a7
Key  [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key  [ 6]:fd2b79282408ddd4ea0aa7511133336f
round   4:991dccb3201b5b1c4ceff65a3711e1e9
Key  [ 7]:331227756638a41d57b0f7e071ee2a98
Key  [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round   5:18768c7964818805fe4c6ecae8a38599
Key  [ 9]:669291b0752e63f806fce76f10e119c8
Key  [10]:ef8bdd46be8ee0277e9b78adef1ec154
round   6:82f9aa127a72632af43d1a17e7bd3a09
Key  [11]:f3902eb06dc409cfd78384624964bf51
Key  [12]:7d72702b21f97984a721c99b0498239d
round   7:1543d7870bf2d6d6efab3cbf62dca97d
Key  [13]:532e60bceaf902c52a06c2c283ecfa32
```

*Sample Data*

```
Key  [14]:181715e5192efb2a64129668cf5d9dd4
round   8:eee3e8744a5f8896de95831ed837ffd5
Key  [15]:83017c1434342d4290e961578790f451
Key  [16]:2603532f365604646ff65803795ccce5
Key  [17]:882f7c907b565ea58dae1c928a0dcf41
K_enc    :cc802aecc7312285912e90af6a1e1154
 ----------------------------------------
rand     :950e604e655ea3800fe3eb4a28918087
aco      :68f4f472b5586ac5850f5f74
key      :34e86915d20c485090a6977931f96df5
round   1:950e604e655ea3800fe3eb4a28918087
Key  [ 1]:34e86915d20c485090a6977931f96df5
Key  [ 2]:8de2595003f9928efaf37e5229935bdb
round   2:d46f5a04c967f55840f83d1cdb5f9afc
Key  [ 3]:46f05ec979a97cb6ddf842ecc159c04a
Key  [ 4]:b468f0190a0a83783521deae8178d071
round   3:e16edede9cb6297f32e1203e442ac73a
Key  [ 5]:8a171624dedbd552356094daaadcf12a
Key  [ 6]:3085e07c85e4b99313f6e0c837b5f819
round   4:805144e55e1ece96683d23366fc7d24b
Key  [ 7]:fe45c27845169a66b679b2097d147715
Key  [ 8]:44e2f0c35f64514e8bec66c5dc24b3ad
round   5:edbaf77af070bd22e9304398471042f1
Key  [ 9]:0d534968f3803b6af447eaf964007e7b
Key  [10]:f5499a32504d739ed0b3c547e84157ba
round   6:0dab1a4c846aef0b65b1498812a73b50
Key  [11]:e17e8e456361c46298e6592a6311f3fb
Key  [12]:ec6d14da05d60e8abac807646931711f
round   7:1e7793cac7f55a8ab48bd33bc9c649e0
Key  [13]:2b53dde3d89e325e5ff808ed505706ae
Key  [14]:41034e5c3fb0c0d4f445f0cf23be79b0
round   8:3723768baa78b6a23ade095d995404da
Key  [15]:e2ca373d405a7abf22b494f28a6fd247
Key  [16]:74e09c9068c0e8f1c6902d1b70537c30
Key  [17]:767a7f1acf75c3585a55dd4a428b2119
round   1:39809afb773efd1b7510cd4cb7c49f34
Key  [ 1]:1d0d48d485abddd3798b483a82a0f878
Key  [ 2]:aed957e600a5aed5217984dd5fef6fd8
round   2:6436ddbabe92655c87a7d0c12ae5e5f6
Key  [ 3]:fee00bb0de89b6ef0a289696a4faa884
Key  [ 4]:33ce2f4411db4dd9b7c42cc586b8a2ba
round   3:cec690f7e0aa5f063062301e049a5cc5
added -> :f7462a0c97e85c1d4572fd52b35efbf1
Key  [ 5]:b5116f5c6c29e05e4acb4d02a46a3318
Key  [ 6]:ff4fa1f0f73d1a3c67bc2298abc768f9
round   4:dcdfe942e9f0163fc24a4718844b417d
Key  [ 7]:5453650c0819e001e48331ad0e9076e0
Key  [ 8]:b4ff8dda778e26c0dce08349b81c09a1
round   5:265a16b2f766afae396e7a98c189fda9
Key  [ 9]:f638fa294427c6ed94300fd823b31d10
Key  [10]:1ccfa0bd86a9879b17d4bc457e3e03d6
```

*Sample Data*

```
round   6:628576b5291d53d1eb8611c8624e863e
Key  [11]:0eaee2ef4602ac9ca19e49d74a76d335
Key  [12]:6e1062f10a16e0d378476da3943842e9
round   7:d7b9c2e9b2d5ea5c27019324cae882b3
Key  [13]:40be960bd22c744c5b23024688e554b9
Key  [14]:95c9902cb3c230b44d14ba909730d211
round   8:97fb6065498385e47eb3df6e2ca439dd
Key  [15]:10d4b6e1d1d6798aa00aa2951e32d58d
Key  [16]:c5d4b91444b83ee578004ab8876ba605
Key  [17]:1663a4f98e2862eddd3ec2fb03dcc8a4
K_enc    :c1beafea6e747e304cf0bd7734b0a9e2
 ----------------------------------------
rand     :6a8ebcf5e6e471505be68d5eb8a3200c
aco      :658d791a9554b77c0b2f7b9f
key      :35cf77b333c294671d426fa79993a133
round   1:6a8ebcf5e6e471505be68d5eb8a3200c
Key  [ 1]:35cf77b333c294671d426fa79993a133
Key  [ 2]:c4524e53b95b4bf2d7b2f095f63545fd
round   2:ade94ec585db0d27e17474b58192c87a
Key  [ 3]:c99776768c6e9f9dd3835c52cea8d18a
Key  [ 4]:f1295db23823ba2792f21217fc01d23f
round   3:da8dc1a10241ef9e6e069267cd2c6825
Key  [ 5]:9083db95a6955235bbfad8aeefec5f0b
Key  [ 6]:8bab6bc253d0d0c7e0107feab728ff68
round   4:e6665ca0772ceecbc21222ff7be074f8
Key  [ 7]:2fa1f4e7a4cf3ccd876ec30d194cf196
Key  [ 8]:267364be247184d5337586a19df8bf84
round   5:a857a9326c9ae908f53fee511c5f4242
Key  [ 9]:9aef21965b1a6fa83948d107026134c7
Key  [10]:d2080c751def5dc0d8ea353cebf7b973
round   6:6678748a1b5f21ac05cf1b117a7c342f
Key  [11]:d709a8ab70b0d5a2516900421024b81e
Key  [12]:493e4843805f1058d605c8d1025f8a56
round   7:766c66fe9c460bb2ae39ec01e435f725
Key  [13]:b1ed21b71daea03f49fe74b2c11fc02b
Key  [14]:0e1ded7ebf23c72324a0165a698c65c7
round   8:396e0ff7b2b9b7a3b35c9810882c7596
Key  [15]:b3bf4841dc92f440fde5f024f9ce8be9
Key  [16]:1c69bc6c2994f4c84f72be8f6b188963
Key  [17]:bb7b66286dd679a471e2792270f3bb4d
round   1:45654f2f26549675287200f07cb10ec9
Key  [ 1]:1e2a5672e66529e4f427b0682a3a34b6
Key  [ 2]:974944f1ce0037b1febcf61a2bc961a2
round   2:990cd869c534e76ed4f4af7b3bfbc6c8
Key  [ 3]:8147631fb1ce95d624b480fc7389f6c4
Key  [ 4]:6e90a2db33d284aa13135f3c032aa4f4
round   3:ceb662f875aa6b94e8192b5989abf975
added -> :8b1bb1d753fe01e1c08b2ba9f55c07bc
Key  [ 5]:cbad246d24e36741c46401e6387a05f9
Key  [ 6]:dcf52aaec5713110345a41342c566fc8
round  4 :d4e000be5de78c0f56ff218f3c1df61b
```

*Sample Data*

```
Key  [ 7]:8197537aa9d27e67d17c16b182c8ec65
Key  [ 8]:d66e00e73d835927a307a3ed79d035d8
round   5:9a4603bdef954cfaade2052604bed4e4
Key  [ 9]:71d46257ecc1022bcd312ce6c114d75c
Key  [10]:f91212fa528379651fbd2c32890c5e5f
round   6:09a0fd197ab81eb933eece2fe0132dbb
Key  [11]:283acc551591fadce821b02fb9491814
Key  [12]:ca5f95688788e20d94822f162b5a3920
round   7:494f455a2e7a5db861ece816d4e363e4
Key  [13]:ba574aef663c462d35399efb999d0e40
Key  [14]:6267afc834513783fef1601955fe0628
round   8:37a819f91c8380fb7880e640e99ca947
Key  [15]:fdcd9be5450eef0f8737e6838cd38e2b
Key  [16]:8cfbd9b8056c6a1ce222b92b94319b38
Key  [17]:4f64c1072c891c39eeb95e63318462e0
K_enc    :a3032b4df1cceba8adc1a04427224299
 ----------------------------------------
rand     :5ecd6d75db322c75b6afbd799cb18668
aco      :63f701c7013238bbf88714ee
key      :b9f90c53206792b1826838b435b87d4d
round   1:5ecd6d75db322c75b6afbd799cb18668
Key  [ 1]:b9f90c53206792b1826838b435b87d4d
Key  [ 2]:15f74bbbde4b9d1e08f858721f131669
round   2:72abb85fc80c15ec2b00d72873ef9ad4
Key  [ 3]:ef7fb29f0b01f82706c7439cc52f2dab
Key  [ 4]:3003a6aecdee06b9ac295cce30dcdb93
round  3 :2f10bab93a0f73742183c68f712dfa24
Key  [ 5]:5fcdbb3afdf7df06754c954fc6340254
Key  [ 6]:ddaa90756635579573fe8ca1f93d4a38
round   4:183b145312fd99d5ad08e7ca4a52f04e
Key  [ 7]:27ca8a7fc703aa61f6d7791fc19f704a
Key  [ 8]:702029d8c6e42950762317e730ec5d18
round   5:cbad52d3a026b2e38b9ae6fefffecc32
Key  [ 9]:ff15eaa3f73f4bc2a6ccfb9ca24ed9c5
Key  [10]:034e745246cd2e2cfc3bda39531ca9c5
round   6:ce5f159d0a1acaacd9fb4643272033a7
Key  [11]:0a4d8ff5673731c3dc8fe87e39a34b77
Key  [12]:637592fab43a19ac0044a21afef455a2
round  7 :8a49424a10c0bea5aba52dbbffcbcce8
Key  [13]:6b3fde58f4f6438843cdbe92667622b8
Key  [14]:a10bfa35013812f39bf2157f1c9fca4e
round   8:f5e12da0e93e26a5850251697ec0b917
Key  [15]:2228fe5384e573f48fdd19ba91f1bf57
Key  [16]:5f174db2bc88925c0fbc6b5485bafc08
Key  [17]:28ff90bd0dc31ea2bb479feb7d8fe029
round   1:0c75eed2b54c1cfb9ff522daef94ed4d
Key  [ 1]:a21ceb92d3c027326b4de775865fe8d0
Key  [ 2]:26f64558a9f0a1652f765efd546f3208
round   2:48d537ac209a6aa07b70000016c602e8
Key  [ 3]:e64f9ef630213260f1f79745a0102ae5
Key  [ 4]:af6a59d7cebfd0182dcca9a537c4add8
```

*Sample Data*

```
round   3:8b6d517ac893743a401b3fb7911b64e1
added -> :87e23fa87ddf90c1df10616d7eaf51ac
Key  [ 5]:9a6304428b45da128ab64c8805c32452
Key  [ 6]:8af4d1e9d80cb73ec6b44e9b6e4f39d8
round   4:9f0512260a2f7a5067efc35bf1706831
Key  [ 7]:79cc2d138606f0fca4e549c34a1e6d19
Key  [ 8]:803dc5cdde0efdbee7a1342b2cd4d344
round   5:0cfd7856edfafac51f29e86365de6f57
Key  [ 9]:e8fa996448e6b6459ab51e7be101325a
Key  [10]:2acc7add7b294acb444cd933f0e74ec9
round   6:2f1fa34bf352dc77c0983a01e8b7d622
Key  [11]:f57de39e42182efd6586b86a90c86bb1
Key  [12]:e418dfd1bb22ebf1bfc309cd27f5266c
round   7:ee4f7a53849bf73a747065d35f3752b1
Key  [13]:80a9959133856586370854db6e0470b3
Key  [14]:f4c1bc2f764a0193749f5fc09011a1ae
round   8:8fec6f7249760ebf69e370e9a4b80a92
Key  [15]:d036cef70d6470c3f52f1b5d25b0c29d
Key  [16]:d0956af6b8700888a1cc88f07ad226dc
Key  [17]:1ce8b39c4c7677373c30849a3ee08794
K_enc    :ea520cfc546b00eb7c3a6cea3ecb39ed
```

# 11 CONNECTIONLESS PERIPHERAL BROADCAST SAMPLE DATA

This section contains an example of the DM3 packet used for the synchronization train (see [Vol 2] Part B, Section 8.11.2).

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 0
TYPE = 1010 (DM3)
FLOW = 0
ARQN = 0
SEQN = 0

Payload:
--------
logical channel = 10 (binary) (L2CAP start or no fragmentation)
payload length = 28 bytes
flow = 0
current CLK = 0x2345678
next Connectionless Peripheral Broadcast instant = 0x23457a0
AFH channel map = all channels used except 16 and 42 to 47
Central BD_ADDR = NAP 0xACDE, UAP 0x48, LAP 0x610316
Connectionless Peripheral Broadcast interval = 564 slots
Connectionless Peripheral Broadcast LT_ADDR = 1
service data = 0x69

AIR DATA

Packet header (including HEC, in transmitted bit order):
--------------------------------------------------------
000000000
000111000111
000
000
000
111111111000000000000111
```

*Sample Data*

```
Payload
-------
The data forming the payload consists of the following 32 octets
(given in hexadecimal) in the order transmitted:

e2 00
78 56 34 02
d0 2b 1a 01
ff ff fe ff ff 03 ff ff ff 7f
16 03 61 48 de ac
34 02
01
69
f2 85

The bit sequence transmitted (including FEC) will therefore begin:

0100011100 01001
0000000001 10101
1110011010 11011

and end:

0100111110 10001
1000010000 00011
```

BR/EDR Controller
Part H

# SECURITY SPECIFICATION

*This Part describes the security system which may be used at the Link Layer. The Encryption, Authentication, and Key Generation schemes are specified. The requirements for the supporting process of random number generation are also specified.*

# CONTENTS

**1 Security overview ..... 1030**
1.1 Pausing encryption and role switch ..... 1031
1.2 Change connection link keys ..... 1031
1.3 Periodically refreshing encryption keys ..... 1032

**2 Random number generation ..... 1033**

**3 Key management ..... 1035**
3.1 Key types ..... 1035
3.2 Key generation and initialization ..... 1037
3.2.1 Generation of initialization key, $K_{init}$ ..... 1037
3.2.2 Authentication ..... 1038
3.2.3 [This section is no longer used] ..... 1038
3.2.4 Generation of a combination key ..... 1038
3.2.5 Generating the encryption key ..... 1040
3.2.6 Point-to-multipoint configuration ..... 1040
3.2.7 Modifying the link keys ..... 1041
3.2.8 Generating a temporary link key ..... 1041

**4 Encryption (E0) ..... 1043**
4.1 Encryption key size negotiation ..... 1043
4.2 Encryption of broadcast messages ..... 1044
4.3 Encryption concept ..... 1044
4.4 Encryption algorithm ..... 1045
4.4.1 The operation of the cipher ..... 1047
4.5 LFSR initialization ..... 1048
4.6 Key stream sequence ..... 1051

**5 Authentication ..... 1052**
5.1 Repeated attempts ..... 1055

**6 The authentication and key-generating functions ..... 1056**
6.1 The authentication function $E_1$ ..... 1056
6.2 The functions $A_r$ and $A'_r$ ..... 1058
6.2.1 The round computations ..... 1059
6.2.2 The substitution boxes “e” and “l” ..... 1059
6.2.3 Key scheduling ..... 1061
6.3 $E_2$-key generation function for authentication ..... 1061
6.4 $E_3$-key generation function for encryption ..... 1063

**7 Secure Simple Pairing ..... 1065**
7.1 Phase 1: Public key exchange ..... 1066
7.2 Phase 2: Authentication stage 1 ..... 1067
7.2.1 Authentication stage 1: Numeric Comparison protocol ..... 1067
7.2.2 Authentication stage 1: Out of Band protocol ..... 1069
7.2.3 Authentication stage 1: Passkey Entry protocol ..... 1071
7.3 Phase 3: Authentication stage 2 ..... 1073
7.4 Phase 4: Link key calculation ..... 1074
7.5 Phase 5: LMP authentication and encryption ..... 1075
7.6 Elliptic curve definition ..... 1075
7.7 Cryptographic function definitions ..... 1076
7.7.1 The Secure Simple Pairing commitment function *f1* .. 1076
7.7.2 The Secure Simple Pairing numeric verification function *g* ..... 1077
7.7.3 The Secure Simple Pairing key derivation function *f2* ..... 1078
7.7.4 The Secure Simple Pairing check function *f3* ..... 1079
7.7.5 [This section is no longer used] ..... 1080
7.7.6 The AES encryption key generation function *h3* ..... 1080
7.7.7 The Device authentication key generation function *h4* ..... 1081
7.7.8 The Device authentication confirmation function *h5* .. 1081

**8 [This section is no longer used] ..... 1083**

**9 AES-CCM encryption for BR/EDR ..... 1084**
9.1 Nonce formats ..... 1084
9.2 Counter mode blocks ..... 1086
9.3 Encryption blocks ..... 1087
9.4 Encryption key size reduction ..... 1088
9.5 Repeated MIC failures ..... 1088

# 1 SECURITY OVERVIEW

Bluetooth wireless technology provides peer-to-peer communications over short distances. In order to provide usage protection and information confidentiality, the system provides security measures both at the application layer and the Link Layer. These measures are designed to be appropriate for a peer environment.

This means that in each device, the authentication and encryption routines are implemented in the same way.

In this Part, the term "random number" includes pseudo-random numbers, subject to the requirements in Section 2.

The security mechanisms used in BR/EDR have evolved over the course of multiple versions of the specification in three phases: legacy, Secure Simple Pairing, and Secure Connections. The encryption, authentication and key generation algorithms associated with each is shown in Table 1.1.

| Security Mechanism | Legacy | Secure Simple Pairing | Secure Connections |
|---|---|---|---|
| Encryption | E0 | E0 | AES-CCM |
| Authentication | SAFER+ | SAFER+ | HMAC-SHA-256 |
| Key Generation | SAFER+ | P-192 ECDH<br>HMAC-SHA-256 | P-256 ECDH<br>HMAC-SHA-256 |

*Table 1.1: Security algorithms*

The legacy encryption, authentication and key generation algorithms are described in Section 3 to 6. The algorithms used for Secure Simple Pairing and Secure Connections are described in Section 7 to 9.

Four different entities are used for maintaining security at the Link Layer: a Bluetooth Device Address, two secret keys, and a random number that shall be regenerated for each new transaction. The four entities and their sizes are summarized in Table 1.2.

| Entity | Size |
|---|---|
| BD_ADDR | 48 bits |
| Private user key, authentication | 128 bits |
| Private user key, encryption configurable length (byte-wise) | 56 to 128 bits |
| RAND | 128 bits |

*Table 1.2: Entities used in authentication and encryption procedures*

The Bluetooth Device Address (BD_ADDR) is the 48-bit address. The BD_ADDR can be obtained via user interactions, or, automatically, via an inquiry routine by a device.

The secret keys are derived during initialization and are never disclosed. The encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits. For the encryption algorithm, the key size may vary between 7 and 16 octets (56 to 128 bits).

The encryption key is entirely different from the authentication key. Each time encryption is activated, a new encryption key shall be generated. Thus, the lifetime of the encryption key does not necessarily correspond to the lifetime of the authentication key.

It is anticipated that the authentication key will be more static in its nature than the encryption key – once established, the particular application running on the device decides when, or if, to change it. To underline the fundamental importance of the authentication key to a specific link, it is often referred to as the link key.

RAND is a random number which can be derived from a random process in the device. This is not a static parameter and will change frequently.

In this part, the terms user and application are used interchangeably to designate the entity that is at either side.

## 1.1 Pausing encryption and role switch

To perform a role switch on a connection encryption must be disabled or paused as the encryption is based off the Central's clock and Bluetooth address information. Unfortunately, if the role switch is required, and encryption is turned off, the device on the other end of the link will not be aware of the reason for disabling encryption, and can therefore take two possible actions: send clear text data, or disconnect. Neither of these possible actions is desirable. When performing a role switch on an encrypted link, the role switch shall be performed as a single operation where possible. If this is not possible because the other device does not support the encryption pause feature, then when a device needs an encrypted link, and encryption is disabled, then the device should not send any user data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.2 Change connection link keys

It is possible to perform a change of connection link keys while a link is encrypted, but it is not possible to use the new link keys until encryption has been stopped and then restarted. As with role switches, disabling encryption and then re-enabling it again can cause user data to be sent in the clear or a disconnection to occur. The use of encryption pausing prevents this problem from occurring. On devices that do not

support the encryption pause feature, when a device needs an encrypted link, and encryption is disabled, then the device should not send any data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.3 Periodically refreshing encryption keys

If both devices support the encryption pause feature, then the encryption keys shall be refreshed by the Link Manager at least once every $2^{28}$ ticks of the Bluetooth clock (about 23.3 hours) when E0 encryption is used and before either the PayloadCounter or dayCounter roll over when AES-CCM encryption is used if it is not refreshed by the Host or the remote Link Manager. To refresh an encryption key, the Host may use the Change Connection Link Key procedure or request an encryption key refresh. If the encryption key has not been refreshed before going stale, a device may disconnect the link.

Note: The roll over will occur after at least $2^{38}$ ticks of the Bluetooth clock or at least 2.72 years.

# 2 RANDOM NUMBER GENERATION

Each device shall have a random number generator. Random numbers are used for many purposes within the security functions – or instance, for the challenge-response scheme, for generating authentication and encryption keys, nonces used in Secure Simple Pairing, for Passkeys used in authentication. A device shall use a random number generator compliant with [FIPS PUB 140-2] (http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf) or a later update.

When using a pseudo-random number generator, the device shall use a seed with at least the minimum entropy required by that pseudo-random number generator.

The random number generator shall be tested against the [FIPS SP800-22] (http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf). This encompasses the verification of the following statistical tests performed on the output of the PRNG as specified by the [FIPS SP800-22]:

1. The Frequency (Monobit) Test
2. Frequency Test within a Block
3. The Runs Test
4. Test for the Longest-Run-of-Ones in a Block
5. The Binary Matrix Rank Test
6. The Discrete Fourier Transform (Spectral) Test
7. The Non-overlapping Template Matching Test
8. The Overlapping Template Matching Test
9. Maurer's "Universal Statistical" Test
10. The Linear Complexity Test
11. The Serial Test
12. The Approximate Entropy Test
13. The Cumulative Sums (Cusums) Test
14. The Random Excursions Test
15. The Random Excursions Variant Test

These tests are part of standard statistical mathematical packages. Some test suites, like the Diehard test suite, can be used to verify FIPS compliance. Alternatively, other tools, such as the DieHarder (http://www.phy.duke.edu/~rgb/General/rand_rate.php) or the available NIST tools (http://csrc.nist.gov/groups/ST/toolkit/random_number.html)

and the corresponding recommendations (http://csrc.nist.gov/publications/nistpubs/800-22-rev1/SP800-22rev1.pdf) may also be used.

# 3 KEY MANAGEMENT

A device shall not allow the end user to set the encryption key size. The Bluetooth Baseband processing shall not accept an encryption key given from higher software layers. Whenever a new encryption key is required, it shall be created as defined in Section 6.4.

Changing a link key shall also be done through the defined Baseband procedures. Depending on what kind of link key it is, different approaches are required. The details are found in Section 3.2.7.

## 3.1 Key types

The link key is a 128-bit random number which is shared between two or more parties and is the base for all security transactions between these parties. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived.

In the following, a session is defined as the time interval for which the device is a member of a particular piconet. Thus, the session terminates when the device disconnects from the piconet.

The link keys are either semi-permanent or temporary. A semi-permanent link key may be stored in non-volatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the devices sharing it. The designation semi-permanent is justified by the possibility of changing it. How to do this is described in Section 3.2.7.

The lifetime of a temporary link key is limited by the lifetime of the current session – it shall not be reused in a later session. Typically, in a point-to-multipoint configuration where the same information is to be distributed securely to several recipients, a common encryption key is useful. To achieve this, the temporary link key replaces the semi-permanent link keys. The details of this procedure are found in Section 3.2.6.

In the following, the current link key is the link key in use at the current moment. It can be semi-permanent or temporary. Thus, the current link key is used for all authentications and all generation of encryption keys in the on-going connection (session).

In order to accommodate different types of applications, three types of link keys have been defined:

- the combination key $K_{AB}$
- the temporary key K_temp
- the initialization key $K_{init}$

In addition to these keys there is an encryption key, denoted K_enc. This key is derived from the current link key. Whenever encryption is activated by an LM command, the encryption key shall be changed automatically. The purpose of separating the authentication key and encryption key is to facilitate the use of a shorter encryption key without weakening the strength of the authentication procedure. There are no governmental restrictions on the strength of authentication algorithms. However, in some countries, such restrictions exist on the strength of encryption algorithms.

The combination key is derived from information in both devices A and B, and is therefore always dependent on two devices. The combination key is derived for each new combination of two devices.

The temporary link key, K_temp, shall only be used during the current session. It shall only replace the original link key temporarily. For example, a Central may use the temporary link key to reach more than one device simultaneously using the same encryption key, see Section 3.2.6.

The initialization key, $K_{init}$, shall be used as the link key during the initialization process when no combination keys have been defined and exchanged yet or when a link key has been lost. The initialization key protects the transfer of initialization parameters. The key is derived from a random number, an L-octet PIN code, and a BD_ADDR. This key shall only be used during initialization.

The PIN may be a fixed number provided with the device (for example when there is no user interface). Alternatively, the PIN can be selected by the user, and then entered in both devices that are to be matched. The latter procedure should be used when both devices have a user interface, for example a phone and a laptop. Entering a PIN in both devices is more secure than using a fixed PIN in one of the devices, and should be used whenever possible. Even if a fixed PIN is used, it shall be possible to change the PIN; this is in order to prevent re-initialization by users who once obtained the PIN. If no PIN is available, a default value of zero may be used. The length of this default PIN is one byte, PIN (default) = 0x00. This default PIN may be provided by the Host.

For many applications the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. Even though this gives sufficient security in many cases, there exist countless other, more sensitive, situations where this is not reliable enough. Therefore, the PIN code may be chosen to be any length

from 1 to 16 octets. For the longer lengths, the devices exchanging PIN codes may use software at the application layer rather than mechanical (i.e. human) interaction. For example, this can be a Diffie-Hellman key agreement, where the exchanged key is passed on to the $K_{init}$ generation process in both devices, just as in the case of a shorter PIN code.

## 3.2 Key generation and initialization

The link keys must be generated and distributed among the devices in order to be used in the authentication procedure. Since the link keys shall be secret, they shall not be obtainable through an inquiry routine in the same way as the Bluetooth Device Addresses. The exchange of the keys takes place during an initialization phase which shall be carried out separately for each two devices that are using authentication and encryption. The initialization procedures consist of the following five parts:

- generation of an initialization key
- generation of link key
- link key exchange
- authentication
- generation of encryption key in each device (optional)

After the initialization procedure, the devices can proceed to communicate, or the link can be disconnected. If encryption is implemented, the $E_0$ algorithm shall be used with the proper encryption key derived from the current link key. For any new connection established between devices A and B, they shall use the common link key for authentication, instead of once more deriving $K_{init}$ from the PIN. A new encryption key derived from that particular link key shall be created next time encryption is activated.

If no link key is available, the LM shall automatically start an initialization procedure.

### 3.2.1 Generation of initialization key, $K_{init}$

A link key is used temporarily during initialization, the initialization key $K_{init}$. This key shall be derived by the $E_{22}$ algorithm from a BD_ADDR, a PIN code, the length of the PIN (in octets), and a random number $IN_RAND$. The principle is depicted in Figure 6.4. The 128-bit output from $E_{22}$ shall be used for key exchange during the generation of a link key. When the devices have performed the link key exchange, the initialization key shall be discarded.

When the initialization key is generated, the PIN is augmented with the BD_ADDR. If one device has a fixed PIN the BD_ADDR of the other device shall be used. If both

devices have a variable PIN the BD_ADDR of the device that received IN_RAND shall be used. If both devices have a fixed PIN they cannot be paired. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used. This procedure ensures that $K_{init}$ depends on the identity of the device with a variable PIN. A fraudulent device may try to test a large number of PINs by claiming another BD_ADDR each time. It is the application's responsibility to take countermeasures against this threat. If the device address is kept fixed, the waiting interval before the next try may be increased exponentially (see Section 5.1).

The details of the $E_{22}$ algorithm can be found in Section 6.3.

### 3.2.2 Authentication

The legacy authentication procedure shall be carried out as described in Section 5. During each legacy authentication, a new AU_RAND_A shall be issued.

For legacy authentication, mutual authentication is achieved by first performing the authentication procedure in one direction and then immediately performing the authentication procedure in the opposite direction.

The secure authentication procedure is always a mutual authentication. Secure authentication shall be carried out as described in Section 7.7.8. During each secure authentication, new AU_RAND_C and AU_RAND_P shall be used.

As a side effect of a successful authentication procedure an auxiliary parameter, the Authenticated Ciphering Offset (ACO), will be computed. The ACO shall be used for ciphering key generation as described in Section 3.2.5.

The claimant/verifier status is determined by the LM.

### 3.2.3 [This section is no longer used]

### 3.2.4 Generation of a combination key

To use a combination key, it is first generated during the initialization procedure. The combination key is the combination of two numbers generated in device A and B, respectively. First, each device shall generate a random number, $LK_RAND_A$ and $LK_RAND_B$. Then, using $E_{21}$ with the random number and their own BD_ADDRs, the two random numbers

$$LK_K_A = E_{21}(LK_RAND_A, BD_ADDR_A), \quad \text{(EQ 1)}$$

and

$$LK_K_B = E_{21}(LK_RAND_B, BD_ADDR_B), \quad \text{(EQ 2)}$$

shall be created in device A and device B, respectively. These numbers constitute the devices' contribution to the combination key that is to be created. Then, the two random numbers *LK_RAND*$_A$ and *LK_RAND*$_B$ shall be exchanged securely by XORing with the current link key, $K$. Thus, device A shall send $CA = K \oplus LK_RAND_A$ to device B, while device B shall send $CB = K \oplus LK_RAND_B$ to device A. If this is done during the initialization phase the link key $K = K_{init}$. If CA is identical to CB, then the devices shall terminate the process of generating the combination key (e.g., by rejecting the following authentication).

When the random numbers $LK_RAND_A$ and $LK_RAND_B$ have been mutually exchanged, each device shall recalculate the other device's contribution to the combination key. This is possible since each device knows the Bluetooth Device Address of the other device. Thus, device A shall calculate (EQ 2) and device B shall calculate (EQ 1). After this, both devices shall XOR the two numbers to generate the 128-bit link key. The result shall be stored in device A as the link key $K_{AB}$ and in device B as the link key $K_{BA}$.
If the resulting link key consists of all zeroes, then the devices shall discard the link key and terminate the process of generating the combination key (see [Vol 2] Part C, Section 4.2.1). When both devices have derived the new combination key, a mutual authentication procedure shall be initiated to confirm the success of the transaction. If this authentication procedure fails, then the device shall discard the link key. The old link key shall be discarded after a successful exchange of a new combination key. The message flow between Central and Peripheral and the principle for creating the combination key is depicted in Figure 3.1.

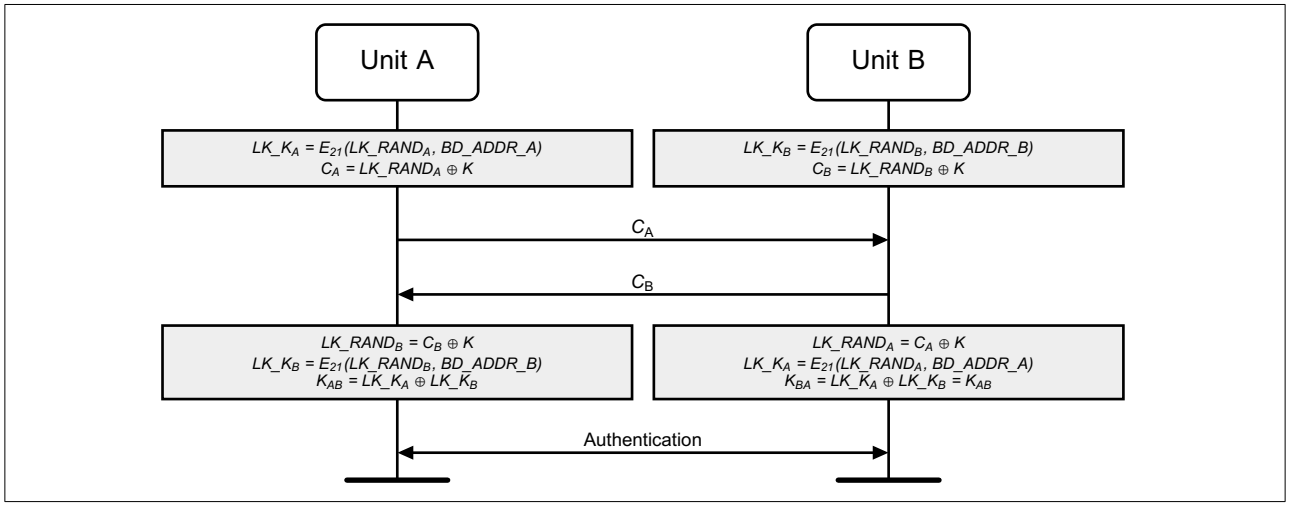



*Figure 3.1: Generating a combination key. The old link key (K) is discarded after the exchange of a new combination key has succeeded.*

### 3.2.5 Generating the encryption key

The encryption key, K_enc, is derived by algorithm $E_3$ from the current link key, a 96-bit Ciphering OFfset number (COF), and a 128-bit random number. The COF is determined in one of two ways. If the current link key is a temporary link key, then COF shall be derived from the Central's BD_ADDR. Otherwise the value of COF shall be set to the value of ACO as computed during the authentication procedure. Therefore:

$$\text{COF} = \begin{cases} \text{BD_ADDR} \parallel \text{BD_ADDR}, & \text{if link key is a temporary key} \\ \text{ACO}, & \text{otherwise.} \end{cases} \quad \text{(EQ 3)}$$

There is an explicit call of $E_3$ when the LM activates encryption. Consequently, the encryption key is automatically changed each time the device enters encryption mode. The details of the key generating function $E_3$ can be found in Section 6.4.

### 3.2.6 Point-to-multipoint configuration

It is possible for the Central to use separate encryption keys for each Peripheral in a point-to-multipoint configuration with ciphering activated. Then, if the application requires more than one Peripheral to listen to the same payload, each Peripheral must be addressed individually. This can cause unwanted capacity loss for the piconet. Moreover, a Peripheral might not be capable of switching between two or more encryption keys in real time (e.g., after looking at the LT_ADDR in the header). Thus, the Central cannot use different encryption keys for broadcast messages and individually addressed traffic. Therefore, the Central may tell several Peripherals to use a common link key (and, hence, indirectly also to use a common encryption key) and may then broadcast the information encrypted. For many applications, this key is only of temporary interest. In the following discussion, this key is denoted by K_temp.

The transfer of necessary parameters shall be protected by the routine described in Section 3.2.8. After the confirmation of successful reception in each Peripheral, the Central shall issue a command to the Peripherals to replace their respective current link key by the new temporary link key. Before encryption can be activated, the Central shall also generate and distribute a common EN_RAND to all participating Peripherals. Using this random number and the newly derived temporary link key, each Peripheral shall generate a new encryption key.

The Central negotiates the encryption key length to use individually with each Peripheral that will use the temporary link key. If the Central has already negotiated with some of these Peripherals, it has knowledge of the sizes that can be accepted. There may be situations where the permitted key lengths of some devices are incompatible. In that case, the Central shall omit the limiting device from the group.

When all Peripherals have received the necessary data, the Central can communicate information on the piconet securely using the encryption key derived from the new

temporary link key. Each Peripheral in possession of the temporary link key can eavesdrop on all encrypted traffic, not only the traffic intended for itself. The Central may tell all participants to fall back to their old link keys simultaneously.

### 3.2.7 Modifying the link keys

If the key change concerns combination keys, then the procedure is straightforward. The change procedure is identical to the procedure described in Figure 3.1, using the current value of the combination key as link key. This procedure can be carried out at any time after the authentication and encryption start. Since the combination key corresponds to a single link, it can be modified each time this link is established. This will improve the security of the system since then old keys lose their validity after each session.

Starting up an entirely new initialization procedure is also possible. In that case, user interaction is necessary since a PIN will be required in the authentication and encryption procedures.

### 3.2.8 Generating a temporary link key

The key-change routines described so far are semi-permanent. To create the temporary link key, which can replace the current link key during a session (see Section 3.2.6), other means are needed. First, the Central shall create a new link key from two 128-bit random numbers, RAND1 and RAND2. This shall be done by

$$\text{K_temp} = E_{22}(\text{RAND1}, \text{RAND2}, 16). \quad \text{(EQ 4)}$$

This key is a 128-bit random number. The reason for using the output of $E_{22}$ and not directly choosing a random number as the key, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the device.

Then, a third random number, RAND, shall be transmitted to the Peripheral. Using $E_{22}$ with the current link key and RAND as inputs, both the Central and the Peripheral shall compute a 128-bit overlay. The Central shall send the bitwise XOR of the overlay and the new link key to the Peripheral. The Peripheral, who knows the overlay, shall recalculate K_temp. To confirm the success of this transaction, the devices shall perform a mutual authentication procedure using the new link key. This procedure shall then be repeated for each Peripheral that receives the new link key. The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the Central falls back to the previous (semi-permanent) link key.

The Central activates encryption by an LM command. Before activating encryption, the Central shall ensure that all Peripherals receive the same random number,

EN_RAND, since the encryption key is derived through the means of $E_3$ individually in all participating devices. Each Peripheral shall compute a new encryption key as follows:

$$\text{K_enc} = E_3(\text{K_temp}, \text{EN_RAND}, \text{COF}) \quad \text{(EQ 5)}$$

where the value of COF shall be derived from the Central's BD_ADDR as specified by equation (EQ 3). The details of the encryption key generating function are described in Section 6.4. The message flow between the Central and the Peripheral when generating the temporary link key is depicted in Figure 3.2.

Note: In this case the ACO produced during the authentication is not used when computing the ciphering key.

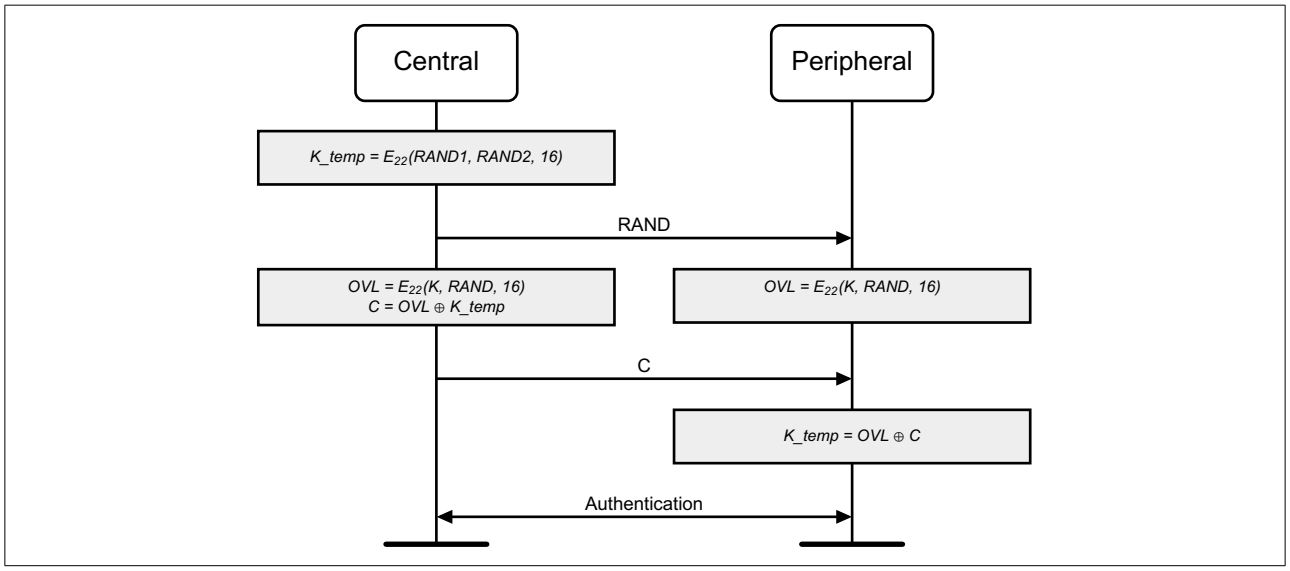



*Figure 3.2: Temporary link key distribution and computation of the corresponding encryption key*

# 4 ENCRYPTION (E0)

User information can be protected by encryption of the packet payload; the access code and the packet header shall never be encrypted. The encryption of the payload shall be carried out with a stream cipher called $E_0$ that shall be re-synchronized for every payload. The overall principle is shown in Figure 4.1.

The stream cipher system $E_0$ shall consist of three parts:

- the first part performs initialization (generation of the session key). The session key generator shall combine the input bits in an appropriate order and shall shift them into the four LFSRs used in the key stream generator.
- the second part generates the key stream bits and shall use a method derived from the summation stream cipher generator attributable to Massey and Rueppel. The second part is the main part of the cipher system, as it will also be used for initialization.
- the third part performs encryption and decryption.

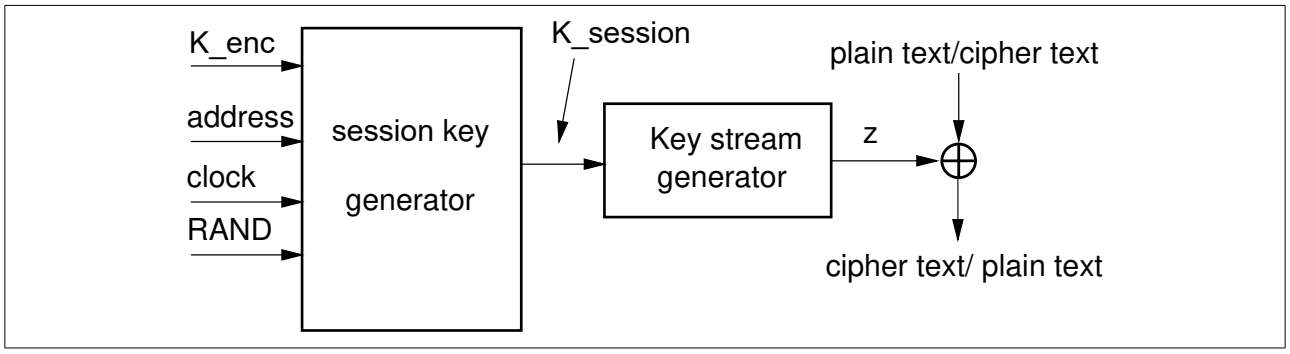



*Figure 4.1: Stream ciphering for Bluetooth with $E_0$.*

## 4.1 Encryption key size negotiation

Each device implementing the Baseband specification shall have a parameter defining the maximal allowed key length, L_max, 7 ≤ L_max ≤ 16 (number of octets in the key). For each application using encryption, a number L_min shall be defined indicating the shortest acceptable key size for that particular application. Before generating the encryption key, the devices involved shall negotiate to decide the key size to use.

The Central shall send a suggested value, L_sug_c, to the Peripheral. Initially, the suggested value shall be set to L_max_c. If L_min_p ≤ L_sug_c, and, the Peripheral supports the suggested length, the Peripheral shall acknowledge and this value shall be the length of the encryption key for this link. However, if both conditions are not fulfilled, the Peripheral shall send a new proposal, L_sug_p < L_sug_c, to the Central. This

value shall be the largest among all supported lengths less than the previous Central suggestion. Then, the Central shall perform the corresponding test on the Peripheral suggestion. This procedure shall be repeated until a key length agreement is reached, or, one device aborts the negotiation. An abort may be caused by lack of support for L_sug and all smaller key lengths, or if L_sug < L_min in one of the devices. In case of an abort link encryption cannot be employed.

The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested key size. However, this is a necessary precaution. Otherwise a fraudulent device could enforce a weak protection on a link by claiming a short maximum key size.

## 4.2 Encryption of broadcast messages

There may be three settings for the Baseband regarding encryption:

1. No encryption.

   This is the default setting. No messages are encrypted

2. Point-to-point only encryption.

   Broadcast messages are not encrypted. This may be enabled either during the connection establishment procedure or after the connection has been established.

3. Point-to-point and broadcast encryption.

   All messages are encrypted. This may be enabled after the connection has been established only. This setting should not be enabled unless all affected links share the same temporary link key as well as the same EN_RAND value, both used in generating the encryption key.

| Broadcast traffic | Individually addressed traffic |
|---|---|
| No encryption | No encryption |
| No encryption | Encryption (semi-permanent link key) |
| Encryption (temporary link key) | Encryption (temporary link key) |

*Table 4.1: Possible encryption modes for a Peripheral in possession of a temporary link key*

## 4.3 Encryption concept

For the encryption routine, a stream cipher algorithm is used in which ciphering bits are XORed with the data stream to be sent over the air interface. The payload is ciphered after the CRC bits are appended, but, prior to the FEC encoding.

Each packet payload shall be ciphered separately. The cipher algorithm $E_0$ uses the Central's Bluetooth Device Address (BD_ADDR_C), 26 bits of the Central real-time clock ($CLK_{26\text{-}1}$) and the encryption key K_enc as input, see Figure 4.2.

The encryption key K_enc is derived from the current link key, COF, and a random number, EN_RAND_C (see Section 6.4). The random number shall be issued by the Central before entering encryption mode.

Note: EN_RAND_C is publicly known since it is transmitted as plain text over the air.

Within the $E_0$ algorithm, the encryption key K_enc is modified into another key denoted K_session. The maximum effective size of this key shall be factory preset and may be set to any multiple of eight between seven and sixteen (i.e., 56 to 128 bits). The procedure for deriving the key is described in Section 4.5.

The $E_0$ algorithm shall be re-initialized at the start of each new packet (i.e. for Central-to-Peripheral as well as for Peripheral-to-Central transmission). By using $CLK_{26\text{-}1}$ at least one bit is changed between two transmissions. Thus, a new keystream is generated after each re-initialization. For packets covering more than a single slot, the Bluetooth clock as found in the first slot shall be used for the entire packet.

The encryption algorithm $E_0$ generates a binary keystream, $K_{cipher}$, which shall be XORed with the data to be encrypted. The cipher is symmetric; decryption shall be performed in exactly the same way using the same key as used for encryption.

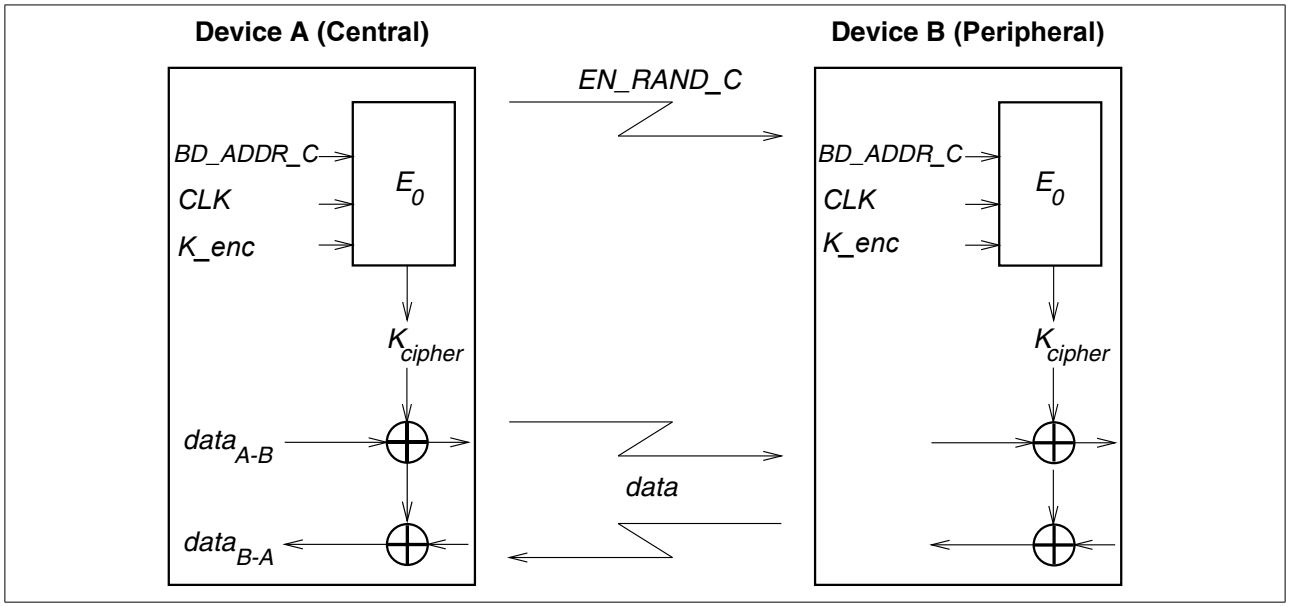



*Figure 4.2: Functional description of the encryption procedure*

## 4.4 Encryption algorithm

The system uses linear feedback shift registers (LFSRs) whose output is combined by a simple finite state machine (called the summation combiner) with 16 states. The output of this state machine is the key stream sequence, or, during initialization phase, the randomized initial start value. The algorithm uses an encryption key K_enc, a

48-bit Bluetooth address, the Central's clock bits $CLK_{26\text{-}1}$, and a 128-bit RAND value. Figure 4.3 shows the setup.

There are four LFSRs ($LFSR_1$ ,...,$LFSR_4$ ) of lengths $L_1 = 25$, $L_2 = 31$, $L_3 = 33$, and, $L_4 = 39$, with feedback polynomials as specified in Table 4.2. The total length of the registers is 128. These polynomials are all primitive. The Hamming weight of all the feedback polynomials is five.

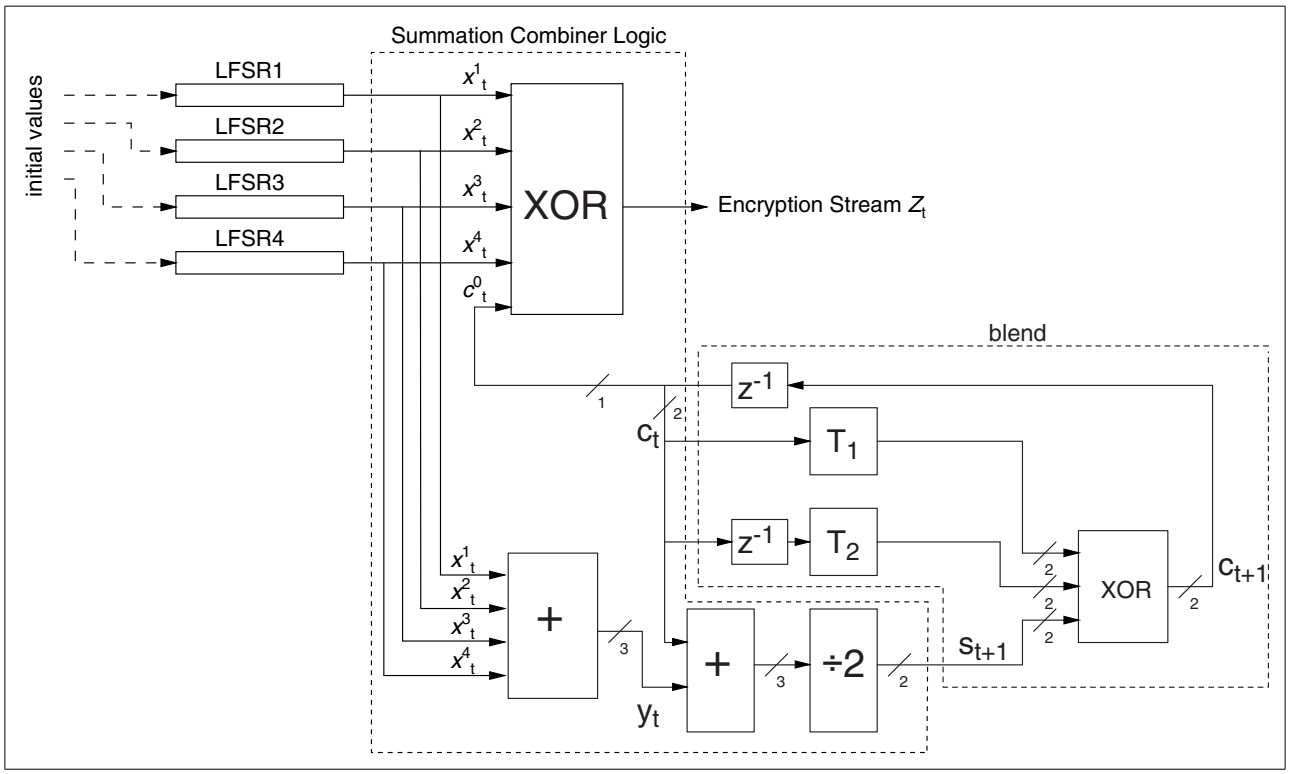



*Figure 4.3: Concept of the $E_0$ encryption algorithm*

| $i$ | $L_i$ | **feedback** $f_i(t)$ | **weight** |
|---|---|---|---|
| 1 | 25 | $t^{25} + t^{20} + t^{12} + t^8 + 1$ | 5 |
| 2 | 31 | $t^{31} + t^{24} + t^{16} + t^{12} + 1$ | 5 |
| 3 | 33 | $t^{33} + t^{28} + t^{24} + t^4 + 1$ | 5 |
| 4 | 39 | $t^{39} + t^{36} + t^{28} + t^4 + 1$ | 5 |

*Table 4.2: The four primitive feedback polynomials*

Let $x_t^i$ denote the $t^{th}$ symbol of LFSR$_i$. The value $y_t$ is derived from the four-tuple $x_t^1, \ldots, x_t^4$ using the following equation:

$$y_t = \sum_{i=1}^{4} x_t^i, \quad \text{(EQ 6)}$$

where the sum is over the integers. Thus $y_t$ can take the values 0,1,2,3, or 4. The output of the summation generator is obtained by the following equations:

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\}, \quad \text{(EQ 7)}$$

$$s_{t+1} = \left(s_{t+1}^1, s_{t+1}^0\right) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\}, \quad \text{(EQ 8)}$$

$$c_{t+1} = \left(c_{t+1}^1, c_{t+1}^0\right) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \quad \text{(EQ 9)}$$

where $T_1[\,.\,]$ and $T_2[\,.\,]$ are two different linear bijections over GF(4). Suppose GF(4) is generated by the irreducible polynomial $x^2 + x + 1$, and let $\alpha$ be a zero of this polynomial in GF(4). The mappings $T_1$ and $T_2$ are now defined as:

$$T_1 : \mathrm{GF}(4) \rightarrow \mathrm{GF}(4)$$
$$x \mapsto x$$
$$T_2 : \mathrm{GF}(4) \rightarrow \mathrm{GF}(4)$$
$$x \mapsto (\alpha + 1)x\,.$$

The elements of GF(4) can be written as binary vectors. This is summarized in Table 4.3.

| $x$ | $T_1[x]$ | $T_2[x]$ |
|---|---|---|
| 00 | 00 | 00 |
| 01 | 01 | 11 |
| 10 | 10 | 01 |
| 11 | 11 | 10 |

*Table 4.3: The mappings $T_1$ and $T_2$*

Since the mappings are linear, they can be implemented using XOR gates; i.e.

$$T_1 : (x_1, x_0) \mapsto (x_1, x_0),$$
$$T_2 : (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0)\,.$$

### 4.4.1 The operation of the cipher

Figure 4.4 gives an overview of the operation in time. The encryption algorithm shall run through the initialization phase before the start of transmission or reception of a new

packet. Thus, for multislot packets the cipher is initialized using the clock value of the first slot in the multislot sequence.

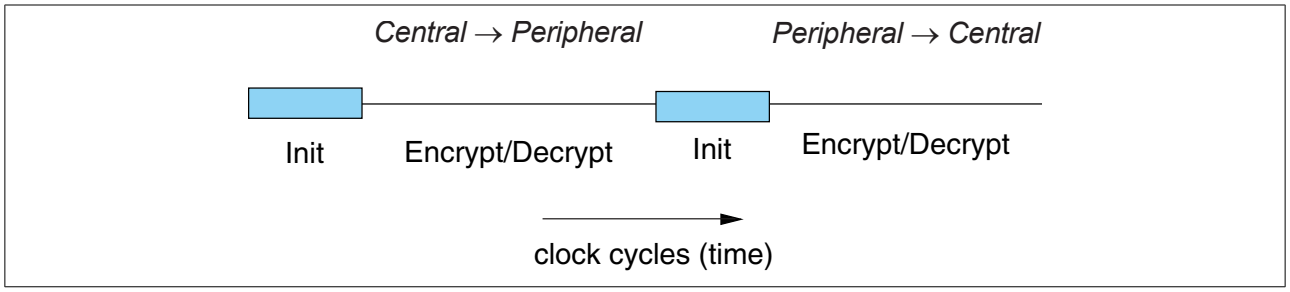



*Figure 4.4: Overview of the operation of encryption. Between each start of a packet (TX or RX), the LFSRs are re-initialized.*

## 4.5 LFSR initialization

The key stream generator is loaded with an initial value for the four LFSRs (in total 128 bits) and the 4 bits that specify the values of $c_0$ and $c_{-1}$. The 132 bit initial value is derived from four inputs by using the key stream generator. The input parameters are the key K_enc, a 128-bit random number RAND, a 48-bit Bluetooth Device Address, and the 26 Central's clock bits $CLK_{26\text{-}1}$.

The effective length of the encryption key may vary between 56 and 128 bits. The actual key length as obtained from $E_3$ is 128 bits. Then, within $E_0$, the key length may be reduced by a *mod* operation between K_enc and a polynomial of desired degree. After reduction, the result is encoded with a block code in order to distribute the starting states more uniformly. The operation shall be as defined in (EQ 10).

When the encryption key has been created the LFSRs are loaded with their initial values. Then, 200 stream cipher bits are created by operating the generator. Of these bits, the last 128 are fed back into the key stream generator as an initial value of the four LFSRs. The values of $c_t$ and $c_{t-1}$ are kept. From this point on, when clocked the generator produces the encryption (decryption) sequence which is bitwise XORed to the transmitted (received) payload data.

In the following, octet $i$ of a binary sequence $X$ is $X[i]$. Bit 0 of $X$ is the LSB. Then, the LSB of $X[i]$ corresponds to bit $8i$ of the sequence $X$, the MSB of $X[i]$ is bit $8i + 7$ of $X$. For instance, bit 24 of the Bluetooth Device Address is the LSB of BD_ADDR[3].

The details of the initialization shall be as follows:

1. Create the encryption key to use from the 128-bit secret key K_enc and the 128-bit publicly known EN_RAND. Let L, $7 \leq L \leq 16$, be the effective key length in number of octets. The resulting encryption key is K_session:

$$\text{K_session}(x) = g_2^{(L)}(x)\left(\text{K_enc}(x) \; mod \; g_1^{(L)}(x)\right) \quad \text{(EQ 10)}$$

   where $\deg\left(g_1^{(L)}(x)\right) = 8L$ and $\deg\left(g_2^{(L)}(x)\right) \leq 128 - 8L$. The polynomials are defined in Table 4.4.

2. Shift the 3 inputs K_session, the Bluetooth Device Address, the clock, and the six-bit constant 111001 into the LFSRs. In total, 208 bits are shifted in:
   a. Open all switches shown in Figure 4.5;
   b. Arrange inputs bits as shown in Figure 4.5; Set the content of all shift register elements to zero. Set $t = 0$.
   c. Start shifting bits into the LFSRs. The rightmost bit at each level of Figure 4.5 is the first bit to enter the corresponding LFSR.
   d. When the first input bit at level i reaches the rightmost position of $LFSR_i$, close the switch of this LFSR.
   e. At $t = 39$ (when the switch of $LFSR_4$ is closed), reset both blend registers $c_{39} = c_{39-1} = 0$; Up to this point, the content of $c_t$ and $c_{t-1}$ has been of no concern. However, their content will now be used in computing the output sequence.
   f. From now on output symbols are generated. The remaining input bits are continuously shifted into their corresponding shift registers. When the last bit has been shifted in, the shift register is clocked with input = 0;

      Note: When finished, $LFSR_1$ has effectively clocked 30 times with feedback closed, $LFSR_2$ has clocked 24 times, $LFSR_3$ has clocked 22 times, and $LFSR_4$ has effectively clocked 16 times with feedback closed.
3. To mix initial data, continue to clock until 200 symbols have been produced with all switches closed ($t = 239$);
4. Keep blend registers $c_t$ and $c_{t-1}$, make a parallel load of the last 128 generated bits into the LFSRs according to Figure 4.6 at $t = 240$;

After the parallel load in item 4, the blend register contents shall be updated for each subsequent clock.

| **L** | deg | $g_1^{(L)}$ | deg | $g_2^{(L)}$ |
|---|---|---|---|---|
| 7 | [56] | 00000000 00000000 01000000 00000095 | [71] | 00000000 000000b3 f7fffce2 79f3a073 |
| 8 | [64] | 00000000 00000001 00000000 0000001b | [63] | 00000000 00000000 a1ab815b c7ec8025 |

| L | deg | $g_1^{(L)}$ | deg | $g_2^{(L)}$ |
|---|---|---|---|---|
| 9 | [72] | 00000000 00000100 00000000 00000609 | [49] | 00000000 00000000 0002c980 11d8b04d |
| 10 | [80] | 00000000 00010000 00000000 00000215 | [42] | 00000000 00000000 0000058e 24f9a4bb |
| 11 | [88] | 00000000 01000000 00000000 0000013b | [35] | 00000000 00000000 0000000c a76024d7 |
| 12 | [96] | 00000001 00000000 00000000 000000dd | [28] | 00000000 00000000 00000000 1c9c26b9 |
| 13 | [104] | 00000100 00000000 00000000 0000049d | [21] | 00000000 00000000 00000000 0026d9e3 |
| 14 | [112] | 00010000 00000000 00000000 0000014f | [14] | 00000000 00000000 00000000 00004377 |
| 15 | [120] | 01000000 00000000 00000000 000000e7 | [7] | 00000000 00000000 00000000 00000089 |
| 16 | [128] | 1 00000000 00000000 00000000 00000000 | [0] | 00000000 00000000 00000000 00000001 |

*Table 4.4: Polynomials used when creating K_session*[1]

[1]All polynomials are in hexadecimal notation. The LSB is in the rightmost position.

In Figure 4.5, all bits are shifted into the LFSRs, starting with the least significant bit (LSB). For instance, from the third octet of the address, BD_ADDR[2], first $BD_ADDR_{16}$ is entered, followed by $BD_ADDR_{17}$, etc. Furthermore, $CL_0$ corresponds to $CLK_1$,..., $CL_{25}$ corresponds to $CLK_{26}$.

Note: The output symbols $x_t^i$, $i = 1, ..., 4$ are taken from the positions 24, 24, 32, and 32 for $LFSR_1$, $LFSR_2$,$LFSR_3$, and $LFSR_4$, respectively (counting the leftmost position as number 1).

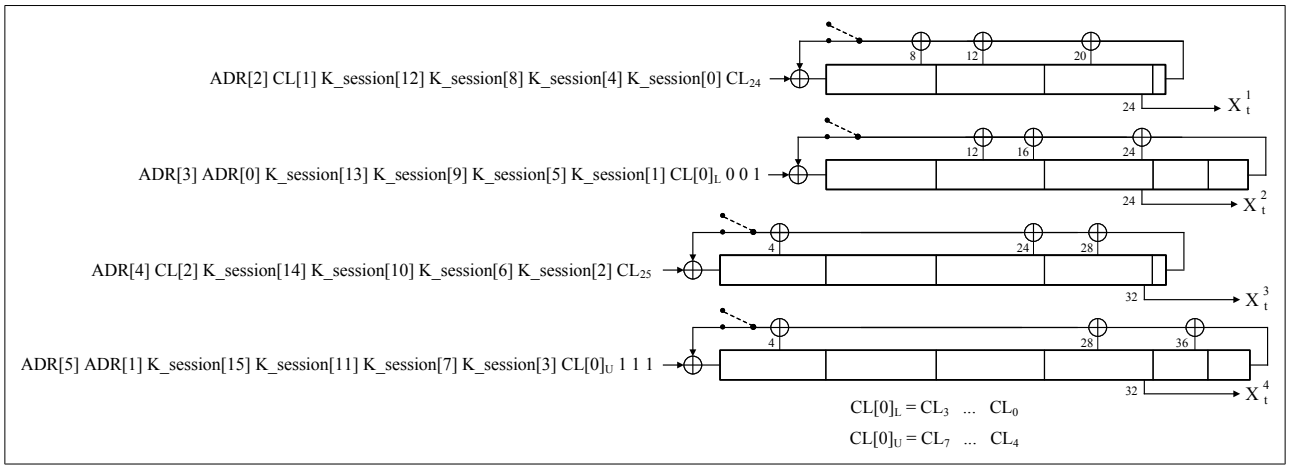



*Figure 4.5: Arranging the input to the LFSRs*

In Figure 4.6, the 128 binary output symbols $Z_0$,..., $Z_{127}$ are arranged in octets denoted Z[0],..., Z[15]. The LSB of Z[0] corresponds to the first of these symbols, the MSB of Z[15] is the last output from the generator. These bits shall be loaded into the LFSRs according to the figure. It is a parallel load and no update of the blend registers is done. The first output symbol is generated at the same time. The octets shall be written into the registers with the LSB in the leftmost position (i.e. the opposite of before). For example, $Z_{24}$ is loaded into position 1 of $LFSR_4$.

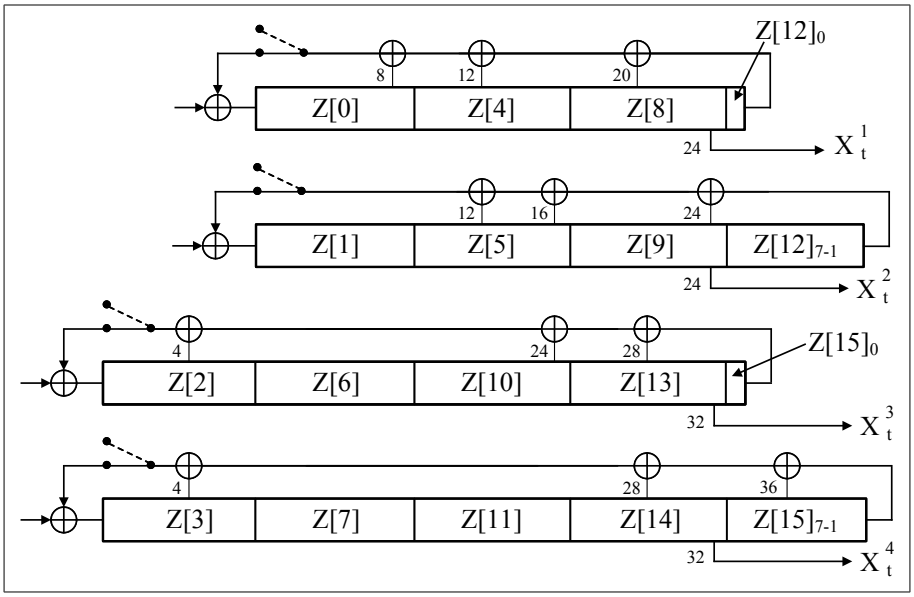



*Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs.*

## 4.6 Key stream sequence

When the initialization is finished, the output from the summation combiner is used for encryption/decryption. The first bit to use shall be the one produced at the parallel load, i.e. at $t = 240$. The circuit shall be run for the entire length of the current payload. Then, before the reverse direction is started, the entire initialization process shall be repeated with updated values on the input parameters.

Sample data of the encryption output sequence can be found in [Vol 2] Part G, Section 1.1. All implementations of encryption shall produce these encryption streams for the given initialization values.

# 5 AUTHENTICATION

Legacy authentication uses a challenge-response scheme in which a claimant's knowledge of a secret key is checked through a 2-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, for example K. In the challenge-response scheme the verifier challenges the claimant to authenticate a random input (the challenge), denoted by AU_RAND_A, with an authentication code, denoted by $E_1$, and return the result SRES to the verifier, see Figure 5.1. This figure also shows that the input to $E_1$ consists of the tuple AU_RAND_A and the Bluetooth Device Address (BD_ADDR_B) of the claimant. The use of this address prevents a simple reflection attack[1]. The secret $K$ shared by devices A and B is the current link key.

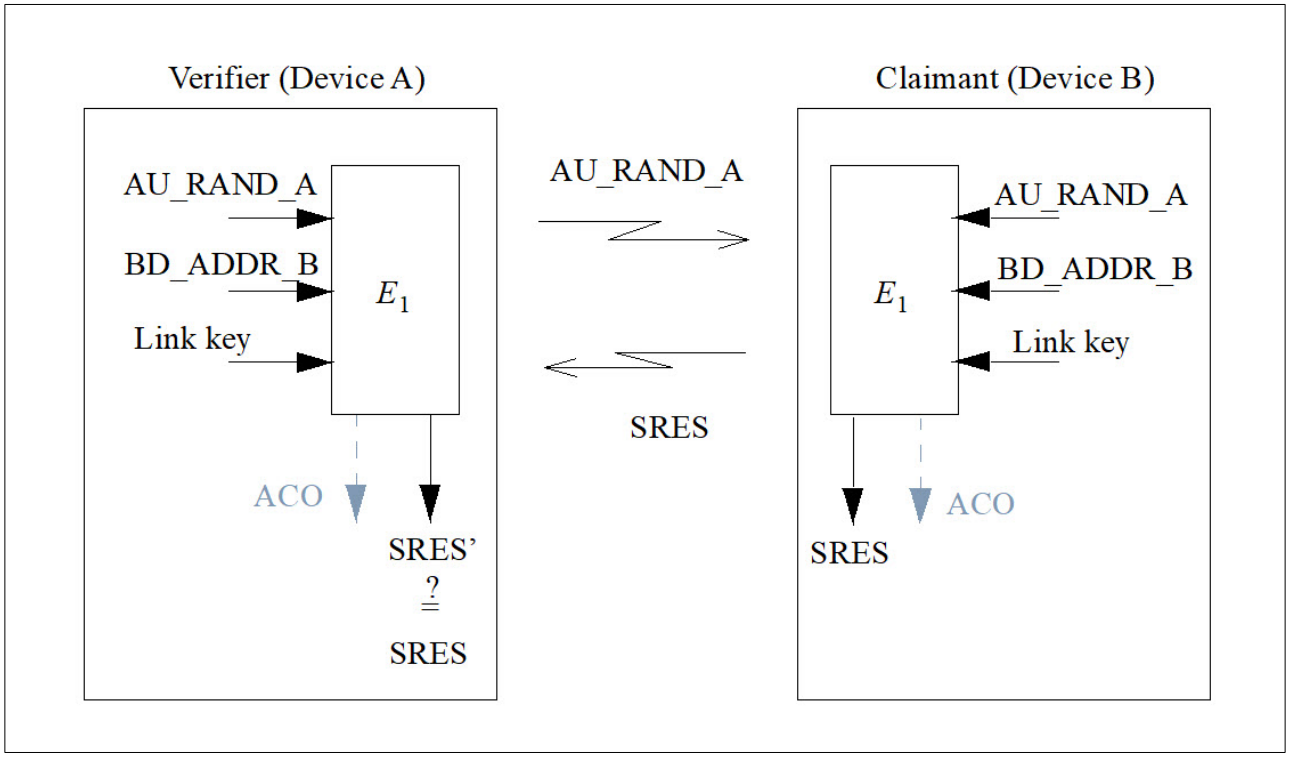



*Figure 5.1: Challenge-response for the Bluetooth.*

The challenge-response scheme for symmetric keys in legacy authentication is depicted in Figure 5.2.

[1]The reflection attack actually forms no threat because all service requests are dealt with on a FIFO basis. When preemption is introduced, this attack is potentially dangerous.

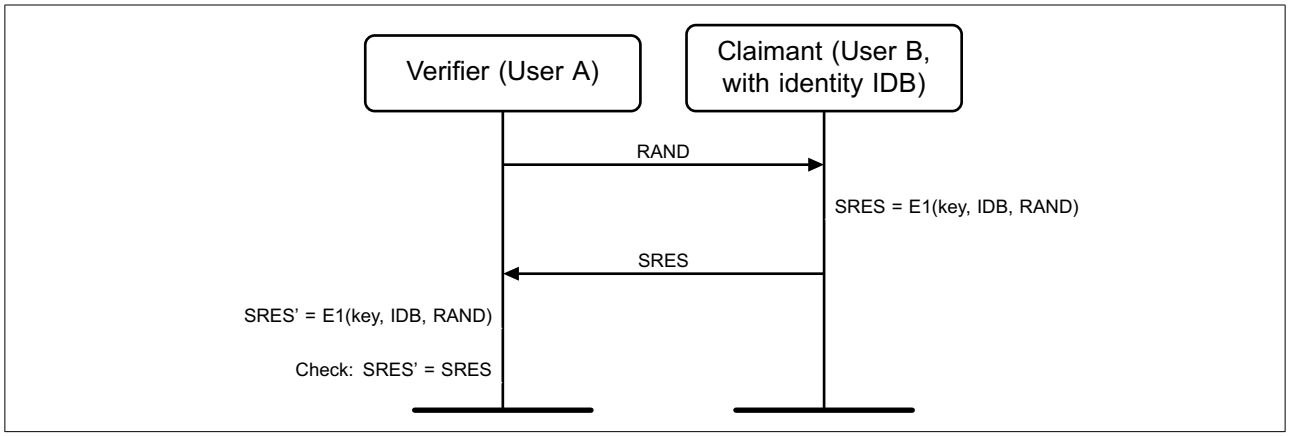



*Figure 5.2: Challenge-response for symmetric key systems.*

In legacy authentication, the verifier is not required to be the Central. The application indicates which device has to be authenticated. Some applications only require a one-way authentication. However, some peer-to-peer communications, should use a mutual authentication in which each device is subsequently the challenger (verifier) in two authentication procedures. The LM shall process authentication preferences from the application to determine in which direction(s) the authentication(s) takes place. For mutual authentication with the devices of Figure 5.1, after device A has successfully authenticated device B, device B could authenticate device A by sending an AU_RAND_B (different from the AU_RAND_A that device A issued) to device A, and deriving the SRES and SRES' from the new AU_RAND_B, the address of device A, and the link key $K_{AB}$.

If a legacy authentication is successful the value of ACO as produced by $E_1$ shall be retained.

Secure Authentication uses a challenge-response scheme in which both devices act as a verifier and claimant in the same sequence where the knowledge of a secret key is checked through a 4-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, for example K. In the challenge-response scheme the Central challenges the Peripheral to authenticate a random input (the challenge), denoted by AU_RAND_C, with an authentication code, denoted by h4 and h5, and return the resulting SRES_P to the verifier, see Figure 5.3. Similarly, the Peripheral challenges the Central to authenticate a random input (the challenge), denoted AU_RAND_P, with an authentication code, denoted by h4 and h5, and return the resulting SRES_C to the verifier. This figure also shows that the inputs to h4 and h5 consist of a secret, a string “btdk”, the Bluetooth Device Address of the Central (BD_ADDR_C), and the Bluetooth Device Address of the Peripheral (BD_ADDR_P). The use of these addresses prevents a simple reflection attack. The secret *K* shared by the Central and Peripheral is the current link key.

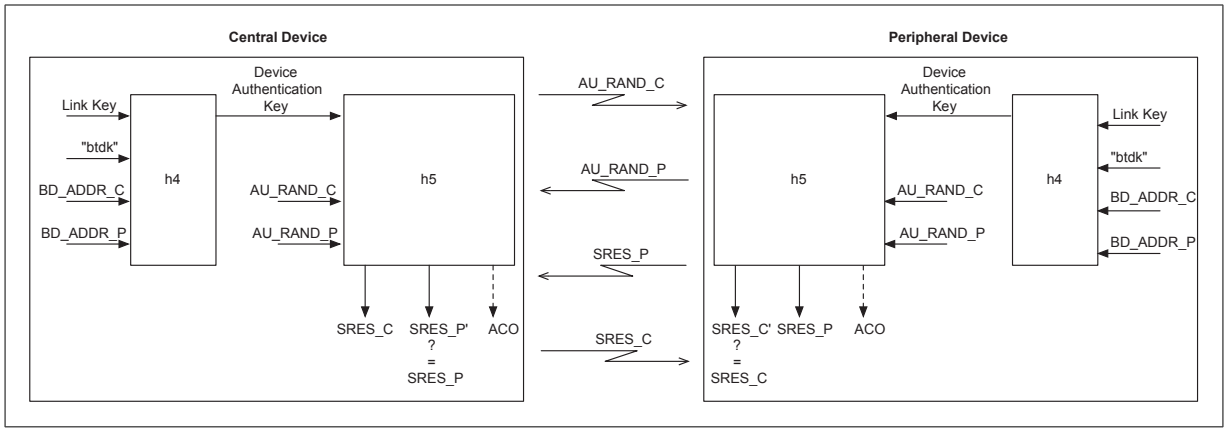



*Figure 5.3: Challenge and response for secure authentication.*

The challenge-response scheme for symmetric keys in secure authentication when initiated by the Central is depicted in Figure 5.4.

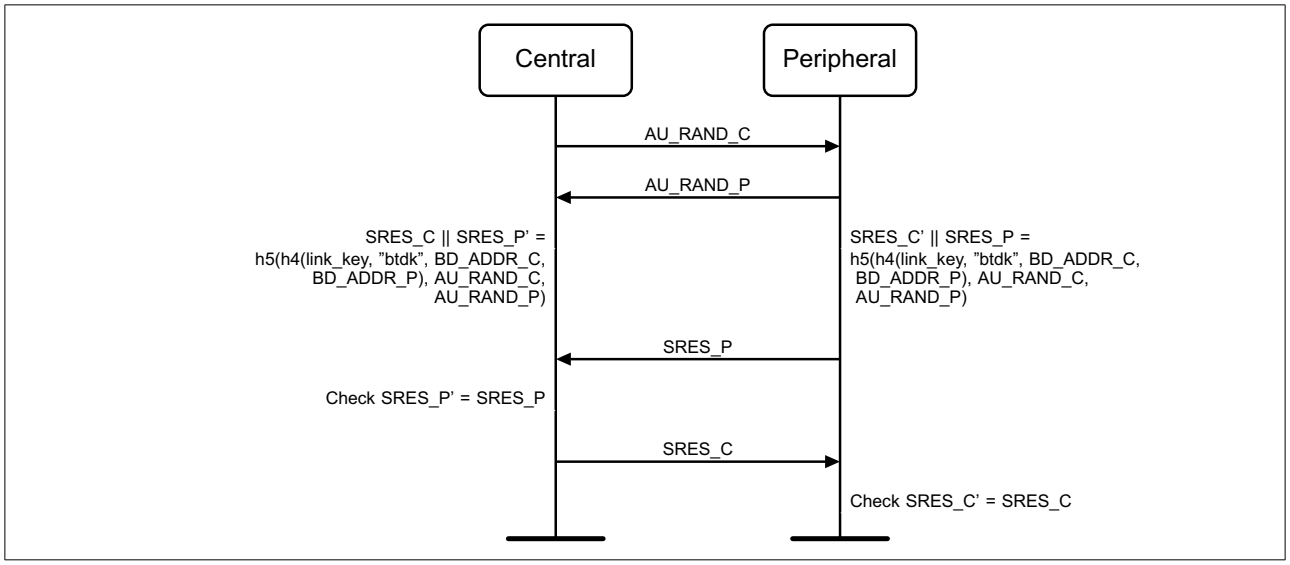



*Figure 5.4: Challenge-response for secure authentication.*

For the purposes of this authentication protocol the roles used are those applying when the initiating device sends its AU_RAND to the responder, irrespective of whether a role switch happens during the protocol. The device that sends AU_RAND_C shall check SRES_P and the device that sends AU_RAND_P shall check SRES_C. Alternatively, the device may reject the request for a role switch during the protocol.

In secure authentication, both the Central and Peripheral are the verifier and claimant. The application indicates which device has to be authenticated, but secure authentication is always a mutual authentication.

If a secure authentication is successful the value of ACO as produced by h5 shall be retained.

## 5.1 Repeated attempts

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure, the waiting interval shall be increased exponentially. For example, after each failure, the waiting interval before a new attempt can be made could be twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure restricts the rate at which an intruder can repeat the authentication procedure with different keys.

A device shall generate a new public/private key-pair after every pairing (successful or failed).

[1]Another appropriate value larger than 1 may be used.

# 6 THE AUTHENTICATION AND KEY-GENERATING FUNCTIONS

This section describes the algorithms used for authentication and key generation.

## 6.1 The authentication function $E_1$

The authentication function $E_1$ is a computationally secure authentication code. $E_1$ uses the encryption function SAFER+. The algorithm is an enhanced version of an existing 64-bit block cipher SAFER-SK128, and it is freely available. In the following discussion, the block cipher will be denoted as the function $A_r$ which maps using a 128-bit key, a 128-bit input to a 128-bit output, i.e.

$$A_r: \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$
$$(k \times x) \mapsto t\,. \qquad \text{(EQ 11)}$$

The details of $A_r$ are given in the next section. The function $E_1$ is constructed using $A_r$ as follows

$$E_1: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32} \times \{0,1\}^{96}$$
$$(K, \text{RAND}, \text{address}) \mapsto (\text{SRES}, \text{ACO}), \qquad \text{(EQ 12)}$$

where $\text{SRES} = Hash(K, \text{RAND}, \text{address}, 6)[0, \ldots, 3]$, where $Hash$ is a keyed hash function defined as[1]

$$Hash: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{128}$$
$$(K, I_1, I_2, L) \mapsto A'_r\left(\left[\widetilde{K}\right], [E(I_2, L) +_{16} (A_r(K, I_1) \oplus I_1)]\right), \qquad \text{(EQ 13)}$$

and where

$$E = \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{8 \times 16}$$
$$(X[0, \ldots, L-1], L) \mapsto (X[i(mod\ \text{L}\ )] \text{ for } i = 0 \ldots 15), \qquad \text{(EQ 14)}$$

is an expansion of the $L$ octet word $X$ into a 128-bit word. The function $A_r$ is evaluated twice for each evaluation of $E_1$. The key $\widetilde{K}$ for the second use of $A_r$ (actually $A'_r$) is offset from $K$ as follows[2]

[1] In this section and in Figure 6.3, the operator $+_{16}$ denotes addition *mod* 256 of each of the 16 octets separately.

[2] The constants are the largest primes below 257 for which 10 is a primitive root.

$$
\begin{aligned}
\tilde{K}\left[0\right] &= (K[0] + 233) \mod 256, & \tilde{K}[1] &= K[1] \oplus 229, \\
\tilde{K}\left[2\right] &= (K[2] + 223) \mod 256, & \tilde{K}[3] &= K[3] \oplus 193, \\
\tilde{K}\left[4\right] &= (K[4] + 179) \mod 256, & \tilde{K}[5] &= K[5] \oplus 167, \\
\tilde{K}\left[6\right] &= (K[6] + 149) \mod 256, & \tilde{K}[7] &= K[7] \oplus 131, \\
\tilde{K}\left[8\right] &= K[8] \oplus 233, & \tilde{K}[9] &= (K[9] + 229) \mod 256, \\
\tilde{K}\left[10\right] &= K[10] \oplus 223, & \tilde{K}[11] &= (K[11] + 193) \mod 256, \\
\tilde{K}\left[12\right] &= K[12] \oplus 179, & \tilde{K}[13] &= (K[13] + 167) \mod 256, \\
\tilde{K}\left[14\right] &= K[14] \oplus 149, & \tilde{K}[15] &= (K[15] + 131) \mod 256,
\end{aligned}
\qquad \text{(EQ 15)}
$$

A data flowchart of the computation of $E_1$ is shown in Figure 6.1. $E_1$ is also used to deliver the parameter ACO (Authenticated Ciphering Offset) that is used in the generation of the ciphering key by $E_3$, see equations (EQ 3) and (EQ 23). The value of ACO is formed by octets 4 to 15 of the output of the hash function defined in (EQ 13):

$$\text{ACO} = Hash(K, \text{RAND}, \text{address}, 6)[4, \ldots, 15]. \qquad \text{(EQ 16)}$$

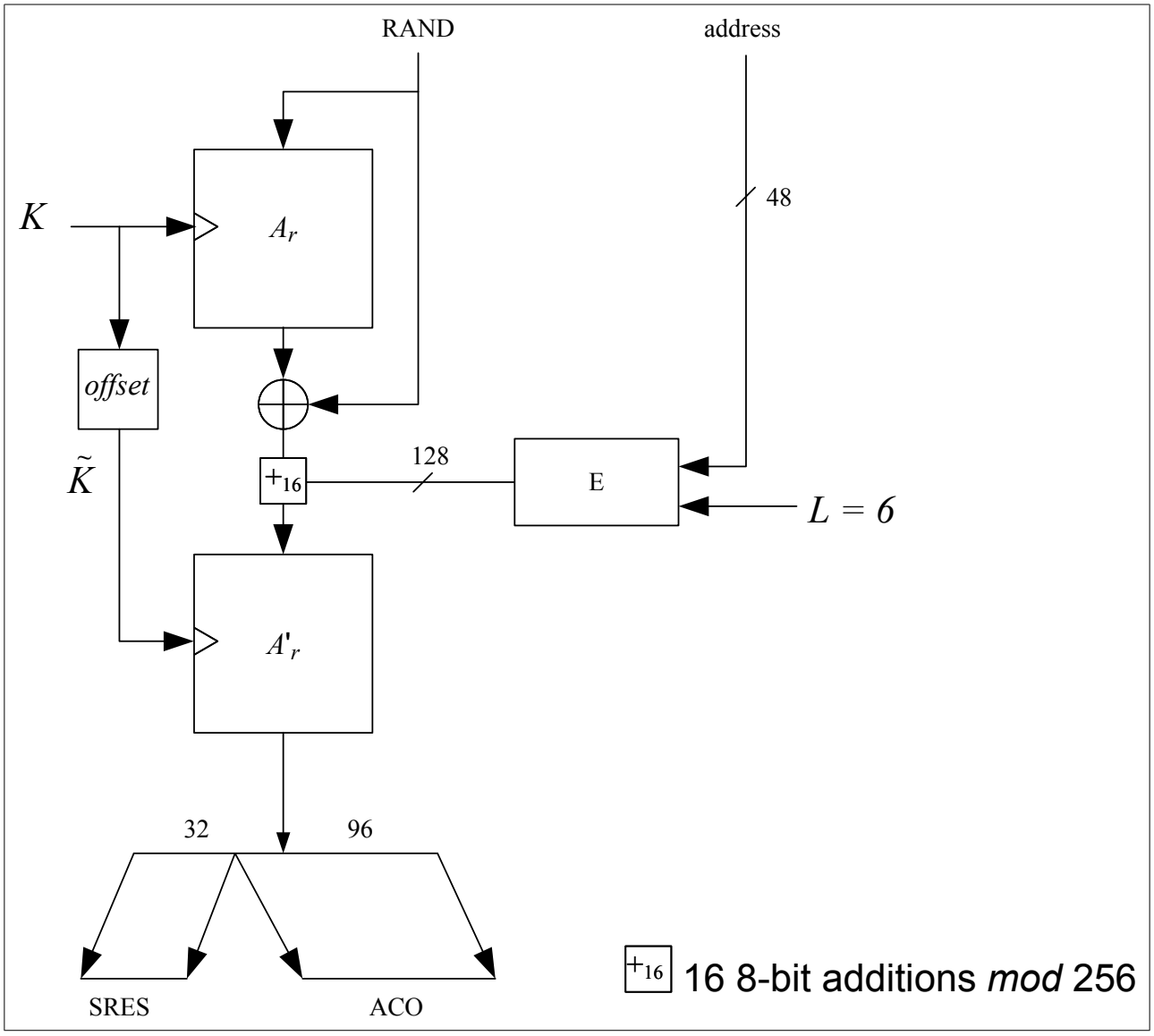



*Figure 6.1: Flow of data for the computation of $E_1$*

## 6.2 The functions $A_r$ and $A'_r$

The function $A_r$ is identical to SAFER+. It consists of a set of 8 layers, (each layer is called a round) and a parallel mechanism for generating the sub keys $K_p[j], p = 1, 2, \ldots, 17$, which are the round keys to be used in each round. The function will produce a 128-bit result from a 128-bit random input string and a 128-bit key. Besides the function $A_r$, a slightly modified version referred to as $A'_r$ is used in which the input of round 1 is added to the input of round 3. This is done to make the modified version non-invertible and prevents the use of $A'_r$ (especially in $E_{2x}$) as an encryption function. See Figure 6.2 for details.

### 6.2.1 The round computations

The computations in each round are a composition of encryption with a round key, substitution, encryption with the next round key, and, finally, a Pseudo Hadamard Transform (PHT). The computations in a round shall be as shown in Figure 6.2. The sub keys for round *r*,*r* = 1, 2, ..., 8 are denoted $K_{2r-1}[j]$, $K_{2r}[j]$, $j$ = 0, 1, ..., 15. After the last round $K_{17}[j]$ is applied identically to all previous odd numbered keys.

### 6.2.2 The substitution boxes “e” and “l”

In Figure 6.2 two boxes are shown, marked “e” and “l”. These boxes implement the same substitutions as are used in SAFER+; i.e. they implement

$$\begin{aligned} e, l &: \{0, \ldots, 255\} \rightarrow \{0, \ldots, 255\}, \\ e &: i \mapsto \left(45^i \ mod \ 257\right) \ mod \ 256, \\ l &: i \mapsto j \text{ s.t. } i = e(j). \end{aligned}$$

Their role, as in the SAFER+ algorithm, is to introduce non-linearity.

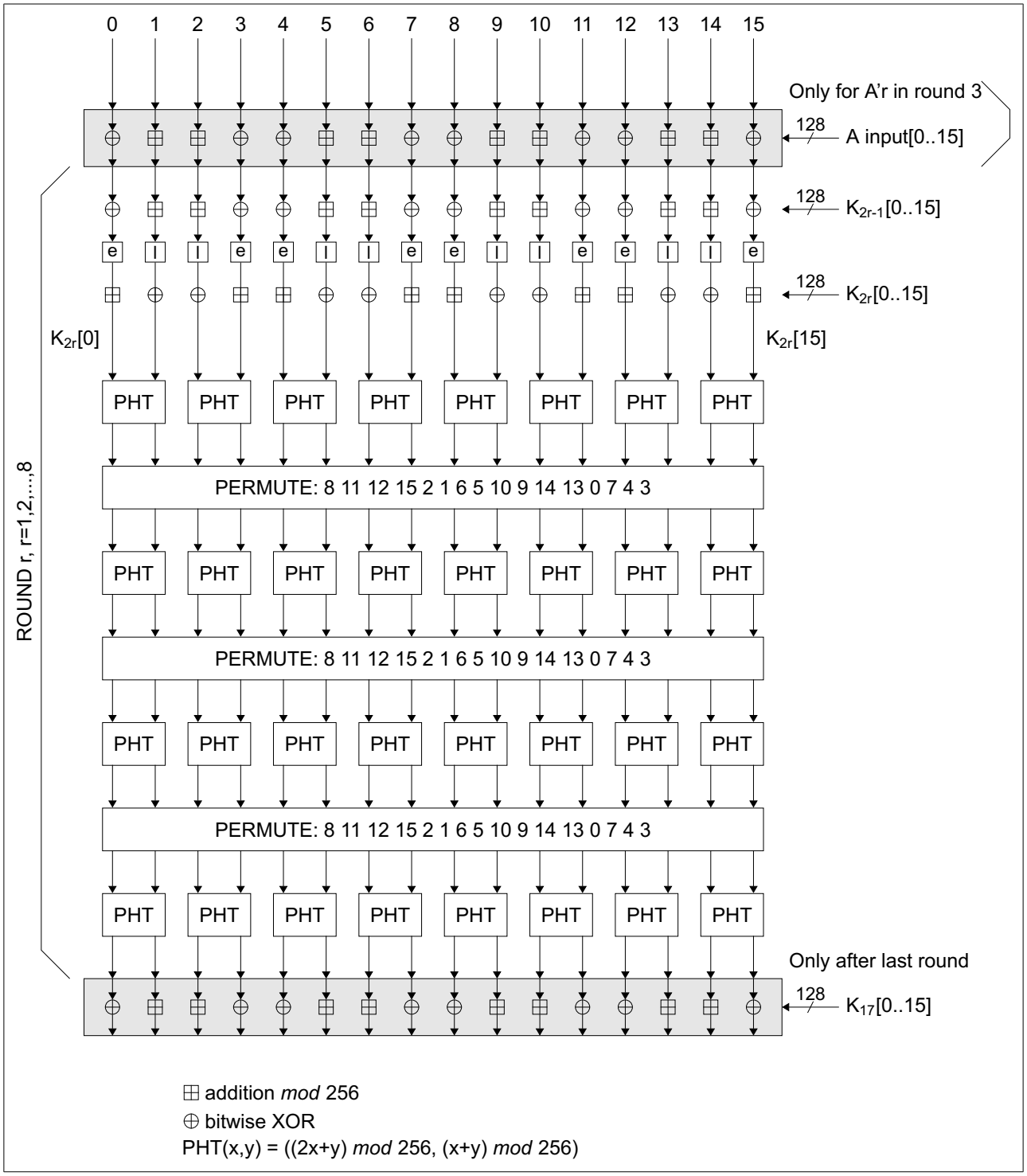



*Figure 6.2: One round in $A_r$ and $A'_r$*

In Section 6.2, the permutation boxes show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, etc.

### 6.2.3 Key scheduling

In each round, 2 batches of 16 octet-wide keys are needed. These round keys are derived as specified by the key scheduling in SAFER+. Figure 6.3 gives an overview of how the round keys $K_p[j]$ are determined. The bias vectors $B_2$, $B_3$, ..., $B_{17}$ shall be computed according to following equation:

$$B_p[i] = \left(\left(45^{\left(45^{17p+i+1} \bmod 257\right)} \bmod 257\right) \bmod 256\right), \text{for } i = 0, \ldots, 15. \quad \text{(EQ 17)}$$

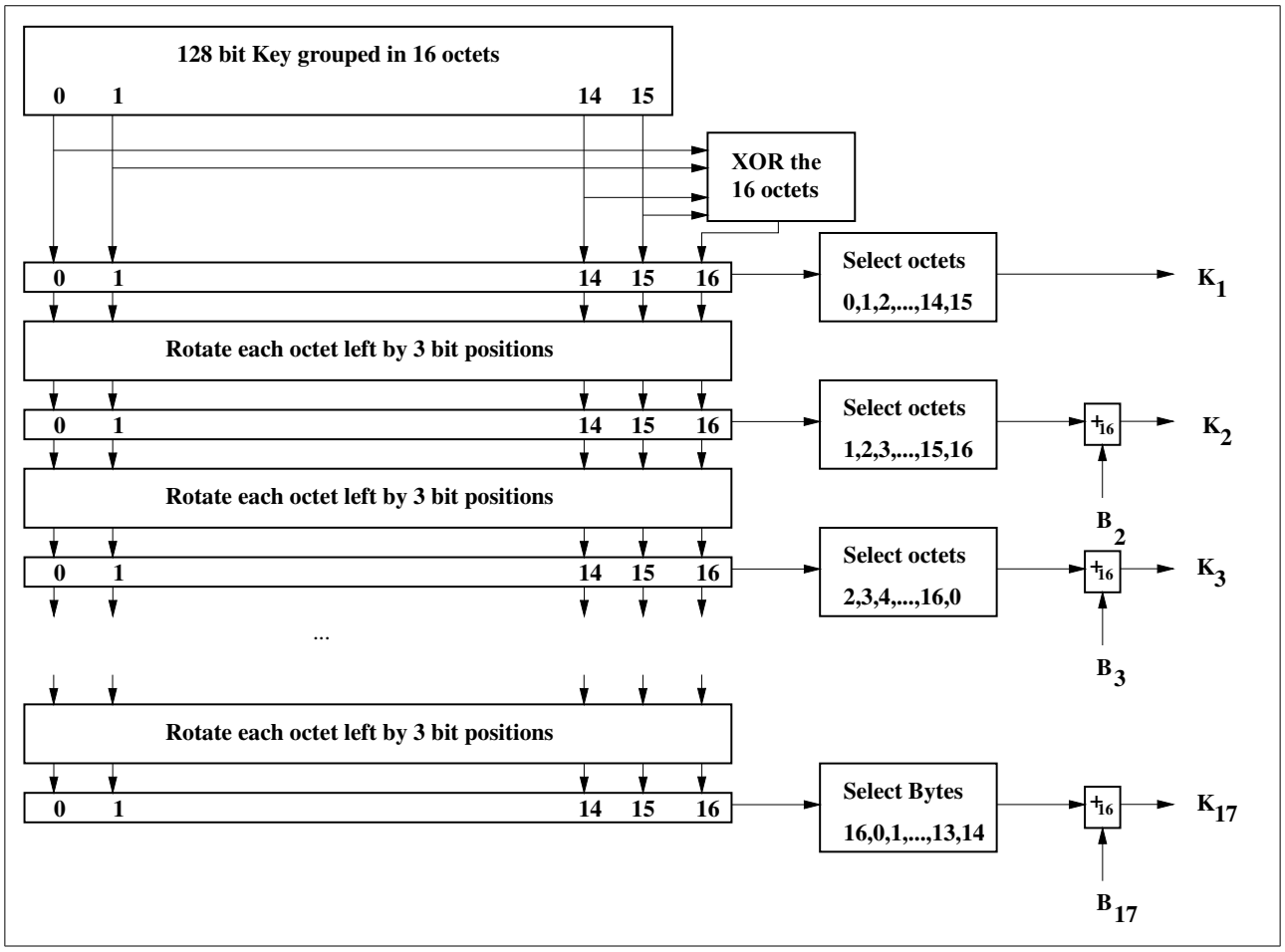



*Figure 6.3: Key scheduling in $A_r$*

## 6.3 $E_2$-key generation function for authentication

The key used for authentication shall be derived through the procedure that is shown in Figure 6.4. The figure shows two modes of operation for the algorithm. In the first mode, $E_{21}$ produces a 128-bit link key, $K$, using a 128-bit RAND value and a 48-bit address. This mode shall be used when creating combination keys. In the second mode, $E_{22}$ produces a 128-bit link key, $K$, using a 128-bit RAND value and an $L$ octet

user PIN. The second mode shall be used to create the initialization key, and also when a temporary link key is to be generated.

When the initialization key is generated, the PIN is augmented with the BD_ADDR, see Section 3.2.1 for which address to use. The augmentation shall always start with the least significant octet of the address immediately following the most significant octet of the PIN. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used.

This key generating algorithm again exploits the cryptographic function $E_2$. for mode 1 (denoted $E_{21}$) is computed according to following equations:

$$E_{21}: \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{128}$$
$$(\text{RAND}, \text{address}) \mapsto A'_r(X, Y) \quad \text{(EQ 18)}$$

where (for mode 1)

$$\begin{cases} X = \text{RAND} \oplus \left(6 \times 2^{120}\right) \\ Y = (\text{address} \parallel \text{address} \parallel \text{address})_{127:0} \end{cases} \quad \text{(EQ 19)}$$

Let $L$ be the number of octets in the user PIN. The augmenting is defined by

$$\text{PIN'} = \text{the L' least significant octets of (BD_ADDR} \parallel \text{PIN)} \quad \text{(EQ 20)}$$

where L' = min { 16, L+6 }.

Then, in mode 2, $E_2$ (denoted $E_{22}$) is

$$E_{22}: \{0,1\}^{8L'} \times \{0,1\}^{128} \times \{1,2,\ldots,16\} \rightarrow \{0,1\}^{128}$$
$$(\text{PIN'}, \text{RAND}, \text{L}') \mapsto A'_r(X, Y) \quad \text{(EQ 21)}$$

where

$$\begin{cases} X = (\text{PIN'} \parallel \text{PIN'} \parallel \text{PIN'})_{127:0} \\ Y = \text{RAND} \oplus \left(\text{L'} \times 2^{120}\right) \end{cases} \quad \text{(EQ 22)}$$

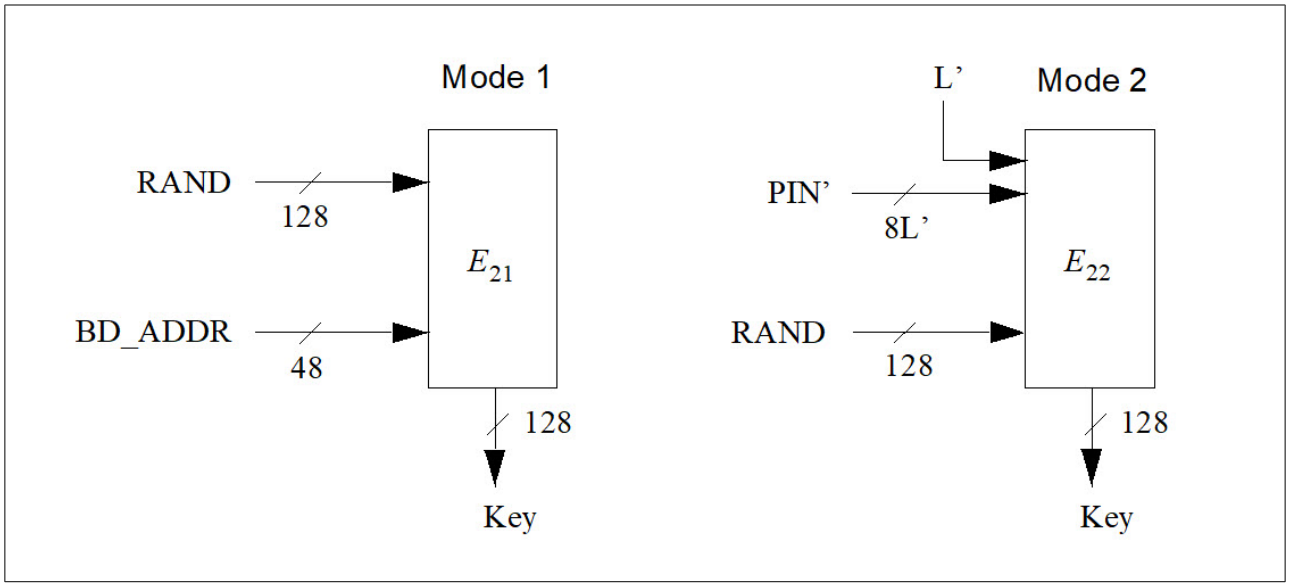



*Figure 6.4: Key generating algorithm $E_2$ and its two modes. Mode 1 is used for combination keys, while mode 2 is used for $K_{init}$ and K_temp*

## 6.4 $E_3$-key generation function for encryption

The ciphering key K_enc used by $E_0$ shall be generated by $E_3$. The function $E_3$ is constructed using $A'_r$ as follows

$$E_3: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{96} \rightarrow \{0,1\}^{128}$$
$$(K,\ \text{RAND},\ \text{COF}) \mapsto Hash(K,\ \text{RAND},\ \text{COF}, 12) \quad \text{(EQ 23)}$$

where *Hash* is the hash function as defined by (EQ 13). The key length produced is 128 bits. However, before use within $E_0$, the encryption key K_enc is shortened to the correct encryption key length, as described in Section 4.5. A block scheme of $E_3$ is depicted in Figure 6.5.

The value of COF is determined as specified by equation (EQ 3).

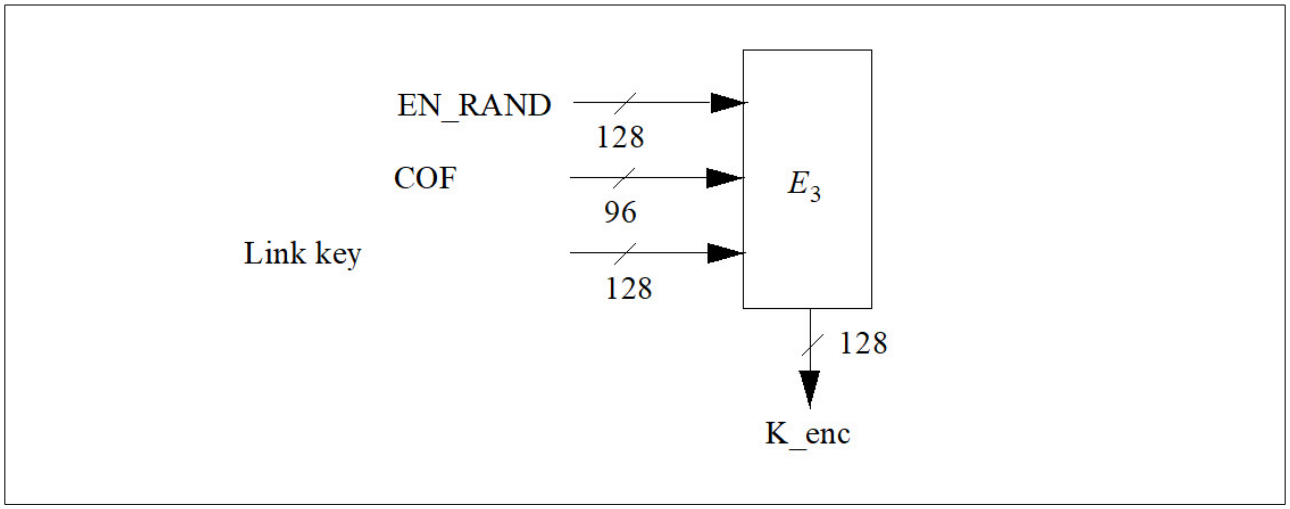



*Figure 6.5: Generation of the encryption key*

# 7 SECURE SIMPLE PAIRING

The Secure Simple Pairing security functions and procedures are described in this section. In addition, a cryptographic analysis of each procedure is provided.

There are five phases of Secure Simple Pairing:

- Phase 1: Public key exchange
- Phase 2: Authentication stage 1
- Phase 3: Authentication stage 2
- Phase 4: Link key calculation
- Phase 5: LMP Authentication and Encryption

Phases 1, 3, 4 and 5 are the same for all protocols whereas phase 2 (Authentication stage 1) is different depending on the protocol used. Distributed through these five phases are 13 steps as shown in Figure 7.1.

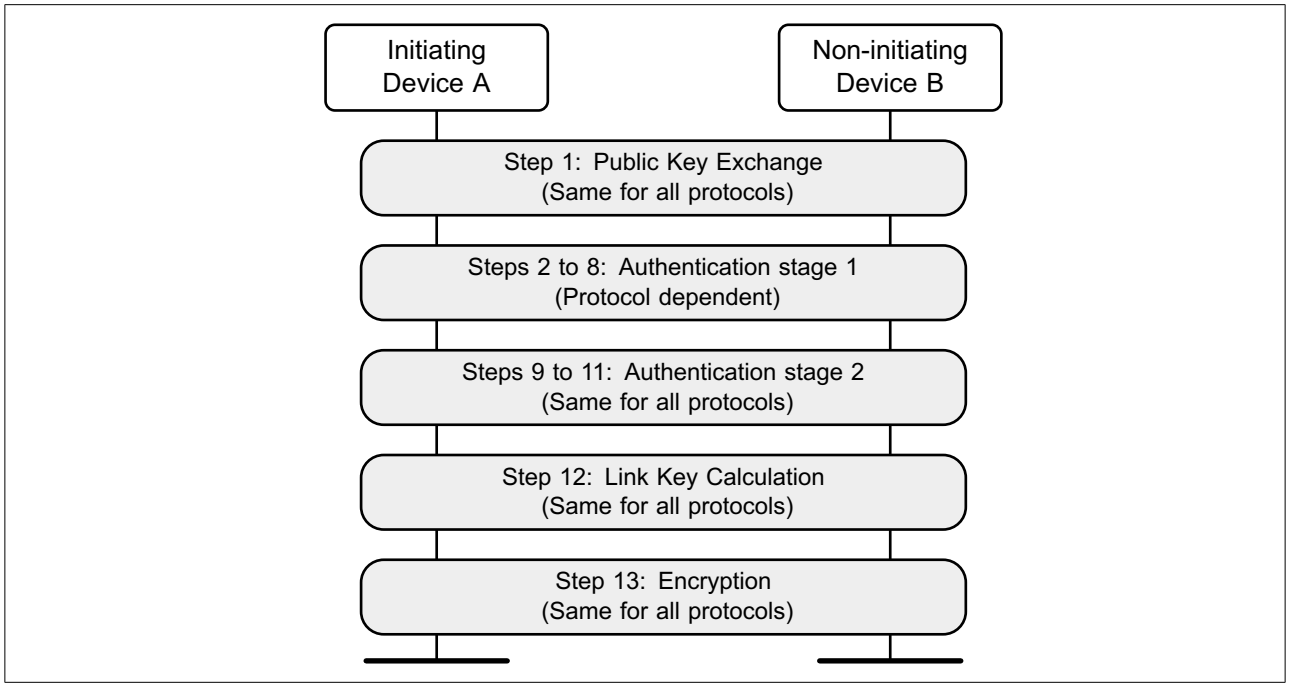



*Figure 7.1: Secure Simple Pairing security phases*

The terminology used in the security sections is defined in Table 7.1:

| Term | Definition |
|---|---|
| Cx | Commitment value from device X |
| Cxi | $i^{th}$ commitment value from device X. Only used in the passkey entry protocol |
| DHKey | Diffie Hellman key |
| Ex | Check value from device X |
| f1( ) | Used to generate the 128-bit commitment values Ca and Cb |
| f2( ) | Used to compute the link key and possible other keys from the DHKey and random nonces |
| f3( ) | Used to compute check values Ea and Eb in Authentication stage 2 |
| g( ) | Used to compute numeric check values |
| IOcapA | IO capabilities of device A |
| IOcapB | IO capabilities of device B |
| LK | Link Key |
| Nx | Nonce (unique random value) from device X |
| Nxi | $i^{th}$ nonce (unique random value) from device X. Only used in the passkey entry protocol |
| PKx | Public Key of device X |
| rx | Random value generated by device X |
| rxi | Bit i of the random value rx. Only used in the passkey entry protocol |
| SKx | Secret (Private) Key of device X |
| Vx | Confirmation value on device X. Only used in the numeric compare protocol. |
| X | BD_ADDR of device X |

*Table 7.1: Terminology*

## 7.1 Phase 1: Public key exchange

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). See Section 5.1 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b). If the two public keys have the same X coordinate and neither is the debug key (see [Vol 4] Part E, Section 7.6.4), each device shall fail the pairing process. These public keys are not regarded as secret although they may identify the devices.

When both devices' Controllers and Hosts support Secure Connections, the P-256 elliptic curve is used. When at least one device's Controller or Host doesn't support Secure Connections, the P-192 elliptic curve is used.

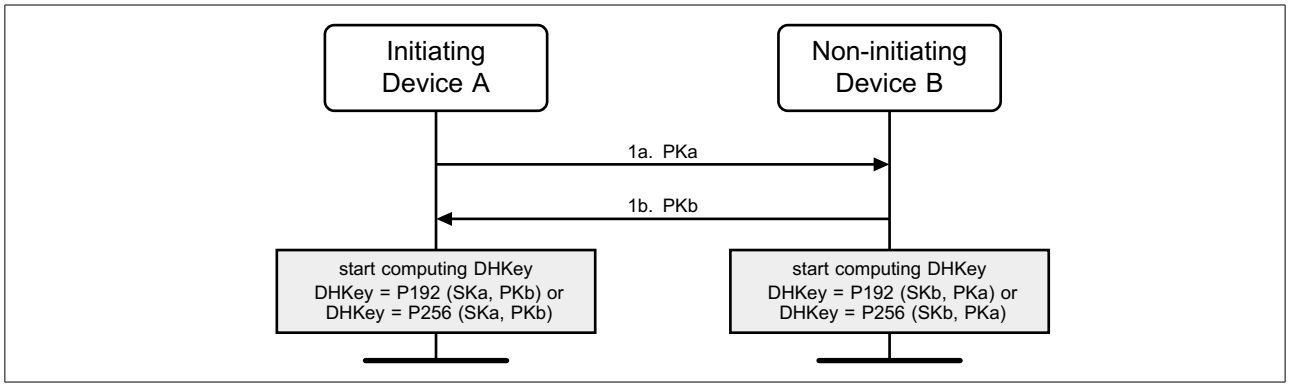



*Figure 7.2: Public key exchange details*

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-192 or P-256) - see Section 7.6.

## 7.2 Phase 2: Authentication stage 1

Authentication stage 1 has three different protocols: Numeric Comparison, Out-of-Band, and Passkey Entry. The Just Works association model shares the Numeric Comparison protocol and does not have a separate protocol.

The protocol is chosen based on the IO capabilities of the two devices.

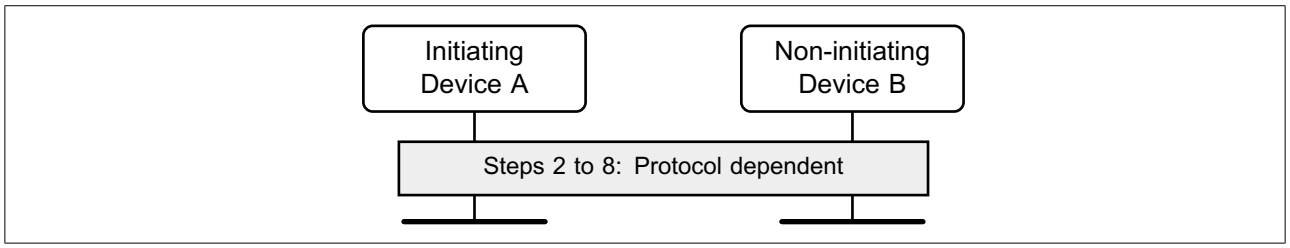



*Figure 7.3: Authentication stage 1 (high level)*

The three protocols are described in the following sections.

### 7.2.1 Authentication stage 1: Numeric Comparison protocol

The Numeric Comparison protocol provides limited protection against active "man-in-the-middle" (MITM) attacks as an active man-in-the-middle will succeed with a probability of 0.000001 on each iteration of the protocol. Provided that there is no MITM at the time the pairing is performed, the shared Link Key that is generated is computationally secure from even a passive eavesdropper that may have been present during the pairing.

The sequence diagram of Authentication stage 1 for the Numeric Comparison protocol from the cryptographic point of view is shown in Figure 7.4.

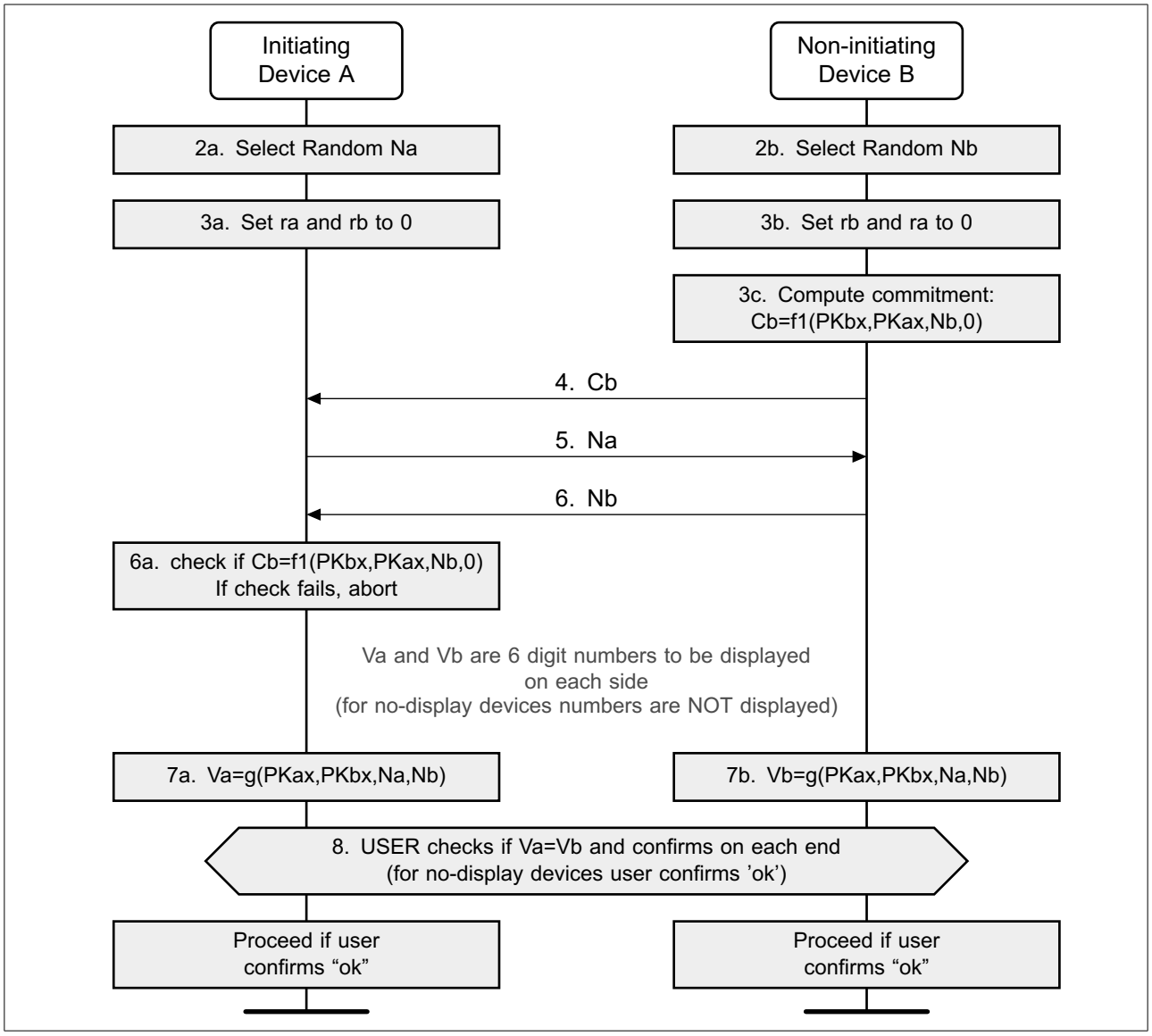



*Figure 7.4: Authentication stage 1: Numeric Comparison protocol details*

After the public keys have been exchanged, each device selects a random 128-bit nonce (step 2). This value is used to mitigate replay attacks and shall be freshly generated with each instantiation of the pairing protocol. This value shall be generated using a random number generator that meets the requirements of Section 2.

Following this the responding device then computes a commitment to the two public keys that have been exchanged and its own nonce value (step 3c). This commitment is computed as a one-way function of these values and is transmitted to the initiating device (step 4). The commitment prevents an attacker from changing these values at a later time.

The initiating and responding devices then exchange their respective nonce values (steps 5 and 6) and the initiating device confirms the commitment (step 6a). A failure at

this point indicates the presence of an attacker or other transmission error and causes the protocol to abort. The protocol may be repeated with or without the generation of new public-private key pairs, but, as stated above, new nonces must be generated if the protocol is repeated.

Assuming that the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices (steps 7a, 7b, and 8). The user is expected to check that these 6-digit values match and to confirm if there is a match. If there is no match, the protocol aborts and, as before, new nonces must be generated if the protocol is to be repeated.

An active MITM must inject its own key material into this process to have any effect other than denial-of-service. A simple MITM attack will result in the two 6-digit display values being different with probability 0.999999. A more sophisticated attack may attempt to engineer the display values to match, but this is thwarted by the commitment sequence. If the attacker first exchanges nonces with the responding device, it must commit to the nonce that it will use with the initiating device before it sees the nonce from the initiating device. If the attacker first exchanges nonces with the initiating device, it must send a nonce to the responding device before seeing the nonce from the responding device. In each case, the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices. It therefore cannot choose its own nonces in such a way as to cause the display values to match.

### 7.2.2 Authentication stage 1: Out of Band protocol

The Out-of-Band protocol is used when authentication information has been received by at least one of the devices and indicated in the OOB Authentication Data Present parameter in the LMP IO capability exchange sequence. The mode in which the discovery of the peer device is first done in-band and then followed by the transmission of authentication parameters through OOB interface is not supported. The sequence diagram for Authentication stage 1 for Out of Band from the cryptographic point of view is shown in Figure 7.5.

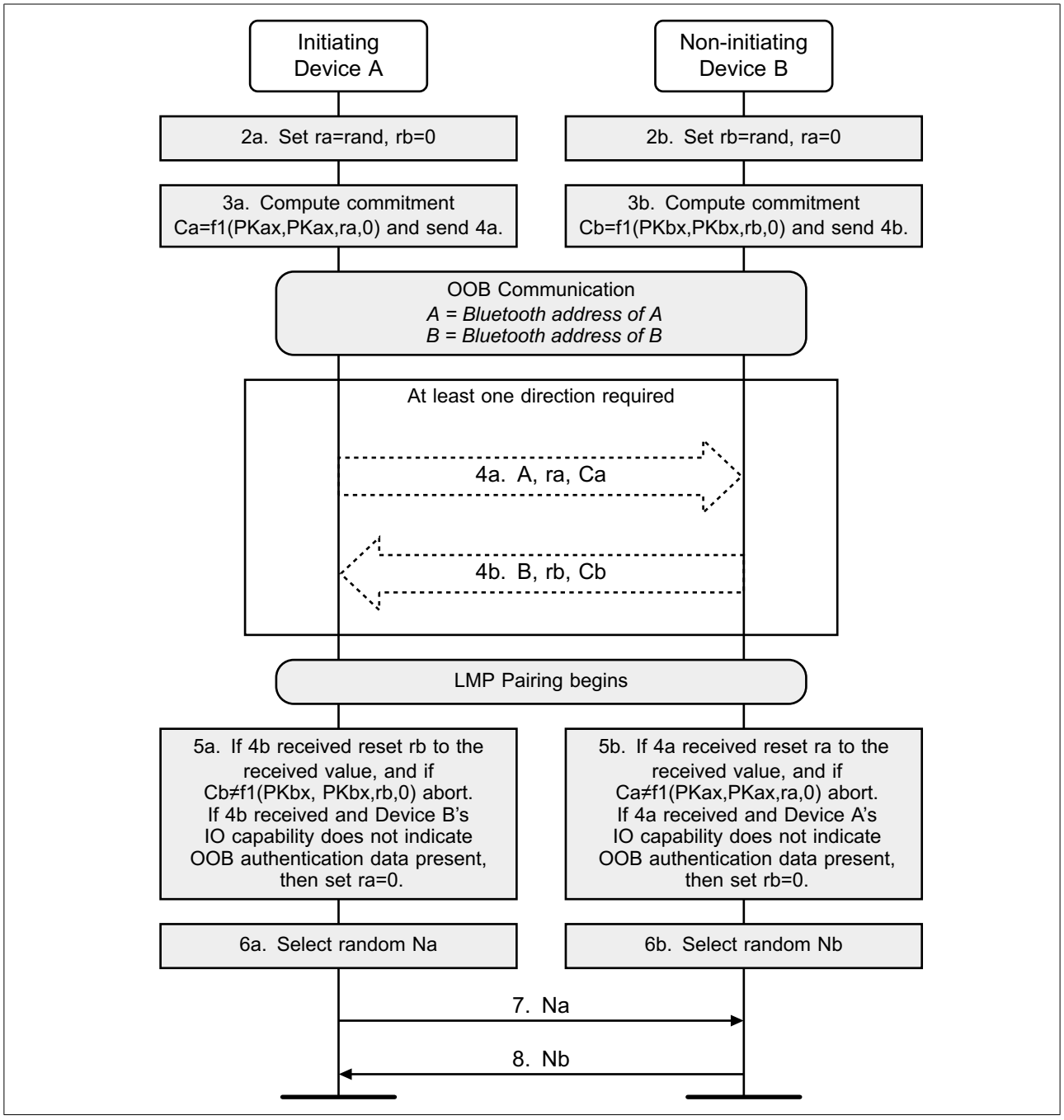



*Figure 7.5: Authentication stage 1: Out of Band details*

**Principle of operation**: If both devices can transmit and/or receive data over an out-of-band channel, then mutual authentication will be based on the commitments of the public keys (Ca and Cb) exchanged OOB in Authentication stage 1.

If OOB communication is possible only in one direction, then authentication of the device sending the OOB communication will be based on the commitment of its public key (Ca or Cb) while authentication of the device receiving the OOB communication

will be based on that device knowing a random number r sent via OOB. In this case, mutual authentication can only be obtained if r is secret: r can be created afresh every time, or access to the device sending r must be restricted. If r is not secret, then mutual authentication cannot be achieved; only the device receiving the OOB information in step 4a or 4b can authenticate the peer. If r is not sent by a device, it is assumed to be 0 by the device receiving the OOB information in step 4a or 4b.

**Roles of A and B**: The OOB Authentication stage 1 protocol is symmetric with respect to the roles of A and B. It does not require that device A always will initiate pairing and it automatically resolves asymmetry in the OOB communication, e.g. in the case when one of the devices has an NFC tag and can only transmit OOB.

**Order of steps**: The public key exchange must happen before the verification step 5. In the diagram the in-band public key exchange between the devices (step 1) is done before the OOB communication (step 4). But when the pairing is initiated by an OOB interface, public key exchange will happen after the OOB communication (step 1 will be between steps 4 and 5).

Values of ra and rb: Since the direction of the peer's OOB interface cannot be verified before the OOB communication takes place, a device shall always generate and if possible transmit through its OOB interface a random number r to the peer. Each device applies the following rules locally to set the values of its own r and the value of the peer's r:

1. Initially, r of the device is set to a random number and r of the peer is set to 0 (step 2).
2. If a device has received OOB, it sets the peer's r value to what was sent by the peer (Step 5).
3. If the remote device's OOB Authentication Data parameter sent in the LMP IO capabilities exchange sequence is set to No OOB Data Received, it sets its own r value to 0 (Step 5)

These rules ensure that when entering Authentication stage 2, both A and B have the same values for ra and rb if OOB communication took place.

### 7.2.3 Authentication stage 1: Passkey Entry protocol

The Passkey Entry protocol is used when LMP IO capability exchange sequence indicates that Passkey Entry shall be used.

The sequence diagram for Authentication stage 1 for Passkey Entry from the cryptographic point of view is shown in Figure 7.6.

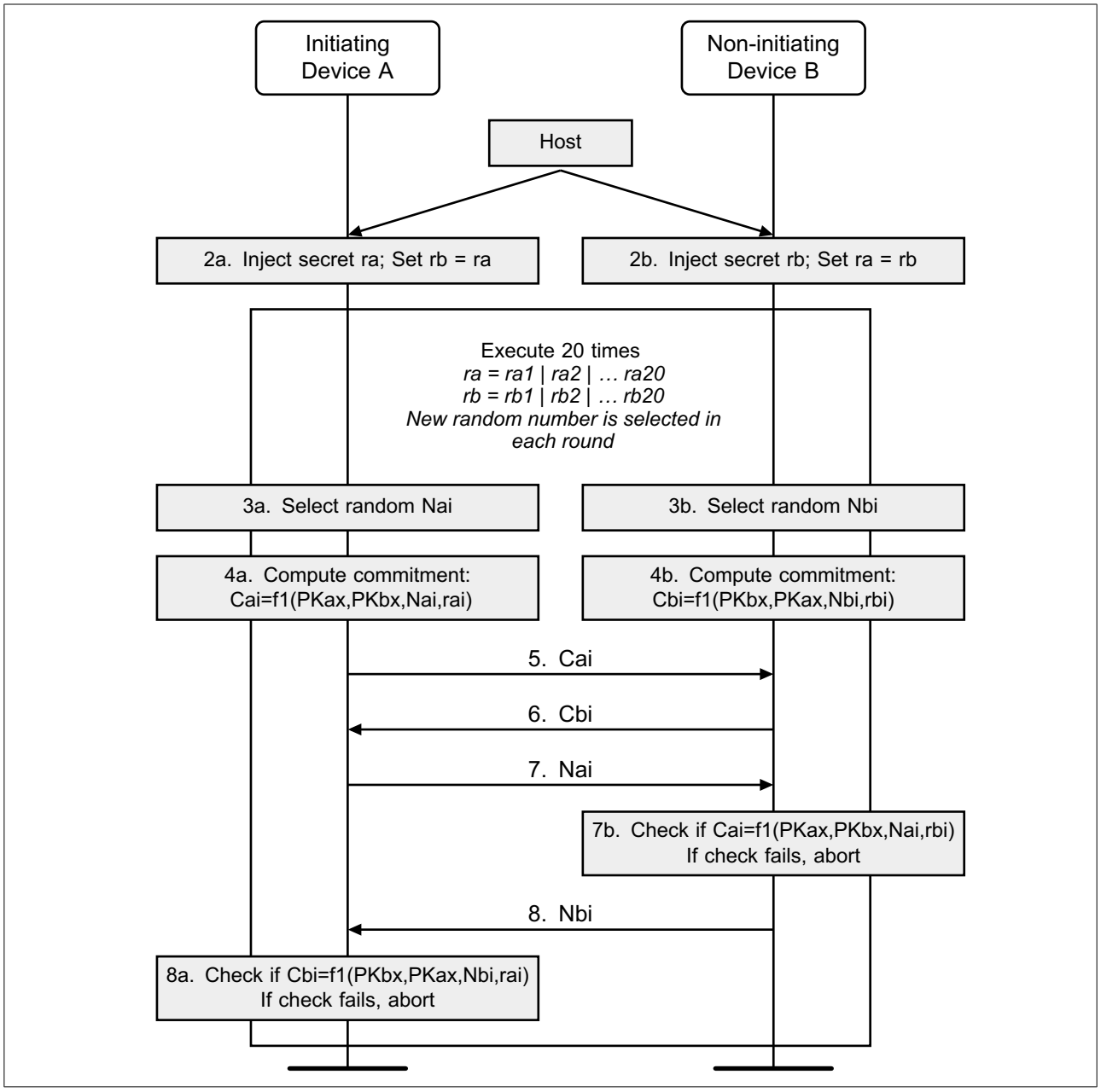



*Figure 7.6: Authentication stage 1: Passkey Entry details*

The user inputs an identical Passkey into both devices. Alternately, the Passkey may be generated and displayed on one device, and the user then inputs it into the other (step 2). This short shared key will be the basis of the mutual authentication of the devices. A new Passkey shall be generated randomly for each pairing procedure.

Steps 3 to 8 are repeated 20 times since a 6-digit Passkey is 20 bits (999999=0xF423F). If the device allows a shorter passkey to be entered, it shall be prefixed with zeros (e.g. “1234” is equivalent to “001234”).

In Steps 3 to 8, each side commits to each bit of the Passkey, using a long nonce (128 bits), and sending the hash of the nonce, the bit of the Passkey, and both public keys to the other party. The parties then take turns revealing their commitments until the entire Passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the Passkey effectively reveals that bit of the Passkey in the process, but the other party then has to reveal the corresponding commitment to show the same bit value for that bit of the Passkey, or else the first party will then abort the protocol, after which no more bits of the Passkey are revealed.

This "gradual disclosure" prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a MITM attack. A MITM attacker with only partial knowledge of the Passkey will only receive one incorrectly-guessed bit of the Passkey before the protocol fails. Hence, a MITM attacker who engages first one side, then the other will only gain an advantage of at most two bits over a simple brute-force guesser, making the probability of success 0.000004 instead of 0.000001.

The long nonce is included in the commitment hash to make it difficult to brute-force even after the protocol has failed. The public Diffie-Hellman values are included to tie the Passkey protocol to the original ECDH key exchange, to prevent a MITM from substituting the attacker's public key on both sides of the ECDH exchange in standard MITM fashion.

At the end of this stage, Na is set to Na20 and Nb is set to Nb20 for use in Authentication stage 2.

## 7.3 Phase 3: Authentication stage 2

The second stage of authentication then confirms that both devices have successfully completed the exchange. This stage is identical in all three protocols and is shown in Figure 7.7.

Each device computes a new confirmation value that includes the previously exchanged values and the newly derived shared key (step 9). The initiating device then transmits its confirmation value which is checked by the responding device (step 10). If this check fails, it indicates that the initiating device has not confirmed the pairing, and the protocol shall be aborted. The responding device then transmits its confirmation value which is checked by the initiating device (step 11). A failure indicates that the responding device has not confirmed the pairing and the protocol should abort.

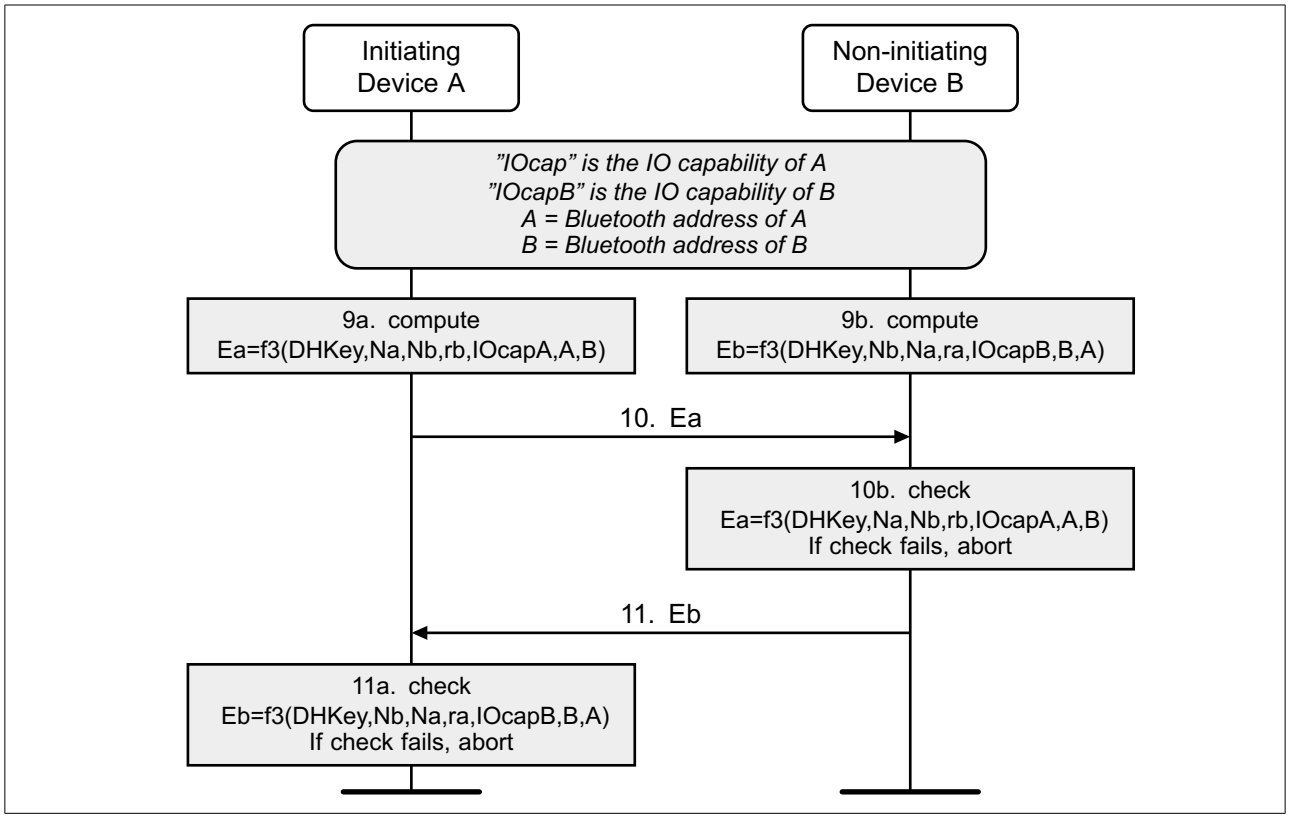



*Figure 7.7: Authentication stage 2*

## 7.4 Phase 4: Link key calculation

Once both sides have confirmed the pairing, a link key is computed from the derived shared key and the publicly exchanged data (step 12). The nonces ensure the freshness of the key even if long-term ECDH values are used by both sides. This link key is used to maintain the pairing.

When computing the link key both parties shall input the parameters in the same order (shown in Figure 7.8) to ensure that both devices compute the same key: Nc is whichever of Na and Nb was generated by the Central and Np is the other, while BD_ADDR_C and BD_ADDR_P are the addresses of the Central and Peripheral respectively.

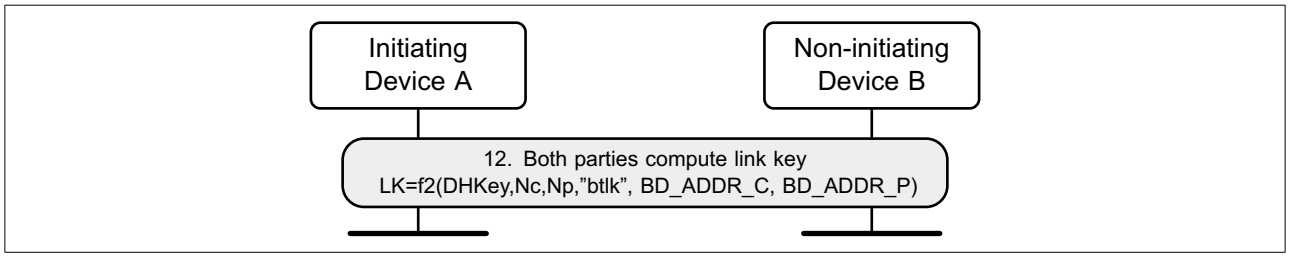



*Figure 7.8: Link key calculation*

## 7.5 Phase 5: LMP authentication and encryption

The final phase in Secure Simple Pairing consists of authentication and generation of the encryption key. This is the same as the final steps in legacy pairing.

## 7.6 Elliptic curve definition

Secure Simple Pairing supports two elliptic curves: P-192 and P-256.

Secure Simple Pairing uses the FIPS P-192 and P-256 curves defined in FIPS 186-4[1]. Elliptic curves are specified by $p$, $a$, $b$ and are of the form:

$$E: y^2 = x^3 + ax + b \ (mod\ p)$$

In NIST P-192 and P-256, $a$ is -3 and the following parameters are given:

- The prime modulus $p$, order $r$, base point x-coordinate $Gx$, base point y-coordinate $Gy$.
- The integers $p$ and $r$ are given in decimal form; bit strings and field elements are given in hex.

**For P-192:**

```
p =6277101735386680763835789423207666416083908700390324961279
r =6277101735386680763835789423176059013767194773182842284081
b =64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
Gx=188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
Gy=07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
```

The function P192 is defined as follows. Given an integer $u$, $0 < u < r$, and a point $V$ on the curve $E$, the value P192($u$,$V$) is computed as the x-coordinate of the $u^{th}$ multiple $uV$ of the point $V$.

The private keys shall be between 1 and $r \div 2$, where $r$ is the order of the Abelian group on the elliptic curve (i.e., between 1 and $2^{192} \div 2$).

**For P-256:**

```
p =115792089210356248762697446949407573530086143415290314195533631308867097853951
r =115792089210356248762697446949407573529996955224135760342422259061068512044369
b =5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b
Gx=6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296
Gy=4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5
```

[1] http://dx.doi.org/10.6028/NIST.FIPS.186-4

The function P256 is defined as follows. Given an integer $u$, $0 < u < r$, and a point $V$ on the curve $E$, the value P256($u$, $V$) is computed as the x-coordinate of the $u^{th}$ multiple $uV$ of the point $V$.

The private keys shall be between 1 and $r \div 2$, where $r$ is the order of the Abelian group on the elliptic curve (i.e., between 1 and $2^{256} \div 2$).

A valid public key $Q = (X_Q, Y_Q)$ is one where $X_Q$ and $Y_Q$ are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b$ (*mod* p) in the relevant curve's finite field.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

## 7.7 Cryptographic function definitions

In addition to computing the Elliptic Curve Diffie Hellman key, the Numeric Comparison, Out-of-Band and Passkey Entry protocols require four cryptographic functions. These functions are known as *f1*, *g*, *f2* and *f3*.

*f1* is used to generate the 128-bit commitment values Ca and Cb

*g* is used to compute the 6-digit numeric check values

*f2* is used to compute the link key and possible other keys from the DHKey and random nonces

*f3* is used to compute check values Ea and Eb in Authentication stage 2.

The basic building block for these functions is based on SHA-256, specified in [FIPS PUB 180-4] (http://dx.doi.org/10.6028/NIST.FIPS.180-4).

Inside the *f1*, *g*, *f2*, and *f3* cryptographic functions, when a multi-octet integer input parameter is used as input to the SHA-256 and HMAC-SHA-256 functions, the most significant octet of the integer parameter shall be the first octet of the stream and the least significant octet of the integer parameter shall be the last octet of the stream. The output of the *f1*, *g*, *f2*, and *f3* cryptographic functions is a multi-octet integer where the first octet out of SHA-256 and HMAC-SHA-256 shall be the MSB and the last octet shall be the LSB of that parameter.

### 7.7.1 The Secure Simple Pairing commitment function *f1*

The commitments are computed with function *f1*. The definition of the Secure Simple Pairing commitment function makes use of the MAC function HMAC based on $SHA\text{-}256_x$, which is denoted as $HMAC\text{-}SHA\text{-}256_X$ with 128-bit key X.

The inputs to the Secure Simple Pairing function *f1* are specified in Table 7.2.

| Input | Bits with P-192 | Bits with P-256 |
|---|---|---|
| U | 192 | 256 |
| V | 192 | 256 |
| X | 128 | 128 |
| Z | 8 | 8 |

*Table 7.2: Inputs to the f1 function*

Z is zero (i.e., 8 bits of zeros) for Numeric Comparison and OOB protocol. In the Passkey protocol the most significant bit of Z is set equal to one and the least significant bit is made up from one bit of the passkey e.g. if the passkey bit is 1 then Z = 0x81 and if the passkey bit is 0 then Z = 0x80.

The output of the Secure Simple Pairing *f1* function is:

$$f1(U, V, X, Z) = \text{HMAC-SHA-256}_X (U \,||\, V \,||\, Z) \div 2^{128}$$

The inputs to *f1* are different depending on the different protocols as specified in Table 7.3.

| Numeric Comparison | Out-Of-Band | Passkey Entry |
|---|---|---|
| Ca = *f1*(PKax, PKbx, Na, 0) | Ca = *f1*(PKax, PKax, ra, 0) | Cai = *f1*(PKax, PKbx, Nai, rai) |
| Cb = *f1*(PKbx, PKax, Nb, 0) | Cb = *f1*(PKbx, PKbx, rb, 0) | Cbi = *f1*(PKbx, PKax, Nbi, rbi) |

*Table 7.3: Inputs to f1 for the different protocols*

where PKax denotes the x-coordinate of the public key PKa of A. Similarly, PKbx denotes the x-coordinate of the public key PKb of B. Nai is the nonce value of $i^{th}$ round. For each round Nai value is a new 128 bit number. Similarly, rai is one bit value of the passkey expanded to 8 bits (either 0x80 or 0x81).

Na and Nb are nonces from Devices A and B. ra and rb are random values generated by devices A and B.

### 7.7.2 The Secure Simple Pairing numeric verification function *g*

The Secure Simple Pairing *g* function is defined as follows:

The inputs to the Secure Simple Pairing function *g* are specified in Table 7.4.

| Input | Bits with P-192 | Bits with P-256 |
|---|---|---|
| U | 192 | 256 |
| V | 192 | 256 |
| X | 128 | 128 |
| Y | 128 | 128 |

*Table 7.4: Inputs to the g function*

The output of the Secure Simple Pairing *g* function is:

$$g(U, V, X, Y) = \text{SHA-256}(U \,||\, V \,||\, X \,||\, Y) \; mod \; 2^{32}$$

The numeric verification value is taken as six least significant digits of the 32-bit integer *g*(PKax, PKbx, Na, Nb) where PKax denotes the x-coordinate of the public key PKa of A and PKbx denotes the x-coordinate of the public key PKb of B.

Output of SHA-256 is truncated to 32 bits by taking the least significant 32 bits of the output of SHA-256. This value is converted to decimal numeric value. The checksum used for numeric comparison is the least significant six digits.

$$Compare\ Value = g\ (U, V, X, Y)\ mod\ 10^6$$

For example, if output = 0x 01 2e b7 2a then decimal value = 19838762 and the checksum used for numeric comparison is 838762.

### 7.7.3 The Secure Simple Pairing key derivation function *f2*

The definition of the Secure Simple Pairing key derivation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_W$ with 192-bit or 256-bit key W.

The inputs to the Secure Simple Pairing function *f2* are specified in Table 7.5.

| Input | Bits with P-192 | Bits with P-256 |
|---|---|---|
| W | 192 | 256 |
| $N_1$ | 128 | 128 |
| $N_2$ | 128 | 128 |
| keyID | 32 | 32 |
| $A_1$ | 48 | 48 |
| $A_2$ | 48 | 48 |

*Table 7.5: Inputs to the f2 function*

The string "btlk" is mapped into a keyID using ASCII as 0x62746C6B.

The output of the Secure Simple Pairing *f2* function is:

$$f2(W, N_1, N_2, KeyID, A_1, A_2) = HMAC\text{-}SHA\text{-}256_W (N1 \,||\, N2 \,||\, KeyID \,||\, A_1 \,||\, A_2) \div 2^{128}$$

The output of *f2* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

The link key is then calculated as:

LK = *f2*(DHKey, Nc, Np, "btlk", BD_ADDR_C, BD_ADDR_P)

Nc is whichever of N1 and N2 was generated by the Central and sent to the Peripheral; Np is the other.

### 7.7.4 The Secure Simple Pairing check function *f3*

The definition of the Secure Simple Pairing *f3* check function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_W$ with 192-bit or 256-bit key W.

The inputs to the Secure Simple Pairing function *f3* are specified in Table 7.6.

| Input | Bits with P-192 | Bits with P-256 |
|---|---|---|
| W | 192 | 256 |
| $N_1$ | 128 | 128 |
| $N_2$ | 128 | 128 |
| IOcap | 24 | 24 |
| $A_1$ | 48 | 48 |
| $A_2$ | 48 | 48 |

*Table 7.6: Inputs to the f3 function*

IOcap is three octets with the most significant octet as the Authentication Requirements parameter, the middle octet as the LMP Out-of-Band Authentication Data parameter, and the least significant octet as the LMP IO capability parameter.

The output of the Secure Simple Pairing *f3* function is:

$$f3(W, N_1, N_2, R, IOcap, A_1, A_2) = HMAC\text{-}SHA\text{-}256W (N_1 \,||\, N_2 \,||\, R \,||\, IOcap \,||\, A_1 \,||\, A_2) \div 2^{128}$$

The output of *f3* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256. The check values are computed with function *f3*. The inputs to *f3* are different depending on the different protocols, as specified in Table 7.7.

| Numeric Comparison | Out-Of-Band | Passkey Entry |
|---|---|---|
| Ea = *f3*(DHKey, Na, Nb, 0, IO-capA, A, B) | Ea = *f3(* DHKey, Na, Nb, rb, IO-capA, A, B) | Ea = *f3(* DHKey, Na20, Nb20, rb, IOcapA, A, B) |
| Eb = *f3*(DHKey, Nb, Na, 0, IO-capB, B, A) | Eb = *f3*(DHKey, Nb, Na, ra, IO-capB, B, A) | Eb = *f3*(DHKey, Nb20, Na20, ra, IOcapB, B, A) |

*Table 7.7: Inputs to f3 for the different protocols*

DHKey denotes the shared secret Diffie-Hellman Key computed as P192(SKa, PKb) or P256(SKa, PKb) by A and as P192(SKb, PKa) or P256(SKb, PKa) by B. IOcapA denotes the IO capability data of A and IOcapB denotes the IO capability data of B. In Passkey Entry, the data ra and rb are 6-digit passkey values which are expressed as a 128-bit integer. For instance, if the 6-digit value of ra is 131313, then

R = 0x 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 f1.

The input A is the BD_ADDR of device A and the input B is the BD_ADDR of device B.

### 7.7.5 [This section is no longer used]

### 7.7.6 The AES encryption key generation function *h3*

AES encryption keys are created using the AES encryption key generation function *h3*. The definition of the AES encryption key generation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-256$_T$ with 128-bit key T.

The inputs to function *h3* are specified in Table 7.8.

| Input | Bits with P-256 |
|---|---|
| T | 128 |
| keyID | 32 |
| $A_1$ | 48 |
| $A_2$ | 48 |
| ACO | 64 |

*Table 7.8: Inputs to the h3 function*

$A_1$ is the BD_ADDR of the Central. $A_2$ is the BD_ADDR of the Peripheral. ACO is the 64 bit ACO output from *h5*. T is the 128 bit Bluetooth Link Key derived from *f2*.

The string “btak” (Bluetooth AES Key) is mapped into a keyID using ASCII as 0x6274616B.

The output of function *h3* is:

$$h3(W, keyID, A_1, A_2, ACO) = HMAC\text{-}SHA\text{-}256_T(KeyID \,||\, A_1 \,||\, A_2 \,||\, ACO) \div 2^{128}$$

The output of *h3* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

### 7.7.7 The Device authentication key generation function *h4*

With Secure Connections, a device authentication key is created using function *h4*. The definition of the device authentication key generation function makes use of the MAC function HMAC based on SHA-256, which is denoted as $HMAC\text{-}SHA\text{-}256_T$ with 128-bit key T.

The inputs to function *h4* are specified in Table 7.9.

| Input | Bits with P-256 |
|---|---|
| T | 128 |
| keyID | 32 |
| $A_1$ | 48 |
| $A_2$ | 48 |

*Table 7.9: Inputs to the h4 function*

$A_1$ is the BD_ADDR of the Central. $A_2$ is the BD_ADDR of the Peripheral. T is the 128 bit Bluetooth Link Key derived from *f2*.

The string “btdk” (Bluetooth Device Key) is mapped into a keyID using ASCII as 0x6274646B.

The output of function *h4* is:

$$h4(W, KeyID, A_1, A_2) = HMAC\text{-}SHA\text{-}256_T(KeyID \,||\, A_1 \,||\, A_2) \div 2^{128}$$

The output of *h4* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

### 7.7.8 The Device authentication confirmation function *h5*

With Secure Connections, device authentication confirmation values are created using function *h5*. The definition of the device authentication confirmation function makes use of the MAC function HMAC based on SHA-256, which is denoted as $HMAC\text{-}SHA\text{-}256_S$ with 128-bit key S.

The inputs to function *h5* are specified in Table 7.10.

| Input | Bits with P-256 |
|---|---|
| S | 128 |
| $R_1$ | 128 |
| $R_2$ | 128 |

*Table 7.10: Inputs to the h5 function*

$R_1$ is the 128 bit random number (AU_RAND_C) from the Central during the Link Manager device authentication sequence. $R_2$ is the 128 bit random number from the Peripheral (AU_RAND_P) during the Link Manager device authentication sequence. S is the 128-bit Bluetooth Device Authentication Key derived from *h4*.

The output of function *h5* is:

$$h5(W, R_1, R_2) = \text{HMAC-SHA-256}_S(R_1 \,||\, R_2) \div 2^{128}$$

The output of *h5* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256. The first 32 bits (leftmost) become the SRES_C. The next 32 bits become the SRES_P. The final 64 bits become the Authentication Ciphering Offset (ACO), which is used in *h3* and as the IV for Encryption Start for the encryption nonce.

# 8 [THIS SECTION IS NO LONGER USED]

# 9 AES-CCM ENCRYPTION FOR BR/EDR

The Baseband provides security using Counter with CBC-MAC (CCM) as defined in IETF RFC 3610 (http://www.ietf.org/rfc/rfc3610.txt) with a modification to the $B_1$ counter mode block format that omits the length of the additional authenticated data. The description of the algorithm can also be found in the NIST Special Publication 800-38C (http://csrc.nist.gov/publications/PubsSPs.html).

Using the notation in [4], CCM has two size parameters, M and L.

The Baseband defines these to be:

M = 4; indicating that the MIC length is 4 octets (32 bits)

Size of M represents a trade-off between message expansion and the probability that an attacker can undetectably modify a message.

L = 2; indicating that the Length field is 2 octets (16 bits)

Size of L requires a trade-off between the maximum message size and the size of the Nonce.

## 9.1 Nonce formats

All of the parameters in the nonce are unique per logical transport. The nonce will be 13 octets and will have two formats. The first format is called the payload counter format and is used for ACL packets on the primary LT_ADDR. The second format is called the clock format and is used for eSCO packets.

The reason for two formats has to do with two factors: potential security attacks and synchronization. Since ACL packets on the primary LT_ADDR carry protocol, they are susceptible to attacks that eSCO packets (only data) are not.

The descriptions of the two nonce formats are provided in Table 9.1.

| Octet | Field | Octets | Payload Counter Format Description | Clock Format Description |
|---|---|---|---|---|
| 0 | Nonce0 | 1 | PayloadCounter[7:0] | CLK[8:1] |
| 1 | Nonce1 | 1 | PayloadCounter[15:8] | CLK[16:9] |
| 2 | Nonce2 | 1 | PayloadCounter[23:16] | CLK[24:17] |

| Octet | Field | Octets | Payload Counter Format Description | Clock Format Description |
|---|---|---|---|---|
| 3 | Nonce3 | 1 | PayloadCounter[31:24] | Bits 2-0:<br>CLK[27:25]<br><br>Bits 7-3:<br>dayCounter[4:0] |
| 4 | Nonce4 | 1 | Bits 3-0: PayloadCounter[35:32]<br><br>Bit 4: zero-length ACL-U continuation packet (1=zero-length ACL-U continuation packet, 0=otherwise)<br><br>Bit 5: direction (0=Central to Peripheral, 1=Peripheral to Central)<br><br>Bits 7-6:<br>nonceType = 0b00 (ACL) | Bits 5-0:<br>dayCounter[10:5]<br><br>Bits 7-6:<br>nonceType =<br>0b01 (eSCO) |
| 5 | Nonce5 | 1 | IV[7:0] | |
| 6 | Nonce6 | 1 | IV[15:8] | |
| 7 | Nonce7 | 1 | IV[23:16] | |
| 8 | Nonce8 | 1 | IV[31:24] | |
| 9 | Nonce9 | 1 | IV[39:32] | |
| 10 | Nonce10 | 1 | IV[47:40] | |
| 11 | Nonce11 | 1 | IV[55:48] | |
| 12 | Nonce12 | 1 | IV[63:56] | |

*Table 9.1: Nonce formats*

In the payload counter format, the PayloadCounter starts at zero for the first encrypted packet in each direction after encryption is started or resumed and increments by one every time an encrypted payload including zero-length payloads is accepted by the remote device.

Note: If, when encryption is being enabled or resumed, a packet is first transmitted unencrypted and then retransmitted encrypted, then the PayloadCounter is incremented by one when the encrypted retransmission of the packet gets accepted by the remote device.

Bit 4 of Octet 4 shall be set to 1 for a zero length ACL-U continuation packet (see [Vol 2] Part B, Section 7.6.2.2), otherwise it shall be set to 0.

In the clock format, the Central's clock (CLK) used for the nonce shall be the value in the first slot of the packet. After a new ACL connection has been established or a role

switch has been successfully performed and when eSCO is successfully established for the first time then the dayCounter value shall be initialized to:

1 if CLK27 in the first clock format nonce is 0, and initialization procedure 2 (see [Vol 2] Part B, Section 8.6.3) is used

0 otherwise.

After the dayCounter has been initialized, it shall increment by one every time the Central's clock (CLK) rolls over to 0x0000000 (approximately every 23.3 hours).

Note: When Security Mode 4 is in use, eSCO will not be established before encryption is started.

The IV is an 8 octet field. For encryption start, all octets of the IV are from the ACO output of the last execution of *h5* prior to the start of encryption. Multiple device authentications may occur prior to starting encryption but only the ACO of the last device authentication is used. After an encryption resume, all 8 octets of the IV are from the EN_RAND sent by the device initiating the encryption pause (see [Vol 2] Part C, Section 4.2.5.5). An encryption pause and resume will be required prior to the PayloadCounter or dayCounter rolling over in order to keep the nonce fresh for an encryption key.

| Octet | IV for Encryption Start | IV After Resume Encryption |
|---|---|---|
| 0 | ACO[0] | EN_RAND[0] |
| 1 | ACO[1] | EN_RAND[1] |
| 2 | ACO[2] | EN_RAND[2] |
| 3 | ACO[3] | EN_RAND[3] |
| 4 | ACO[4] | EN_RAND[4] |
| 5 | ACO[5] | EN_RAND[5] |
| 6 | ACO[6] | EN_RAND[6] |
| 7 | ACO[7] | EN_RAND[7] |

*Table 9.2: IV construction*

## 9.2 Counter mode blocks

For calculating the MIC, the payload is broken into two or more counter mode blocks. The CCM specification refers to these blocks as blocks $B_0$ – $B_n$. $B_0$ applies to the nonce, $B_1$ applies to the associated data {that is packet header and payload header} and additional B blocks are generated as needed for authentication of the payload body.

| Offset (octets) | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Flags | 1 | 0x49 | As per the CCM specification |
| 1 | Nonce | 13 | variable | The nonce as described in Section 9.1.<br>Nonce0 shall have offset 1.<br>Nonce12 shall have offset 13. |
| 14 | Length[MSO] | 1 | variable | The most significant octet of the length of the payload body |
| 15 | Length[LSO] | 1 | variable | The least significant octet of the length of the payload body |

*Table 9.3: $B_0$ counter mode block format*

| Offset | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Packet_Header [MSO] | 1 | Variable | The most significant octet of the packet header:<br>Bit 0: 0 (ARQN masked)<br>Bit 1: 0 (SEQN masked)<br>Bit 7 – Bit 2: 0b000000 |
| 1 | Packet_Header [LSO] | 1 | Variable | The least significant octet of the packet header:<br>Bit 2 – Bit 0: LT_ADDR<br>Bit 6 – Bit 3: TYPE<br>Bit 7: 0 (FLOW masked) |
| 2 | Payload_Header | 1 | Variable | The payload header:<br>Bit 1 – Bit 0: LLID<br>Bit 2: 0 (FLOW masked)<br>Bit 7 – Bit 3: 0b00000 |
| 3 | Padding | 13 | 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 | These octets are only used to pad the block. They are not part of the packet and never transmitted. |

*Table 9.4: $B_1$ counter mode block format*

## 9.3 Encryption blocks

The CCM algorithm uses the $A_i$ blocks to generate a keystream that is used to encrypt the MIC and payload body. Block $A_0$ is always used to encrypt and decrypt the MIC,

when present. Block $A_1$ is always used to encrypt and decrypt the first 16 octets of the payload body. Subsequent blocks are always used to encrypt and decrypt the rest of the payload body as needed.

| Offset (octets) | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Flags | 1 | 0x01 | As per the CCM specification |
| 1 | Nonce | 13 | variable | The nonce as described in Section 9.1.<br>Nonce0 shall have offset 1.<br>Nonce12 shall have offset 13. |
| 14 | i[MSO] | 1 | variable | The most significant octet of the counter i |
| 15 | i[LSO] | 1 | variable | The least significant octet of the counter i |

*Table 9.5: Encryption mode block format*

## 9.4 Encryption key size reduction

When the devices have negotiated a key size shorter than the maximum length, the key will be shortened by replacing the appropriate number of least significant octets of the key with 0x00.

For example, if a 128-bit encryption key is

0x12345678_9ABCDEF0_12345678_9ABCDEF0

and it is reduced to 7 octets (56 bits), then the resulting key is

0x12345678_9ABCDE00_00000000_00000000.

## 9.5 Repeated MIC failures

Anytime the MIC check fails and the CRC passes on a given packet, it is considered an authentication failure. No more than three authentication failures shall be permitted during the lifetime of an encryption key with a given IV. The third authentication failure shall initiate an encryption key refresh (see [Vol 2] Part C, Section 4.2.5.8). If a fourth authentication failure occurs prior to the encryption key refresh procedure completing, the link shall be disconnected with reason code *Rejected due to Security Reasons* (0x0E).

Note: The MIC is not checked when the CRC is invalid.

# Host

**Specification of the *Bluetooth*® System**

**Volume 3**

**Covered Core Package Version: v6.3**

**Version Date: 2026-05-05**

Bluetooth SIG Proprietary

Host
Part A

# LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

*The Bluetooth Logical Link Control and Adaptation Protocol (L2CAP) supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. The protocol state machine, packet format, and composition are described in this Part of the specification.*

# CONTENTS

**1 Introduction ..... 1096**
1.1 L2CAP features ..... 1096
1.2 Assumptions ..... 1099
1.3 Scope ..... 1100
1.4 Terminology ..... 1100

**2 General operation ..... 1104**
2.1 Channel identifiers ..... 1104
2.2 Operation between devices ..... 1107
2.3 Operation between layers ..... 1108
2.4 Modes of operation ..... 1109
2.5 Mapping channels to logical links ..... 1111

**3 Data packet format ..... 1112**
3.1 Connection-oriented channels in Basic L2CAP mode ..... 1112
3.2 Connectionless data channel in Basic L2CAP mode ..... 1113
3.3 Connection-oriented channel in Retransmission/Flow Control/Streaming modes ..... 1114
3.3.1 L2CAP header fields ..... 1114
3.3.2 Control field ..... 1115
3.3.3 L2CAP SDU Length field (2 octets) ..... 1119
3.3.4 Information Payload field ..... 1119
3.3.5 Frame Check Sequence (2 octets) ..... 1119
3.3.6 Invalid Frame Detection (Retransmission and Flow Control modes) ..... 1120
3.3.7 Invalid Frame Detection algorithm ..... 1121
3.4 Connection-oriented channels in LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode 1122
3.4.1 L2CAP Header fields ..... 1122
3.4.2 L2CAP SDU Length field (2 octets) ..... 1123
3.4.3 Information Payload field ..... 1123

**4 Signaling packet formats ..... 1124**
4.1 L2CAP_COMMAND_REJECT_RSP (code 0x01) ..... 1127
4.2 L2CAP_CONNECTION_REQ (code 0x02) ..... 1128
4.3 L2CAP_CONNECTION_RSP (code 0x03) ..... 1129
4.4 L2CAP_CONFIGURATION_REQ (code 0x04) ..... 1131
4.5 L2CAP_CONFIGURATION_RSP (code 0x05) ..... 1133
4.6 L2CAP_DISCONNECTION_REQ (code 0x06) ..... 1136
4.7 L2CAP_DISCONNECTION_RSP (code 0x07) ..... 1136

4.8 L2CAP_ECHO_REQ (code 0x08) ..... 1137
4.9 L2CAP_ECHO_RSP (code 0x09) ..... 1137
4.10 L2CAP_INFORMATION_REQ (code 0x0A) ..... 1138
4.11 L2CAP_INFORMATION_RSP (code 0x0B) ..... 1139
4.12 Extended Feature Mask ..... 1140
4.13 Fixed Channels Supported over BR/EDR ..... 1141
4.14 [This section is no longer used] ..... 1142
4.15 [This section is no longer used] ..... 1142
4.16 [This section is no longer used] ..... 1142
4.17 [This section is no longer used] ..... 1142
4.18 [This section is no longer used] ..... 1142
4.19 [This section is no longer used] ..... 1142
4.20 L2CAP_CONNECTION_PARAMETER_UPDATE_REQ (code 0x12) ..... 1142
4.21 L2CAP_CONNECTION_PARAMETER_UPDATE_RSP (code 0x13) ..... 1143
4.22 L2CAP_LE_CREDIT_BASED_CONNECTION_REQ (code 0x14) ..... 1144
4.23 L2CAP_LE_CREDIT_BASED_CONNECTION_RSP (code 0x15) ..... 1145
4.24 L2CAP_FLOW_CONTROL_CREDIT_IND (code 0x16) ..... 1147
4.25 L2CAP_CREDIT_BASED_CONNECTION_REQ (code 0x17) . 1148
4.26 L2CAP_CREDIT_BASED_CONNECTION_RSP (code 0x18) .. 1149
4.27 L2CAP_CREDIT_BASED_RECONFIGURE_REQ (code 0x19) ..... 1151
4.28 L2CAP_CREDIT_BASED_RECONFIGURE_RSP (code 0x1A) ..... 1152

**5 Configuration parameter options ..... 1153**
5.1 Maximum Transmission Unit (MTU) ..... 1153
5.2 Flush Timeout option ..... 1155
5.3 Quality of Service (QoS) option ..... 1156
5.4 Retransmission and Flow Control option ..... 1160
5.5 Frame Check Sequence (FCS) option ..... 1165
5.6 Extended Flow Specification option ..... 1166
5.7 Extended Window Size option ..... 1171

**6 State machine ..... 1173**
6.1 General rules for the state machine ..... 1173
6.1.1 CLOSED state ..... 1174
6.1.2 WAIT_CONNECT_RSP state ..... 1175
6.1.3 WAIT_CONNECT state ..... 1176
6.1.4 CONFIG state ..... 1176

6.1.5 OPEN state ..... 1181
6.1.6 WAIT_DISCONNECT state ..... 1182
6.1.7 [This section is no longer used] ..... 1182
6.1.8 [This section is no longer used] ..... 1182
6.1.9 [This section is no longer used] ..... 1182
6.1.10 [This section is no longer used] ..... 1182
6.1.11 [This section is no longer used] ..... 1182
6.1.12 [This section is no longer used] ..... 1182
6.2 Timers events ..... 1182
6.2.1 RTX ..... 1182
6.2.2 ERTX ..... 1183

**7 General procedures ..... 1186**
7.1 Configuration process ..... 1186
7.1.1 Request Path ..... 1187
7.1.2 Response Path ..... 1188
7.1.3 Lockstep Configuration process ..... 1188
7.1.4 Standard Configuration process ..... 1191
7.2 Fragmentation and recombination ..... 1193
7.2.1 Fragmentation of L2CAP PDUs ..... 1193
7.2.2 Recombination of L2CAP PDUs ..... 1194
7.3 Encapsulation of SDUs ..... 1195
7.3.1 Segmentation of L2CAP SDUs ..... 1195
7.3.2 Reassembly of L2CAP SDUs ..... 1196
7.3.3 Segmentation and fragmentation ..... 1196
7.4 Delivery of erroneous L2CAP SDUs ..... 1197
7.5 Operation with flushing on ACL-U logical links ..... 1198
7.6 Connectionless data channel ..... 1199
7.7 Operation collision resolution ..... 1201
7.8 [This section is no longer used] ..... 1201
7.9 Prioritizing data over HCI ..... 1201
7.10 Supporting Extended Flow Specification for BR/EDR and BR/EDR/LE Controllers ..... 1201
7.11 Enhanced Credit-Based Flow Control Reconfiguration ..... 1203
7.12 Operation with flushing on LE-U logical links ..... 1203

**8 Procedures for Flow Control and Retransmission ..... 1204**
8.1 Information retrieval ..... 1204
8.2 Function of PDU Types for Flow Control and Retransmission ..... 1204
8.2.1 Information frame (I-frame) ..... 1204
8.2.2 Supervisory frame (S-frame) ..... 1204
8.2.2.1 Receiver Ready (RR) ..... 1205
8.2.2.2 Reject (REJ) ..... 1205

8.3 Variables and sequence numbers ..... 1205
8.3.1 Sending peer ..... 1206
8.3.1.1 Send sequence number TxSeq ..... 1206
8.3.1.2 Send state variable NextTxSeq ..... 1206
8.3.1.3 Acknowledge state variable ExpectedAckSeq ..... 1206
8.3.2 Receiving peer ..... 1207
8.3.2.1 Receive sequence number ReqSeq ..... 1207
8.3.2.2 Receive state variable ExpectedTxSeq ..... 1208
8.3.2.3 Buffer state variable BufferSeq ..... 1208
8.4 Retransmission mode ..... 1209
8.4.1 Transmitting frames ..... 1209
8.4.1.1 Last received R was set to zero ..... 1209
8.4.1.2 Last received R was set to one ..... 1211
8.4.2 Receiving I-frames ..... 1211
8.4.3 I-frames pulled by the SDU reassembly function ..... 1211
8.4.4 Sending and receiving acknowledgments ..... 1211
8.4.4.1 Sending acknowledgments ..... 1212
8.4.4.2 Receiving acknowledgments ..... 1212
8.4.5 Receiving REJ frames ..... 1213
8.4.6 Waiting acknowledgments ..... 1213
8.4.7 Exception conditions ..... 1213
8.4.7.1 TxSeq sequence error ..... 1213
8.4.7.2 ReqSeq sequence error ..... 1214
8.4.7.3 Timer recovery error ..... 1214
8.4.7.4 Invalid frame ..... 1214
8.5 Flow Control mode ..... 1215
8.5.1 Transmitting I-frames ..... 1215
8.5.2 Receiving I-frames ..... 1216
8.5.3 I-frames pulled by the SDU reassembly function ..... 1216
8.5.4 Sending and receiving acknowledgments ..... 1216
8.5.4.1 Sending acknowledgments ..... 1216
8.5.4.2 Receiving acknowledgments ..... 1216
8.5.5 Waiting acknowledgments ..... 1217
8.5.6 Exception conditions ..... 1217
8.5.6.1 TxSeq sequence error ..... 1217
8.5.6.2 ReqSeq sequence error ..... 1218
8.5.6.3 Invalid frame ..... 1218
8.6 Enhanced Retransmission mode ..... 1218
8.6.1 Function of PDU types ..... 1218
8.6.1.1 Receiver Ready (RR) ..... 1219
8.6.1.2 Reject (REJ) ..... 1219
8.6.1.3 Receiver Not Ready (RNR) ..... 1219

8.6.1.4 Selective Reject (SREJ) ..... 1219
8.6.1.5 Functions of the Poll (P) and Final (F) bits . 1220
8.6.2 Rules for timers ..... 1220
8.6.2.1 Timer rules for ACL-U logical links ..... 1220
8.6.2.2 [This section is no longer used] ..... 1221
8.6.2.3 [This section is no longer used] ..... 1221
8.6.3 General rules for the state machine ..... 1221
8.6.4 State diagram ..... 1222
8.6.5 States tables ..... 1223
8.6.5.1 State machines ..... 1223
8.6.5.2 States ..... 1224
8.6.5.3 Variables and timers ..... 1224
8.6.5.4 Events ..... 1227
8.6.5.5 Conditions ..... 1228
8.6.5.6 Actions ..... 1230
8.6.5.7 XMIT state table ..... 1235
8.6.5.8 WAIT_F state table ..... 1236
8.6.5.9 RECV state table ..... 1236
8.6.5.10 REJ_SENT state table ..... 1240
8.6.5.11 SREJ_SENT state table ..... 1243
8.7 Streaming mode ..... 1247
8.7.1 Transmitting I-frames ..... 1247
8.7.2 Receiving I-frames ..... 1247
8.7.3 Exception conditions ..... 1248
8.7.3.1 TxSeq sequence error ..... 1248

**9 [This section is no longer used] ..... 1249**

**10 Procedures for Credit Based Flow Control ..... 1250**
10.1 LE Credit Based Flow Control mode ..... 1250
10.2 Enhanced Credit Based Flow Control Mode ..... 1251

**Appendix A Configuration MSCs ..... 1253**

**Appendix B Changes to signaling packet names ..... 1256**

# 1 INTRODUCTION

This section of the Bluetooth Specification defines the Logical Link Control and Adaptation Layer Protocol, referred to as L2CAP. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability and segmentation and reassembly operation. L2CAP permits higher level protocols and applications to transmit and receive upper layer data packets (L2CAP Service Data Units, SDU) up to 64 kilobytes in length. L2CAP also permits per-channel flow control and retransmission.

The L2CAP layer provides logical channels, named L2CAP channels, which are multiplexed over one or more logical links.

## 1.1 L2CAP features

The functional requirements for L2CAP include protocol/channel multiplexing, segmentation and reassembly (SAR), per-channel flow control, and error control. L2CAP sits above a lower layer composed of one of the following:

1. BR/EDR Controller
2. BR/EDR/LE Controller (supporting BR/EDR and LE)
3. LE Controller (supporting LE only)

L2CAP interfaces with upper layer protocols.

Figure 1.1 breaks down L2CAP into its architectural components. The Channel Manager provides the control plane functionality and is responsible for all internal signaling, L2CAP peer-to-peer signaling and signaling with higher and lower layers. It performs the state machine functionality described in Section 6 and uses message formats described in Section 4, and Section 5. The Retransmission and Flow Control block provides per-channel flow control and error recovery using packet retransmission. The Resource Manager is responsible for providing a frame relay service to the Channel Manager, the Retransmission and Flow Control block and those application data streams that do not require Retransmission and Flow Control services. It is responsible for coordinating the transmission and reception of packets related to multiple L2CAP channels over the facilities offered at the lower layer interface.

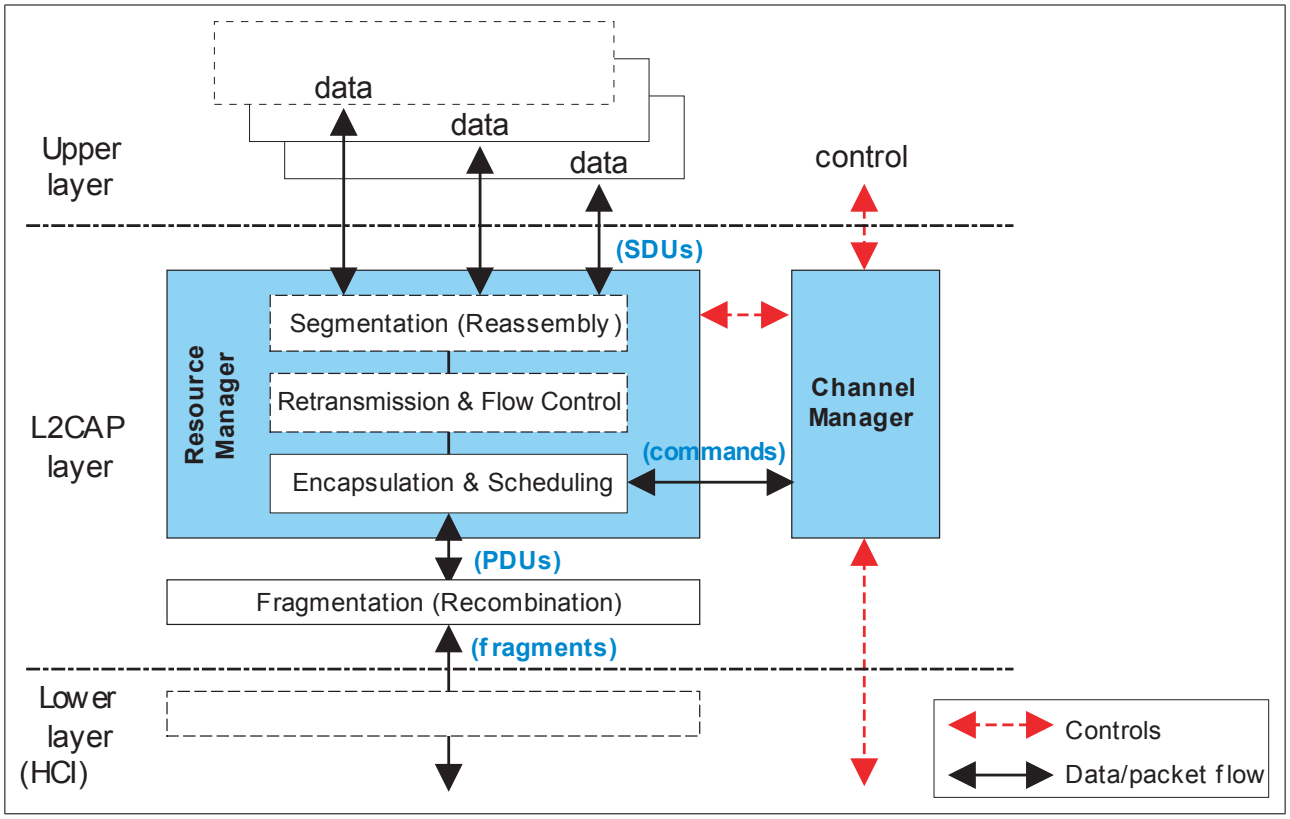



*Figure 1.1: L2CAP architectural blocks*

- *Protocol/channel multiplexing*

  L2CAP supports multiplexing over individual Controllers. An L2CAP channel shall operate over one Controller. An L2CAP channel cannot move between Controllers.

  During channel setup, protocol multiplexing capability is used to route the connection to the correct upper layer protocol.

  For data transfer, logical channel multiplexing is needed to distinguish between multiple upper layer entities. There may be more than one upper layer entity using the same protocol.

- *Segmentation and reassembly*

  With the frame relay service offered by the Resource Manager, the length of transport frames is controlled by the individual applications running over L2CAP. Many multiplexed applications are better served if L2CAP has control over the PDU length. This provides the following benefits:

  a. Segmentation will allow the interleaving of application data units in order to satisfy latency requirements.

  b. Memory and buffer management is easier when L2CAP controls the packet size.

  c. Error correction by retransmission can be made more efficient.

d. The amount of data that is destroyed when an L2CAP PDU is corrupted or lost can be made smaller than the application's data unit.

e. The application is decoupled from the segmentation required to map the application packets into the lower layer packets.

- *Flow control per L2CAP channel*

  Controllers provide error and flow control for data going over the air and HCI flow control exists for data going over an HCI transport. When several data streams run over the same Controller using separate L2CAP channels, each channel requires individual flow control. A window based flow control scheme is provided.

- *Error control and retransmissions*

  Some applications require a residual error rate much smaller than some Controllers can deliver. L2CAP provides error checks and retransmissions of L2CAP PDUs. The error checking in L2CAP protects against errors due to Controllers falsely accepting packets that contain errors but pass Controller-based integrity checks. L2CAP error checking and retransmission also protect against loss of packets due to flushing by the Controller. The error control works in conjunction with flow control in the sense that the flow control mechanism will throttle retransmissions as well as first transmissions.

- *Support for Streaming*

  Streaming applications such as audio set up an L2CAP channel with an agreed-upon data rate and must prevent flow control mechanisms, including those in the Controller, from altering the flow of data. A flush timeout is used to keep data flowing on the transmit side. Streaming mode is used to stop HCI and Controller based flow control from being applied on the receiving side.

- *Fragmentation and Recombination*

  Some Controllers may have limited transmission capabilities and may require fragment sizes different from those created by L2CAP segmentation. Therefore layers below L2CAP may further fragment and recombine L2CAP PDUs to create fragments which fit each layer’s capabilities. During transmission of an L2CAP PDU, many different levels of fragmentation and recombination may occur in both peer devices.

  The HCI driver or Controller may fragment L2CAP PDUs to honor packet size constraints of a Host Controller Interface transport scheme. This results in HCI ACL Data packet payloads carrying start and continuation fragments of the L2CAP PDU. Similarly the Controller may fragment L2CAP PDUs to map them into Controller packets. This may result in Controller packet payloads carrying start and continuation fragments of the L2CAP PDU.

  Each layer of the protocol stack may pass on different sized fragments of L2CAP PDUs, and the size of fragments created by a layer may be different in each peer

device. However the PDU is fragmented within the stack, the receiving L2CAP entity still recombines the fragments to obtain the original L2CAP PDU.

- *Quality of Service*

  The L2CAP connection establishment process allows the exchange of information regarding the quality of service (QoS) expected between two Bluetooth devices. Each L2CAP implementation monitors the resources used by the protocol and ensures that QoS contracts are honored.

  L2CAP may support both ACL-U isochronous (Guaranteed) and ACL-U asynchronous (Best Effort) data flows over the same ACL logical link by marking packets as automatically-flushable or non-automatically-flushable by setting the appropriate value for the Packet_Boundary_Flag in the HCI ACL Data packet (see [Vol 4] Part E, Section 5.4.2). Automatically-flushable L2CAP packets are flushed according to the automatic flush timeout set for the ACL logical link on which the L2CAP channels are mapped (see [Vol 4] Part E, Section 6.19). Non-automatically-flushable L2CAP packets are not affected by the automatic flush timeout and will not be flushed. L2CAP packets for BR/EDR logical links can also be flushed by using the HCI_Flush command (see [Vol 4] Part E, Section 7.3.4).

  Note: The terms "Guaranteed" and "Best Effort" are used as a shorthand to indicate that the implementation will or will not attempt to provide some level of quality of service. They do not make any statement as to what actual quality is provided; in particular, "Guaranteed" does not mean that successful delivery of data is guaranteed. Similarly, a "guarantee" is a set of resources (such as bandwidth and buffer space) allocated to a "Guaranteed" connection and does not mean that a specific data rate and latency will be provided under all possible circumstances.

## 1.2 Assumptions

The protocol is designed based on the following assumptions:

1. Controllers provide orderly delivery of data packets, although there might be individual packet corruption and duplicates. For devices with a BR/EDR or BR/EDR/LE Controller, no more than one ACL-U logical link exists between any two devices. For devices with a BR/EDR/LE or LE Controller, no more than one LE-U logical link exists between any two devices.
2. Controllers always provide the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Unidirectional traffic does not require duplex channels.
3. The L2CAP layer provides a channel with a degree of reliability based on the mechanisms available in Controllers and with additional packet segmentation, error detection, and retransmission that can be enabled in the enhanced L2CAP layer. Some Controllers perform data integrity checks and resend data until it has

been successfully acknowledged or a timeout occurs. Other Controllers will resend data up to a certain number of times whereupon the data is flushed. Because acknowledgments may be lost, timeouts may occur even after the data has been successfully sent.

Note: The use of Baseband Broadcast packets in a BR/EDR or BR/EDR/LE Controller is unreliable and all broadcasts start the first segment of an L2CAP packet with the same sequence bit.

4. Controllers provide error and flow control for data going over the air and HCI flow control exists for data going over an HCI transport but some applications will require better error control than some Controllers provide. The Flow and Error Control block provides four modes. Enhanced Retransmission mode and Retransmission mode offer segmentation, flow control and L2CAP PDU retransmissions. Flow control mode offers just the segmentation and flow control functions. Streaming mode offers segmentation and receiver side packet flushing.

## 1.3 Scope

The following features are outside the scope of L2CAP's responsibilities:

- L2CAP does not transport synchronous data designated for SCO or eSCO logical transports.
- L2CAP does not support a reliable broadcast channel. See Section 3.2.

## 1.4 Terminology

The following formal definitions apply:

| Term | Description |
|---|---|
| Upper layer | The system layer above the L2CAP layer, which exchanges data with L2CAP in the form of SDUs. The upper layer may be represented by an application or higher protocol entity known as the Service Level Protocol. The interface of the L2CAP layer with the upper layer is not specified. |
| Lower layer | The system layer below the L2CAP layer, which exchanges data with the L2CAP layer in the form of PDUs, or fragments of PDUs. The lower layer is mainly represented within the Controller, however a Host Controller interface (HCI) may be involved, such that an HCI Host driver could also be seen as the lower layer. Except for the HCI functional specification (in case HCI is involved) the interface between L2CAP and the lower layer is not specified. |
| L2CAP channel | The logical connection between two endpoints in peer devices, characterized by their Channel Identifiers (CID), which is multiplexed over one Controller based logical link. |

| Term | Description |
|---|---|
| SDU, or L2CAP SDU | Service Data Unit: a packet of data that L2CAP exchanges with the upper layer and transports transparently over an L2CAP channel using the procedures specified here. The term SDU is associated with data originating from upper layer entities only, i.e. does not include any protocol information generated by L2CAP procedures. |
| Segment, or SDU segment | A part of an SDU, as resulting from the Segmentation procedure. An SDU may be split into one or more segments.<br>Note: This term is relevant only to Enhanced Retransmission mode, Streaming mode, Retransmission mode, Flow Control mode, LE Credit Based Flow Control mode, and Enhanced Credit Based Flow Control mode, not to the Basic L2CAP mode. |
| Segmentation | A procedure used in the L2CAP Retransmission and Flow Control modes, resulting in an SDU being split into one or more smaller units, called Segments, as appropriate for the transport over an L2CAP channel.<br>Note: This term is relevant only to the Enhanced Retransmission mode, Streaming mode, Retransmission mode, Flow Control mode, LE Credit Based Flow Control mode, and Enhanced Credit Based Flow Control mode, not to the Basic L2CAP mode. |
| Reassembly | The reverse procedure corresponding to Segmentation, resulting in an SDU being re-established from the segments received over an L2CAP channel, for use by the upper layer. The interface between the L2CAP and the upper layer is not specified; therefore, reassembly may actually occur within an upper layer entity although it is conceptually part of the L2CAP layer.<br>Note: This term is relevant only to Enhanced Retransmission mode, Streaming mode, Retransmission mode, Flow Control mode, LE Credit Based Flow Control mode, and Enhanced Credit Based Flow Control mode, not to the Basic L2CAP mode. |
| PDU, or L2CAP PDU | Protocol Data Unit: a packet of data containing L2CAP protocol information fields, control information, and/or upper layer information data. A PDU is always started by a Basic L2CAP header. Types of PDUs are: B-frames, I-frames, S-frames, C-frames, G-frames, and K-frames. |
| Basic L2CAP header | Minimum L2CAP protocol information that is present in the beginning of each PDU: a PDU length field and a field containing the Channel Identifier (CID). |
| Basic information frame (B-frame) | A B-frame is a PDU used in the Basic L2CAP mode for L2CAP data packets. It contains a complete SDU as its payload, encapsulated by a Basic L2CAP header. |
| Information frame (I-frame) | An I-frame is a PDU used in Enhanced Retransmission mode, Streaming mode, Retransmission mode, and Flow Control mode. It contains an SDU segment and additional protocol information, encapsulated by a Basic L2CAP header. |
| Supervisory frame (S-frame) | An S-frame is a PDU used in Enhanced Retransmission mode, Retransmission mode, and Flow Control mode. It contains protocol information only, encapsulated by a Basic L2CAP header, and no SDU data. |
| Control frame (C-frame) | A C-frame is a PDU that contains L2CAP signaling messages exchanged between the peer L2CAP entities. C-frames are exclusively used on the L2CAP signaling channels. |

| Term | Description |
|---|---|
| Group frame (G-frame) | A G-frame is a PDU exclusively used on the Connectionless L2CAP channel. It is encapsulated by a Basic L2CAP header and contains the PSM followed by the completed SDU. G-frames may be used to broadcast data to active Peripherals via Active Peripheral Broadcast or to send unicast data to a single remote device. |
| Credit-based frame (K-frame) | A K-frame is a PDU used in LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode. It contains an SDU segment and additional protocol information, encapsulated by a Basic L2CAP header. |
| Fragment | A part of a PDU, as resulting from a fragmentation operation. Fragments are used only in the delivery of data to and from the lower layer. They are not used for peer-to-peer transportation. A fragment may be a Start or Continuation Fragment with respect to the L2CAP PDU. A fragment does not contain any protocol information beyond the PDU; the distinction of start and continuation fragments is transported by lower layer protocol provisions.<br>Note: Start Fragments always either begin with the first octet of the Basic L2CAP header of a PDU or they have a length of zero (see [Vol 2] Part B, Section 6.6.2). |
| Fragmentation | A procedure used to split L2CAP PDUs to smaller parts, named fragments, appropriate for delivery to the lower layer transport. Although described within the L2CAP layer, fragmentation may actually occur in an HCI Host driver, and/or in a Controller, to accommodate the L2CAP PDU transport to HCI ACL Data packet or Controller packet sizes.<br>Fragmentation of PDUs may be applied in all L2CAP modes. |
| Recombination | The reverse procedure corresponding to fragmentation, resulting in an L2CAP PDU re-established from fragments. In the receive path, full or partial recombination operations may occur in the Controller and/or the Host, and the location of recombination does not necessarily correspond to where fragmentations occurs on the transmit side. |
| Maximum Transmission Unit (MTU) | The maximum size of an SDU, in octets, that the upper layer entity is capable of accepting. |
| Payload Size | The amount of SDU data in a single PDU, in octets; equivalently, the number of octets in the Information Payload field of a PDU. |
| Maximum PDU payload Size (MPS) | The maximum payload size in octets that the L2CAP layer entity is capable of accepting.<br>In the absence of segmentation, or in the Basic L2CAP mode, the Maximum Transmission Unit is the same as the Maximum PDU payload Size and the two configuration parameters shall be set to the same value. |
| Signaling MTU ($MTU_{sig}$) | The maximum size of command information that the L2CAP layer entity is capable of accepting. The $MTU_{sig}$, refers to the signaling channel only and corresponds to the maximum size of a C-frame, omitting the size of the Basic L2CAP header. The $MTU_{sig}$ value of a peer is discovered when a C-frame that is too large is rejected by the peer. |

| Term | Description |
|---|---|
| Connection-less MTU ($MTU_{cnl}$) | The maximum size of the connection packet information that the L2CAP layer entity is capable of accepting. The $MTU_{cnl}$ refers to the connectionless channel only and corresponds to the maximum G-frame, omitting the size of the Basic L2CAP header and the PSM which immediately follows it. The $MTU_{cnl}$ of a peer can be discovered by sending an L2CAP_INFORMATION_REQ packet. |
| MaxTransmit | In Enhanced Retransmission mode and Retransmission mode, MaxTransmit controls the number of transmissions of a PDU that L2CAP is allowed to try before assuming that the PDU (and the link) is lost. The minimum value is 1 (only 1 transmission permitted). In Enhanced Retransmission mode a value 0 means infinite transmissions.<br><br>Note: Setting MaxTransmit to 1 prohibits PDU retransmissions. Failure of a single PDU will cause the link to drop. By comparison, in Flow Control mode, failure of a single PDU will not necessarily cause the link to drop. |

*Table 1.1: Terminology*

# 2 GENERAL OPERATION

L2CAP is based around the concept of *'channels'*. Each one of the endpoints of an L2CAP channel is referred to by a *channel identifier (CID).*

## 2.1 Channel identifiers

A channel identifier (CID) is the local name representing a logical channel endpoint on the device. The scope of CIDs is related to the logical link as shown in Figure 2.1. The null identifier (0x0000) shall not be used as a destination endpoint. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. These channels are referred to as fixed channels. An implementation of L2CAP shall support the fixed channel 0x0001 (see Table 2.1) on all ACL-U logical links and the fixed channels 0x0004, 0x0005, and 0x0006 (see Table 2.3) on all LE-U logical links. Other fixed channels may be supported on each logical link. The L2CAP_INFORMATION_REQ / L2CAP_INFORMATION_RSP mechanism (described in Section 4.10 and Section 4.11) shall be used to determine which fixed channels a remote device supports over the ACL-U logical link. Each fixed channel has the same CID at each end of the logical link carrying the channel.

The characteristics of each fixed channel are defined on a per channel basis. Fixed channel characteristics include configuration parameters (e.g., reliability, MTU size, QoS), security, and the ability to change parameters using the L2CAP configuration mechanism. Table 2.1 lists the defined fixed channels, provides a reference to where the associated channel characteristics are defined and specifies the logical link over which the channel may operate. Fixed channels are available as soon as the ACL-U or LE-U logical link is set up. All initialization that is normally performed when a channel is created shall be performed for each of the supported fixed channels when the ACL-U or LE-U logical link is set up. Fixed channels shall only run over ACL-U, APB-U, or LE-U logical links.

Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that two simultaneously active L2CAP channels on the same logical link (i.e., to the same peer device using the same transport) shall not share the same CID. A different CID name space exists for each ACL-U, APB-U, and LE-U logical link. Table 2.1 to Table 2.3 summarize the definition and partitioning of the CID name space for each type of logical link.

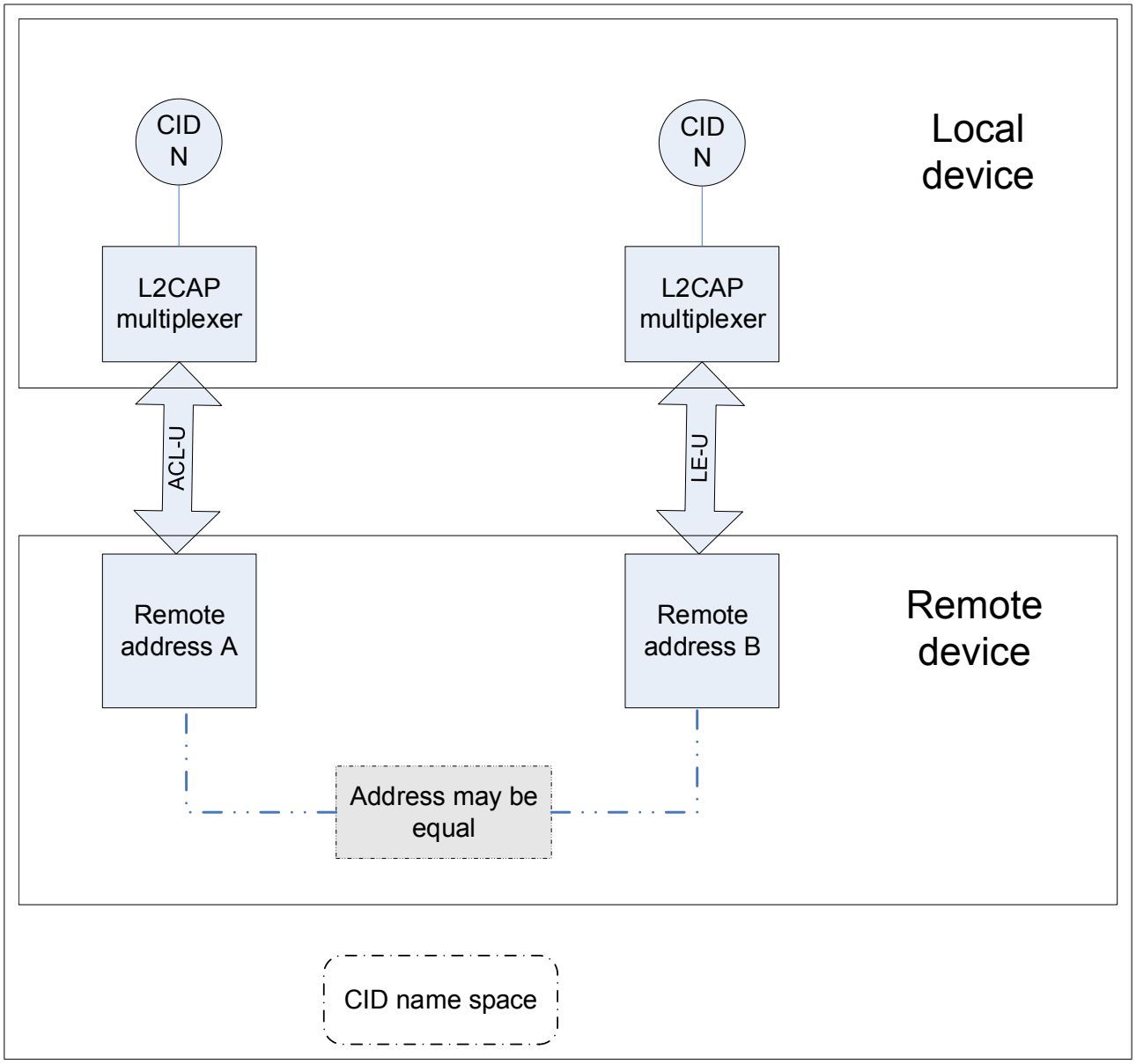



*Figure 2.1: Dynamically allocated CID assignments*

Assignment of dynamically allocated CIDs is relative to a particular logical link and a device can assign CIDs independently from other devices. Thus, even if the same CID value has been assigned to (remote) channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device. Further, even if the same CID value has been assigned to (remote) channel endpoints by the same remote device, these can be distinguished because they will be bound to a different logical link.

The CID name space for ACL-U logical links is as follows:

| CID | Description | Channel Characteristics |
|---|---|---|
| 0x0000 | Null identifier | Not allowed |
| 0x0001 | L2CAP Signaling channel | See Section 4 |
| 0x0002 | Connectionless channel | See Section 7.6 |
| 0x0003 | Previously used | Not applicable |
| 0x0007 | BR/EDR Security Manager | Used only for cross-transport key derivation; see [Vol 3] Part H, Section 2.3.5.7 |
| 0x003F | Previously used | Not applicable |
| 0x0040 to 0xFFFF | Dynamically allocated | Communicated using L2CAP configuration mechanism (see Section 7.1) or the L2CAP credit based create connection mechanism (see Section 4.25) |
| All other values | Reserved for future use | Not applicable |

*Table 2.1: CID name space on ACL-U logical links*

The CID name space for APB-U logical links is as follows:

| CID | Description | Channel Characteristics |
|---|---|---|
| 0x0000 | Null identifier | Not allowed |
| 0x0002 | Connectionless channel | See Section 7.6 |
| All other values | Reserved for future use | Not applicable |

*Table 2.2: CID name space on APB-U logical links*

The CID name space for LE-U logical links is as follows:

| CID | Description | Channel Characteristics |
|---|---|---|
| 0x0000 | Null identifier | Not Allowed |
| 0x0004 | Attribute Protocol | See [Vol 3] Part F |
| 0x0005 | L2CAP LE Signaling channel | See Section 4 |
| 0x0006 | Security Manager protocol | See [Vol 3] Part H |
| 0x0020 to 0x003E | Assigned Numbers | |

| CID | Description | Channel Characteristics |
|---|---|---|
| 0x0040 to 0x007F | Dynamically allocated | Communicated using the L2CAP LE credit based create connection mechanism (see Section 4.22) or the L2CAP credit based create connection mechanism (see Section 4.25) |
| All other values | Reserved for future use | Not applicable |

*Table 2.3: CID name space on LE-U logical links*

## 2.2 Operation between devices

Figure 2.2 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID, combined with the logical link, identifies each endpoint of the channel. When used for broadcast transmissions, the connectionless channel restricts data flow to a single direction. The connectionless channel may be used to transmit data to all active Peripherals, using Active Peripheral Broadcast. When used for unicast transmissions the connectionless channel may be used in either direction between a Central and a Peripheral.

There are also a number of CIDs reserved for special purposes. The L2CAP signaling channels are examples of reserved channels. Fixed channel 0x0001 is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of connection-oriented channels and to discover characteristics of the connectionless channel operating over the ACL-U logical link.

The L2CAP Signaling channel (0x0001) and all supported fixed channels are available immediately when the ACL-U logical link is established between two devices. Another CID (0x0002) is reserved for all incoming and outgoing connectionless data traffic, whether broadcast or unicast. Connectionless data traffic may flow immediately once the ACL-U logical link is established between two devices and once the transmitting device has determined that the remote device supports connectionless traffic.

The L2CAP LE Signaling channel (0x0005) and all supported fixed channels are available immediately when the LE-U logical link is established between two devices.

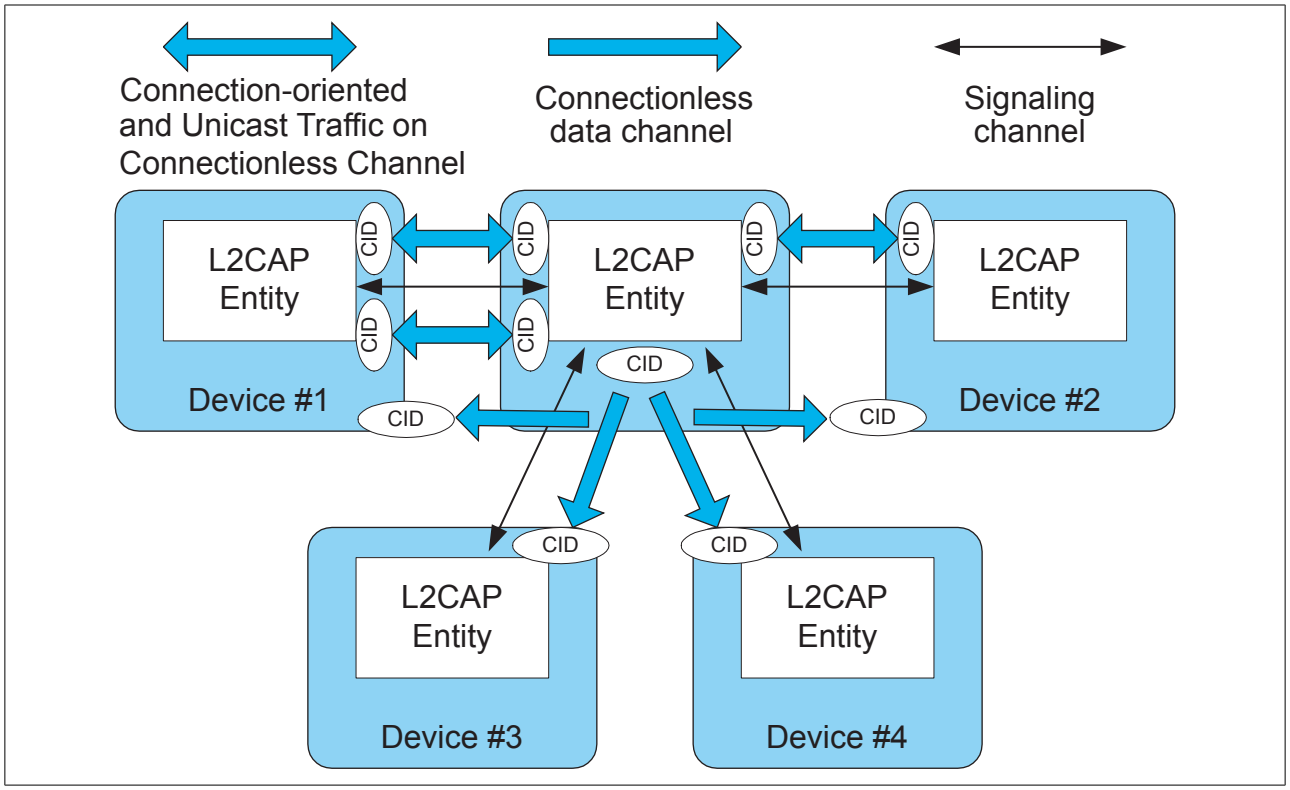



*Figure 2.2: Channels between devices*

Table 2.4 describes the various channel types and their source and destination identifiers. A dynamically allocated CID is allocated to identify, along with the logical link, the local endpoint and shall be in the range 0x0040 to 0xFFFF for ACL-U, 0x0040 to 0x007F for LE-U. Section 6 describes the state machine associated with each connection-oriented channel with a dynamically allocated CID. Section 3.1 and Section 3.3 describe the packet format associated with connection-oriented channels. Section 3.2 describes the packet format associated with the connectionless channel.

| Channel Type | Local CID (sending) | Remote CID (receiving) |
|---|---|---|
| Connection-oriented | Dynamically allocated and fixed | Dynamically allocated and fixed |
| Connectionless data | 0x0002 (fixed) | 0x0002 (fixed) |
| L2CAP Signaling | 0x0001 and 0x0005 (fixed) | 0x0001 and 0x0005 (fixed) |

*Table 2.4: Types of channel identifiers*

## 2.3 Operation between layers

L2CAP implementations should follow the general architecture described below. L2CAP implementations transfer data between upper layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation shall also support a set of signaling commands for use between L2CAP implementations. L2CAP implementations should

also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is implementation specific.

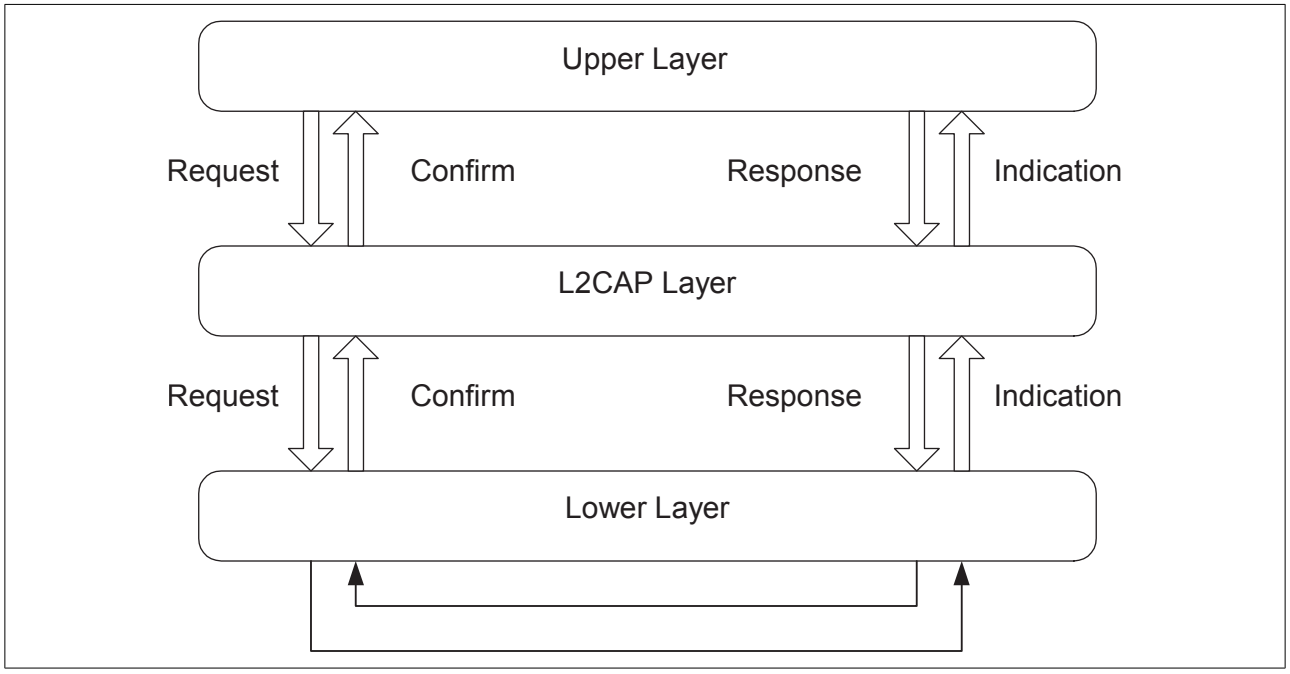



*Figure 2.3: L2CAP transaction model*

## 2.4 Modes of operation

L2CAP channels may operate in one of several different modes as selected for each L2CAP channel.

The modes are:

- Basic L2CAP mode (BR/EDR and some LE fixed channels only)
- Flow Control mode (BR/EDR only)
- Retransmission mode (BR/EDR only)
- Enhanced Retransmission mode (BR/EDR only)
- Streaming mode (BR/EDR only)
- LE Credit Based Flow Control mode (LE only)
- Enhanced Credit Based Flow Control mode (BR/EDR and LE)

Support for Basic L2CAP mode is mandatory. Support for any other mode is optional.

The modes are enabled using the configuration procedure described in Section 7.1. The Basic L2CAP mode shall be the default mode, which is used when no other

mode is agreed. Enhanced Retransmission mode shall be used for ACL-U logical links operating as described in Section 7.10. Enhanced Retransmission mode should be enabled for reliable channels created over ACL-U logical links not operating as described in Section 7.10. Streaming mode shall be used for ACL-U logical links operating as described in Section 7.10. Streaming mode should be enabled for streaming applications created over ACL-U logical links not operating as described in Section 7.10. Enhanced Retransmission mode, Enhanced Credit Based Flow Control mode, or Streaming mode should be enabled when supported by both L2CAP entities. Flow Control mode and Retransmission mode shall only be enabled when communicating with L2CAP entities that do not support Enhanced Retransmission mode, Enhanced Credit Based Flow Control mode, or Streaming mode.

In Flow Control mode, Retransmission mode, and Enhanced Retransmission mode, PDUs exchanged with a peer entity are numbered and acknowledged. The sequence numbers in the PDUs are used to control buffering, and a TxWindow size is used to limit the required buffer space and/or provide a method for flow control.

In Flow Control mode no retransmissions take place, but missing PDUs are detected and can be reported as lost.

In Retransmission mode a timer is used to ensure that all PDUs are delivered to the peer, by retransmitting PDUs as needed. A go-back-n repeat mechanism is used to simplify the protocol and limit the buffering requirements.

Enhanced Retransmission mode is similar to Retransmission mode. It adds the ability to set a POLL bit to solicit a response from a remote L2CAP entity, adds the SREJ S-frame to improve the efficiency of error recovery and adds an RNR S-frame to replace the R-bit for reporting a local busy condition.

Streaming mode is for real-time isochronous traffic. PDUs are numbered but are not acknowledged. A finite flush timeout is set on the sending side to flush packets that are not sent in a timely manner. On the receiving side if the receive buffers are full when a new PDU is received then a previously received PDU is overwritten by the newly received PDU. Missing PDUs can be detected and reported as lost. TxWindow size is not used in Streaming mode.

LE Credit Based Flow Control mode is used for LE L2CAP connection-oriented channels for flow control using a credit based scheme for L2CAP data (i.e. not signaling packets).

Enhanced Credit Based Flow Control mode is used for L2CAP connection-oriented channels on both LE and BR/EDR for flow control using a credit-based scheme for L2CAP data (i.e. not signaling packets). A BR/EDR/LE device that implements Enhanced Credit Based Flow Control mode may support it on BR/EDR only, on LE only, or on both.

Enhanced Retransmission mode should be used instead of Enhanced Credit Based Flow Control mode for data being transmitted over BR/EDR.

Care should be taken in selecting the parameters used for Enhanced Retransmission mode and Streaming mode when they are used beneath legacy profile implementations to ensure that performance is not negatively impacted relative to the performance achieved when using the same profile with Basic mode on an ACL-U logical link. It may be preferable to configure Basic mode to minimize the risk of negative performance impacts.

## 2.5 Mapping channels to logical links

L2CAP maps channels to Controller logical links, which in turn run over Controller physical links. All logical links going between a local Controller and remote Controller run over a single physical link. There is one ACL-U logical link per BR/EDR physical link and one LE-U logical link per LE physical link.

All Best Effort and Guaranteed channels going over a BR/EDR physical link between two devices shall be mapped to a single ACL-U logical link. All channels going over an LE physical link between two devices shall be treated as best effort and mapped to a single LE-U logical link.

When a Guaranteed channel is created, a corresponding Guaranteed logical link shall be created to carry the channel traffic. Creation of a Guaranteed logical link involves admission control. Admission control is verifying that the guarantee can be achieved without compromising existing guarantees. For a BR/EDR Controller, admission control (creation of a Guaranteed logical link) shall be performed by the L2CAP layer.

# 3 DATA PACKET FORMAT

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless. All channels other than the L2CAP connectionless channel (CID 0x0002) and the two L2CAP signaling channels (CIDs 0x0001 and 0x0005) are connection-oriented. All L2CAP layer packet fields shall use little-endian byte order with the exception of the information payload field. The endianness of higher layer protocols encapsulated within L2CAP information payload is protocol-specific.

If a PDU is received on a CID that is not assigned or is reserved for future use on the logical link type, the recipient shall ignore that PDU.

In all L2CAP PDUs, the PDU Length field indicates the size of the entire L2CAP PDU in octets, omitting the 4 octets of the Basic L2CAP header. Therefore a PDU cannot be more than 65539 octets long.

In PDUs that contain an Information Payload field, the number of octets in that field (the payload size) shall not be greater than the peer device's MPS for the channel.

## 3.1 Connection-oriented channels in Basic L2CAP mode

Figure 3.1 illustrates the format of the L2CAP PDU used on connection-oriented channels. In basic L2CAP mode, the L2CAP PDU on a connection-oriented channel is also referred to as a "B-frame".

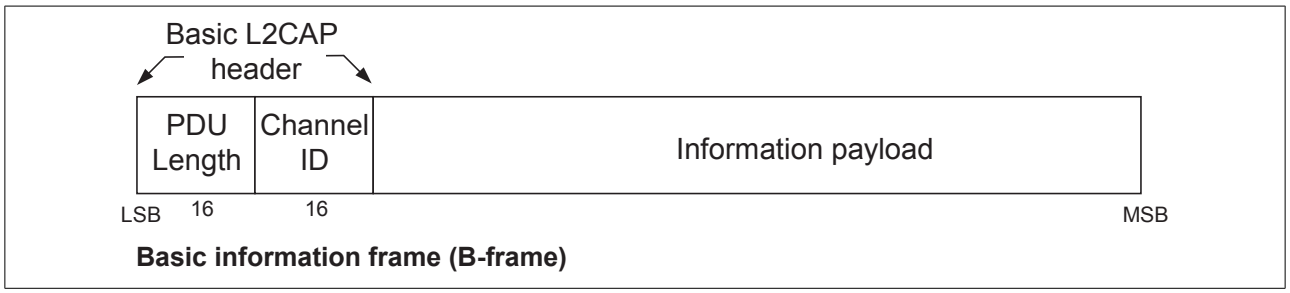



*Figure 3.1: L2CAP PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits)*

The fields shown are:

- *PDU Length: 2 octets (16 bits)*

  For B-frames, the PDU Length equals the payload size and can be up to 65535 octets. The PDU Length field is used for recombination and serves as a simple integrity check of the recombined L2CAP packet on the receiving end.

- *Channel ID: 2 octets*

  The channel ID (CID) identifies the destination channel endpoint of the packet.

- *Information payload: 0 to 65535 octets*

  This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The payload size shall not be greater than the peer device's MTU for the channel. The MTU for channels with dynamically allocated CIDs is determined during channel configuration (see Section 5.1). The minimum supported MTU values for the signaling PDUs are shown in Table 4.1.

## 3.2 Connectionless data channel in Basic L2CAP mode

Figure 3.2 illustrates the L2CAP PDU format within a connectionless data channel. Here, the L2CAP PDU is also referred to as a "G-frame".

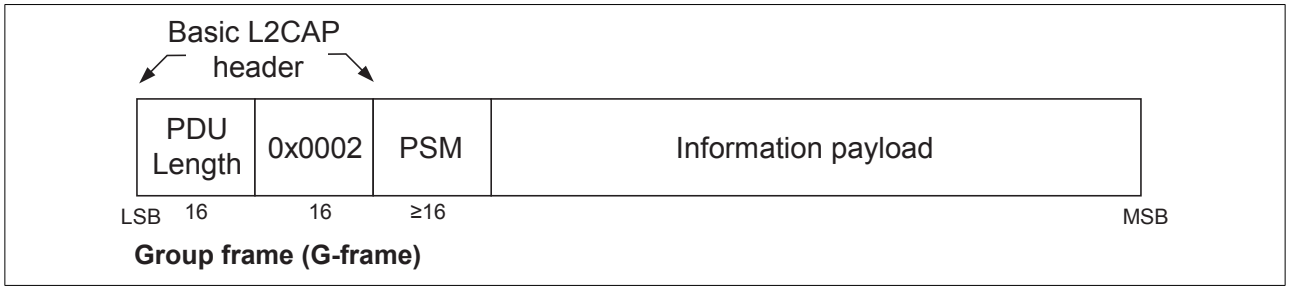



*Figure 3.2: L2CAP PDU format on the connectionless channel*

The fields shown are:

- *PDU Length: 2 octets*

  For G-frames, the PDU Length equals the payload size plus the number of octets in the PSM.

- *Channel ID: 2 octets*

  Channel ID (0x0002) reserved for connectionless traffic.

- *Protocol/Service Multiplexer (PSM): 2 octets (minimum)*

  For information on the PSM field see Section 4.2.

- *Information payload: 0 to 65533 octets*

  This parameter contains the payload information to be distributed to all Peripherals in the piconet for broadcast connectionless traffic, or to a specific remote device for data sent via the L2CAP connectionless channel. The payload size shall not be greater than the peer device's MTU for the channel. Implementations shall support a connectionless MTU ($MTU_{cnl}$) of 48 octets on the connectionless channel. Devices may also explicitly change to a larger or smaller connectionless MTU ($MTU_{cnl}$).

Note: The maximum size of the Information payload field decreases accordingly if the PSM field is extended beyond the two octet minimum.

## 3.3 Connection-oriented channel in Retransmission/Flow Control/ Streaming modes

To support flow control, retransmissions, and streaming, L2CAP PDU types with protocol elements in addition to the Basic L2CAP header are defined. The information frames (I-frames) are used for information transfer between L2CAP entities. The supervisory frames (S-frames) are used to acknowledge I-frames and request retransmission of I-frames. Figure 3.3 illustrates these frames.

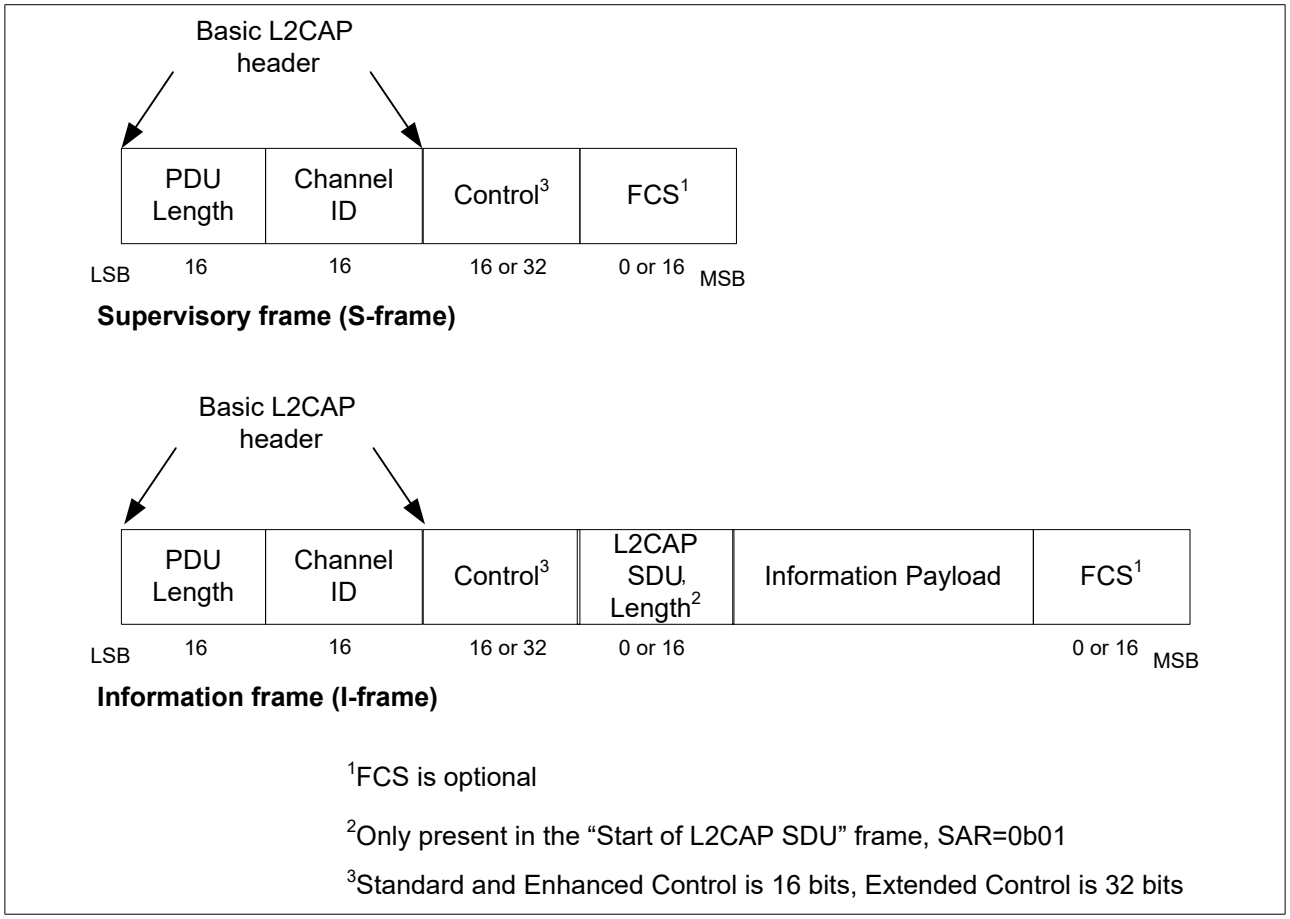



*Figure 3.3: L2CAP PDU formats in Flow Control and Retransmission modes*

### 3.3.1 L2CAP header fields

- *PDU Length: 2 octets*

  For I-frames and S-frames, the PDU Length equals the sum of the octet lengths of the Control, L2CAP SDU Length (when present), Information Payload, and frame check sequence (FCS) (when present) fields. The PDU Length field of an S-frame therefore equals 2, 4, or 6.

The maximum number of Information octets (the payload size) in one I-frame is based on which fields are present and the type of the Control Field. The maximum number of Information octets in an I-frame with a Standard Control field is as follows:

| | |
|---|---|
| L2CAP SDU Length present and FCS present | 65529 octets |
| L2CAP SDU Length present and FCS not present | 65531 octets |
| L2CAP SDU Length not present and FCS present | 65531 octets |
| L2CAP SDU Length not present and FCS not present | 65533 octets |

The maximum number of Information octets in an I-frame with an Extended Control field is as follows:

| | |
|---|---|
| L2CAP SDU Length present and FCS present | 65527 octets |
| L2CAP SDU Length present and FCS not present | 65529 octets |
| L2CAP SDU Length not present and FCS present | 65529 octets |
| L2CAP SDU Length not present and FCS not present | 65531 octets |

- *Channel ID: 2 octets*

  This field contains the Channel Identification (CID).

### 3.3.2 Control field

The Control Field identifies whether the frame is an S-frame or I-frame and contains various information about the frame. There are three different Control Field formats: the Standard Control Field, the Enhanced Control Field, and the Extended Control Field. Which format is used is determined by the mode and is not indicated within the frame. The Standard Control Field shall be used for Retransmission mode and Flow Control mode. The Enhanced and Extended Control Fields shall be used for Enhanced Retransmission mode and Streaming mode. The Enhanced Control Fields shall be used until the first successful use of the Extended Window Size option (see Section 5.7) and Extended Control Fields thereafter. The Control Field will contain sequence numbers where applicable. Its coding is shown in Figure 3.4 to Figure 3.9. There are two different frame types, Information frame types and Supervisory frame types. Information and Supervisory frames types are distinguished by the least significant bit in the Control Field.

- *Information frame format (I-frame)*

  The I-frames are used to transfer information between L2CAP entities. Each I-frame has a TxSeq(Send sequence number), ReqSeq(Receive sequence number) which can acknowledge additional I-frames received by the data Link Layer entity. Each I-frame with a Standard Control field has a retransmission bit (R bit) that affects whether I-frames are retransmitted. Each I-frame with an Enhanced Control Field or an Extended Control Field has an F-bit that is used in Poll/Final bit functions.

The SAR field in the I-frame is used for segmentation and reassembly control. The L2CAP SDU Length field specifies the length of an SDU, including the aggregate length across all segments if segmented.

- *Supervisory frame format (S-frame)*

  S-frames are used to acknowledge I-frames and request retransmission of I-frames. Each S-frame has an ReqSeq sequence number which may acknowledge additional I-frames received by the data Link Layer entity. Each S-frame with a Standard Control Field has a retransmission bit (R bit) that affects whether I-frames are retransmitted. Each S-frame with an Enhanced Control field or an Extended Control Field has a Poll bit (P-bit) and a Final bit (F-bit) and does not have an R-bit.

  Defined types of S-frames are RR (Receiver Ready), REJ (Reject), RNR (Receiver Not Ready) and SREJ (Selective Reject).

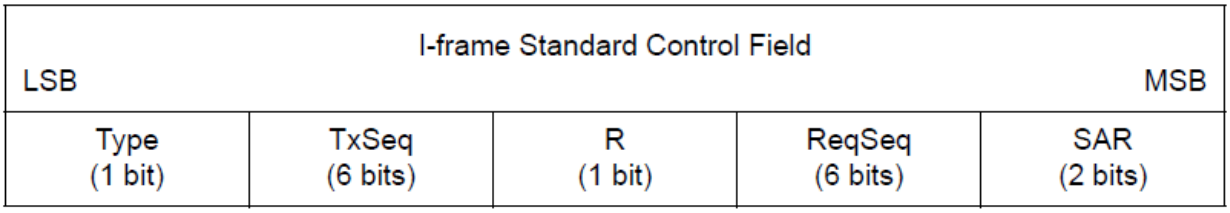



*Figure 3.4: I-frame Standard Control Field format*

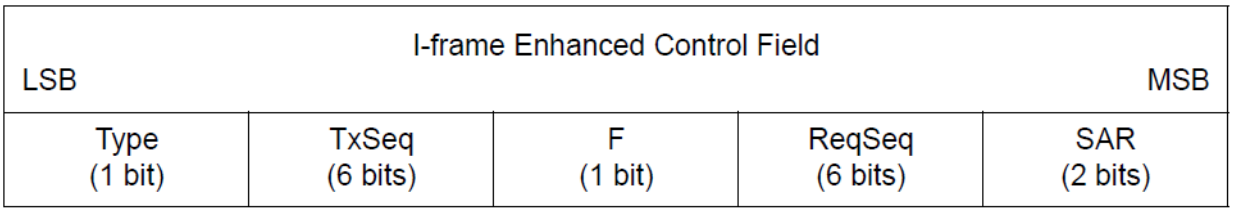



*Figure 3.5: I-frame Enhanced Control Field format*

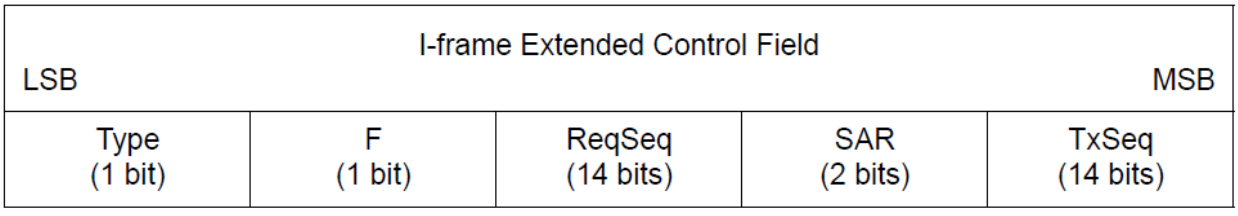



*Figure 3.6: I-frame Extended Control Field format*

| S-frame Standard Control Field | | | | | | |
|---|---|---|---|---|---|---|
| LSB | | | | | | MSB |
| Type (1 bit) | RFU (1 bit) | S (2 bits) | RFU (3 bits) | R (1 bit) | ReqSeq (6 bits) | RFU (2 bits) |

*Figure 3.7: S-frame Standard Control Field format*

| S-frame Enhanced Control Field | | | | | | | |
|---|---|---|---|---|---|---|---|
| LSB | | | | | | | MSB |
| Type (1 bit) | RFU (1 bit) | S (2 bits) | P (1 bit) | RFU (2 bits) | F (1 bit) | ReqSeq (6 bits) | RFU (2 bits) |

*Figure 3.8: S-frame Enhanced Control Field format*

| S-frame Extended Control Field | | | | | |
|---|---|---|---|---|---|
| LSB | | | | | MSB |
| Type (1 bit) | F (1 bit) | ReqSeq (14 bits) | S (2 bits) | P (1 bit) | RFU (13 bits) |

*Figure 3.9: S-frame Extended Control Field format*

- *Type*

  The type bit shall be 0 for an I-frame and 1 for an S-frame.

- *Send Sequence Number - TxSeq*

  The send sequence number is used to number each I-frame, to enable sequencing and retransmission.

- *Receive Sequence Number - ReqSeq*

  The receive sequence number is used by the receiver side to acknowledge I-frames, and in the REJ and SREJ frames to request the retransmission of an I-frame with a specific send sequence number.

- *Retransmission Disable Bit - R*

  The Retransmission Disable bit is used to implement Flow Control. The receiver sets the bit when its internal receive buffer is full, this happens when one or more I-frames have been received but the SDU reassembly function has not yet pulled all the frames received. When the sender receives a frame with the Retransmission

Disable bit set it shall disable the RetransmissionTimer, this causes the sender to stop retransmitting I-frames.

R=0: Normal operation. Sender uses the RetransmissionTimer to control retransmission of I-frames. Sender does not use the MonitorTimer.

R=1: Receiver side requests sender to postpone retransmission of I-frames. Sender monitors signaling with the MonitorTimer. Sender does not use the RetransmissionTimer.

The functions of ReqSeq and R are independent.

- *Segmentation and Reassembly - SAR*

  The SAR bits define whether an L2CAP SDU is segmented. For segmented SDUs, the SAR bits also define which part of an SDU is in this I-frame, thus allowing one L2CAP SDU to span several I-frames.

  An I-frame with SAR=”Start of L2CAP SDU” also contains an L2CAP SDU Length field, specifying the number of information octets in the total L2CAP SDU. The encoding of the SAR bits is shown in Table 3.1.

| | |
|---|---|
| 0b00 | Unsegmented L2CAP SDU |
| 0b01 | Start of L2CAP SDU |
| 0b10 | End of L2CAP SDU |
| 0b11 | Continuation of L2CAP SDU |

*Table 3.1: SAR control element format*

- *Supervisory function - S*

  The S-bits mark the type of S-frame. There are four types defined: RR (Receiver Ready), REJ (Reject), RNR (Receiver Not Ready) and SREJ (Selective Reject). The encoding is shown in Table 3.2.

| | |
|---|---|
| 0b00 | RR - Receiver Ready |
| 0b01 | REJ - Reject |
| 0b10 | RNR - Receiver Not Ready |
| 0b11 | SREJ - Select Reject |

*Table 3.2: S control element format: type of S-frame*

- *Poll - P*

  The P-bit is set to 1 to solicit a response from the receiver. The receiver shall respond immediately with a frame with the F-bit set to 1.

- *Final - F*

  The F-bit is set to 1 in response to an S-frame with the P bit set to 1.

### 3.3.3 L2CAP SDU Length field (2 octets)

When an SDU spans more than one I-frame, the first I-frame in the sequence shall be identified by SAR=0b01=”Start of L2CAP SDU”. The L2CAP SDU Length field shall specify the total number of octets in the SDU and shall not be greater than the peer device's MTU for the channel. The L2CAP SDU Length field shall be present in I-frames with SAR=0b01 (Start of L2CAP SDU) and shall not be used in any other I-frames. When the SDU is unsegmented (SAR=0b00), the L2CAP SDU Length field is not needed and shall not be present.

### 3.3.4 Information Payload field

The information payload field consists of an integer number of octets. The maximum number of octets in this field is the same as the negotiated value of the MPS configuration parameter. The maximum number of octets in this field also depends on which other fields are present (see Section 3.3.1). If SAR=0b00 (Unsegmented L2CAP SDU) then the payload size shall not be greater than the peer device's MTU for the channel.

### 3.3.5 Frame Check Sequence (2 octets)

The Frame Check Sequence (FCS) is 2 octets. This field is mandatory in Retransmission and Flow Control modes. Whether it is present or absent in Enhanced Retransmission and Streaming modes is configurable (see Section 5.5).

The FCS is constructed using the generator polynomial $g(D) = D^{16} + D^{15} + D^2 + 1$ (see Figure 3.10). The 16 bit LFSR is initially loaded with the value 0x0000, as depicted in Figure 3.11. The switch S is set in position 1 while data is shifted in, LSB first for each octet. After the last bit has entered the LFSR, the switch is set in position 2, and the register contents are transmitted from right to left (i.e. starting with position 15, then position 14, etc.). The FCS covers the Basic L2CAP header, Control, L2CAP SDU Length, and Information Payload fields, if present, as shown in Figure 3.3.

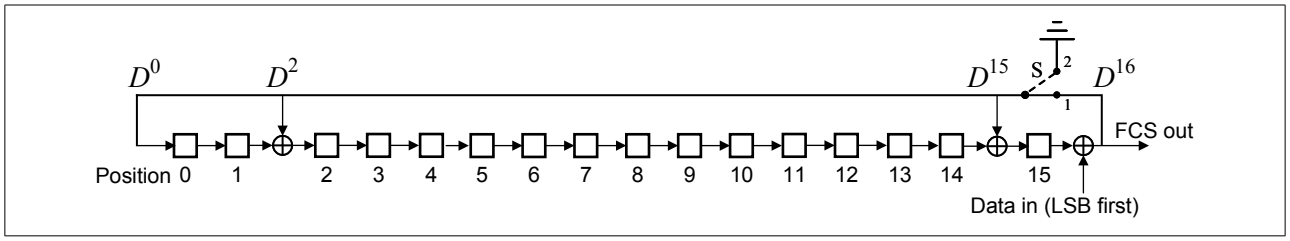



*Figure 3.10: The LFSR circuit generating the FCS*

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| LFSR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

*Figure 3.11: Initial state of the FCS generating circuit*

Examples of FCS calculation, $g(D) = D^{16} + D^{15} + D^2 + 1$:

1. ***I Frame***

   PDU Length = 14
   Channel ID = 0x0040
   Control = 0x0002 (SAR=0b00, ReqSeq=0b000000, R=0, TxSeq=0b000001)
   Information Payload = 00 01 02 03 04 05 06 07 08 09 (10 octets, hexadecimal notation)
   ==> FCS = 0x6138
   ==> Data to Send = 0E 00 40 00 02 00 00 01 02 03 04 05 06 07 08 09 38 61 (hexadecimal notation)

2. ***RR Frame***

   PDU Length = 4
   Channel ID = 0x0040
   Control = 0x0101 (ReqSeq=0b000001, R=0, S=0b00)
   ==> FCS = 0x14D4
   ==> Data to Send = 04 00 40 00 01 01 D4 14 (hexadecimal notation)

### 3.3.6 Invalid Frame Detection (Retransmission and Flow Control modes)

For Retransmission mode and Flow Control mode, a received PDU shall be regarded as invalid if one of the following conditions occurs:

1. Contains an unknown CID.
2. Contains an FCS error.
3. I-frame with a payload size greater than the maximum PDU payload size (MPS).
4. I-frame that has fewer than 8 octets (i.e., PDU Length less than 4).
5. I-frame with SAR=0b01 (Start of L2CAP SDU) that has fewer than 10 octets (i.e., PDU Length less than 6).
6. I-frame with SAR bits that do not correspond to a normal sequence of either unsegmented or start, continuation, end for the given CID.
7. S-frame where the PDU Length field is not equal to 4.

These error conditions may be used for error reporting.

### 3.3.7 Invalid Frame Detection algorithm

For Enhanced Retransmission mode and Streaming mode the following algorithm shall be used for received PDUs. It may be used for Retransmission mode and Flow Control mode:

1. Check the CID. If the PDU contains an unknown CID then it shall be ignored.
2. Check the FCS. If the PDU contains an FCS error then it shall be ignored. If the channel is configured to use "No FCS" then the PDU is considered to have a good FCS (no FCS error).
3. Check the following conditions. If one of the conditions occurs the channel shall be closed or in the case of fixed channels the ACL shall be disconnected.
    a. I-frame with a payload size greater than the maximum PDU payload size (MPS).
    b. I-frame that has fewer than the required number of octets. If the channel is configured to use a Standard or Enhanced Control Field then the required number of octets is 6 if "No FCS" is configured; otherwise, it is 8. If the channel is configured to use the Extended Control Field then the required number of octets is 8 if "No FCS" is configured; otherwise, it is 10.
    c. S-frame where the PDU Length field is invalid. If the channel is configured to use a Standard or Enhanced Control Field then the PDU Length field shall be 2 if "No FCS" is configured; otherwise, the PDU Length field shall be 4. If the channel is configured to use the Extended Control Field then the PDU Length field shall be 4 if "No FCS" is configured; otherwise, the PDU Length field shall be 6.
4. Check the SAR bits. The SAR check is performed after the frame has been successfully received in the correct sequence. The PDU is invalid if one of the following conditions occurs:
    a. I-frame with SAR=0b01 (Start of L2CAP SDU) that has fewer than the required number of octets. If the channel is configured to use a Standard or Enhanced Control field then the required number of octets is 8 if "No FCS" is configured; otherwise, the required number of octets is 10. If the channel is configured to use an Extended Control field then the required number of octets is 10 if "No FCS" is configured; otherwise, the required number of octets is 12.
    b. I-frame with SAR bits that do not correspond to a normal sequence of either unsegmented or start, continuation, end for the given CID.
    c. I-frame with SAR= 0b01 (Start of L2CAP SDU) where the value in the L2CAP SDU Length field exceeds the configured MTU.
5. If the I-frame has been received in the correct sequence and is invalid as described in 4 then the channel shall be closed or in the case of fixed channels the ACL

shall be disconnected. For Streaming mode and Flow Control mode if one or more I-frames are missing from a sequence of I-frames using SAR bits of start, continuation and end then received I-frames in the sequence may be ignored. For Flow Control mode and Streaming mode I-frames received out of sequence with SAR bits of unsegmented may be accepted.

If the algorithm is used for Retransmission mode or Flow control mode then it shall be used instead of Invalid Frame detection described in Section 3.3.6.

These error conditions may be used for error reporting.

## 3.4 Connection-oriented channels in LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode

To support LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode, an L2CAP PDU type with protocol elements in addition to the Basic L2CAP header is defined. In LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode, the L2CAP PDU on a connection-oriented channel is a Credit-based frame (K-frame) as illustrated in Figure 3.12.

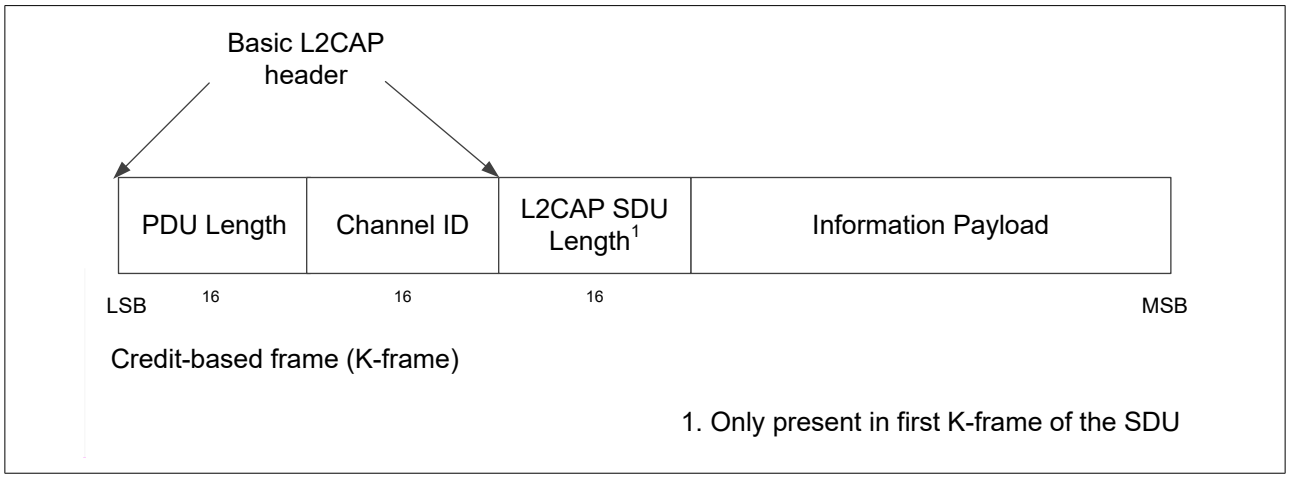



*Figure 3.12: L2CAP PDU format in LE Credit Based Flow Control and Enhanced Credit Based Flow Control modes*

### 3.4.1 L2CAP Header fields

- *PDU Length: 2 octets*

  For K-frames, the PDU Length field equals the payload size plus the octet length of the L2CAP SDU Length (when present).

- *Channel ID: 2 octets*

  The channel ID (CID) identifies the destination channel endpoint of the packet.

### 3.4.2 L2CAP SDU Length field (2 octets)

The first K-frame of the SDU shall contain the L2CAP SDU Length field that shall specify the total number of octets in the SDU. The value shall not be greater than the peer device's MTU for the channel. All subsequent K-frames that are part of the same SDU shall not contain the L2CAP SDU Length field.

### 3.4.3 Information Payload field

The information payload field consists of an integer number of octets.

The number of octets contained in the first K-frame information payload of the SDU is equal to the PDU Length minus 2 octets. All subsequent K-frames of the same SDU contain the number of octets in the information payload equal to the PDU Length.

If the SDU length field value exceeds the receiver's MTU, the receiver shall disconnect the channel. If the payload size of any K-frame exceeds the receiver's MPS, the receiver shall disconnect the channel. If the sum of the payload sizes for the K-frames exceeds the specified SDU length, the receiver shall disconnect the channel.

# 4 SIGNALING PACKET FORMATS

This section describes the signaling commands passed between two L2CAP entities on peer devices. All signaling commands are sent over a signaling channel. The signaling channel for managing channels over ACL-U logical links shall use CID 0x0001 and the signaling channel for managing channels over LE-U logical links shall use CID 0x0005. Signaling channels are available as soon as the lower layer logical transport is set up and L2CAP traffic is enabled. Figure 4.1 illustrates the general format of L2CAP PDUs containing signaling commands (C-frames). Multiple commands may be sent in a single C-frame over fixed channel CID 0x0001 while only one command per C-frame shall be sent over fixed channel CID 0x0005. Commands take the form of requests, responses, and indications. All L2CAP implementations shall support the reception of C-frames with a payload size that does not exceed the signaling MTU. The minimum supported payload size for the C-frame ($MTU_{sig}$) is defined in Table 4.1. L2CAP implementations should not use C-frames that exceed the $MTU_{sig}$ of the peer device. If a device receives a C-frame that exceeds its $MTU_{sig}$ then it shall send an L2CAP_COMMAND_REJECT_RSP packet containing the supported $MTU_{sig}$. Implementations shall be able to handle the reception of multiple commands in an L2CAP packet sent over fixed channel CID 0x0001.

Note: The name of a signalling packet has a suffix indicating its type: _REQ for requests, _RSP for responses, and _IND for indications.

A signaling packet that is not correctly formed is invalid behavior (see [Vol 1] Part E, Section 2.7). Examples of signaling packets that are not correctly formed include:

- The packet is less than 4 octets long.
- The Data Length field is not correct for the type of packet, such as an L2CAP_CONNECTION_REQ packet with a Data Length other than 4 or an L2CAP_INFORMATION_RSP with InfoType = 0x0003, Result = 0x0000, and Data Length not equal to 12.
- The PDU Length of the C-frame does not equal the sum of the sizes of the contained packets.
- The Reason or Status field has an unknown value.
- A C-frame on fixed channel 0x0005 contains more than one signaling packet.

| Logical Link | Minimum Supported Payload Size for the C-frame ($MTU_{sig}$) |
|---|---|
| ACL-U not supporting Extended Flow Specification | 48 octets |
| ACL-U supporting the Extended Flow Specification feature | 672 octets |
| LE-U | 23 octets |

*Table 4.1: Minimum signaling MTU*

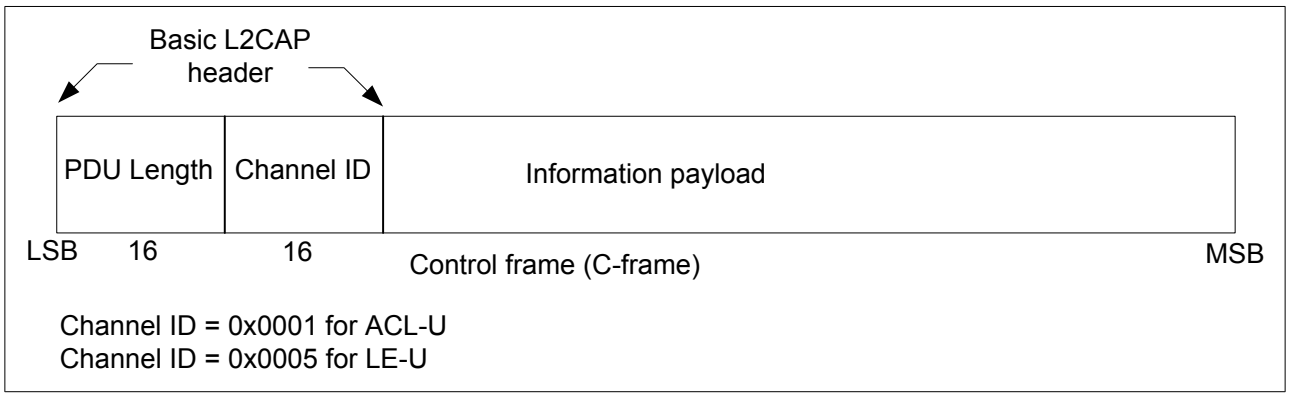



*Figure 4.1: L2CAP PDU format on a signaling channel*

For C-frames, the PDU Length equals the payload size.

Figure 4.2 displays the general format of all signaling commands.

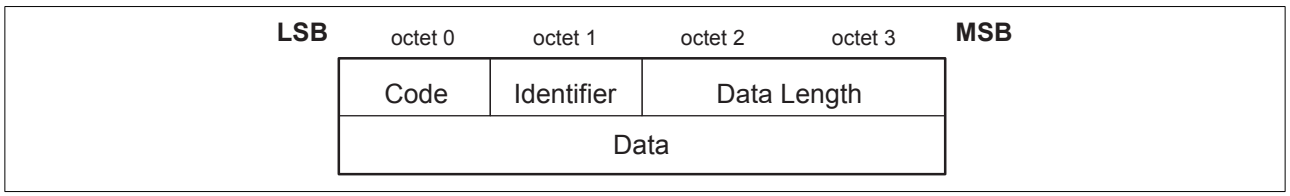



*Figure 4.2: Command format*

The fields shown are:

- *Code (1 octet)*

  The Code field is one octet long and identifies the type of command. When a packet is received with a Code field that is unknown or disallowed on the signaling channel it is received on, an L2CAP_COMMAND_REJECT_RSP packet (defined in Section 4.1) is sent in response.

  Table 4.2 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

| Code | Description | CIDs on which Code is Allowed |
|---|---|---|
| 0x01 | L2CAP_COMMAND_REJECT_RSP | 0x0001 and 0x0005 |
| 0x02 | L2CAP_CONNECTION_REQ | 0x0001 |
| 0x03 | L2CAP_CONNECTION_RSP | 0x0001 |
| 0x04 | L2CAP_CONFIGURATION_REQ | 0x0001 |
| 0x05 | L2CAP_CONFIGURATION_RSP | 0x0001 |
| 0x06 | L2CAP_DISCONNECTION_REQ | 0x0001 and 0x0005 |
| 0x07 | L2CAP_DISCONNECTION_RSP | 0x0001 and 0x0005 |
| 0x08 | L2CAP_ECHO_REQ | 0x0001 |
| 0x09 | L2CAP_ECHO_RSP | 0x0001 |
| 0x0A | L2CAP_INFORMATION_REQ | 0x0001 |
| 0x0B | L2CAP_INFORMATION_RSP | 0x0001 |
| 0x0C to 0x11 | Previously used | None |
| 0x12 | L2CAP_CONNECTION_PARAMETER_UPDATE_REQ | 0x0005 |
| 0x13 | L2CAP_CONNECTION_PARAMETER_UPDATE_RSP | 0x0005 |
| 0x14 | L2CAP_LE_CREDIT_BASED_CONNECTION_REQ | 0x0005 |
| 0x15 | L2CAP_LE_CREDIT_BASED_CONNECTION_RSP | 0x0005 |
| 0x16 | L2CAP_FLOW_CONTROL_CREDIT_IND | 0x0001 and 0x0005 |
| 0x17 | L2CAP_CREDIT_BASED_CONNECTION_REQ | 0x0001 and 0x0005 |
| 0x18 | L2CAP_CREDIT_BASED_CONNECTION_RSP | 0x0001 and 0x0005 |
| 0x19 | L2CAP_CREDIT_BASED_RECONFIGURE_REQ | 0x0001 and 0x0005 |
| 0x1A | L2CAP_CREDIT_BASED_RECONFIGURE_RSP | 0x0001 and 0x0005 |
| Other | Reserved for future use | Any |

*Table 4.2: Signaling command codes*

- *Identifier (1 octet)*

  The Identifier field is one octet long and matches responses with requests. The requesting device sets this field and the responding device uses the same value in its response. Within each signaling channel a different Identifier shall be used for each successive request or indication. Following the original transmission of an Identifier in a command, the Identifier may be recycled if all other Identifiers have subsequently been used.

  RTX and ERTX timers are used to determine when the remote end point is not responding to signaling requests. On the expiration of a RTX or ERTX timer, the same identifier shall be used if a duplicate request is re-sent as stated in Section 6.2.

A device receiving a duplicate request on a particular signaling channel should reply with a duplicate response on the same signaling channel. A command response with an invalid identifier or duplicate response or indication shall be silently discarded. Signaling identifier 0x00 is an invalid identifier and shall never be used in any command.

- *Data Length (2 octets)*

  The Data Length field is two octets long and indicates the size in octets of the data field of the command only, i.e., it does not cover the Code, Identifier, and Data Length fields.

- *Data (0 or more octets)*

  The Data field is variable in length. The Code field determines the format of the Data field. The Data Length field specifies the length of the Data field.

## 4.1 L2CAP_COMMAND_REJECT_RSP (code 0x01)

An L2CAP_COMMAND_REJECT_RSP packet shall be sent in response to a command packet with an unknown command code or when sending the corresponding response is inappropriate. Figure 4.3 defines the format of the packet. The identifier shall match the identifier of the command packet being rejected. Implementations shall always send these packets in response to unidentified signaling packets. L2CAP_COMMAND_REJECT_RSP packets should not be sent in response to an identified response packet.

When multiple commands are included in an L2CAP packet and the packet exceeds the signaling MTU ($MTU_{sig}$) of the receiver, a single L2CAP_COMMAND_REJECT_RSP packet shall be sent in response. The identifier shall match the first request command in the L2CAP packet. If only responses are recognized, the packet shall be silently discarded.

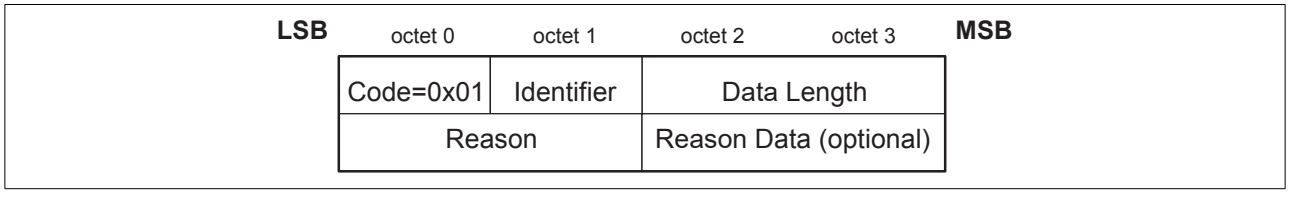



*Figure 4.3: L2CAP_COMMAND_REJECT_RSP packet*

The data fields are:

- *Reason (2 octets)*

  The Reason field describes why the request packet was rejected, and is set to one of the Reason codes in Table 4.3.

| Reason value | Description |
|---|---|
| 0x0000 | Command not understood |
| 0x0001 | Signaling MTU exceeded |
| 0x0002 | Invalid CID in request |
| Other | Reserved for future use |

*Table 4.3: Reason code descriptions*

- *Reason Data (0 or more octets)*

  The length and content of the Reason Data field depends on the Reason code. If the Reason code is 0x0000, “Command not understood”, no Reason Data field is used. If the Reason code is 0x0001, “Signaling MTU Exceeded”, the 2-octet Reason Data field represents the maximum signaling MTU the sender of this packet can accept.

  If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. The Reason Data field shall be 4 octets containing the local (first) and remote (second) channel endpoints (relative to the sender of the L2CAP_COMMAND_REJECT_RSP packet) of the disputed channel. The remote endpoint is the source CID from the rejected command. The local endpoint is the destination CID from the rejected command. If the rejected command contains only one of the channel endpoints, the other one shall be replaced by the null CID 0x0000.

| Reason value | Reason Data length | Reason Data value |
|---|---|---|
| 0x0000 | 0 octets | |
| 0x0001 | 2 octets | Actual $MTU_{sig}$ |
| 0x0002 | 4 octets | Requested CIDs |

*Table 4.4: Reason data values*

## 4.2 L2CAP_CONNECTION_REQ (code 0x02)

L2CAP_CONNECTION_REQ packets are sent to create an L2CAP channel between two devices. The L2CAP channel shall be established before configuration begins. Figure 4.4 defines the format of the packet.

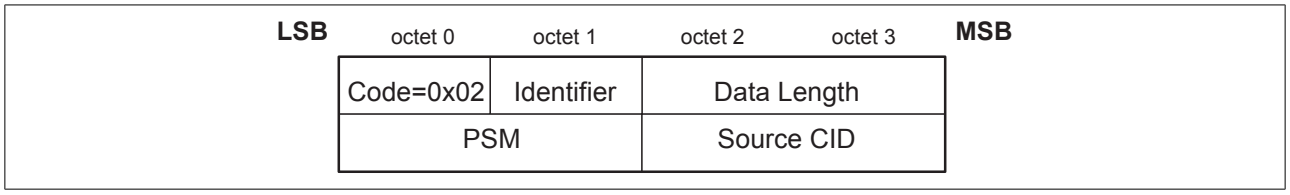



*Figure 4.4: L2CAP_CONNECTION_REQ packet*

The data fields are:

- *Protocol/Service Multiplexer - PSM (2 octets (minimum))*

  The PSM field is at least two octets in length. All PSM values shall have the least significant bit of the most significant octet equal to 0 and the least significant bit of all other octets equal to 1.

  Note: This means that all PSMs are odd numbers and that the end of a PSM can be easily found by searching for the first even octet.

  PSM values are separated into two ranges. Valid values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the Service Discovery protocol (SDP). The dynamically assigned values may be used to support multiple implementations of a particular protocol.

  PSM values in the first range are defined in Assigned Numbers.

| Range | Type | Server Usage | Client Usage |
|---|---|---|---|
| 0x0001 to 0x0EFF (Note [1]) | Fixed, SIG assigned | PSM is fixed for all implementations. | PSM may be obtained via SDP or may be assumed for a fixed service. Protocol used is indicated by the PSM. |
| >0x1000 | Dynamic | PSM may be fixed for a given implementation or may be assigned at the time the service is registered in SDP. | PSM shall be obtained via SDP upon every reconnection. PSM for one direction will typically be different from the other direction. |

*Table 4.5: PSM ranges and usage*

[1] Since PSMs are odd and the least significant bit of the most significant byte is zero, the following ranges do not contain valid PSMs: 0x0100-0x01FF, 0x0300-0x03FF, 0x0500-0x05FF, 0x0700-0x07FF, 0x0900-0x09FF, 0x0B00-0x0BFF, 0x0D00-0x0DFF. All even values are also not valid as PSMs.

- *Source CID - SCID (2 octets)*

  The source CID is two octets in length and represents a channel endpoint on the device sending the request. Once the channel has been configured, data packets flowing to the sender of the request shall be sent to this CID. Thus, the Source CID represents the channel endpoint on the device sending the request and receiving the response. The value of the Source CID shall be from the dynamically allocated range as defined in Table 2.1 and shall not be already allocated to a different channel on the same logical link on the device sending the request.

## 4.3 L2CAP_CONNECTION_RSP (code 0x03)

When a device receives an L2CAP_CONNECTION_REQ packet, it shall send an L2CAP_CONNECTION_RSP packet. Figure 4.5 defines the format of the packet.

A device sending an L2CAP_CONNECTION_REQ packet to an implementation conforming to a version of the specification lower than version 4.2 shall accept an L2CAP_COMMAND_REJECT_RSP (Reason 0x0002 – Invalid CID In Request) packet under conditions now covered by result codes 0x0006 and 0x0007.

If the device sends an L2CAP_CONNECTION_RSP packet with result code "pending", then it shall subsequently send another L2CAP_CONNECTION_RSP (see also [Vol 3] Part C, Section 5.2.2.2).

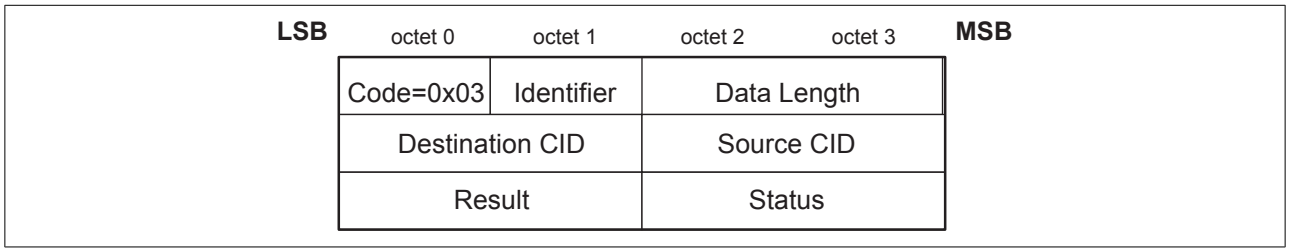



*Figure 4.5: L2CAP_CONNECTION_RSP packet*

The data fields are:

- *Destination Channel Identifier - DCID (2 octets)*

  This field contains the channel endpoint on the device sending this response packet. Thus, the Destination CID represents the channel endpoint on the device receiving the request and sending the response. The value of the Destination CID shall be from the dynamically allocated range as defined in Table 2.1 and shall not be already allocated to a different channel on the same logical link on the device sending the response.

- *Source Channel Identifier - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this response packet. This is copied from the SCID field of the L2CAP_CONNECTION_REQ packet.

- *Result (2 octets)*

  The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed or is pending. A logical channel is established on the receipt of a successful result unless the DCID field is outside of the dynamically allocated range (see Table 2.1) or is already allocated on the device sending the response. Table 4.6 defines values for this field. If the result field does not indicate the connection was successful, the DCID and SCID fields may be invalid and shall be ignored.

| Value | Description |
|---|---|
| 0x0000 | Connection successful |
| 0x0001 | Connection pending |

| Value | Description |
|---|---|
| 0x0002 | Connection refused – PSM not supported |
| 0x0003 | Connection refused – security block |
| 0x0004 | Connection refused – no resources available |
| 0x0006 | Connection refused – invalid Source CID |
| 0x0007 | Connection refused – Source CID already allocated |
| Other | Reserved for future use |

*Table 4.6: Result values for the L2CAP_CONNECTION_RSP packet*

- *Status (2 octets)*

  Only defined for Result = Pending. Indicates the status of the connection. The status is set to one of the values shown in Table 4.7.

| Value | Description |
|---|---|
| 0x0000 | No further information available |
| 0x0001 | Authentication pending |
| 0x0002 | Authorization pending |
| Other | Reserved for future use |

*Table 4.7: Status values for the L2CAP_CONNECTION_RSP packet*

## 4.4 L2CAP_CONFIGURATION_REQ (code 0x04)

L2CAP_CONFIGURATION_REQ packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to re-negotiate this contract whenever appropriate. The contract consists of a set of configuration parameter options defined in Section 5. All parameter options have default values and can have previously agreed values which are values that were accepted in a previous configuration process or in a previous step in the current configuration process. The only parameters that should be included in the L2CAP_CONFIGURATION_REQ packet are those that require different values than the default or previously agreed values.

If no parameters need to be negotiated or specified then no options shall be inserted and the continuation flag (C) shall be set to zero. Any missing configuration parameters are assumed to have their most recently explicitly or implicitly accepted values. Even if all default values are acceptable, an L2CAP_CONFIGURATION_REQ packet with no options shall be sent. Implicitly accepted values are default values for the configuration parameters that have not been explicitly negotiated for the specific channel under configuration.

See Section 7.1 for details of the configuration procedure.

Figure 4.6 defines the format of the packet.

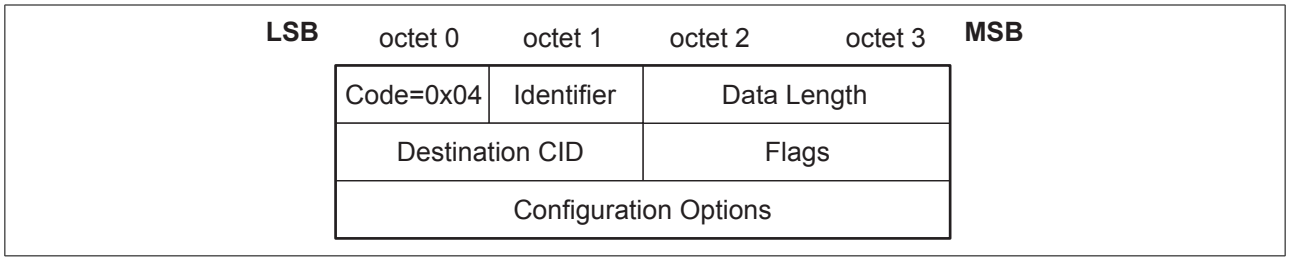



*Figure 4.6: L2CAP_CONFIGURATION_REQ packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field contains the channel endpoint on the device receiving this request packet.

- *Flags (2 octets)*

  Figure 4.7 shows the two-octet Flags field. Note the most significant bit is shown on the left.

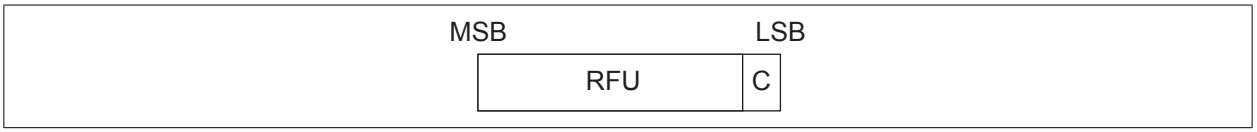



*Figure 4.7: L2CAP_CONFIGURATION_REQ Flags field format*

Only one flag is defined, the Continuation flag (C).

When both L2CAP entities support the Extended Flow Specification option, the Continuation flag shall not be used and shall be set to zero in all L2CAP_CONFIGURATION_REQ and L2CAP_CONFIGURATION_RSP packets.

When all configuration options cannot fit into an L2CAP_CONFIGURATION_REQ packet with a payload size that does not exceed the receiver's $MTU_{sig}$, the options shall be passed in multiple L2CAP_CONFIGURATION_REQ packets. If all options fit into the receiver's $MTU_{sig}$, then they shall be sent in a single L2CAP_CONFIGURATION_REQ packet with the continuation flag set to zero. Each L2CAP_CONFIGURATION_REQ packet shall only contain complete options - partially formed options shall not be sent in a packet. Each L2CAP_CONFIGURATION_REQ packet shall be tagged with a different Identifier and shall be matched with an L2CAP_CONFIGURATION_RSP packet with the same Identifier.

When used in the L2CAP_CONFIGURATION_REQ packet, the continuation flag indicates the responder should expect to receive multiple request packets. The responder shall reply to each L2CAP_CONFIGURATION_REQ packet. The responder may reply to each L2CAP_CONFIGURATION_REQ packet with an L2CAP_CONFIGURATION_RSP packet containing the same option(s)

present in the request (except for those error conditions more appropriate for an L2CAP_COMMAND_REJECT_RSP packet), or the responder may reply with a "Success" L2CAP_CONFIGURATION_RSP packet containing no options, delaying those options until the full request has been received. The L2CAP_CONFIGURATION_REQ packet with the continuation flag cleared shall be treated as the L2CAP_CONFIGURATION_REQ event in the channel state machine.

When used in the L2CAP_CONFIGURATION_RSP packet, the continuation flag shall be set to one if the flag is set to one in the request. If the continuation flag is set to one in the response when the matching request has the flag set to zero, it indicates the responder has additional options to send to the requestor. In this situation, the requestor shall send null-option L2CAP_CONFIGURATION_REQ packets (with continuation flag set to zero) to the responder until the responder replies with an L2CAP_CONFIGURATION_RSP packet where the continuation flag is set to zero. The L2CAP_CONFIGURATION_RSP packet with the continuation flag set to zero shall be treated as the L2CAP_CONFIGURATION_RSP event in the channel state machine.

The result of the configuration transaction is success if all the individual result values are success, and is failure otherwise.

Other flags are reserved for future use.

- *Configuration Options*

  A list of the parameters and their values to be negotiated shall be provided in the Configuration Options field. These are defined in Section 5; in addition, as described in that section, an implementation shall be prepared to receive any number of unknown options. An L2CAP_CONFIGURATION_REQ packet may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the Data Length field is set to 0x0004.

## 4.5 L2CAP_CONFIGURATION_RSP (code 0x05)

L2CAP_CONFIGURATION_RSP packets shall be sent in reply to L2CAP_CONFIGURATION_REQ packets except when the error condition is covered by an L2CAP_COMMAND_REJECT_RSP packet response. Each configuration parameter value (if any is present) in an L2CAP_CONFIGURATION_RSP packet reflects an ‘adjustment’ to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding L2CAP_CONFIGURATION_REQ packet. For example, if an L2CAP_CONFIGURATION_REQ packet relates to traffic flowing from device A to device B, the sender of the L2CAP_CONFIGURATION_RSP packet may adjust this value for the same traffic flowing from device A to device B, but the response cannot adjust the value in the reverse direction.

The options sent in the L2CAP_CONFIGURATION_RSP packet depend on the value in the Result field. Figure 4.8 defines the format of the packet. See also Section 7.1 for details of the configuration process, including use of the result code "pending".

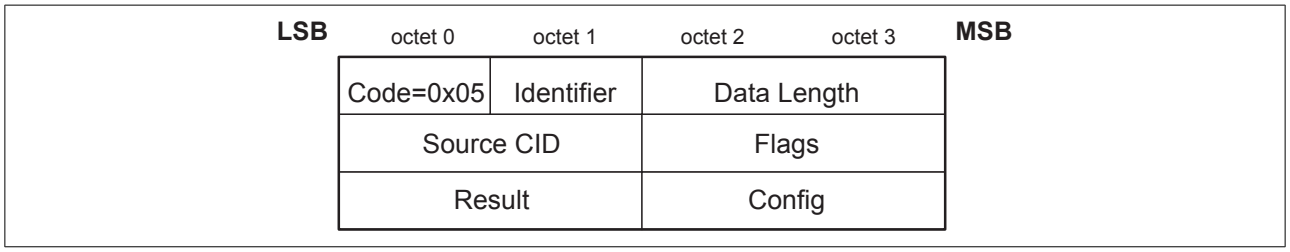



*Figure 4.8: L2CAP_CONFIGURATION_RSP packet*

The data fields are:

- *Source CID - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this response packet. The device receiving the L2CAP_CONFIGURATION_RSP packet shall check that the Identifier field matches the same field in the corresponding L2CAP_CONFIGURATION_REQ packet and the SCID matches its local CID paired with the original DCID.

- *Flags (2 octets)*

  Figure 4.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

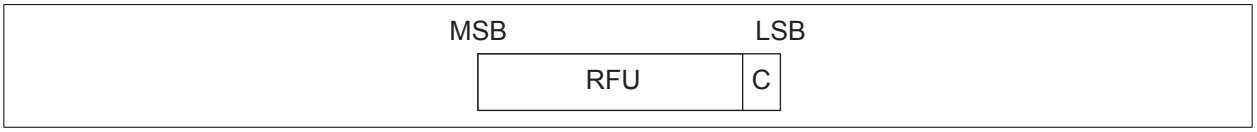



*Figure 4.9: L2CAP_CONFIGURATION_RSP Flags field format*

  Only one flag is defined, the Continuation flag (C).

  When both L2CAP entities support the Extended Flow Specification option, the Continuation flag shall not be used and shall be set to zero in all L2CAP_CONFIGURATION_REQ and L2CAP_CONFIGURATION_RSP packets.

  More L2CAP_CONFIGURATION_RSP packets will follow when C is set to one. This flag indicates that the parameters included in the response are a partial subset of parameters being sent by the device sending the L2CAP_CONFIGURATION_RSP packet.

  The other flag bits are reserved for future use.

- *Result (2 octets)*

  The Result field indicates whether or not the request was acceptable. See Table 4.8 for possible result codes.

| Result | Description |
|---|---|
| 0x0000 | Success |
| 0x0001 | Failure – unacceptable parameters |
| 0x0002 | Failure – rejected (no reason provided) |
| 0x0003 | Failure – unknown options |
| 0x0004 | Pending |
| 0x0005 | Failure - flow spec rejected |
| Other | Reserved for future use |

*Table 4.8: L2CAP_CONFIGURATION_RSP result codes*

- *Configuration options*

  This field contains the list of parameters being configured. These are defined in Section 5. On a successful result (Result=0x0000) and pending result (Result=0x0004), these parameters contain the return values for any wild card parameter values (see Section 5.3) and “adjustments” to non-negotiated configuration parameter values contained in the request. A response with a successful result is also referred to as a positive response.

  On an unacceptable parameters failure (Result=0x0001) the rejected parameters shall be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters in the L2CAP_CONFIGURATION_REQ packet are assumed to have their default value or previously agreed value and they too shall be included in the L2CAP_CONFIGURATION_RSP packet if they need to be changed. A response with an unacceptable parameters failure is also referred to as a negative response.

  On an unknown option failure (Result=0x0003), the option(s) that contain an option type field that is not understood by the recipient of the L2CAP_CONFIGURATION_REQ packet shall be included in the L2CAP_CONFIGURATION_RSP packet unless they are hints. Hints are those options in the L2CAP_CONFIGURATION_REQ packet that are skipped if not understood (see Section 5). Hints shall not be included in the L2CAP_CONFIGURATION_RSP packet and shall not be the sole cause for rejecting the L2CAP_CONFIGURATION_REQ packet.

  On a flow spec rejected failure (Result=0x0005), an Extended Flow Spec option may be included to reflect the QoS level that would be acceptable (see Section 7.1.3).

  The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is implementation specific.

## 4.6 L2CAP_DISCONNECTION_REQ (code 0x06)

Terminating an L2CAP channel requires that an L2CAP_DISCONNECTION_REQ packet be sent and acknowledged by an L2CAP_DISCONNECTION_RSP packet. Figure 4.10 defines the format of the packet. The receiver shall not initiate a disconnection if the source or destination CIDs do not match.

Once an L2CAP_DISCONNECTION_REQ packet is issued, all incoming data in transit on this L2CAP channel shall be discarded and any new additional outgoing data shall be discarded. Once an L2CAP_DISCONNECTION_REQ packet for a channel has been received, all data queued to be sent out on that channel shall be discarded.

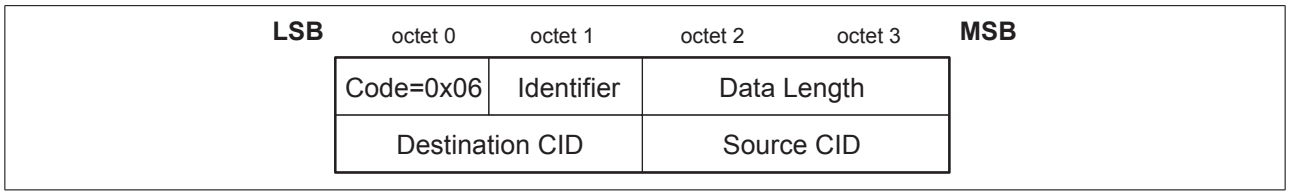



*Figure 4.10: L2CAP_DISCONNECTION_REQ packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device receiving this request.

- *Source CID - SCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device sending this request.

The SCID and DCID are relative to the sender of this request and shall match those of the channel to be disconnected. If the DCID is not recognized by the receiver of this message, an L2CAP_COMMAND_REJECT_RSP packet with ‘invalid CID’ result code shall be sent in response. If the receiver finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.

## 4.7 L2CAP_DISCONNECTION_RSP (code 0x07)

L2CAP_DISCONNECTION_RSP packets shall be sent in response to each valid L2CAP_DISCONNECTION_REQ packet. Figure 4.11 defines the format of the packet.

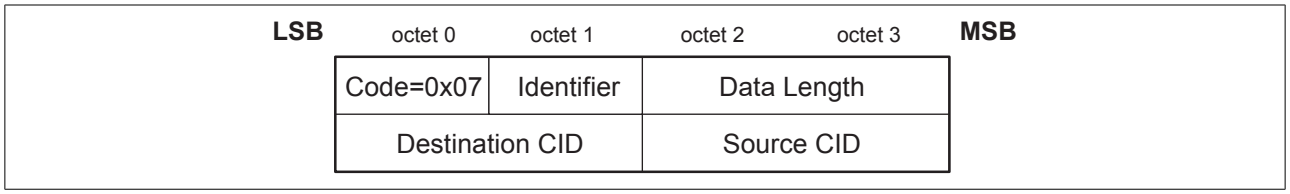



*Figure 4.11: L2CAP_DISCONNECTION_RSP packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field identifies the channel endpoint on the device sending the L2CAP_DISCONNECTION_RSP packet.

- *Source CID - SCID (2 octets)*

  This field identifies the channel endpoint on the device receiving the L2CAP_DISCONNECTION_RSP packet.

  The DCID and the SCID (which are relative to the sender of the L2CAP_DISCONNECTION_REQ packet), and the Identifier fields shall match those of the corresponding L2CAP_DISCONNECTION_REQ packet. If the CIDs do not match, the L2CAP_DISCONNECTION_RSP packet should be silently discarded at the receiver.

## 4.8 L2CAP_ECHO_REQ (code 0x08)

L2CAP_ECHO_REQ packets are used to request a response from a remote L2CAP entity. Figure 4.12 defines the format of the packet. These requests may be used for testing the link or for passing vendor specific information using the optional data field. L2CAP entities shall respond to a valid L2CAP_ECHO_REQ packet with an L2CAP_ECHO_RSP packet. The Echo Data field is optional and implementation specific. L2CAP entities should ignore the contents of this field if present.

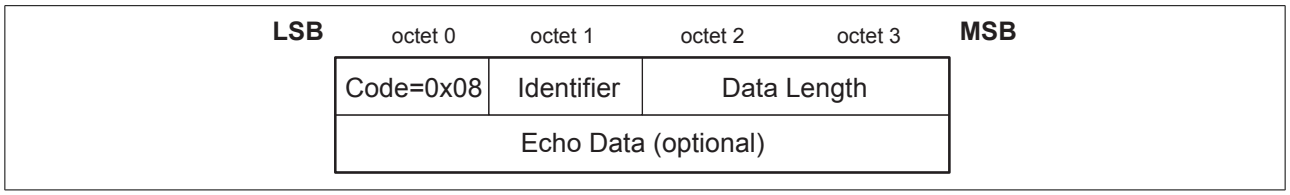



*Figure 4.12: L2CAP_ECHO_REQ packet*

## 4.9 L2CAP_ECHO_RSP (code 0x09)

An L2CAP_ECHO_RSP packet shall be sent upon receiving a valid L2CAP_ECHO_REQ packet. Figure 4.13 defines the format of the packet. The identifier in the L2CAP_ECHO_RSP packet shall match the identifier sent in the

L2CAP_ECHO_REQ packet. The Echo Data field is optional and implementation specific. It may contain the contents of the Echo Data field in the L2CAP_ECHO_REQ packet, different data, or no data at all. L2CAP entities should ignore the contents of this field if present.

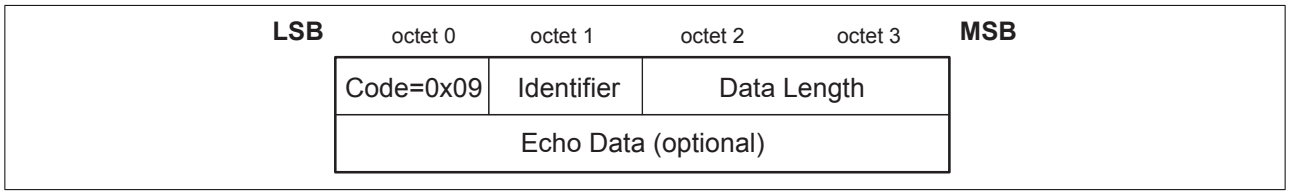



*Figure 4.13: L2CAP_ECHO_RSP packet*

## 4.10 L2CAP_INFORMATION_REQ (code 0x0A)

L2CAP_INFORMATION_REQ packets are used to request implementation specific information from a remote L2CAP entity. Figure 4.14 defines the format of the packet. L2CAP implementations shall respond to a valid L2CAP_INFORMATION_REQ packet with an L2CAP_INFORMATION_RSP packet. It is optional to send L2CAP_INFORMATION_REQ packets.

An L2CAP implementation shall only use optional features or attribute ranges for which the remote L2CAP entity has indicated support through an L2CAP_INFORMATION_RSP packet. Until an L2CAP_INFORMATION_RSP packet which indicates support for optional features or ranges has been received only mandatory features and ranges shall be used.

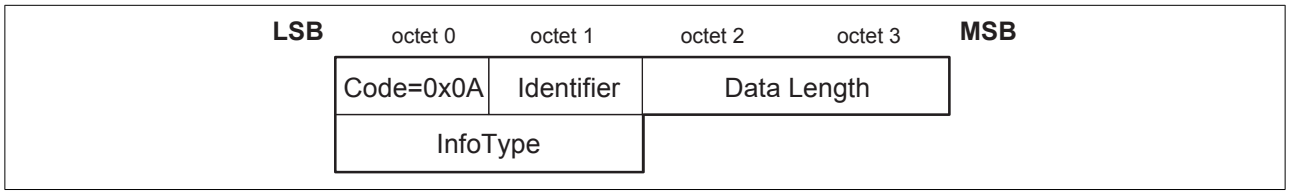



*Figure 4.14: L2CAP_INFORMATION_REQ packet*

The data field is:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information being requested. See Section 4.11 for details on the type of information requested.

| Value | Description |
|---|---|
| 0x0001 | Connectionless MTU |
| 0x0002 | Extended features supported |

| Value | Description |
|---|---|
| 0x0003 | Fixed channels supported over BR/EDR |
| Other | Reserved for future use |

*Table 4.9: InfoType definitions*

L2CAP entities shall not send an L2CAP_INFORMATION_REQ packet with InfoType 0x0003 over fixed channel CID 0x0001 until first verifying that the Fixed Channels bit is set in the Extended feature mask of the remote device. Support for fixed channels is mandatory for devices with BR/EDR/LE or LE Controllers. L2CAP_INFORMATION_REQ and L2CAP_INFORMATION_RSP packets shall not be used over fixed channel CID 0x0005.

## 4.11 L2CAP_INFORMATION_RSP (code 0x0B)

An L2CAP_INFORMATION_RSP packet shall be sent upon receiving a valid L2CAP_INFORMATION_REQ packet. Figure 4.15 defines the format of the packet. The identifier in the L2CAP_INFORMATION_RSP packet shall match the identifier sent in the L2CAP_INFORMATION_REQ packet. The Info field shall contain the value associated with the InfoType field sent in the L2CAP_INFORMATION_REQ packet, or shall be empty if the InfoType is not supported.

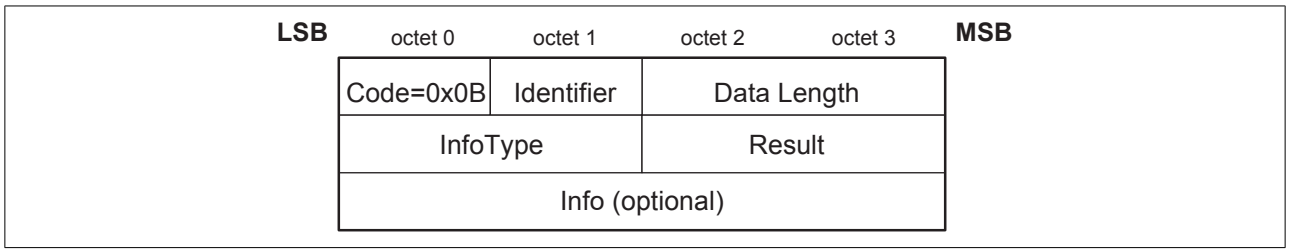



*Figure 4.15: L2CAP_INFORMATION_RSP packet*

The data fields are:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information that was requested. This value shall be copied from the InfoType field in the L2CAP_INFORMATION_REQ packet.

- *Result (2 octets)*

  The Result contains information about the success of the request. If result is "Success," the data field contains the information as specified in Table 4.11. If result is "Not supported," no data shall be returned.

| Value | Description |
|---|---|
| 0x0000 | Success |
| 0x0001 | Not supported |
| Other | Reserved for future use |

*Table 4.10: Result values for the L2CAP_INFORMATION_RSP packet*

- *Info (0 or more octets)*

  The contents of the Info field depends on the InfoType.

  For InfoType = 0x0001, the field contains the remote entity's 2-octet acceptable connectionless MTU. The default value is defined in Section 3.2.

  For InfoType = 0x0002, the Info field contains the 4 octet L2CAP extended feature mask. The feature mask refers to the extended features that the L2CAP entity sending the L2CAP_INFORMATION_RSP packet supports. The feature bits contained in the L2CAP feature mask are specified in Section 4.12.

  For InfoType = 0x0003, the Info field contains an 8 octet bit map that indicates which Fixed L2CAP Channels (i.e., the L2CAP channels that use a CID from 0x0000 to 0x003F) are supported over BR/EDR. A list of available fixed channels is provided in Table 2.1 in Section 2.1. A detailed description of this Info field is given in Section 4.13.

  A device sending an L2CAP_INFORMATION_REQ packet with InfoType = 0x0002 to an implementation conforming to a version lower than 3.0 shall accept an L2CAP_INFORMATION_RSP packet with result code "Not supported" as indicating an all-zero feature mask.

| InfoType | Info | Info length (octets) |
|---|---|---|
| 0x0001 | Connectionless MTU | 2 |
| 0x0002 | Extended feature mask | 4 |
| 0x0003 | Fixed channels supported over BR/EDR | 8 |

*Table 4.11: L2CAP_INFORMATION_RSP Info fields*

## 4.12 Extended Feature Mask

The features are represented as a bit mask in the L2CAP_INFORMATION_RSP packet's Info field (see Section 4.11). For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. All unknown or unassigned feature bits are reserved for future use.

The feature mask shown in Table 4.12 consists of 4 octets (numbered octet 0 to 3), with bit numbers 0 to 7 each.

| No. | Supported feature | Octet | Bit |
|---|---|---|---|
| 0 | Flow control mode | 0 | 0 |
| 1 | Retransmission mode | 0 | 1 |
| 2 | Bi-directional QoS[1] | 0 | 2 |
| 3 | Enhanced Retransmission mode | 0 | 3 |
| 4 | Streaming mode | 0 | 4 |
| 5 | FCS Option[2] | 0 | 5 |
| 6 | Extended Flow Specification for BR/EDR | 0 | 6 |
| 7 | Fixed Channels supported over BR/EDR | 0 | 7 |
| 8 | Extended Window Size | 1 | 0 |
| 9 | Unicast Connectionless Data Reception | 1 | 1 |
| 10 | Enhanced Credit Based Flow Control mode over BR/EDR | 1 | 2 |
| 31 | Reserved for feature mask extension | 3 | 7 |
| All others | Reserved for future use | All others | |

*Table 4.12: Extended feature mask*

[1]Peer side supports upper layer control of the Link Manager's Bi-directional QoS, see Section 5.3 for more details.

[2]Peer side supports negotiating omitting FCS; see Section 5.5 for more details and the required behavior if this bit is not set.

## 4.13 Fixed Channels Supported over BR/EDR

Each fixed channel supported over BR/EDR is represented by a single bit in an 8 octet bit mask. The bit associated with a channel shall be set to 1 if the L2CAP entity supports that channel. The L2CAP Signaling channel is mandatory and therefore the bit associated with that channel shall be set to 1. Table 4.13 shows the format of the bit mask.

| CID | Fixed Channel | Value | Octet | Bit |
|---|---|---|---|---|
| 0x0000 | Null identifier | Shall be set to 0 | 0 | 0 |
| 0x0001 | L2CAP Signaling channel | Shall be set to 1 | 0 | 1 |
| 0x0002 | Connectionless reception | 0 – if not supported<br>1 – if supported | 0 | 2 |
| 0x0003 | Previously used | | 0 | 3 |
| 0x0004 to 0x0006 | Reserved for future use | | 0 | 4-6 |
| 0x0007 | BR/EDR Security Manager | 0 – if not supported<br>1 – if supported | 0 | 7 |

| CID | Fixed Channel | Value | Octet | Bit |
|---|---|---|---|---|
| 0x0008 to 0x003E | Reserved for future use | | 1-6<br>7 | 0-7<br>0-6 |
| 0x003F | Previously used | | 7 | 7 |
| Other | Reserved for future use | | Other | |

*Table 4.13: Fixed Channels Supported bit mask*

An L2CAP entity shall not transmit on any fixed channel (with the exception of the L2CAP Signaling channel) until it has received a Fixed Channels Supported InfoType from the peer L2CAP entity indicating support for that channel, or has received a valid packet from the remote device on that fixed channel. All packets received on a non-supported fixed channel shall be ignored.

## 4.14 [This section is no longer used]

## 4.15 [This section is no longer used]

## 4.16 [This section is no longer used]

## 4.17 [This section is no longer used]

## 4.18 [This section is no longer used]

## 4.19 [This section is no longer used]

## 4.20 L2CAP_CONNECTION_PARAMETER_UPDATE_REQ (code 0x12)

This command shall only be sent from the Peripheral to the Central and only if one or more of the Peripheral's Controller, the Central's Controller, the Peripheral's Host and the Central's Host do not support the Connection Parameters Request Link Layer Control procedure ([Vol 6] Part B, Section 5.1.7). If a Peripheral's Host receives an L2CAP_CONNECTION_PARAMETER_UPDATE_REQ packet it shall respond with an L2CAP_COMMAND_REJECT_RSP packet with reason 0x0000 (Command not understood). Figure 4.16 defines the format of the packet.

The L2CAP_CONNECTION_PARAMETER_UPDATE_REQ packet allows the Peripheral's Host to request a set of new connection parameters. When the Central's Host receives an L2CAP_CONNECTION_PARAMETER_UPDATE_REQ packet, depending on the parameters of other connections, the Central's Host may accept the requested parameters and deliver the requested parameters to its Controller or reject the request. In devices supporting HCI, the Central's Host delivers the requested parameters to its Controller using the HCI_LE_Connection_Update command (see [Vol 4] Part E, Section 7.8.18). If the Central's Host accepts the requested

parameters it shall send the L2CAP_CONNECTION_PARAMETER_UPDATE_RSP packet with result 0x0000 (Parameters accepted) otherwise it shall set the result to 0x0001 (request rejected).

The Peripheral's Host will receive an indication from the Peripheral's Controller when the connection parameters have been updated. In devices supporting HCI, this notification will be in the form of an HCI_LE_Connection_Update_Complete event (see [Vol 4] Part E, Section 7.7.65.3). If the Central's Controller rejects the updated connection parameters no indication from the Peripheral's Controller will be sent to the Peripheral's Host.

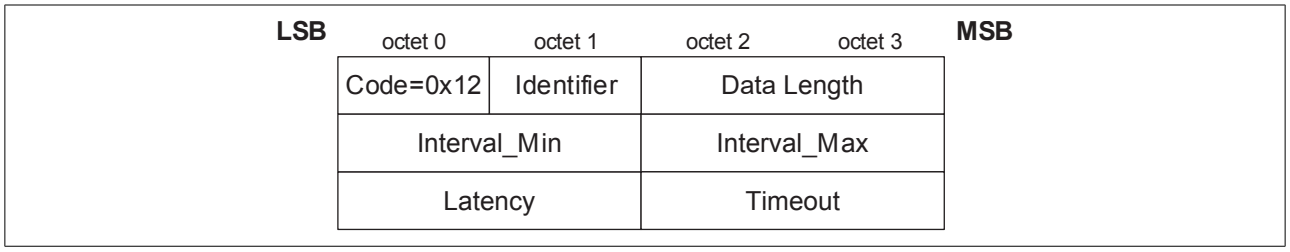



*Figure 4.16: L2CAP_CONNECTION_PARAMETER_UPDATE_REQ packet*

The data fields shall have the same meanings and requirements as the fields of the LL_CONNECTION_PARAM_REQ PDU (see [Vol 6] Part B, Section 2.4.2.16) with the same names.

## 4.21 L2CAP_CONNECTION_PARAMETER_UPDATE_RSP (code 0x13)

This response shall only be sent from the Central to the Peripheral.

The L2CAP_CONNECTION_PARAMETER_UPDATE_RSP packet shall be sent by the Central's Host when it receives an L2CAP_CONNECTION_PARAMETER_UPDATE_REQ packet. Figure 4.17 defines the format of the packet. If the Central's Host accepts the request it shall send the connection parameter update to its Controller.

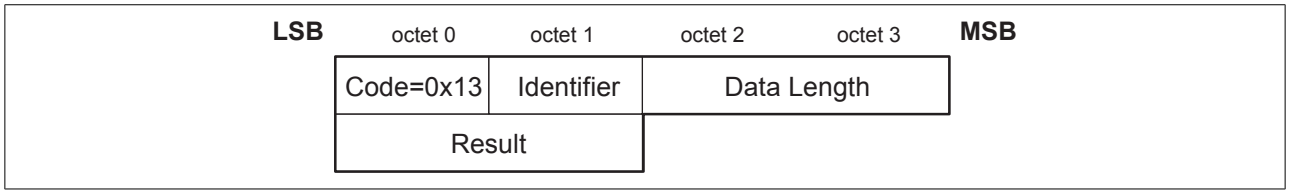



*Figure 4.17: L2CAP_CONNECTION_PARAMETER_UPDATE_RSP packet*

The data field is:

- *Result (2 octets)*

  The result field indicates the response to the request. The result value of 0x0000 indicates that the Central's Host has accepted the connection parameters while 0x0001 indicates that the Central's Host has rejected the connection parameters.

| Result | Description |
|---|---|
| 0x0000 | Connection Parameters accepted |
| 0x0001 | Connection Parameters rejected |
| Other | Reserved for future use |

*Table 4.14: Result values for the L2CAP_CONNECTION_PARAMETER_UPDATE_RSP packet*

## 4.22 L2CAP_LE_CREDIT_BASED_CONNECTION_REQ (code 0x14)

L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packets are sent to create and configure an L2CAP channel between two devices using LE Credit Based Flow Control mode. Figure 4.18 defines the format of the packet.

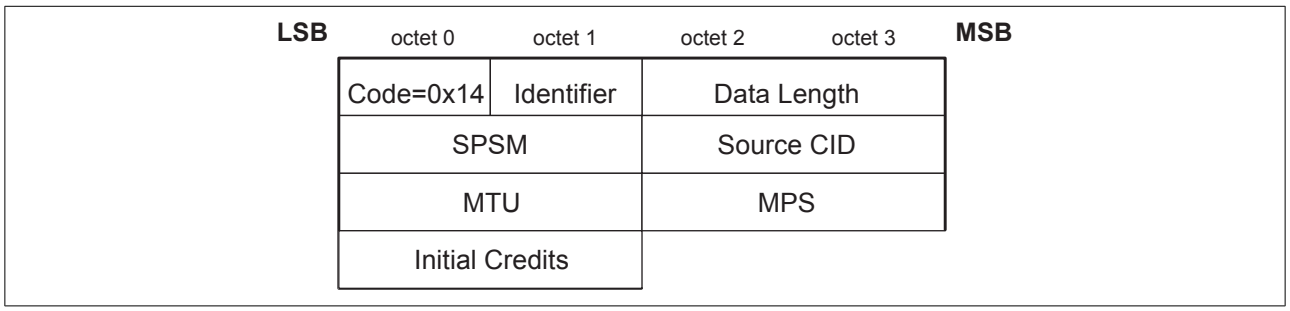



*Figure 4.18: L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet*

The data fields are:

- *Simplified Protocol/Service Multiplexer – SPSM*[1] *(2 octets)*

  The SPSM field is two octets in length. SPSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. Values in the second range are dynamically allocated and used in conjunction with services defined in the GATT Server. The dynamically assigned values may be used to support multiple implementations of a particular protocol.

  Note: Unlike a normal PSM (see Section 4.2), the length of an SPSM is not variable and each octet can be either odd or even.

  SPSM values are defined in Table 4.15.

[1]This was called LE_PSM in versions 4.1 to 5.1.

| Range | Type | Server Usage | Client Usage |
|---|---|---|---|
| 0x0001 to 0x007F | Fixed, SIG assigned | SPSM is fixed for all implementations | SPSM may be assumed for fixed service. Protocol used is indicated by the SPSM as defined in Assigned Numbers. |
| 0x0080 to 0x00FF | Dynamic | SPSM may be fixed for a given implementation or may be assigned at the time the service is registered in GATT | SPSM shall be obtained from the service in GATT upon every reconnection. SPSM for one direction will typically be different from the other direction. |
| Other | RFU | Not applicable | Not applicable |

*Table 4.15: L2CAP_LE_CREDIT_BASED_CONNECTION_REQ SPSM ranges*

- *Source CID – SCID (2 octets)*

  The source CID is two octets in length and represents a channel endpoint on the device sending the request. Once the channel has been created, data packets flowing to the sender of the request shall be sent to this CID. Thus, the Source CID represents the channel endpoint on the device sending the request and receiving the response. The value of the Source CID shall be from the dynamically allocated range as defined in Table 2.3 and shall not be already allocated to a different channel on the same logical link on the device sending the request.

- *Maximum Transmission Unit – MTU (2 octets)*

  The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet can receive on this channel. L2CAP implementations shall support a minimum MTU size of 23 octets.

- *Maximum PDU Payload Size – MPS (2 octets)*

  The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet is capable of receiving on this channel. L2CAP implementations shall support a minimum MPS of 23 octets and may support an MPS up to 65533 octets.

- *Initial Credits – (2 octets)*

  The initial credit value indicates the number of K-frames that the peer device can send to the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet. The initial credit value shall be in the range 0 to 65535.

## 4.23 L2CAP_LE_CREDIT_BASED_CONNECTION_RSP (code 0x15)

When a device receives an L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet, it shall send an L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet. Figure 4.19 defines the format of the packet.

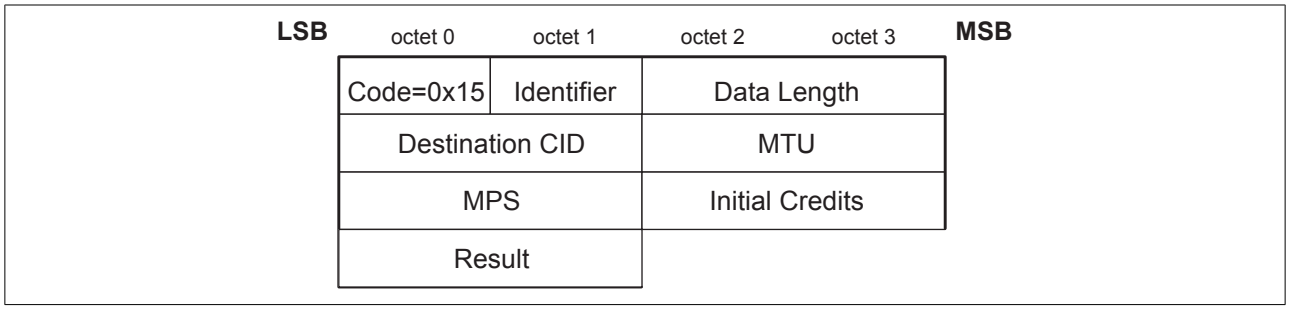



*Figure 4.19: L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet*

The data fields are:

- *Destination CID – DCID (2 octets)*

  The destination CID is two octets in length and represents a channel endpoint on the device sending the response. Once the channel has been created, data packets flowing to the sender of the response shall be sent to this CID. Thus, the destination CID represents the channel endpoint on the device receiving the request and sending the response. The value of the Destination CID shall be from the dynamically allocated range as defined in Table 2.3 and shall not be already allocated to a different channel on the same logical link on the device sending the response.

- *Maximum Transmission Unit – MTU (2 octets)*

  The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet can receive on this channel. L2CAP implementations shall support a minimum MTU size of 23 octets.

- *Maximum PDU Payload Size – MPS (2 octets)*

  The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet is capable of receiving on this channel. L2CAP implementations shall support a minimum MPS of 23 octets and may support an MPS up to 65533 octets.

- *Initial Credits – (2 octets)*

  The initial credit value indicates the number of K-frames that the peer device can send to the L2CAP layer entity sending the L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet. The initial credit value shall be in the range 0 to 65535.

- *Result – (2 octets)*

  The result field indicates the outcome of the connection request. A result value of 0x0000 indicates success while a non-zero value indicates the connection request was refused. A logical channel is established on the receipt of a successful result.

Table 4.16 defines values for this field. The DCID, MTU, MPS and Initial Credits fields shall be ignored when the result field indicates the connection was refused.

| Value | Description |
|---|---|
| 0x0000 | Connection successful |
| 0x0002 | Connection refused – SPSM not supported |
| 0x0004 | Connection refused – no resources available |
| 0x0005 | Connection refused – insufficient authentication |
| 0x0006 | Connection refused – insufficient authorization |
| 0x0007 | Connection refused – encryption key size too short[1] |
| 0x0008 | Connection refused – insufficient encryption |
| 0x0009 | Connection refused – invalid Source CID |
| 0x000A | Connection refused – Source CID already allocated |
| 0x000B | Connection refused – unacceptable parameters |
| Other | Reserved for future use |

*Table 4.16: Result values for the L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet*

[1]This was previously "Connection refused - insufficient encryption key size".

## 4.24 L2CAP_FLOW_CONTROL_CREDIT_IND (code 0x16)

A device shall send an L2CAP_FLOW_CONTROL_CREDIT_IND packet when it is capable of receiving additional K-frames (for example after it has processed one or more K-frames) in LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode. Figure 4.20 defines the format of the packet.

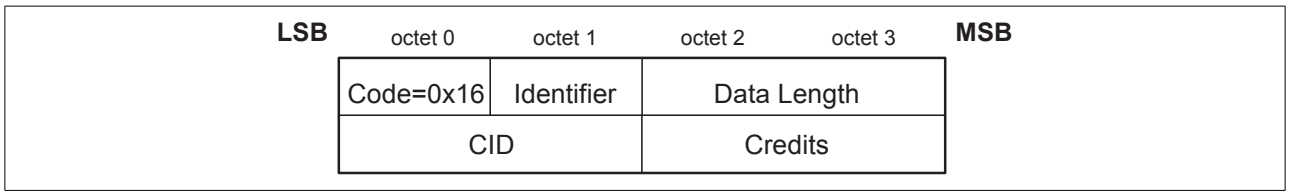



*Figure 4.20: L2CAP_FLOW_CONTROL_CREDIT_IND packet*

The data fields are:

- *CID – (2 octets)*

  The CID is two octets in length and represents the source channel endpoint of the device sending the L2CAP_FLOW_CONTROL_CREDIT_IND packet. For example, a received L2CAP_FLOW_CONTROL_CREDIT_IND packet with a given CID (0x0042) would provide credits for the receiving device's destination CID (0x0042).

- *Credits – (2 octets)*

  The credit value field represents number of credits the receiving device can increment, corresponding to the number of K-frames that can be sent to the peer device sending the L2CAP_FLOW_CONTROL_CREDIT_IND packet. The credit value field shall be a number between 1 and 65535.

## 4.25 L2CAP_CREDIT_BASED_CONNECTION_REQ (code 0x17)

L2CAP_CREDIT_BASED_CONNECTION_REQ packets are sent to create and configure up to five L2CAP channels between two devices. Figure 4.21 defines the format of the packet.

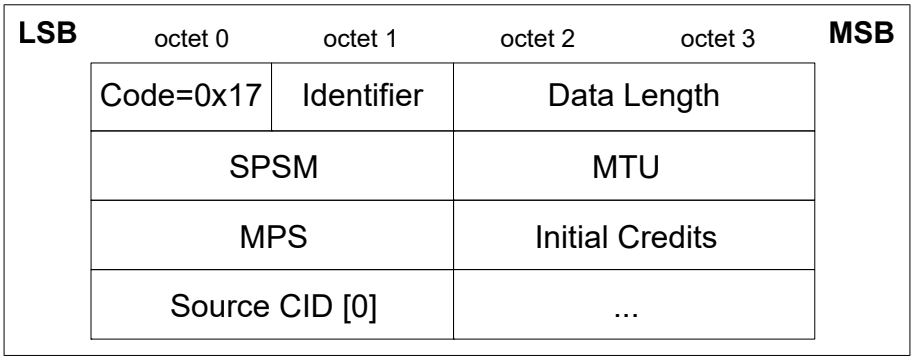



*Figure 4.21: L2CAP_CREDIT_BASED_CONNECTION_REQ packet*

The data fields are:

- *Simplified Protocol/Service Multiplexer – SPSM (2 octets)*

  The SPSM is defined in Section 4.22.

- *Maximum Transmission Unit – MTU (2 octets)*

  The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_CONNECTION_REQ packet can receive on each of the Source CID channels. L2CAP implementations shall support a minimum MTU size of 64 octets for these channels.

- *Maximum PDU Payload Size – MPS (2 octets)*

  The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_CONNECTION_REQ packet is capable of receiving on each of the Source CID channels. L2CAP implementations shall support a minimum MPS of 64 octets and may support an MPS up to 65533 octets for these channels.

- *Initial Credits – (2 octets)*

  The Initial Credit field value indicates the number of K-frames that the peer device can send to the L2CAP layer entity sending the

L2CAP_CREDIT_BASED_CONNECTION_REQ packet on each of the Source CID channels. The initial credit value shall be in the range of 1 to 65535.

- *Source CID – (2 to 10 octets)*

  The Source CID is an array of up to 5 two-octet values and represents the channel endpoints on the device sending the request. Once a channel has been created, data packets flowing to the sender of the request shall be sent to these CIDs. Each entry in the array shall be non-zero and represents a request for a channel. The value of each Source CID shall be from the dynamically allocated range as defined in Table 2.1 or Table 2.3 (depending on the transport in use) and shall not be already allocated to a different channel on the same logical link on the device sending the request.

## 4.26 L2CAP_CREDIT_BASED_CONNECTION_RSP (code 0x18)

When a device receives an L2CAP_CREDIT_BASED_CONNECTION_REQ packet, it shall send an L2CAP_CREDIT_BASED_CONNECTION_RSP packet. If the device sends an L2CAP_CREDIT_BASED_CONNECTION_RSP packet with a result code "pending", then it shall subsequently send another L2CAP_CREDIT_BASED_CONNECTION_RSP (see also [Vol 3] Part C, Section 5.2.2.2). Figure 4.22 defines the format of the packet.

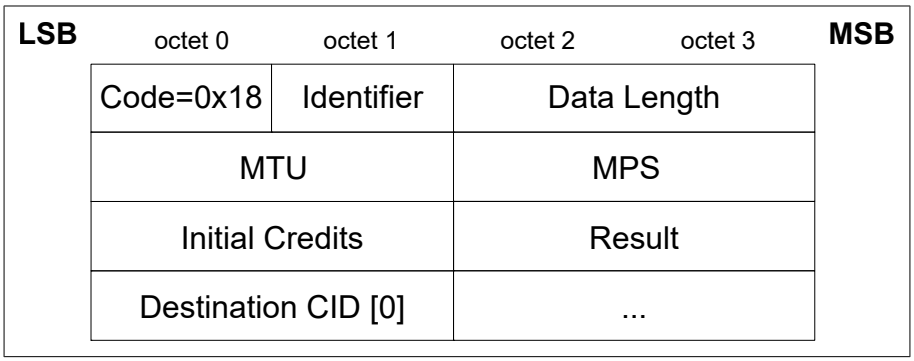



*Figure 4.22: L2CAP_CREDIT_BASED_CONNECTION_RSP packet*

The data fields are:

- *Maximum Transmission Unit – MTU (2 octets)*

  The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_CONNECTION_RSP packet can receive on each of the Destination CID channels. L2CAP implementations shall support a minimum MTU size of 64 octets.

- *Maximum PDU Payload Size – MPS (2 octets)*

  The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_CONNECTION_RSP packet is capable of receiving on each of the Destination CID channels. L2CAP

implementations shall support a minimum MPS of 64 octets and may support an MPS up to 65533 octets.

- *Initial Credits – (2 octets)*

  The Initial Credit field value indicates the number of K-frames that the peer device can send to the L2CAP layer entity sending the L2CAP_CREDIT_BASED_CONNECTION_RSP packet on each of the Destination CID channels. The initial credit value shall be in the range of 1 to 65535.

- *Result – (2 octets)*

  The Result field indicates the outcome of the connection request. Table 4.17 defines values for this field. The Destination CID, MTU, MPS and Initial Credits fields shall be ignored when the Result field indicates that all connections were refused or all connections are pending.

| Value | Description |
|---|---|
| 0x0000 | All connections successful |
| 0x0002 | All connections refused – SPSM not supported |
| 0x0004 | Some connections refused – insufficient resources available |
| 0x0005 | All connections refused – insufficient authentication |
| 0x0006 | All connections refused – insufficient authorization |
| 0x0007 | All connections refused – encryption key size too short[1] |
| 0x0008 | All connections refused – insufficient encryption |
| 0x0009 | Some connections refused – invalid Source CID |
| 0x000A | Some connections refused – Source CID already allocated |
| 0x000B | All connections refused – unacceptable parameters |
| 0x000C | All connections refused – invalid parameters |
| 0x000D | All connections pending – no further information available |
| 0x000E | All connections pending – authentication pending |
| 0x000F | All connections pending – authorization pending |
| Other | Reserved for future use |

*Table 4.17: Result values for the L2CAP_CREDIT_BASED_CONNECTION_RSP packet*

[1]This was previously "All connections refused - insufficient encryption key size".

- *Destination CID – (2 to 10 octets)*

  The Destination CID is an array of up to 5 two-octet values and represents the channel endpoints on the device sending the L2CAP_CREDIT_BASED_CONNECTION_RSP packet. The value of the Destination CID shall be from the dynamically allocated range as defined in Table 2.1 or Table 2.3

(depending on the transport in use) and shall not be already allocated to a different channel on the same logical link on the device sending the response. Once a channel has been created, data packets flowing to the sender of the response shall be sent to these CIDs. The order of the Destination CIDs shall correspond to the order of the Source IDs in the corresponding L2CAP_CREDIT_BASED_CONNECTION_REQ packet. If a Destination CID is non-zero, the channel was established. If a Destination CID is 0x0000, the channel was not established. If a device receives an L2CAP_CREDIT_BASED_CONNECTION_RSP packet with an already-assigned Destination CID, then both the original channel and the new channel shall not be used.

## 4.27 L2CAP_CREDIT_BASED_RECONFIGURE_REQ (code 0x19)

A device shall send an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet when its receive MTU or MPS values have changed compared to when the channel was created or last reconfigured. Figure 4.23 defines the format of the packet.

The current MTU and MPS values of the channels may be different before this packet is sent.

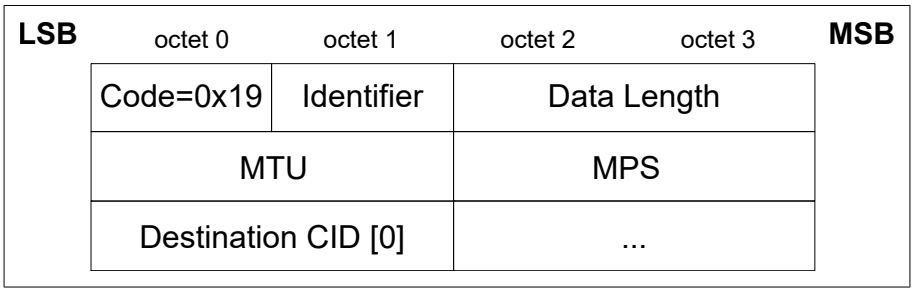



*Figure 4.23: L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet*

The data fields are:

- *Maximum Transmission Unit – MTU (2 octets)*

  The MTU field specifies the maximum SDU size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet can receive on each of the Destination CID channels. The MTU field shall be greater than or equal to the greatest current MTU size of these channels.

- *Maximum PDU Payload Size – MPS (2 octets)*

  The MPS field specifies the maximum PDU payload size (in octets) that the L2CAP layer entity sending the L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet is capable of receiving on each of the Destination CID channels. If more than one channel is being configured, the MPS field shall be greater than or equal to the current MPS size of each of these channels. If only one channel is being configured, the MPS field may be less than the current MPS of that channel.

- *Destination CID – (2 to 10 octets)*

  The Destination CID is an array of up to 5 two-octet values which shall be non-zero and represent the channel endpoints on the device sending the L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet.

## 4.28 L2CAP_CREDIT_BASED_RECONFIGURE_RSP (code 0x1A)

When a device receives an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet, it shall send an L2CAP_CREDIT_BASED_RECONFIGURE_RSP packet. Figure 4.24 defines the format of the packet.

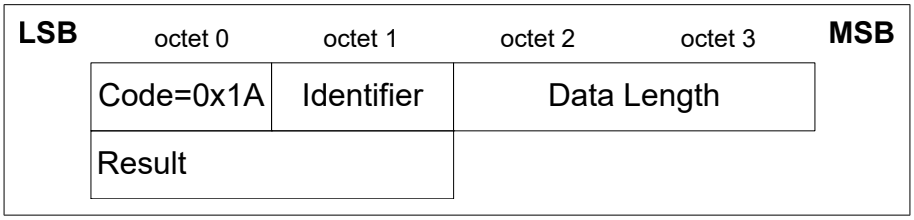



*Figure 4.24: L2CAP_CREDIT_BASED_RECONFIGURE_RSP*

The data fields are:

- *Result (2 octets)*

The Result field contains information about the success of the request.

| Value | Description |
|---|---|
| 0x0000 | Reconfiguration successful |
| 0x0001 | Reconfiguration failed - reduction in size of MTU not allowed |
| 0x0002 | Reconfiguration failed - reduction in size of MPS not allowed for more than one channel at a time |
| 0x0003 | Reconfiguration failed - one or more Destination CIDs invalid |
| 0x0004 | Reconfiguration failed - other unacceptable parameters |
| All other values | Reserved for future use |

*Table 4.18: Result values for the L2CAP_CREDIT_BASED_RECONFIGURE_RSP packet*

# 5 CONFIGURATION PARAMETER OPTIONS

Options are a mechanism to extend the configuration parameters. Options shall be transmitted as information elements containing an option type, an option length, and one or more option data fields. Figure 5.1 illustrates the format of an option.

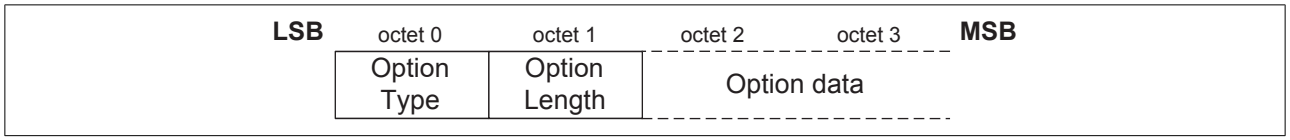



*Figure 5.1: Configuration option format*

The configuration option fields are:

- *Option Type (1 octet)*

  The Option Type field defines the parameters being configured. If the option is not recognized (e.g. because the option is defined in a higher version of the specification than the version the implementation conforms to) then:

  – If the most significant bit of the type is 0 (i.e. types 0x00 to 0x7F), the recipient shall refuse the entire configuration request.

  – If the most significant bit of the type is 1 (i.e. types 0x80 to 0xFF), the recipient shall ignore the option and continue processing with the next option (if any).

- *Option Length (1 octet)*

  The Option Length field defines the number of octets in the option data. Thus an option type without option data has a length of 0.

- *Option data*

  The contents of this field are dependent on the option type.

## 5.1 Maximum Transmission Unit (MTU)

This option specifies the maximum SDU size the sender of this option is capable of accepting for a channel. The Option Type is 0x01, and the Option Length is 2 octets, carrying the two-octet MTU size value as the only information element (see Figure 5.2).

MTU is not a negotiated value, it is an informational parameter that each device can specify independently. It indicates to the remote device that the local device can receive, in this channel, an MTU larger than the minimum required. All L2CAP implementations shall support a minimum MTU of 48 octets over the ACL-U logical link and 23 octets over the LE-U logical link; however, some protocols and profiles explicitly require support for a larger MTU. The minimum MTU for a channel is the larger of the L2CAP

minimum 48 octet MTU and any MTU explicitly required by the protocols and profiles using that channel.

Note: The MTU is only affected by the profile directly using the channel. For example, if a service discovery transaction is initiated by a non service discovery profile, that profile does not affect the MTU of the L2CAP channel used for service discovery.

The following rules shall be used when responding to an L2CAP_CONFIGURATION_REQ packet specifying the MTU for a channel:

- A request specifying any MTU greater than or equal to the minimum MTU for the channel shall be accepted.
- A request specifying an MTU smaller than the minimum MTU for the channel may be rejected.

The signaling described in Section 4.5 may be used to reject an MTU smaller than the minimum MTU for a channel. The "failure-unacceptable parameters" result sent to reject the MTU shall include the proposed value of MTU that the remote device intends to transmit. It is implementation specific whether the local device continues the configuration process or disconnects the channel.

If the remote device sends a positive L2CAP_CONFIGURATION_RSP packet it should include the actual MTU to be used on this channel for traffic flowing into the local device. Following the above rules, the actual MTU cannot be less than 48 bytes.This is the minimum of the MTU in the L2CAP_CONFIGURATION_REQ packet and the outgoing MTU capability of the device sending the L2CAP_CONFIGURATION_RSP packet. The new agreed value (the default value in a future re-configuration) is the value specified in the response.

An MTU option received in a negative L2CAP_CONFIGURATION_RSP packet where the MTU option is not in error should be interpreted in the same way as it is in a positive L2CAP_CONFIGURATION_RSP packet (e.g. the case where another configuration option value is unacceptable but the negative L2CAP_CONFIGURATION_RSP packet contains the MTU option in addition to the unacceptable option).

The MTU to be used on this channel for the traffic flowing in the opposite direction will be established when the remote device sends its own L2CAP_CONFIGURATION_REQ packet as explained in Section 4.4.

If the configured mode is Enhanced Retransmission mode or Streaming mode then MTU shall not be reconfigured to a smaller size.

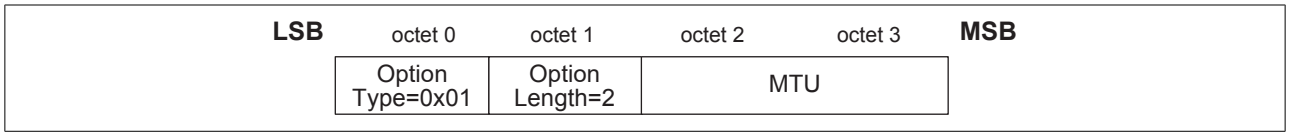



*Figure 5.2: MTU option format*

The option data field is:

- *Maximum Transmission Unit - MTU (2 octets)*

  The MTU field is the maximum SDU size, in octets, that the originator of the request can accept for this channel. The MTU is asymmetric and the sender of the request shall specify the MTU it can receive on this channel if it differs from the default value. L2CAP implementations shall support a minimum MTU size of 48 octets. The default value is 672 octets.

## 5.2 Flush Timeout option

This option is used to inform the recipient of the Flush Timeout the sender is going to use. This option shall not be used if the Extended Flow Specification is used. The Flush Timeout is defined in the BR/EDR Baseband specification [Vol 2] Part B, Section 3.3. The Option Type is 0x02 and the Option Length is 2 octets (see Figure 5.3). The Flush Timeout option is negotiable.

If the remote device returns a negative response to this option and the local device cannot honor the proposed value, then it shall either continue the configuration process by sending a new request with the original value, or disconnect the channel. The flush timeout applies to all channels on the same ACL logical transport but may be overridden on a packet by packet basis by marking individual L2CAP packets as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data packet (see Section 1.1).

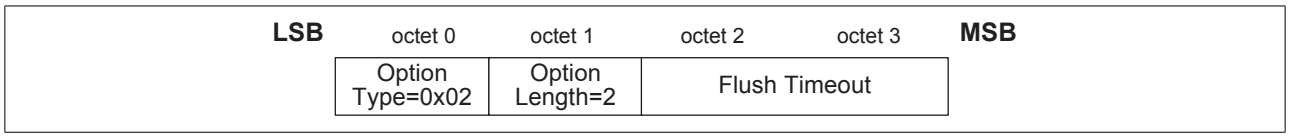



*Figure 5.3: Flush Timeout option format*

The option data field is:

- *Flush Timeout*

  This value is the Flush Timeout in milliseconds. This is an asymmetric value and the sender of the request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

Possible values are:

0x0001 - no retransmissions at the Baseband level should be performed since the minimum polling interval is 1.25 ms.

0x0002 to 0xFFFE - Flush Timeout used by the Baseband.

0xFFFF - an infinite amount of retransmissions. This is also referred to as a 'reliable channel'. In this case, the Baseband shall continue retransmissions until physical link loss is declared by link manager timeouts.

## 5.3 Quality of Service (QoS) option

This option specifies a flow specification similar to RFC 1363[1]. Although the RFC flow specification addresses only the transmit characteristics, the Bluetooth QoS interface can handle the two directions (Tx and Rx) in the negotiation as described below.

If no QoS configuration parameter is negotiated the link shall assume the default parameters. The Option Type is 0x03. This option shall not be used if the Extended Flow Specification option is used. The QoS option is negotiable. Figure 5.4 specifies the format of this option.

In an L2CAP_CONFIGURATION_REQ packet, this option describes the outgoing traffic flow from the device sending the request. In a positive L2CAP_CONFIGURATION_RSP packet, this option describes the incoming traffic flow agreement to the device sending the response. In a negative L2CAP_CONFIGURATION_RSP packet, this option describes the preferred incoming traffic flow to the device sending the response.

L2CAP implementations are only required to support 'Best Effort' service, support for any other service type is optional. Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort shall be assumed. If any QoS guarantees are required then a QoS configuration request shall be sent.

The remote device's L2CAP_CONFIGURATION_RSP packet contains information that depends on the value of the result field (see Section 4.5). If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is "Failure – unacceptable parameters", the response shall include a list of outgoing flow specification parameters and parameter values that would make a new L2CAP_CONNECTION_REQ packet from the local device acceptable by the remote device. Both explicitly referenced in an L2CAP_CONFIGURATION_REQ packet or implied configuration parameters can be included in an L2CAP_CONFIGURATION_RSP packet. Recall that any missing configuration parameters from an L2CAP_CONFIGURATION_REQ packet are assumed to have their most recently accepted values.

[1]Internet Engineering Task Force, "A Proposed Flow Specification", RFC 1363, September 1992.

If an L2CAP_CONFIGURATION_REQ packet contains any QoS option parameters set to “do not care” then the L2CAP_CONFIGURATION_RSP packet shall set the same parameters to “do not care”. This rule applies for Best Effort and, if the parameter is allowed to be set to the "do not care" value, for Guaranteed Service.

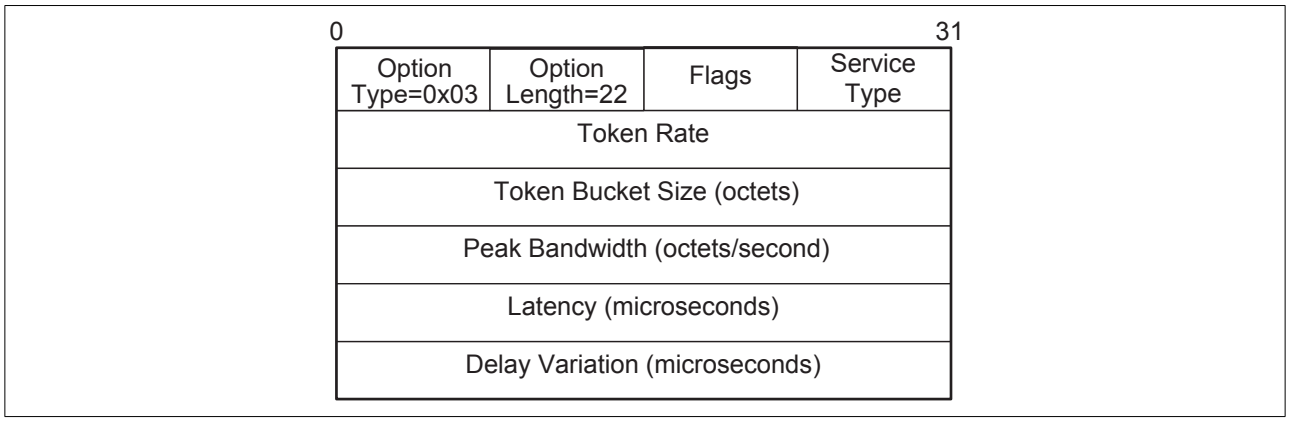



*Figure 5.4: Quality of Service (QoS) option format containing Flow Specification*

The option data fields are:

- *Flags (1 octet)*

  Reserved for future use.

- *Service Type (1 octet)*

  This field indicates the level of service required. Table 5.1 defines the different services available. The default value is ‘Best effort’.

  If ‘Best effort’ is selected, the remaining parameters should be treated as optional by the remote device. The remote device may choose to ignore the fields, try to satisfy the parameters but provide no response (QoS option omitted in the response message), or respond with the settings it will try to meet.

  If ‘No traffic’ is selected, the remainder of the fields shall be ignored because there is no data being sent across the channel in the outgoing direction.

| Value | Description |
|---|---|
| 0x00 | No traffic |
| 0x01 | Best effort (Default) |
| 0x02 | Guaranteed |
| Other | Reserved for future use |

*Table 5.1: Service type definitions*

- *Token Rate (4 octets)*

  The value of this field represents the average data rate with which the application transmits data. The application may send data at this rate continuously. On a short

time scale the application may send data in excess of the average data rate, dependent on the specified Token Bucket Size and Peak Bandwidth (see below). The Token Bucket Size and Peak Bandwidth allow the application to transmit data in a 'bursty' fashion.

The Token Rate signaled between two L2CAP peers specifies the data transmitted by the application and shall not include the L2CAP protocol overhead. The Token Rate signaled over the interface between L2CAP and the Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Rate value signaled over this interface may also include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

The Token Rate is the rate with which traffic credits are provided. Credits can be accumulated up to the Token Bucket Size. Traffic credits are consumed when data is transmitted by the application. When traffic is transmitted, and there are insufficient credits available, the traffic is non-conformant. The Quality of Service guarantees are only provided for conformant traffic. For non-conformant traffic there may be insufficient resources such as bandwidth and buffer space. Furthermore non-conformant traffic may violate the QoS guarantees of other traffic flows.

The Token Rate is specified in octets per second. The value 0x00000000 indicates no token rate is specified. This is the default value and means “do not care”. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token rate available. The meaning of this value depends on the service type. For best effort, the value is a hint that the application requires as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

- *Token Bucket Size (4 octets)*

  The Token Bucket Size specifies a limit on the 'burstiness' with which the application may transmit data. The application may offer a burst of data equal to the Token Bucket Size instantaneously, limited by the Peak Bandwidth (see below). The Token Bucket Size is specified in octets.

  The Token Bucket Size signaled between two L2CAP peers specifies the data transmitted by the application and shall not include the L2CAP protocol overhead. The Token Bucket Size signaled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Bucket Size value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The value of 0x00000000 means that no token bucket is needed; this is the default value. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token bucket available. The meaning of this value depends on the service type. For best effort, the value

indicates the application requires a bucket as big as possible. For Guaranteed service the value represents the maximum L2CAP SDU size.

The Token Bucket Size is a property of the traffic carried over the L2CAP channel. The Maximum Transmission Unit (MTU) is a property of an L2CAP implementation. For the Guaranteed service the Token Bucket Size shall be smaller or equal to the MTU.

- *Peak Bandwidth (4 octets)*

  The value of this field, expressed in octets per second, limits how fast packets from applications may be sent back-to-back. Some systems can take advantage of this information, resulting in more efficient resource allocation.

  The Peak Bandwidth signaled between two L2CAP peers specifies the data transmitted by the application and shall not include the L2CAP protocol overhead. The Peak Bandwidth signaled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Peak Bandwidth value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The value of 0x00000000 means "do not care." This states that the device has no preference on incoming maximum bandwidth, and is the default value. When the Guaranteed service is selected, the default value shall not be used.

- *Access Latency (4 octets)*

  The value of this field is the maximum acceptable delay of an L2CAP packet to the air-interface. The precise interpretation of this number depends on over which interface this flow parameter is signaled. When signaled between two L2CAP peers, the Access Latency is the maximum acceptable delay between the instant when the L2CAP SDU is received from the upper layer and the start of the L2CAP SDU transmission over the air. When signaled over the interface between L2CAP and the Link Manager, it is the maximum delay between the instant the first fragment of an L2CAP PDU is stored in the Controller buffer and the initial transmission of the L2CAP packet on the air.

  Thus the Access Latency value may be different when signaled between L2CAP and the Link Manager to account for any queuing delay at the L2CAP transmit side. Furthermore the Access Latency value may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The Access Latency is expressed in microseconds. The value 0xFFFFFFFF means “do not care” and is the default value. When the Guaranteed service is selected, the default value shall not be used.

- *Delay Variation (4 octets)*

  The value of this field is the difference, in microseconds, between the maximum and minimum possible delay of an L2CAP SDU between two L2CAP peers. The Delay

Variation is a purely informational parameter. The value 0xFFFFFFFF means “do not care” and is the default value.

## 5.4 Retransmission and Flow Control option

This option specifies whether retransmission and flow control is used. If the feature is used both incoming and outgoing parameters are specified by this option. The Retransmission and Flow Control option contains both negotiable parameters and non-negotiable parameters. The mode parameter controls both incoming and outgoing data flow (i.e. both directions have to agree) and is negotiable. The other parameters control incoming data flow and are non-negotiable. Figure 5.5 specifies the format of this option.

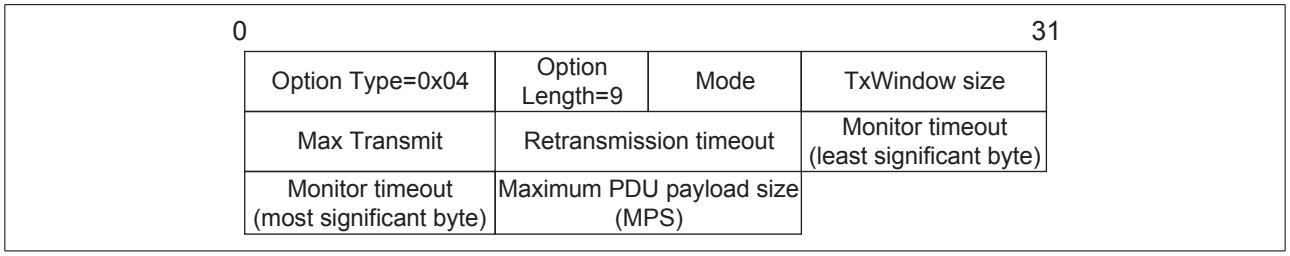



*Figure 5.5: Retransmission and Flow Control option format*

The option data fields are:

- *Mode (1 octet)*

  The field contains the requested mode of the link. Possible values are shown in Table 5.2.

| Value | Description |
|---|---|
| 0x00 | L2CAP Basic mode |
| 0x01 | Retransmission mode |
| 0x02 | Flow control mode |
| 0x03 | Enhanced Retransmission mode |
| 0x04 | Streaming mode |
| Other | Reserved for future use |

*Table 5.2: Mode definitions*

  The Basic L2CAP mode is the default. If Basic L2CAP mode is requested then all other parameters shall be ignored.

  Enhanced Retransmission mode should be enabled if a reliable channel has been requested. Enhanced Retransmission mode shall only be sent to an L2CAP entity that has previously advertised support for the mode in its Extended Feature Mask (see Section 4.12).

Streaming mode should be enabled if a finite L2CAP Flush Timeout is set on an L2CAP connection. Streaming mode shall only be sent to a device that has previously advertised support for the mode in the Extended Feature Mask (see Section 4.12).

Flow Control mode and Retransmission mode shall only be used for backwards compatibility with L2CAP entities that do not support Enhanced Retransmission mode or Streaming mode.

- *TxWindow size (1 octet)*

  This field specifies the size of the transmission window for Flow Control mode, Retransmission mode, and Enhanced Retransmission mode. The range is 1 to 32 for Flow Control mode and Retransmission mode. The range is 1 to 63 for Enhanced Retransmission mode.

  In Retransmission mode and Flow Control mode this parameter should be negotiated to reflect the buffer sizes allocated for the connection on both sides. In general, the Tx Window size should be made as large as possible to maximize channel use. Tx Window size also controls the delay on flow control action. The transmitting device can send as many PDUs fit within the window.

  In Enhanced Retransmission mode this value indicates the maximum number of I-frames that the sender of the option can receive without acknowledging some of the received frames. It is not negotiated. It is an informational parameter that each L2CAP entity can specify separately. In general, the TxWindow size should be made as large as possible to maximize channel use. The transmitting L2CAP entity can send as many PDUs as will fit within the receiving L2CAP entity's TxWindow. TxWindow size values in an L2CAP_CONFIGURATION_RSP packet indicate the maximum number of packets the sender can send before it requires an acknowledgment. In other words it represents the number of unacknowledged packets the sender can hold. The value sent in an L2CAP_CONFIGURATION_RSP packet shall be less than or equal to the TxWindow size sent in the L2CAP_CONFIGURATION_REQ packet. The receiver of this option in the L2CAP_CONFIGURATION_RSP packet may use this value as part of its acknowledgment algorithm.

  In Streaming mode this value is reserved for future use.

- *MaxTransmit (1 octet)*

  This field controls the number of transmissions of a single I-frame that L2CAP is allowed to try in Retransmission mode and Enhanced Retransmission mode. The minimum value is 1 (one transmission is permitted).

  MaxTransmit controls the number of retransmissions that L2CAP is allowed to try in Retransmission mode and Enhanced Retransmission mode before accepting that a packet and the channel is lost. When a packet is lost after being transmitted MaxTransmit times the channel shall be disconnected by sending a Disconnect request (see Section 4.6). In Enhanced Retransmission mode MaxTransmit controls

the number of retransmissions for I-frames and S-frames with P-bit set to 1. The sender of the option in an L2CAP_CONFIGURATION_REQ packet specifies the value that shall be used by the receiver of the option. MaxTransmit values in an L2CAP_CONFIGURATION_RSP packet shall be ignored. Lower values might be appropriate for services requiring low latency. Higher values will be suitable for a link requiring robust operation. A value of 1 means that no retransmissions will be made but also means that the channel will be disconnected as soon as a packet is lost. MaxTransmit shall not be set to zero in Retransmission mode. In Enhanced Retransmission mode a value of zero for MaxTransmit means infinite retransmissions.

In Streaming mode this value is reserved for future use.

- *Retransmission timeout (2 octets)*

  This is the value in milliseconds of the retransmission timeout (this value is used to initialize the RetransmissionTimer).

  The purpose of this timer in Retransmission mode is to activate a retransmission in some exceptional cases. In such cases, any delay requirements on the channel may be broken, so the value of the timer should be set high enough to avoid unnecessary retransmissions due to delayed acknowledgments. Suitable values could be 100s of milliseconds and up.

  The purpose of this timer in flow control mode is to supervise I-frame transmissions. If an acknowledgment for an I-frame is not received within the time specified by the RetransmissionTimer value, either because the I-frame has been lost or the acknowledgment has been lost, the timeout will cause the transmitting side to continue transmissions. Suitable values are implementation dependent.

  The purpose of this timer in Enhanced Retransmission mode is to detect lost I-frames and initiate appropriate error recovery. The value used for the Retransmission timeout is specified in Section 8.6.2. The value sent in an L2CAP_CONFIGURATION_REQ packet is also specified in Section 8.6.2. A value for the Retransmission timeout shall be sent in a positive L2CAP_CONFIGURATION_RSP packet and indicates the value that will be used by the sender of the L2CAP_CONFIGURATION_RSP packet.

  In Streaming mode this value is reserved for future use.

- *Monitor timeout (2 octets)*

  In Retransmission mode this is the value in milliseconds of the interval at which S-frames should be transmitted on the return channel when no frames are received on the forward channel. (this value is used to initialize the MonitorTimer, see below).

  This timer ensures that lost acknowledgments are retransmitted. Its main use is to recover Retransmission Disable Bit changes in lost frames when no data is being sent. The timer shall be started immediately upon transitioning to the open state. It shall remain active as long as the connection is in the open state and the retransmission timer is not active. Upon expiration of the Monitor timer an S-frame

shall be sent and the timer shall be restarted. If the monitor timer is already active when an S-frame is sent, the timer shall be restarted. An idle connection will have periodic monitor traffic sent in both directions. The value for this timeout should also be set to 100s of milliseconds or higher.

In Enhanced Retransmission mode the Monitor timeout is used to detect lost S-frames with P-bit set to 1. If the timeout occurs before a response with the F-bit set to 1 is received the S-frame is resent. The value used for the Monitor timeout is specified in Section 8.6.3. The value sent in an L2CAP_CONFIGURATION_REQ packet is also specified in Section 8.6.2. A value for the Monitor timeout shall be sent in a positive L2CAP_CONFIGURATION_RSP packet and indicates the value that will be used by the sender of the L2CAP_CONFIGURATION_RSP packet.

In Streaming mode this value is reserved for future use.

- *Maximum PDU payload Size - MPS (2 octets)*

  The maximum payload size that the L2CAP layer entity sending the option in an L2CAP_CONFIGURATION_REQ packet is capable of accepting, i.e. the MPS corresponds to the maximum PDU payload size. Each device specifies the value separately. An MPS value sent in a positive L2CAP_CONFIGURATION_RSP packet is the actual MPS the receiver of the L2CAP_CONFIGURATION_REQ packet will use on this channel for traffic flowing into the local device. An MPS value sent in a positive L2CAP_CONFIGURATION_RSP packet shall be equal to or smaller than the value sent in the L2CAP_CONFIGURATION_REQ packet.

When using Retransmission mode and Flow Control mode the settings are configured separately for the two directions of an L2CAP connection. For example, in operating with an L2CAP entity implementing a lower version of the specification, an L2CAP connection can be configured as Flow Control mode in one direction and Retransmission mode in the other direction. If Basic L2CAP mode is configured in one direction and Retransmission mode or Flow control mode is configured in the other direction on the same L2CAP channel then the channel shall not be used.

Note: This asymmetric configuration only occurs during configuration.

When using Enhanced Retransmission mode or Streaming mode, both directions of the L2CAP connection shall be configured to the same mode. A precedence algorithm shall be used by both devices so a mode conflict can be resolved in a quick and deterministic manner.

There are two operating states:

- A device has a preferred mode but is willing to use another mode (known as "state 1"), and
- A device requires a specific mode (known as "state 2"). This includes cases where channels are created over ACL-U logical links operating as described in Section 7.10.

In state 1, Basic L2CAP mode has the highest precedence and shall take precedence over Enhanced Retransmission mode and Streaming mode. Enhanced Retransmission mode has the second highest precedence and shall take precedence over all other modes except Basic L2CAP mode. Streaming mode shall have the next level of precedence after Enhanced Retransmission mode.

In state 2, a layer above L2CAP requires Enhanced Retransmission mode or Streaming mode. In this case, the required mode takes precedence over all other modes.

A device does not know in which state the remote device is operating so the state 1 precedence algorithm assumes that the remote device may be a state 2 device. If the mode proposed by the remote device has a higher precedence (according to the state 1 precedence) then the algorithm will operate such that creation of a channel using the remote device's mode has higher priority than disconnecting the channel.

The algorithm for state 1 devices is divided into two parts. Part one covers the case where the remote device proposes a mode with a higher precedence than the state 1 local device. Part two covers the case where the remote device proposes a mode with a lower precedence than the state 1 local device. Part one of the algorithm is as follows:

- When the remote device receives the L2CAP_CONFIGURATION_REQ packet from the local device it will either reject the local device's L2CAP_CONFIGURATION_REQ packet by sending a negative L2CAP_CONFIGURATION_RSP packet or disconnect the channel. The negative L2CAP_CONFIGURATION_RSP packet will contain the remote device's desired mode.
- Upon receipt of the negative L2CAP_CONFIGURATION_RSP packet the local device shall either send a second L2CAP_CONFIGURATION_REQ packet proposing the mode contained in the remote device's negative L2CAP_CONFIGURATION_RSP packet or disconnect the channel.
- When the local device receives the L2CAP_CONFIGURATION_REQ packet from the remote device it shall send a positive L2CAP_CONFIGURATION_RSP packet or disconnect the channel.
- If the mode in the remote Device's negative L2CAP_CONFIGURATION_RSP packet does not match the mode in the remote device's L2CAP_CONFIGURATION_REQ packet then the local device shall disconnect the channel.

Part two of the algorithm is as follows:

- When the local device receives the L2CAP_CONFIGURATION_REQ packet from the remote device it shall reject the L2CAP_CONFIGURATION_REQ packet by sending a negative L2CAP_CONFIGURATION_RSP packet proposing its desired mode. The local device's desired mode shall be the same mode it sent in its L2CAP_CONFIGURATION_REQ packet. Upon receiving the negative

L2CAP_CONFIGURATION_RSP packet the remote device will either send a second L2CAP_CONFIGURATION_REQ packet or disconnect the channel.

- If the local device receives a second L2CAP_CONFIGURATION_REQ packet from the remote device that does not contain the desired mode then the local device shall disconnect the channel.
- If the local device receives a negative L2CAP_CONFIGURATION_RSP packet then it shall disconnect the channel.

An example of the algorithm for state 1 devices is as follows:

- The remote device proposes Basic L2CAP mode in an L2CAP_CONFIGURATION_REQ packet and the local device proposes Enhanced Retransmission mode or Streaming mode. The remote device rejects the local device's L2CAP_CONFIGURATION_REQ packet by sending a negative L2CAP_CONFIGURATION_RSP packet proposing Basic mode. The local device will send a second L2CAP_CONFIGURATION_REQ packet proposing Basic L2CAP mode or disconnect the channel. If the local device sends a second L2CAP_CONFIGURATION_REQ packet that does not propose Basic L2CAP mode then the remote device will disconnect the channel. If the local device rejects the remote device's L2CAP_CONFIGURATION_REQ packet then the remote device will disconnect the channel.

The algorithm for state 2 devices is as follows:

- If the local device proposes a mode in an L2CAP_CONFIGURATION_REQ packet and the remote device proposes a different mode or rejects the local device's L2CAP_CONFIGURATION_REQ packet then the local device shall disconnect the channel.

For Enhanced Retransmission mode and Streaming mode the Retransmission timeout and Monitor Timeout parameters of the Retransmission and Flow Control option parameters may be changed but all other parameters shall not be changed in a subsequent reconfiguration after the channel has reached the OPEN state.

## 5.5 Frame Check Sequence (FCS) option

This option is used to specify the type of Frame Check Sequence (FCS) that will be included on S/I-frames that are sent. It is non-negotiable. The FCS option shall only be used when the mode is configured to, or in an L2CAP_CONFIGURATION_REQ packet that contains an option configuring it to, Enhanced Retransmission mode or Streaming mode. This option shall only be used if the peer L2CAP entity has indicated support for the FCS Option in the Extended Features Mask (see Section 4.12).

Figure 5.6 specifies the format of this option. The Option Type is 0x05. The value 0x00 (No FCS) indicates that the sender does not require the FCS to be included in

S/I-frames. "No FCS" shall only be used if both L2CAP entities send the FCS Option with value "No FCS" in an L2CAP_CONFIGURATION_REQ packet. If one L2CAP entity sends the FCS Option with "No FCS" in an L2CAP_CONFIGURATION_REQ packet and the other L2CAP sends the FCS Option with a value other than "No FCS" then the default shall be used. If one or both L2CAP entities do not send the FCS option in an L2CAP_CONFIGURATION_REQ packet then the default shall be used.

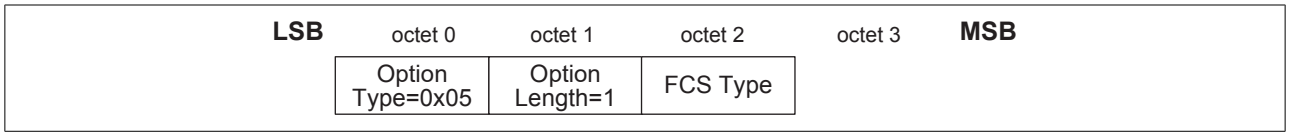



*Figure 5.6: FCS option format*

The FCS types are shown in Table 5.3

| Value | Description |
|---|---|
| 0x00 | No FCS |
| 0x01 | 16-bit FCS defined in Section 3.3.5 (default) |
| Other | Reserved for future use |

*Table 5.3: FCS types*

## 5.6 Extended Flow Specification option

This option specifies a flow specification for requesting a desired Quality of Service (QoS) on a channel. It is non-negotiable. Extended Flow Specification may be supported on channels created over ACL-U logical links (see Section 7.10). If both devices show support for Extended Flow Specification for BR/EDR in the Extended Feature mask (see Table 4.12) then all channels created between the two devices shall use an Extended Flow Specification. The Quality of Service option and Flush Timeout option shall not be used if the Extended Flow Specification is used.

The parameters in the Extended Flow Specification option specify the traffic stream in the outgoing direction (transmitted traffic). The Option Type is 0x06. Figure 5.7 specifies the format of this option.

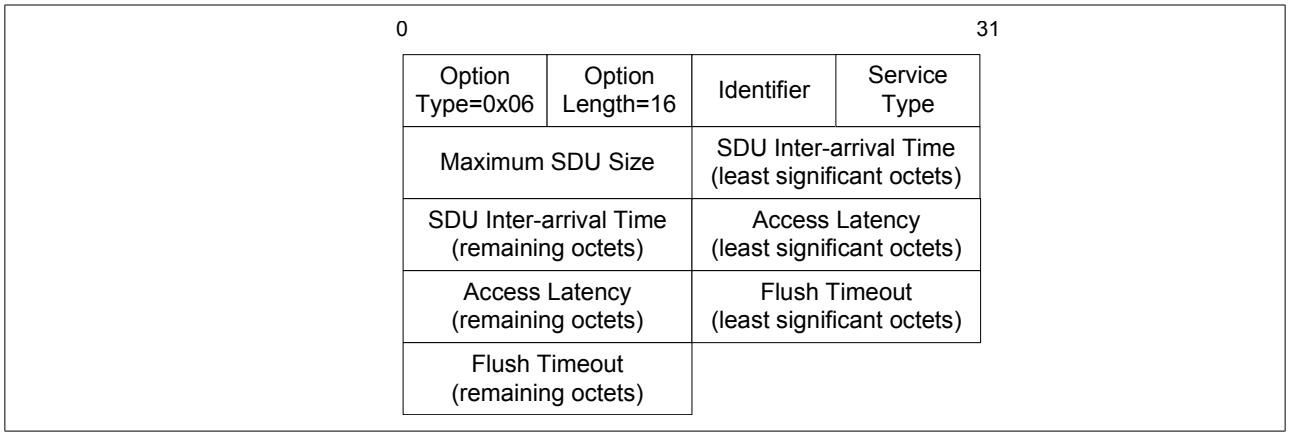



*Figure 5.7: Extended Flow Specification option format*

If no Extended Flow Specification is provided by the upper layer, an Extended Flow Specification with following default values shall be used:

| Qos Parameter | Default Value |
|---|---|
| Identifier | 0x01 |
| Service Type | Best Effort |
| Maximum SDU size | 0xFFFF |
| SDU Inter-arrival Time | 0xFFFFFFFF |
| Access Latency | 0xFFFFFFFF |
| Flush Timeout | 0xFFFFFFFF |

*Table 5.4: Default values for Extended Flow Specification*

For a Best Effort channel no latency, air-time or bandwidth guarantees shall be assumed.

The parameters of the Extended Flow Specification are shown in Table 5.5:

| QoS parameter | Parameter Size in Octets | Unit |
|---|---|---|
| Identifier | 1 | *none* |
| Service Type | 1 | *none* |
| Maximum SDU Size | 2 | octets |
| SDU Inter-arrival Time | 4 | microseconds |
| Access Latency | 4 | microseconds |
| Flush Timeout | 4 | microseconds |

*Table 5.5: Traffic parameters for L2CAP QoS configuration*

- *Identifier (1 octet)*

  This field provides a unique identifier for the flow specification. This identifier is used by some Controllers in the process of setting up the QoS request. Each active flow specification sent by a device to configure an L2CAP channel shall have a unique Identifier. An Identifier can be reused when the L2CAP channel associated with the flow spec is disconnected. The Identifier shall be unique within the scope of a physical link. Extended Flow Specifications for channels on different physical links may have the same Identifier. The Identifier for an Extended Flow Specification with Service Type Best Effort shall be 0x01.

  Since the Identifier for an Extended Flow Specification with Service Type Best Effort is fixed to 0x01 it is possible to generate a Best Effort Extended Flow Specification for the remote device without performing the Lockstep Configuration process. (The Lockstep Configuration process is described in Section 7.1.3).

- *Service Type (1 octet)*

  This field indicates the level of service required. Table 5.6 defines the different Service Types values. The default value is 'Best effort'. If 'Best effort' is selected then Access Latency and Flush Timeout shall both be set to 0xFFFFFFFF. Maximum SDU size and SDU Inter-arrival Time are used to indicate the maximum data rate that the application can deliver to L2CAP for transmission. The remote device should respond with lower settings indicating the maximum rate at which it can receive data (for example, maximum rate data it can write to a mass storage device, etc.). Values of 0xFFFF for Maximum SDU size and 0xFFFFFFFF for SDU Inter-arrival time are used when the actual values are not known. If Maximum SDU size is set to 0xFFFF then SDU Inter-arrival time shall be set to 0xFFFFFFFF, if SDU Inter-arrival time is set to 0xFFFFFFFF then Maximum SDU size shall be set to 0xFFFF. This tells the Controller to allocate as much bandwidth as possible.

If “Guaranteed” is selected the QoS parameters can be used to identify different types of Guaranteed traffic.

- **Guaranteed bandwidth** traffic is traffic with a minimum data rate but no particular latency requested. Latency will be related to the link supervision timeout. For this type of traffic Access Latency is set to 0xFFFFFFFF.
- **Guaranteed Latency** traffic is traffic with only latency requirements. For this type of traffic SDU Inter-arrival time is set to 0xFFFFFFFF. HID interrupt channel and AVRCP are examples of this type of traffic.
- **Both Guaranteed Latency and Bandwidth** traffic has both a latency and bandwidth requirement. An example is Audio/Video streaming.

If 'No Traffic' is selected the remainder of the fields shall be ignored because there is no data being sent across the channel in the outgoing direction.

| Value | Description |
|---|---|
| 0x00 | No Traffic |
| 0x01 | Best effort (Default) |
| 0x02 | Guaranteed |
| Other | Reserved for future use |

*Table 5.6: Service type definitions*

A channel shall not be configured as “Best Effort” in one direction and “Guaranteed” in the other direction. If a channel is configured in this way it shall be disconnected. A channel may be configured as “No Traffic” in one direction and “Best Effort” in the other direction. The “No Traffic” refers to the application traffic not the Enhanced Retransmission mode supervisory traffic. A channel configured in this way is considered to have a service type of Best Effort. A channel may be configured as “No Traffic” in one direction and “Guaranteed” in the other direction. A channel configure in this way is considered to have a service type of Guaranteed.

Once configured the service type of a channel shall not be changed during reconfiguration.

- *Maximum SDU Size (2 octets)*

  The Maximum SDU Size parameter specifies the maximum size of the SDUs transmitted by the application. If the Service Type is “Guaranteed” then traffic submitted to L2CAP with a larger size is considered non-conformant. QoS guarantees are only provided for conformant traffic.

- *SDU Inter-arrival time (4 octets)*

  The SDU Inter-arrival time parameter specifies the time between consecutive SDUs generated by the application. For streaming traffic, SDU Inter-arrival time should be set to the average time between SDUs. For variable rate traffic and best effort traffic, SDU Inter-arrival time should be set to the minimum time between SDUs. If the Service Type is “Guaranteed” then traffic submitted to L2CAP with a smaller interval, is considered non-conformant. QoS guarantees are only provided for conformant traffic.

- *Access Latency (4 octets)*

  The Access Latency parameter specifies the maximum delay between consecutive transmission opportunities on the air-interface for the connection. Access latency is based on the time base of the Controller, which is not necessarily synchronous to the time base being used by the Host or the application.

  For streaming traffic (such as in A2DP), the Access Latency should be set to indicate the time budgeted for transmission of the data over the air, and would normally be

roughly equal to the Flush Timeout minus the duration of streaming data which can be stored in the receive side application buffers.

For non-streaming, bursty traffic (such as in HID and AVRCP), the Access Latency parameter value sent in the L2CAP_CONFIGURATION_REQ packet should be set to the desired latency, minus any HCI transport delays and any other stack delays that may be expected on the device and on target Host systems. The remote device receiving the L2CAP_CONFIGURATION_REQ packet may send an Access Latency parameter value in the L2CAP_CONFIGURATION_RSP packet which is equal to or lower than the value it received. The remote device may send a lower value to account for other traffic it may be carrying, or overhead activities it may be carrying out.

If HCI is used then the Host should take into account the latency of the HCI transport when determining the value for the Access Latency. For example if the application requires an Access Latency of 20 ms and the HCI transport has a latency of 5 ms then the value for Access Latency should be 15 ms.

- *Flush Timeout (4 Octets)*

  The Flush Timeout defines a maximum period after which all segments of the SDU are flushed from L2CAP and the Controller. A Flush Timeout value of 0xFFFFFFFF indicates that data will not be discarded by the transmitting side, even if the link becomes congested, and thus in this case data is treated as reliable and is never flushed.

  The device receiving the L2CAP_CONFIGURATION_REQ packet with Flush Timeout set to 0xFFFFFFFF should not modify this parameter. The Flush Timeout for a “Best Effort” channel shall be set to 0xFFFFFFFF.

  However, if the Traffic Type is “Guaranteed” and the transmit side buffer is limited, then the Flush Timeout parameter given in the L2CAP_CONFIGURATION_REQ packet may be set to a value corresponding to the duration of streaming data which the transmit buffer can hold before it must begin discarding data. The side receiving the L2CAP_CONFIGURATION_REQ packet may then set the Flush Timeout parameter in the L2CAP_CONFIGURATION_RSP packet to a lower value if the receive side buffer is smaller than the transmit side buffer.

  Note: The total available buffer space is typically a combination of application buffers, any buffers maintained by the L2CAP implementation, and HCI buffers provided by the Controller.

  The Flush Timeout should normally be set to a value which is larger than the Access Latency, and which also accounts for buffers maintained by the application on the receive side such as de-jitter buffers used in audio and video streaming applications. In general, the Flush Timeout value should be selected to ensure that the Flush Timeout expires when the application buffers are about to be exhausted.

Flush Timeout may be set to 0x00000000 to indicate that no retransmissions are to occur. Data may be flushed immediately after transmission in this case. This behavior is useful in applications such as gaming where the tolerable latency is on the order of a few milliseconds, and hence the information contained in a packet will become stale very rapidly. In such applications, it is preferable to send “fresher” data if the last SDU submitted for transmission was not transmitted as a result of interference.

## 5.7 Extended Window Size option

This option is used to negotiate the maximum extended window size. The Option Type is 0x07 and the option is non-negotiable. Figure 5.8 specifies the format of this option.

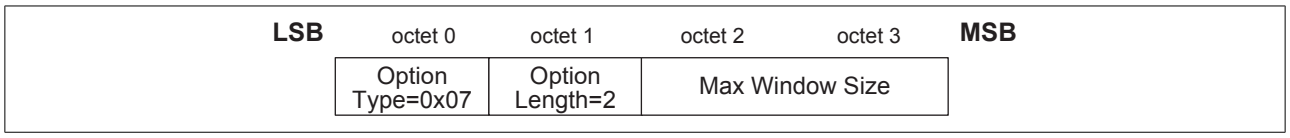



*Figure 5.8: Extended Window Size option format*

The allowable values for the Maximum Extended Window Size are:

| Value | Description |
|---|---|
| 0x0000 | Invalid value for Enhanced Retransmission mode<br>Valid for Streaming mode |
| 0x0001 to 0x3FFF | Valid maximum window size (frames) for Enhanced Retransmission mode. Invalid for Streaming mode |
| Other | Reserved for future use |

*Table 5.7: Extended Window Size values*

This option shall only be sent if the peer L2CAP entity has indicated support for the Extended Window size feature in the Extended Features Mask (see Section 4.12).

This option shall be ignored if the channel is configured to use Basic L2CAP mode (see Section 5.4).

For Enhanced Retransmission mode, this option has the same directional semantics as the Retransmission and Flow Control option (see Section 5.4). The sender of an L2CAP_CONFIGURATION_REQ packet containing this option is suggesting the maximum window size (possibly based on its own internal L2CAP receive buffer resources) that the peer L2CAP entity should use when sending data.

For Streaming mode, this option is used to enable use of the Extended Control Field and if sent, shall have a value of 0.

If this option is successfully configured then the Maximum Window Size negotiated using the Retransmission and Flow Control option (see Section 5.4) shall be ignored.

If this option is successfully configured in either direction then both L2CAP entities shall use the Extended Control Fields in all S/I-frames (see Section 3.3.2).

If the option is only configured in one direction then the maximum window size for the opposite direction shall be taken from the maximum window size value in the existing Retransmission and Flow Control option. In this configuration both L2CAP entities shall use the extended control fields in all S/I-frames.

# 6 STATE MACHINE

The state machine does not necessarily represent all possible scenarios.

## 6.1 General rules for the state machine

It is implementation specific, and outside the scope of the specification, how the transmissions are triggered.

“Ignore” means that the signal can be silently discarded.

The following states have been defined to clarify the protocol; the actual number of states and naming in a given implementation is outside the scope of the specification:

- CLOSED – channel not connected.
- WAIT_CONNECT – a connection request has been received, but only a connection response with indication “pending” can be sent.
- WAIT_CONNECT_RSP – a connection request has been sent, pending a positive connect response.
- CONFIG – the different options are being negotiated for both sides; this state comprises a number of substates, see Section 6.1.3.
- OPEN – user data transfer state.
- WAIT_DISCONNECT – a disconnect request has been sent, pending a disconnect response.

In the following state tables and descriptions, the L2CAP_Data message corresponds to one of the PDU formats used on connection-oriented data channels as described in Section 3, including PDUs containing B-frames, I-frames, or S-frames.

Some state transitions and actions are triggered only by internal events effecting one of the L2CAP entity implementations, not by preceding L2CAP signaling messages. It is implementation-specific and out of the scope of the specification, how these internal events are realized; just for the clarity of specifying the state machine, the following abstract internal events are used in the state event tables, as far as needed:

- *OpenChannel_Req* – a local L2CAP entity is requested to set up a new connection-oriented channel.
- *OpenChannel_Rsp* – a local L2CAP entity is requested to finally accept or refuse a pending connection request.
- *ConfigureChannel_Req* – a local L2CAP entity is requested to initiate an outgoing configuration request.

- *CloseChannel_Req* – a local L2CAP entity is requested to close a channel.
- *SendData_Req* – a local L2CAP entity is requested to transmit an SDU.
- *ReconfigureChannel_Req* – a local L2CAP entity is requested to reconfigure the parameters of a connection-oriented channel.
- *ControllerLogicalLinkInd* – a Controller indicates the acceptance or rejection of a logical link request with an Extended Flow Specification.

There is a single state machine for each L2CAP connection-oriented channel that is active. A state machine is created for each new L2CAP_CONNECTION_REQ received. The state machine always starts in the CLOSED state.

To simplify the state event tables, the RTX and ERTX timers, as well as the handling of request retransmissions are described in Section 6.2 and not included in the state tables.

L2CAP messages not bound to a specific data channel and thus not impacting a channel state (e.g. L2CAP_INFORMATION_REQ, L2CAP_ECHO_REQ) are not covered in this section.

The following states and transitions are illustrated in Figure 6.1.

### 6.1.1 CLOSED state

| Event | Condition | Action | Next State |
|---|---|---|---|
| *OpenChannel_req* | - | Send L2CAP_CONNEC-TION_REQ | WAIT_-CONNECT_RSP |
| L2CAP_CONNEC-TION_REQ | Normal, connection is possible | Send L2CAP_CONNEC-TION_RSP (success) | CONFIG (substate WAIT_CONFIG) |
| L2CAP_CONNEC-TION_REQ | Need to indicate pending | L2CAP_CONNEC-TION_RSP (pending) | WAIT_CONNECT |
| L2CAP_CONNEC-TION_REQ | No resource, not approved, etc. | Send L2CAP_CONNEC-TION_RSP (refused) | CLOSED |
| L2CAP_CONNEC-TION_RSP | - | Ignore | CLOSED |
| L2CAP_CONFIGURA-TION_REQ | - | Send L2CAP_COM-MAND_REJECT_RSP (with reason Invalid CID) | CLOSED |
| L2CAP_CONFIGURA-TION_RSP | - | Ignore | CLOSED |

| Event | Condition | Action | Next State |
|---|---|---|---|
| L2CAP_DISCONNEC-TION_REQ | - | Send L2CAP_DISCONNEC-TION_RSP | CLOSED |
| L2CAP_DISCONNEC-TION_RSP | - | Ignore | CLOSED |
| L2CAP_Data | - | Ignore | CLOSED |

*Table 6.1: CLOSED state event table*

Note: The L2CAP_CONNECTION_REQ message is not mentioned in any of the other states apart from the CLOSED state, as it triggers the establishment of a new channel, thus the branch into a new instance of the state machine.

### 6.1.2 WAIT_CONNECT_RSP state

| Event | Condition | Action | Next State |
|---|---|---|---|
| L2CAP_CONNEC-TION_RSP | Success indica-ted in result | Send L2CAP_CONFIGURA-TION_REQ | CONFIG (substate WAIT_CONFIG) |
| L2CAP_CONNEC-TION_RSP | Result pending | - | WAIT_CONNECT_RSP |
| L2CAP_CONNEC-TION_RSP | Remote side refu-ses connection | - | CLOSED |
| L2CAP_CONFIGU-RATION_REQ | - | Send L2CAP_COMMAND_-REJECT_RSP (with reason Invalid CID) | WAIT_CONNECT_RSP |
| L2CAP_CONFIGU-RATION_RSP | - | Ignore | WAIT_CONNECT_RSP |
| L2CAP_DISCON-NECTION_RSP | - | Ignore | WAIT_CONNECT_RSP |
| L2CAP_Data | - | Ignore | WAIT_CONNECT_RSP |

*Table 6.2: WAIT_CONNECT_RSP state event table*

Note: An L2CAP_DISCONNECTION_REQ message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

### 6.1.3 WAIT_CONNECT state

| Event | Condition | Action | Next State |
|---|---|---|---|
| *OpenChannel_Rsp* | Pending connection request is finally acceptable | Send L2CAP_CONNECTION_RSP (success) | CONFIG (substate WAIT_CONFIG) |
| *OpenChannel_Rsp* | Pending connection request is finally refused | Send L2CAP_CONNECTION_RSP (refused) | CLOSED |
| L2CAP_CONNECTION_RSP | - | Ignore | WAIT_CONNECT |
| L2CAP_CONFIGURATION_RSP | - | Ignore | WAIT_CONNECT |
| L2CAP_DISCONNECTION_RSP | - | Ignore | WAIT_CONNECT |
| L2CAP_Data | - | Ignore | WAIT_CONNECT |

*Table 6.3: WAIT_CONNECT state event table*

Note: An L2CAP_DISCONNECTION_REQ or L2CAP_CONFIGURATION_REQ message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

### 6.1.4 CONFIG state

Two configuration processes exist as described in Section 7.1. The configuration processes are the Standard process and the Lockstep process.

In the Standard and Lockstep configuration processes both L2CAP entities initiate a configuration request during the configuration process. This means that each device adopts an initiator role for the outgoing configuration request, and an acceptor role for the incoming configuration request. Configurations in both directions may occur sequentially, but can also occur in parallel.

In the Lockstep configuration process both L2CAP entities send L2CAP_CONFIGURATION_REQ packets and receive L2CAP_CONFIGURATION_RSP packets with a pending result code before submitting the flow specifications to their local Controllers. A final L2CAP_CONFIGURATION_RSP packet is sent by each L2CAP entity indicating the response from its local Controller.

The following substates are distinguished within the CONFIG state:

- WAIT_CONFIG – a device has sent or received a connection response, but has neither initiated a configuration request yet, nor received a configuration request with acceptable parameters.
- WAIT_SEND_CONFIG – for the initiator path, a configuration request has not yet been initiated, while for the response path, a request with acceptable options has been received.
- WAIT_CONFIG_REQ_RSP – for the initiator path, a request has been sent but a positive response has not yet been received, and for the acceptor path, a request with acceptable options has not yet been received.
- WAIT_CONFIG_RSP – the acceptor path is complete after having responded to acceptable options, but for the initiator path, a positive response on the recent request has not yet been received.
- WAIT_CONFIG_REQ – the initiator path is complete after having received a positive response, but for the acceptor path, a request with acceptable options has not yet been received.
- WAIT_IND_FINAL_RSP – for both the initiator and acceptor, the Extended Flow Specification has been sent to the local Controller but neither a response from Controller nor the final configuration response has yet been received.
- WAIT_FINAL_RSP – the device received an indication from the Controller accepting the Extended Flow Specification and has sent a positive response. It is waiting for the remote device to send a configuration response.
- WAIT_CONTROL_IND – the device has received a positive response and is waiting for its Controller to accept or reject the Extended Flow Specification.

According to Section 6.1.1 and Section 6.1.2, the CONFIG state is entered via WAIT_CONFIG substate from either the CLOSED state, the WAIT_CONNECT state, or the WAIT_CONNECT_RSP state. The CONFIG state is left for the OPEN state if both the initiator and acceptor paths complete successfully.

For better overview, separate tables are given: Table 6.4 shows the success transitions; therein, transitions on one of the minimum paths (no previous non-success transitions) are shaded. Table 6.5 shows the non-success transitions within the configuration process, and Table 6.6 shows further transition cause by events not belonging to the configuration process itself. The following configuration states and transitions are illustrated in Figure 6.2.

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG | *ConfigureChannel_ Req* | Standard process | Send L2CAP_CONFIGURATION_REQ | WAIT_CONFIG_REQ_RSP |
| WAIT_CONFIG | *ConfigureChannel_ Req* | Lockstep process | Send L2CAP_CONFIGURATION_REQ (Ext Flow Spec plus other options) | WAIT_CONFIG_REQ_RSP |
| WAIT_CONFIG | L2CAP_CONFIGURATION_REQ | Options acceptable<br>Standard process | Send L2CAP_CONFIGURATION_RSP (success) | WAIT_SEND_CONFIG |
| WAIT_CONFIG | L2CAP_CONFIGURATION_REQ | Options acceptable<br>Lockstep process | Send L2CAP_CONFIGURATION_RSP (pending) | WAIT_SEND_CONFIG |
| WAIT_CONFIG_REQ_RSP | L2CAP_CONFIGURATION_REQ | Options acceptable | Send L2CAP_CONFIGURATION_RSP (success) | WAIT_CONFIG_RSP |
| WAIT_CONFIG_REQ_RSP | L2CAP_CONFIGURATION_RSP | Remote side accepts options | (continue waiting for configuration request) | WAIT_CONFIG_REQ |
| WAIT_CONFIG_REQ | L2CAP_CONFIGURATION_REQ | Options acceptable<br>Standard process | Send L2CAP_CONFIGURATION_RSP (success) | OPEN |
| WAIT_CONFIG_REQ | L2CAP_CONFIGURATION_REQ | Options acceptable<br>Lockstep process | Send L2CAP_CONFIGURATION_RSP (pending) | WAIT_IND_FINAL_RSP |
| WAIT_SEND_CONFIG | *ConfigureChannel_ Req* | *none* | Send L2CAP_CONFIGURATION_REQ | WAIT_CONFIG_RSP |
| WAIT_CONFIG_RSP | L2CAP_CONFIGURATION_RSP | Remote side accepts options<br>Standard process | *none* | OPEN |

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG_RSP | L2CAP_CONFIG-URATION_RSP | Remote side accepts op-tion<br>Lockstep proc | *none* | WAIT_IND_-FINAL_RSP |
| WAIT_IND_FINAL_-RSP | ControllerLogical LinkInd | Reject | Send L2CAP_CONFIG-URATION_RSP (fail) | WAIT_CONFIG |
| WAIT_IND_FINAL_-RSP | ControllerLogical LinkInd | Accept | Send L2CAP_CONFIG-URATION_RSP (success) | WAIT_FINAL_-RSP |
| WAIT_IND_FINAL_-RSP | L2CAP_CONFIG-URATION_RSP | Remote side fail | *none* | WAIT_CONFIG |
| WAIT_IND_FINAL_-RSP | L2CAP_CONFIG-URATION_RSP | Remote side success | *none* | WAIT_-CONTROL_IND |
| WAIT_FINAL_RSP | L2CAP_CONFIG-URATION_RSP | Remote side fail | *none* | WAIT_CONFIG |
| WAIT_FINAL_RSP | L2CAP_CONFIG-URATION_RSP | Remote side success | *none* | OPEN |
| WAIT_CONTROL_IND | ControllerLogical LinkInd | Reject | Send L2CAP_CONFIG-URATION_RSP (fail) | WAIT_CONFIG |
| WAIT_CONTROL_IND | ControllerLogical LinkInd | Accept | Send L2CAP_CONFIG-URATION_RSP (success) | OPEN |

*Table 6.4: CONFIG state event table*

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG | L2CAP_CONFIG-URATION_REQ | Options not acceptable | Send L2CAP_CONFIG-URATION_RSP (fail) | WAIT_CONFIG |
| WAIT_CONFIG | L2CAP_CONFIG-URATION_RSP | *none* | Ignore | WAIT_CONFIG |
| WAIT_SEND_CONFIG | L2CAP_CONFIG-URATION_RSP | *none* | Ignore | WAIT_SEND_-CONFIG |

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG_-REQ_RSP | L2CAP_CONFIG-URATION_REQ | Options not acceptable | Send L2CAP_CONFIG-URATION_RSP (fail) | WAIT_CONFIG_-REQ_RSP |
| WAIT_CONFIG_-REQ_RSP | L2CAP_CONFIG-URATION_RSP | Remote side rejects op-tions | Send L2CAP_CONFIG-URATION_REQ (new options) | WAIT_CONFIG_-REQ_RSP |
| WAIT_CONFIG_REQ | L2CAP_CONFIG-URATION_REQ | Options not acceptable | Send L2CAP_CONFIG-URATION_RSP (fail) | WAIT_CONFIG_-REQ |
| WAIT_CONFIG_REQ | L2CAP_CONFIG-URATION_RSP | *none* | Ignore | WAIT_CONFIG_-REQ |
| WAIT_CONFIG_RSP | L2CAP_CONFIG-URATION_RSP | Remote side rejects op-tions | Send L2CAP_CONFIG-URATION_REQ (new options) | WAIT_CONFIG_-RSP |

*Table 6.5: CONFIG state/substates event table: non-success transitions within configuration process*

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| CONFIG (any sub-state) | *CloseChan-nel_Req* | Any internal rea-son to stop | Send L2CAP_DISCON-NECTION_REQ | WAIT_-DISCONNECT |
| CONFIG (any sub-state) | L2CAP_DIS CONNECTI ON_REQ | *none* | Send L2CAP_DISCON-NECTION_RSP | CLOSED |
| CONFIG (any sub-state) | L2CAP_DIS CONNECTI ON_RSP | *none* | Ignore | CONFIG (remain in substate) |
| CONFIG (any sub-state) | L2CAP_Da-ta | *none* | Process the PDU | CONFIG (remain in substate) |

*Table 6.6: CONFIG state/substates event table: events not related to configuration process*

**Notes:**

- Receiving data PDUs (L2CAP_Data) in CONFIG state should be relevant only in case of a transition to a reconfiguration procedure (from OPEN state). Discarding the received data is allowed only in Retransmission mode and Enhanced Retransmission mode. Discarding an S-frame is allowed but not recommended. If a S-frame is discarded, the monitor timer will cause a new S-frame to be sent after a time out.
- Indicating a failure in a configuration response does not necessarily imply a failure of the overall configuration procedure; instead, based on the information received in the negative response, a modified configuration request may be triggered.

### 6.1.5 OPEN state

| Event | Condition | Action | Next State |
|---|---|---|---|
| *SendData_req* | *none* | Send L2CAP_Data packet according to configured mode | OPEN |
| *ReconfigureChannel_Req* | *none* | Complete outgoing SDU<br>Send L2CAP_CONFIGURATION_REQ | CONFIG (substate WAIT_CONFIG_REQ_RSP) |
| *CloseChannel_Req* | *none* | Send L2CAP_DISCONNECTION_REQ | WAIT_-DISCONNECT |
| L2CAP_CONNECTION_RSP | *none* | Ignore | OPEN |
| L2CAP_CONFIGURATION_-REQ | Incoming config options acceptable | Complete outgoing SDU<br>Send L2CAP_CONFIGURATION_RSP (ok) | CONFIG (substate WAIT_SEND_-CONFIG) |
| L2CAP_CONFIGURATION_-REQ | Incoming config options not acceptable | Complete outgoing SDU<br>Send L2CAP_CONFIGURATION_RSP (fail) | OPEN |
| L2CAP_DISCONNECTION_-REQ | *none* | Send L2CAP_DISCONNECTION_RSP | CLOSED |
| L2CAP_DISCONNECTION_-RSP | *none* | Ignore | OPEN |
| L2CAP_Data | *none* | Process the PDU | OPEN |

*Table 6.7: OPEN state event table*

The outgoing SDU shall be completed from the view of the remote entity. Therefore all PDUs forming the SDU shall have been reliably transmitted by the local entity and acknowledged by the remote entity, before entering the configuration state.

### 6.1.6 WAIT_DISCONNECT state

| Event | Condition | Action | Next State |
|---|---|---|---|
| L2CAP_CONNECTION_RSP | *none* | Ignore | WAIT_DIS-CONNECT |
| L2CAP_CONFIGURATION_-REQ | *none* | Send L2CAP_COMMAND_-REJECT_RSP with reason Invalid CID | WAIT_DIS-CONNECT |
| L2CAP_CONFIGURATION_-RSP | *none* | Ignore | WAIT_DIS-CONNECT |
| L2CAP_DISCONNECTION_-REQ | *none* | Send L2CAP_DISCONNECTION_-RSP | CLOSED |
| L2CAP_DISCONNECTION_-RSP | *none* | *none* | CLOSED |
| L2CAP_Data | *none* | Ignore | WAIT_DIS-CONNECT |

*Table 6.8: WAIT_DISCONNECT state event table*

### 6.1.7 [This section is no longer used]

### 6.1.8 [This section is no longer used]

### 6.1.9 [This section is no longer used]

### 6.1.10 [This section is no longer used]

### 6.1.11 [This section is no longer used]

### 6.1.12 [This section is no longer used]

## 6.2 Timers events

### 6.2.1 RTX

The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signaling requests. This timer is started when a signaling request (see Section 7) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value shall be reset to a new value at least double the previous value. When retransmitting the Request message, the context of the same state shall be assumed as with the original transmission. If a Request message is received that is identified as a duplicate (retransmission), it shall be processed in the context of the same state which applied when the original Request message was received.

Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before terminating the channel identified by the Requests. The exception is fixed channel CIDs since they can never be terminated. On the LE transport, the Peripheral's Host may disconnect the link on the expiry of the RTX timer. The decision should be based on the flush timeout of the signaling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level.

For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level. When terminating the channel, it is not necessary to send an L2CAP_DISCONNECTION_REQ and enter WAIT_DISCONNECT state. Channels can be transitioned directly to the CLOSED state.

The value of this timer is implementation-dependent but the minimum initial value is 1 second and the maximum initial value is 60 seconds. One RTX timer shall exist for each outstanding signaling request, including each L2CAP_ECHO_REQ. The timer disappears on the final expiration, when the response is received, or the physical link is lost. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 60 seconds.

### 6.2.2 ERTX

The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing additional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an L2CAP_CONNECTION_RSP event with a “connect pending” result (0x0001) is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be disconnected.

If a duplicate Request is sent, the particular ERTX timer disappears, replaced by a new RTX timer and the whole timing procedure restarts as described previously for the RTX timer.

The value of this timer is implementation-dependent but the minimum initial value is 60 seconds and the maximum initial value is 300 seconds. Similar to RTX, there shall be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 300 seconds. When terminating the channel, it is not necessary to send an L2CAP_DISCONNECTION_REQ and enter WAIT_DISCONNECT state. Channels should be transitioned directly to the CLOSED state.

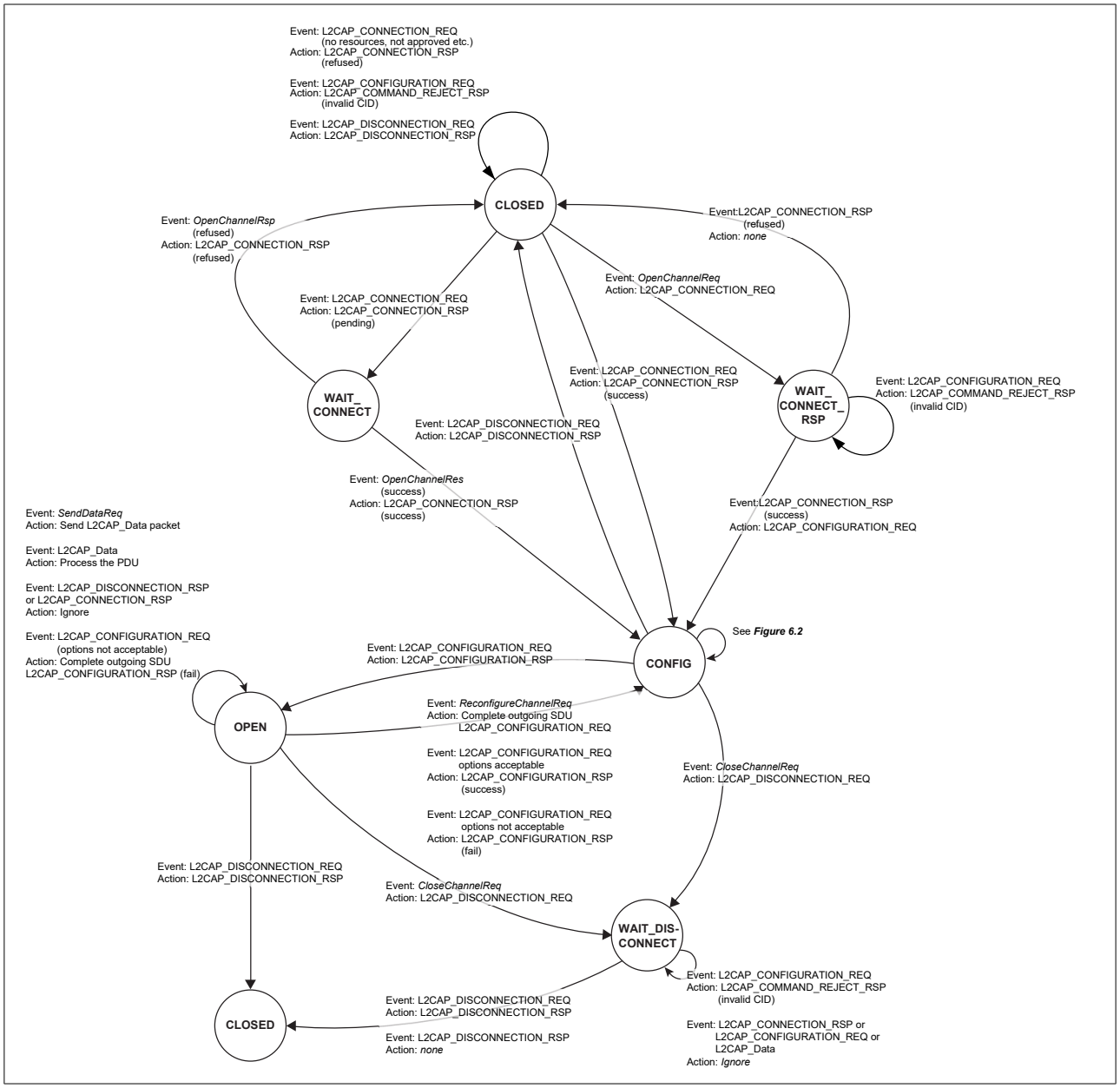



*Figure 6.1: States and transitions*

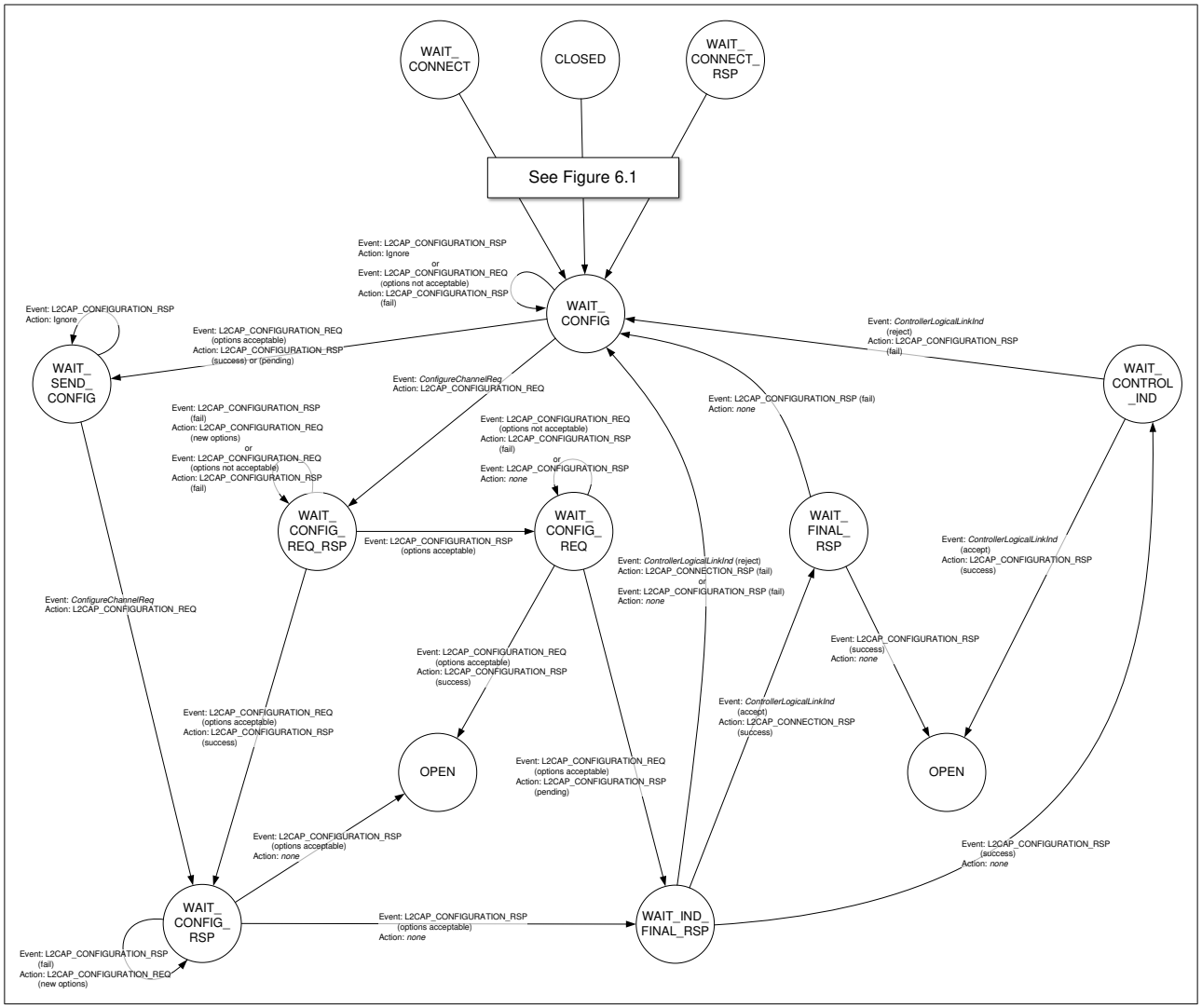



*Figure 6.2: Configuration states and transitions*

# 7 GENERAL PROCEDURES

This section describes the general operation of L2CAP, including the configuration process, the handling and the processing of user data for transportation over the air interface. This section also describes the operation of L2CAP features including the delivery of erroneous packets, the flushing of expired data and operation in connectionless mode, operation collision resolution, aggregation of best effort flow specifications, and prioritizing data over HCI.

Procedures for the flow control and retransmission modes are described in Section 8.

## 7.1 Configuration process

Configuration consists of two processes, the Standard process and the Lockstep process. The Lockstep process shall be used if both L2CAP entities support the Extended Flow Specification option otherwise the Standard process shall be used.

For both processes, configuring the channel parameters shall be done independently for both directions. Both configurations may be done in parallel.

Each configuration parameter is one-directional. The configuration parameters describe the non default parameters the device sending an L2CAP_CONFIGURATION_REQ packet will accept. The configuration request cannot request a change in the parameters the device receiving the request will accept.

If a device needs to establish the value of a configuration parameter the remote device will accept, then it must wait for an L2CAP_CONFIGURATION_REQ packet containing that configuration parameter to be sent from the remote device.

The Lockstep process is used when the channel parameters include an Extended Flow Specification option. The Lockstep process can be divided into two phases. In the first phase, each L2CAP entity shall receive an Extended Flow Specification option along with all non-default parameters from its peer L2CAP entity and then present the pair of Extended Flow specifications (local and remote) to its local Controller. In the second phase, each L2CAP entity shall report the result from its local Controller to the peer L2CAP entity. The Lockstep process is described in Section 7.1.3. The Standard process is described in Section 7.1.4.

Both the Lockstep and Standard processes can be abstracted into the initial Request configuration path and a Response configuration path, followed by the reverse direction phase. Reconfiguration follows a similar two-phase process by requiring configuration in both directions.

During a reconfiguration session, all data traffic on the channel should be suspended pending the outcome of the configuration. If an L2CAP entity receives an L2CAP_CONFIGURATION_REQ packet while it is waiting for a response it shall not block sending the L2CAP_CONFIGURATION_RSP packet, otherwise the configuration process may deadlock.

### 7.1.1 Request Path

The Request Path can configure the following:

- requester's incoming MTU
- requester's outgoing flush timeout
- requester's outgoing QoS
- requester's incoming and outgoing flow and error control information
- requester's incoming and outgoing Frame Check Sequence option
- requester's outgoing QoS using Extended Flow Specification option
- requester's incoming Extended Window size option plus incoming and outgoing frame format.

Table 7.1 defines the configuration options that may be placed in an L2CAP_CONFIGURATION_REQ packet.

| Parameter | Description |
|---|---|
| MTU | Incoming MTU information |
| FlushTO | Outgoing flush timeout[1] |
| QoS | Outgoing QoS information[1] |
| RFCMode | Incoming and outgoing Retransmission and Flow Control mode |
| FCS | Incoming and outgoing Frame Check Sequence |
| ExtFlowSpec | Outgoing QoS information[1] |
| ExtWindow | Incoming Extended Window size plus incoming and outgoing frame format |

*Table 7.1: Parameters allowed in request*

[1]FlushTO, QoS and ExtFlowSpec are considered QoS information. ExtFlowSpec is used instead of FlushTO and QoS when both devices support ExtFlowSpec.

The state machine for the configuration process is described in Section 6.

### 7.1.2 Response Path

The Response Path can configure the following:

- responder's outgoing MTU, that is the remote side's incoming MTU
- remote side's flush timeout
- responder's incoming QoS Flow Specification (remote side's outgoing QoS Flow Specification)
- responder's incoming and outgoing Flow and Error Control information
- responder's incoming QoS Extended Flow Specification (remote side's outgoing QoS Flow Specification)
- responder's outgoing Extended Window size.

For the Standard process, if a request-oriented parameter is not present in the Request message (reverts to last agreed value), the remote side may negotiate for a non-default value by including the proposed value in a negative Response message. Table 7.2 defines the configuration options that may be placed in an L2CAP_CONFIGURATION_RSP packet.

| Parameter | Description |
|---|---|
| MTU | Outgoing MTU information |
| FlushTO | Incoming flush timeout[1] |
| QoS | Incoming QoS information[1] |
| RFCMode | Incoming and outgoing Retransmission and Flow Control mode |
| ExtFlowSpec | Incoming QoS information[1] |
| ExtWindow | Outgoing Extended Window size |

*Table 7.2: Parameters allowed in response*

[1]FlushTO, QoS and ExtFlowSpec are considered QoS information. ExtFlowSpec is used instead of FlushTO and QoS when both devices support ExtFLowSpec

### 7.1.3 Lockstep Configuration process

L2CAP_CONFIGURATION_REQ packets are sent to establish or change the channel parameters including the QoS contract between two L2CAP entities. An Extended Flow Specification option along with any non-default parameters shall be sent in an L2CAP_CONFIGURATION_REQ packet following the sending or receipt of a Connect Response which accepts an L2CAP Connect Request for the channel being configured. Unlike the Standard process, negotiation involving the sending of multiple L2CAP_CONFIGURATION_REQ packets shall not be performed. In other words, only one L2CAP_CONFIGURATION_REQ packet shall be sent by each L2CAP entity during

the Lockstep configuration process. The Lockstep process shall only be used during channel reconfiguration when an Extended Flow Specification option is present in the L2CAP_CONFIGURATION_REQ packet used for reconfiguration.

The Extended Flow Specification shall only be sent for channels created on the ACL-U logical link if both the local and remote L2CAP entities have indicated support for the Extended Flow Specification for BR/EDR in their Extended Features masks. The Extended Features mask shall be obtained via the L2CAP_INFORMATION_REQ/L2CAP_INFORMATION_RSP signaling mechanism (InfoType = 0x0002) prior to the issuance of the L2CAP_CONFIGURATION_REQ packet. If an L2CAP_CONFIGURATION_REQ packet is sent containing the Extended Flow Specification option then the Quality of Service option and Flush Timeout option shall not be included.

L2CAP_CONFIGURATION_RSP packets shall be sent in reply to L2CAP_CONFIGURATION_REQ packets except when the error condition is covered by an L2CAP_COMMAND_REJECT_RSP response. Although the L2CAP Configuration signaling mechanism allows for the use of wildcards, wildcard values are not supported in the Extended Flow Specification since each parameter represents a property of the traffic in one direction only, and no negotiation is intended to be performed at the L2CAP Configuration level.

The recipient of an L2CAP_CONFIGURATION_REQ packet shall perform all necessary checks with the Controller to validate that the requested QoS can be granted. In the case of a BR/EDR or BR/EDR/LE Controller the L2CAP layer performs the Controller checks. If HCI is used then the L2CAP entity should check that the requested QoS can be achieved over the HCI transport. In order to perform these checks, the recipient needs to have the QoS parameters for both directions. In order for each side to determine when to perform relevant Controller checks, each side will reply with a result “Pending” (0x0004). If no parameters are sent in the L2CAP_CONFIGURATION_RSP packet with result “Pending” then the parameters sent in the L2CAP_CONFIGURATION_REQ packet have been accepted without change. This Lockstep procedure results in both L2CAP entities performing the following:

- Receiving an L2CAP_CONFIGURATION_REQ packet containing the Extended Flow Specification option along with all non-default parameters
- Sending an L2CAP_CONFIGURATION_RSP packet with any modifications to the Extended Flow Specification option and allowed modifications to non-default parameters (result “Pending”)
- Sending an L2CAP_CONFIGURATION_REQ packet containing the Extended Flow Specification option along with all non-default parameters

- Receiving an L2CAP_CONFIGURATION_RSP packet containing any modifications to the Extended Flow Specification option and allowed modifications to non-default parameters (result “Pending”).

The ERTX timer is used when an L2CAP_CONFIGURATION_RSP packet is received with result “Pending” (see Section 6.2.2).

If an L2CAP_CONFIGURATION_RSP packet with result “success” is received before an L2CAP_CONFIGURATION_RSP packet with result “pending” is received the recipient shall disconnect the channel. This is a violation of the Lockstep configuration process.

If a device sends an Extended Flow Specification option in an L2CAP_CONFIGURATION_REQ packet with service type “Best Effort” and receives an L2CAP_CONFIGURATION_REQ packet with service type “Guaranteed,” the channel shall be disconnected. If a device sends an Extended Flow Specification in an L2CAP_CONFIGURATION_REQ packet with type “Guaranteed” and receives an L2CAP_CONFIGURATION_REQ packet with service type “Best Effort,” the channel shall be disconnected.

If the service type is “Best Effort” then values for certain parameters may be sent in an L2CAP_CONFIGURATION_RSP packet with result “Pending” to indicate the maximum bandwidth the sender of the L2CAP_CONFIGURATION_RSP packet is capable to receive. The values sent in an L2CAP_CONFIGURATION_RSP packet with result “Pending” and service type Best Effort shall be in accordance with Table 7.3.

| Parameter | Changes permitted by responder |
|---|---|
| Maximum SDU Size | May decrease only |
| SDU Inter-arrival Time | May increase only |
| Access Latency | None |
| Flush Timeout | None |

*Table 7.3: Permitted parameter in L2CAP_CONFIGURATION_RSP packets for service type “Best Effort”*

After the L2CAP_CONFIGURATION_RSP packet is received with result “Pending,” L2CAP may issue the necessary checks with the Controller.

If the Controller cannot support the Extended Flow Specifications with service type “Guaranteed,” then the recipient of the L2CAP_CONFIGURATION_REQ packet shall send an L2CAP_CONFIGURATION_RSP packet indicating a result code of “Failure - flow spec rejected” (0x0005). If the Controller indicates that it can support the Extended Flow Specifications, then the recipient of the L2CAP_CONFIGURATION_REQ packet shall send an L2CAP_CONFIGURATION_RSP packet with result code of “Success” (0x0000) with no parameters.

If the Result of the L2CAP_CONFIGURATION_RSP packet is Failure (0x0005) for service type “Guaranteed” then an Extended Flow Specification option may be sent in the L2CAP_CONFIGURATION_RSP packet. The Extended Flow Specification parameters sent in the L2CAP_CONFIGURATION_RSP packet may be changed to reflect a QoS level that would be acceptable, but shall only be changed in accordance with Table 7.4.

| Parameter | Changes permitted by responder |
|---|---|
| Maximum SDU Size | None |
| SDU Inter-arrival Time | None |
| Access Latency | May decrease only |
| Flush Timeout | May decrease only (unless set to 0xFFFFFFFF, in which case no change is permitted) |

*Table 7.4: Permitted parameter changes in L2CAP_CONFIGURATION_RSP packets for service type “Guaranteed”*

If an L2CAP_CONFIGURATION_RSP packet is received containing the Extended Flow Specification option with the same values sent earlier, the upper layer shall be notified of the error.

### 7.1.4 Standard Configuration process

For the Standard process the following general procedure shall be used for each direction:

1. Local device informs the remote device of the parameters that the local device will accept using an L2CAP_CONFIGURATION_REQ packet.
2. Remote device responds, agreeing or disagreeing with these values, including the default ones, using an L2CAP_CONFIGURATION_RSP packet.
3. The local and remote devices repeat steps (1) and (2) until agreement on all parameters is reached.

The decision on the amount of time (or messages) spent configuring the channel parameters before terminating the configuration is left to the implementation, but it shall not last more than 120 seconds.

There are two types of configuration parameters: negotiable and non-negotiable.

Negotiable parameters are those that a remote side receiving an L2CAP_CONFIGURATION_REQ packet can disagree with by sending an L2CAP_CONFIGURATION_RSP packet with the *Unacceptable Parameters* (0x0001) result code, proposing new values that can be accepted. Non-negotiable parameters

are only informational and the recipient of an L2CAP_CONFIGURATION_REQ packet cannot disagree with them but can provide adjustments to the values in a positive L2CAP_CONFIGURATION_RSP packet. Section 5 identifies each parameter as negotiable or non-negotiable.

Note: MTU is non-negotiable but can be rejected if a value lower than the mandated minimum is proposed (See Section 5.1).

The following rules shall be used for parameter negotiation in the Request Path:

1. An L2CAP entity shall send at least one L2CAP_CONFIGURATION_REQ packet as part of initial configuration or reconfiguration. If all default or previously agreed values are acceptable, an L2CAP_CONFIGURATION_REQ packet with no options shall be sent.
2. When an L2CAP entity receives a positive L2CAP_CONFIGURATION_RSP packet from the remote device it shall consider all configuration parameters explicitly contained in the L2CAP_CONFIGURATION_REQ packet along with the default and previously agreed values not explicitly contained in the L2CAP_CONFIGURATION_REQ packet as accepted by the remote device.
3. When an L2CAP entity receives a negative L2CAP_CONFIGURATION_RSP packet and sends a new L2CAP_CONFIGURATION_REQ packet it shall include all the options it sent in the previous L2CAP_CONFIGURATION_REQ packet with the same values except the negotiable options that were explicitly rejected in the negative L2CAP_CONFIGURATION_RSP packet which will have new values.

   Note: The resending of all the options provides backwards compatibility with remote implementations that don’t assume rule 4 when responding.
4. Negotiable options not included in a negative L2CAP_CONFIGURATION_RSP packet are considered accepted by the remote device.

   If non-negotiable options are included in a negative L2CAP_CONFIGURATION_RSP packet, then they should be processed as if the response was positive.

The following rules shall be used for parameter negotiation in the Response Path:

1. A positive L2CAP_CONFIGURATION_RSP packet accepts the values of all configuration parameters explicitly contained in the received L2CAP_CONFIGURATION_REQ packet and the default and previously agreed values not explicitly provided.
2. An L2CAP Entity shall send a negative L2CAP_CONFIGURATION_RSP packet to reject negotiable parameter values that are unacceptable, be it values explicitly provided in the received L2CAP_CONFIGURATION_REQ packet or

previously agreed or default values. The rejected parameters sent in a negative L2CAP_CONFIGURATION_RSP packet shall have values that are acceptable to the L2CAP entity sending the negative L2CAP_CONFIGURATION_RSP packet.

3. All negotiable options being rejected shall be rejected in the same negative L2CAP_CONFIGURATION_RSP packet.
4. The only options allowed in a negative L2CAP_CONFIGURATION_RSP packet are the negotiable options being rejected. No wildcards or adjustments to non-negotiable options shall be in a negative L2CAP_CONFIGURATION_RSP packet.
5. Negotiable options not included in the negative L2CAP_CONFIGURATION_RSP packet are considered accepted.

## 7.2 Fragmentation and recombination

Fragmentation is the breaking down of PDUs into smaller pieces for delivery from L2CAP to the lower layer. Recombination is the process of reassembling a PDU from fragments delivered up from the lower layer. Fragmentation and Recombination may be applied to any L2CAP PDUs.

### 7.2.1 Fragmentation of L2CAP PDUs

An L2CAP implementation may fragment any L2CAP PDU for delivery to the lower layers, whether directly to the Controller or over HCI. All fragments associated with a specific L2CAP PDU shall be sent to the Controller, in the same order as their contents occur in the PDU, before any other fragment associated with the same logical link. Fragments associated with different logical links may be interleaved. (See [Vol 2] Part B, Section 5.3 and [Vol 6] Part B, Section 4.5.17 for related requirements on the Controller.)

For example, in the BR/EDR Controller the two LLID bits defined in the first octet of Baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP PDUs. LLID shall be 0b10 for the first segment in an L2CAP PDU and 0b01 for a continuation segment. An illustration of fragmentation for a BR/EDR Controller is shown in Figure 7.1. An example of how fragmentation might be used in a device with HCI is shown in Figure 7.2.

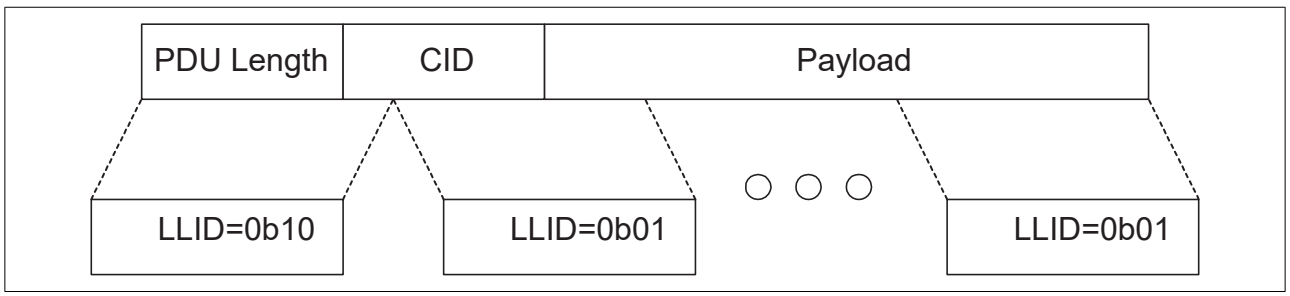



*Figure 7.1: L2CAP fragmentation in a BR/EDR Controller*

Note: The BR/EDR Link Controller is able to impose a different fragmentation on the PDU by using “start” and “continuation” indications as fragments are translated into Baseband packets. Thus, both L2CAP and the BR/EDR Link Controller use the same mechanism to control the size of fragments.

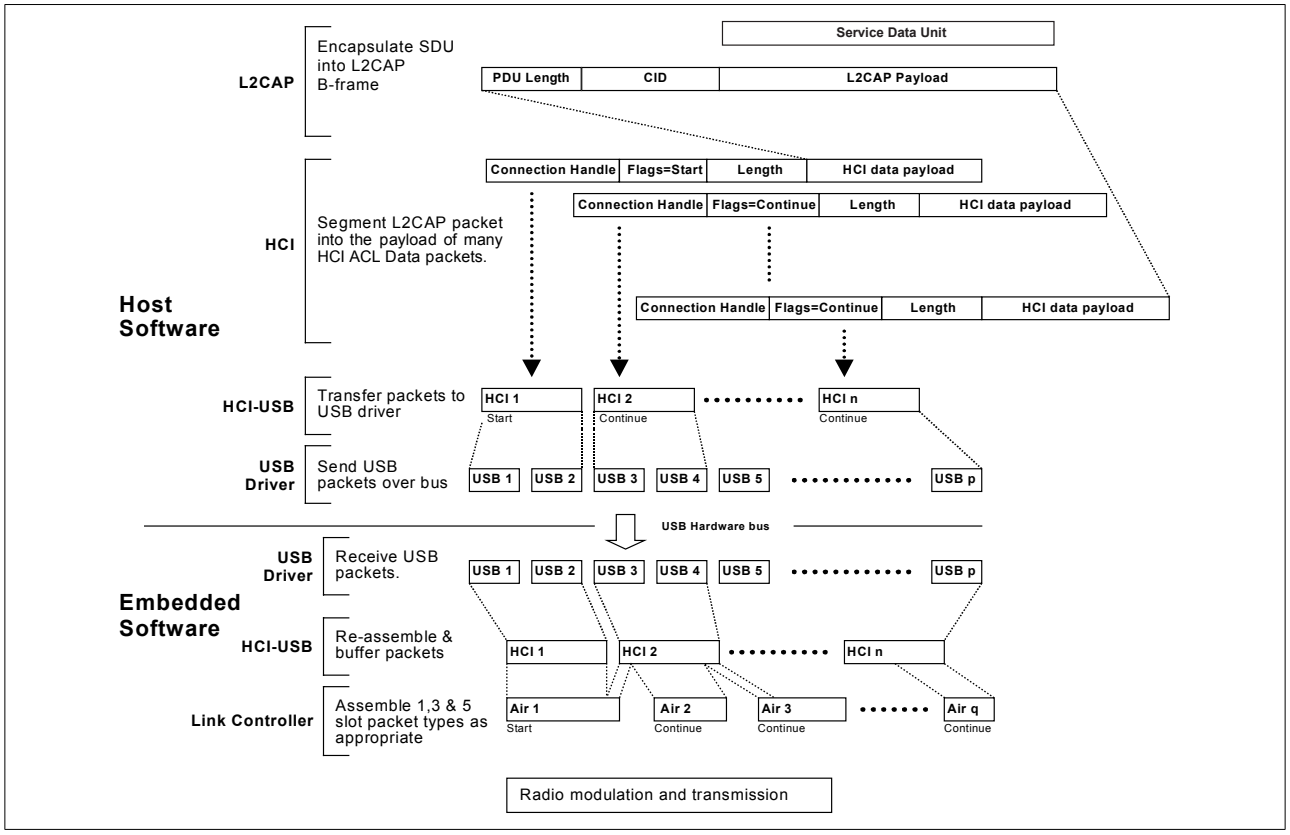



*Figure 7.2: Example of fragmentation processes in a device with a BR/EDR Controller and USB HCI transport*

### 7.2.2 Recombination of L2CAP PDUs

The Controller will provide fragments of L2CAP PDUs to the L2CAP implementation. It is the responsibility of L2CAP to reassemble PDUs and SDUs and to check the length field of the SDUs. The L2CAP layer shall reassemble the fragments it receives from the Controller into complete PDUs, which may then further be combined into SDUs. (See [Vol 2] Part B, Section 5.3 and [Vol 6] Part B, Section 4.5.17 for related requirements on the Controller.)

An L2CAP implementation shall use the PDU Length field in the header of L2CAP PDUs, see Section 3, as a consistency check and shall discard any L2CAP PDUs that fail to match the PDU Length field. If channel reliability is not needed, packets with invalid lengths may be silently discarded. For reliable channels using Basic mode, an L2CAP implementation shall indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as

specified in Section 5.2. For higher data integrity L2CAP should be operated in the Enhanced Retransmission mode.

## 7.3 Encapsulation of SDUs

All SDUs are encapsulated into one or more L2CAP PDUs.

In Basic L2CAP mode, an SDU shall be encapsulated with a minimum of L2CAP protocol elements, resulting in a type of L2CAP PDU called a Basic Information Frame (B-frame).

Segmentation and Reassembly operations are only used in Enhanced Retransmission mode, Streaming mode, Retransmission mode, Flow Control mode, LE Credit Based Flow Control mode, and Enhanced Credit Based Flow Control mode. SDUs may be segmented into a number of smaller packets called SDU segments. Each segment shall be encapsulated with L2CAP protocol elements resulting in an L2CAP PDU called an Information Frame (I-frame) or a Credit-based Frame (K-frame).

The maximum size of an SDU segment shall be given by the Maximum PDU Payload Size (MPS). The MPS parameter may be exported using an implementation specific interface to the upper layer.

The specification does not describe a service interface with the upper layer, nor does it assume any specific buffer management scheme of a Host implementation. Consequently, a reassembly buffer may be part of the upper layer entity. It is assumed that SDU boundaries are preserved between peer upper layer entities.

### 7.3.1 Segmentation of L2CAP SDUs

In Flow Control, Streaming, or Retransmission modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP protocol elements (header and checksum fields) to form I-frames. I-frames are subject to flow control and may be subject to retransmission procedures. The header carries a 2 bit SAR field that is used to identify whether the I-frame is a 'start', 'end' or 'continuation' packet or whether it carries a complete, unsegmented SDU.

In LE Credit Based Flow Control or Enhanced Credit Based Control modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP protocol elements (header fields) to form K-frames. The header of the first segment contains the length of the entire SDU.

Figure 7.3 illustrates segmentation and fragmentation.

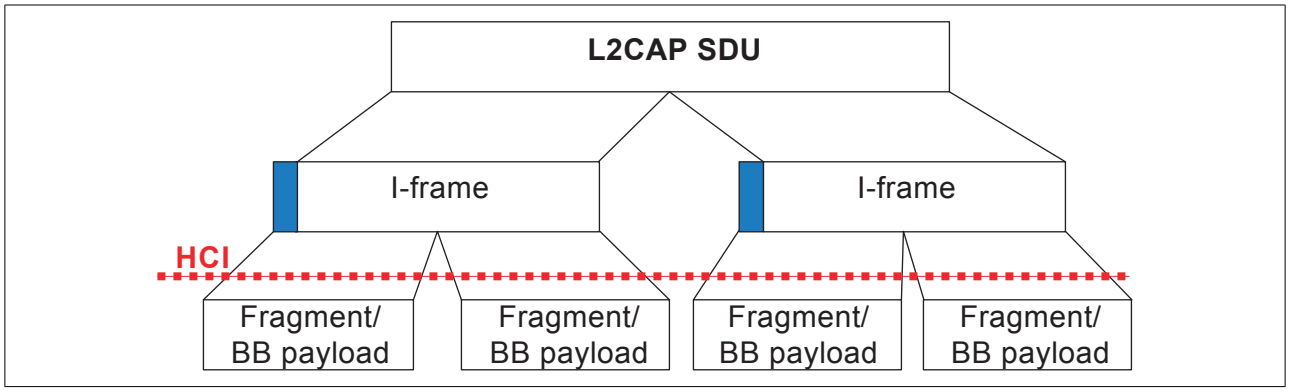



*Figure 7.3: Segmentation and fragmentation of an SDU in a BR/EDR Controller*

### 7.3.2 Reassembly of L2CAP SDUs

The receiving side uses the SAR field of incoming 'I-frames' for the reassembly process. The L2CAP SDU Length field, present in the “start of SDU” I-frame, is an extra integrity check, and together with the sequence numbers may be used to indicate lost L2CAP SDUs to the application.

The receiving side uses the SDU length of the first K-frame and the PDU length of each K-frame for the reassembly process.

Figure 7.3 illustrates segmentation and fragmentation.

### 7.3.3 Segmentation and fragmentation

Figure 7.4 illustrates the use of segmentation and fragmentation operations to transmit a single SDU. While SDUs and L2CAP PDUs are transported in peer-to-peer fashion, the fragment size used by the Fragmentation and Recombination routines are implementation specific and may be different in the sender and the receiver. The over-the-air sequence of packet fragments as created by the sender is common to both devices.

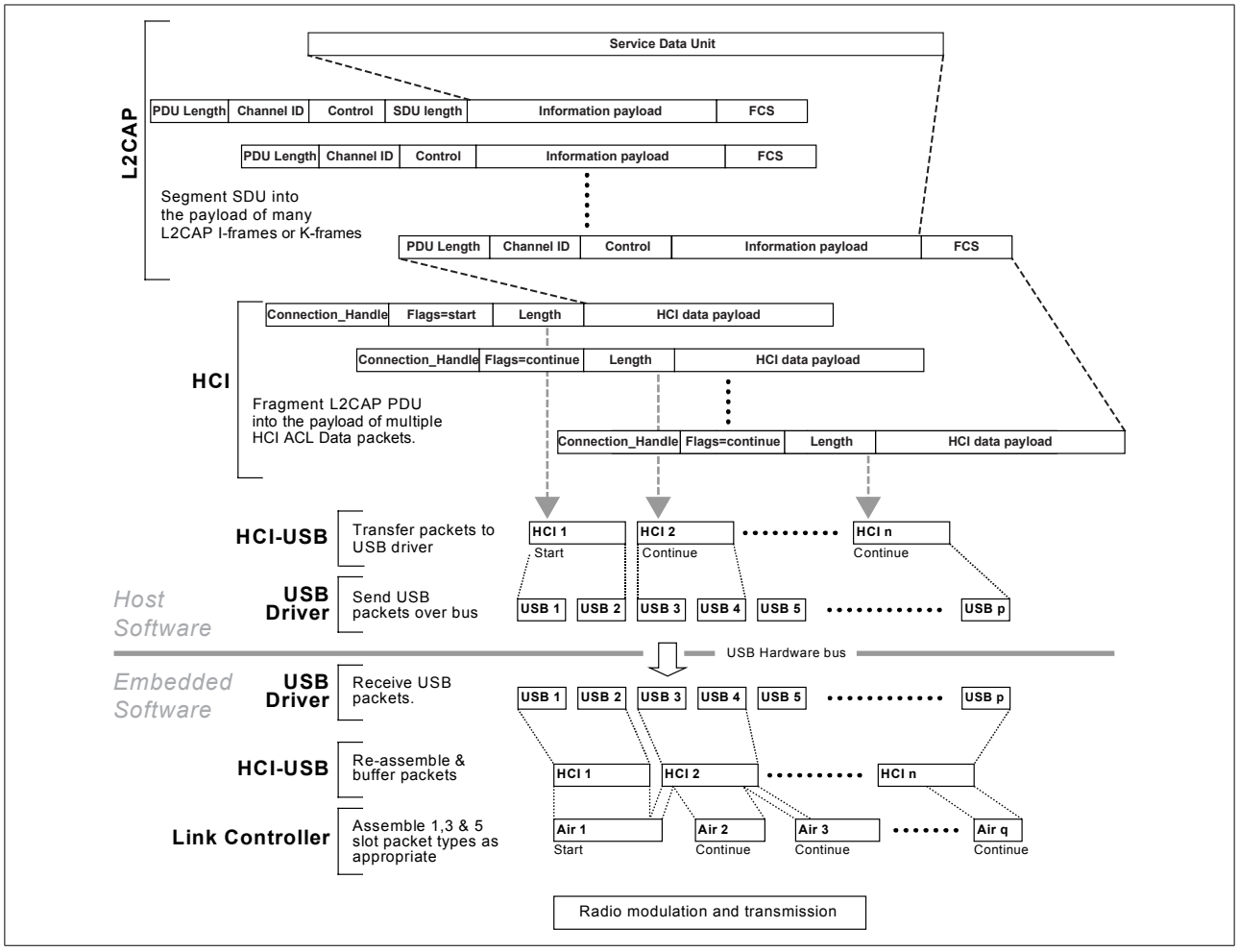



*Figure 7.4: Example of segmentation and fragment processes in a device with a BR/EDR Controller and USB HCI Transport*[1]

## 7.4 Delivery of erroneous L2CAP SDUs

Some applications may require corrupted or incomplete L2CAP SDUs to be delivered to the upper layer. If delivery of erroneous L2CAP SDUs is enabled, the receiving side will pass information to the upper layer on which parts of the L2CAP SDU (i.e., which L2CAP frames) have been lost, failed the error check, or passed the error check. If delivery of erroneous L2CAP SDUs is disabled, the receiver shall discard any L2CAP SDU segment with any missing frames or any frames failing the error checks. L2CAP SDUs whose SDU Length field (if provided) does not equal the sum of the segment payload sizes shall also be discarded.

[1]For simplicity, the stripping of any additional HCI and USB specific information fields prior to the creation of the Baseband packets (Air_1, Air_2, etc.) is not shown in the figure.

## 7.5 Operation with flushing on ACL-U logical links

In the L2CAP configuration using either the Flush Timeout option or the Extended Flow Specification option, the Flush timeout may be set separately per L2CAP channel, but in the BR/EDR Baseband, the flush mechanisms operate per ACL logical transport.

When there is more than one L2CAP channel mapped to the same ACL logical transport, the automatic flush timeout does not discriminate between L2CAP channels. The automatic flush timeout also applies to unicast data sent via the L2CAP connectionless channel. The exception is packets marked as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data packet (see Section 1.1). The automatic flush timeout flushes a specific automatically-flushable L2CAP PDU. The HCI_Flush command flushes all outstanding L2CAP PDUs for the ACL logical transport including L2CAP PDUs marked as non-automatically-flushable. Therefore, care has to be taken when using the Automatic Flush Timeout and the HCI_Flush command. The HCI_Enhanced_Flush command should be used instead.

All packets associated with a reliable connection shall be marked as non-automatically-flushable (if it is mapped to an ACL logical transport with a finite automatic flush timeout) or L2CAP Enhanced Retransmission mode or Retransmission mode shall be used. In Enhanced Retransmission mode or Retransmission mode, loss of flushed L2CAP PDUs on the channel is detected by the L2CAP ARQ mechanism and they are retransmitted. L2CAP Enhanced Retransmission mode or Retransmission mode can be used for other purposes such as the need for a residual error rate that is much smaller than the Baseband can deliver. In this situation L2CAP Enhanced Retransmission mode or Retransmission mode and the Non-Flushable Packet Boundary Flag feature can be used at the same time.

If it is desired to send unicast data via the L2CAP connectionless channel which is not subject to automatic flushing, then the data should be marked as non-automatically flushable if it is mapped to an ACL logical transport with a finite automatic flush timeout. Unicast data sent via the L2CAP connectionless channel may be marked flushable.

There is only one automatic flush timeout setting per ACL logical transport. Therefore, all time bounded L2CAP channels on an ACL logical transport with an automatic flush timeout setting should configure the same flush timeout value at the L2CAP level. The flush timeout setting for the ACL logical transport also applies to unicast data sent via the L2CAP connectionless channel which are marked flushable.

If Automatic Flush Timeout is used, then it should be taken into account that it only flushes one L2CAP PDU. If one PDU has timed out and needs flushing, then other automatically-flushable packets on the same logical transport are also likely to need flushing. Therefore, flushing can be handled by the HCI_Enhanced_Flush command so that all outstanding automatically-flushable L2CAP PDUs are flushed.

When both reliable and isochronous data is to be sent over the same ACL logical transport, an infinite Automatic Flush Timeout can be used. In this case the isochronous data can be flushed using the HCI_Enhanced_Flush command with Packet_Type set to “Automatically-Flushable Only,” thus preserving the reliable data.

## 7.6 Connectionless data channel

In addition to connection-oriented channels, L2CAP also provides a connectionless channel. Data sent on the connectionless channel shall only be sent over the BR/EDR radio. The connectionless channel allows broadcast transmissions from the Central to all members of the piconet or unicast transmissions from either a Central or Peripheral to a single remote device. Data sent through the connectionless channel is sent in a best-effort manner. The connectionless channel provided by L2CAP has no quality of service.

While L2CAP itself does not provide retransmission for data sent via the connectionless channel, the Baseband does provide an ARQ scheme for unicast data (see [Vol 2] Part B, Section 7.6). If a higher degree of reliability is desired for unicast data sent via the connectionless L2CAP channel than is provided by the Baseband ARQ scheme then an ARQ scheme should be implemented at a higher layer.

No acknowledgment is provided by the Baseband for broadcast transmissions and hence broadcast transmissions sent via the connectionless L2CAP channel are unreliable and hence might or might not reach each member of the piconet.

The receiving L2CAP entity may silently discard data received via the connectionless L2CAP channel if the data packet is addressed to a PSM and no application is registered to receive data on that PSM.

L2CAP does not provide flow control for either connection-oriented L2CAP channels operating in Basic mode or for traffic on the connectionless L2CAP channel. Hence if data received by L2CAP is not accepted by the target applications in a timely manner, congestion can occur in a receiving L2CAP implementation. For connection-oriented channels, the receiving L2CAP entity may elect to close a channel if the target application for that channel does not accept the data in a timely manner. Since this option is not available for unicast data received via the connectionless L2CAP channel, the receiving L2CAP entity may instead elect to de-register the application receiving the data or to disconnect the underlying physical link. An application shall be notified if it is de-registered. If the underlying physical link is disconnected then all applications using that physical link shall be notified.

Unicast data shall only be sent via the connectionless L2CAP channel to a remote device if the remote device indicates support for Unicast Connectionless Data Reception in the L2CAP Extended Features Mask.[1]

The L2CAP Extended Features Mask should be retrieved using the L2CAP_INFORMATION_REQ packet to determine if Unicast Connectionless Data Reception is supported. Support for Unicast Connectionless Data Reception can be inferred from information obtained via a SDP or EIR. For example, if a service is found via SDP or EIR that is known to mandate the support of UCD, then it may be assumed that the device indicating support for the service supports Unicast Connectionless Data Reception.

Unicast data sent via the L2CAP connectionless channel are subject to automatic flushing and hence the packet boundary flags should be set appropriately if the Controller supports the Packet Boundary Flag feature. See Section 7.5 for further details. Since the receiving L2CAP entity has no mechanism to enable it to know whether received packets were originally marked as flushable or non-flushable on the transmitting device, the receiving L2CAP entity should treat all unicast packets received via the connectionless L2CAP packet as non-flushable.

If encryption is required for a given profile, then the profile or application shall ensure that authentication is performed and encryption is enabled prior to sending any unicast data on the connectionless L2CAP channel by using Security mode 4 as defined in GAP ([Vol 3] Part C, Section 5.2.2). There is no mechanism provided in the specification to prevent reception of unencrypted data on the connectionless L2CAP channel. An application which requires received data to be encrypted must ignore any unencrypted data it receives over the connectionless channel.

Broadcast transmissions to the connectionless channel are sent with the broadcast LT_ADDR and hence may be received by any of the Peripherals in the piconet. If it is desirable to restrict the reception of the transmitted data to only a subset of the Peripherals in the piconet, then higher level encryption may be used to support private communication.

The Central will not receive transmissions broadcast on the connectionless channel. Therefore, higher layer protocols must loopback any broadcast data traffic being sent to the Central if required.

The connectionless data channel shall not be used with Enhanced Retransmission mode, Retransmission mode or Flow Control mode.

[1]The Unicast Connectionless Data Reception bit in the L2CAP Extended Features Mask does not in any way indicate support or lack of support for reception of broadcast data on the connectionless L2CAP channel even though both broadcast data and unicast data are sent and received using the same CID (0x0002). For historical reasons, there is no bit to indicate support for sending or receiving of broadcast data on the connectionless L2CAP channel.

## 7.7 Operation collision resolution

When two devices request the same operation by sending a request packet with the same code, a collision can occur. Some operations require collision resolution. The description of each operation in Section 4 will indicate if collision resolution is required. Both devices must know which request will be rejected. The following algorithm shall be used by both devices to determine which request to reject.

1. Set i=0 (representing the least significant octet of the BD_ADDR).
2. Compare the octet[i] of the BD_ADDR of both devices. If the octets are not equal go to step 4.
3. Increment i by 1. Go to step 2.
4. The device with the larger BD_ADDR octet shall reject the request from the other device.

## 7.8 [This section is no longer used]

## 7.9 Prioritizing data over HCI

In order for Guaranteed channels to meet their guarantees, L2CAP should prioritize traffic over the HCI transport in devices that support HCI. Packets for Guaranteed channels should receive higher priority than packets for Best Effort channels.

## 7.10 Supporting Extended Flow Specification for BR/EDR and BR/EDR/LE Controllers

If both the local L2CAP entity and the remote L2CAP entity indicate support for Extended Flow Specification for BR/EDR in the Extended Feature Mask then all channels created between the two devices shall send an Extended Flow Specification option and shall use the Lockstep configuration procedure. In addition all channels shall use Enhanced Retransmission mode or Streaming mode. If one or both L2CAP entities do not indicate support for Extended Flow Specification for BR/EDR in the Extended Feature Mask then the Lockstep configuration procedure shall not be used and the Extended Flow Specification option shall not be sent for channels created over the ACL-U logical link between the two devices.

The L2CAP entity shall perform admission control for Guaranteed channels during the Lockstep configuration procedure (see Section 7.1.3). Admission control is performed to determine if the requested Guaranteed QoS can be achieved by the BR/EDR or BR/EDR/LE Controller over an ACL-U logical link without compromising existing Guaranteed channels running on the Controller. If HCI is used then the L2CAP entity shall also verify that the QoS can be achieved over the HCI transport.

In performing admission control the L2CAP layer shall reject a Guaranteed QoS request that causes at least one of the following rules to be violated.

- The total guaranteed data rate in both directions shall not exceed two times the highest Symmetric Max of an ACL-U logical link over the BR/EDR or BR/EDR/LE Controller (see [Vol 2] Part B, Section 6.7). For example, for a Basic Rate Controller the highest Symmetric Max Rate is 433.9 kb/s (DH5) from [Vol 2] Part B, Table 6.8. Two times that value is 867.8 kb/s.
- The total guaranteed data rate in any one direction shall not exceed the highest Asymmetric Max Rate of an ACL-U logical link (see [Vol 2] Part B, Section 6.7). For example, the highest Asymmetric Max Rate for Basic Rate is 723.2 kb/s (DH5 packet) from [Vol 2] Part B, Table 6.8.

The data rate of a Guaranteed channel is calculated by dividing the Maximum SDU size in an Extended Flow Specification by the SDU Inter-arrival time. Total guaranteed data rate in both directions is calculated by taking the sum of the data rates in both directions for all existing Guaranteed channels plus the data rate in both directions of the requested Guaranteed channel (i.e. data rates from the both outgoing and incoming Extended Flow Specifications). Total guaranteed data rate for one direction is calculated by taking the sum of the data rates in one direction for all existing Guaranteed channels plus the data rate in the same direction of the requested Guaranteed channel.

An L2CAP entity should use additional information for admission control that may result in a Guaranteed QoS request being rejected even if none of the rules are violated. This allows the L2CAP entity to account for things such as CQDDR, SCO/eSCO channels, LMP traffic, page scanning, etc.

In order to meet the access latency and/or data rate required by a Guaranteed channel, the L2CAP entity should:

- Prioritize traffic over the HCI transport in devices that support HCI.
- Give conformant packets for Guaranteed channels higher priority than packets for Best Effort channels. Packets for Best Effort channels should have higher priority than non-conformant packets for Guaranteed channels. (See Section 5.6 for a discussion of non-conformant and conformant traffic).
- Use the SAR feature to restrict the PDU sizes of Best Effort SDUs so that they do not prevent the Controller from sending the SDUs of the Guaranteed channel within the time periods required by its Extended Flow Specification.

When a channel is reconfigured the Lockstep configuration process is only used when an Extended Flow Specification option is present in the L2CAP_CONFIGURATION_REQ packet as described in Section 7.1.3.

## 7.11 Enhanced Credit-Based Flow Control Reconfiguration

In Enhanced Credit Based Flow Control mode, each individual channel has its own MTU and MPS; both of these values can be reconfigured independently.

To perform reconfiguration, a device shall send an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet to the peer device with the new proposed MTU and MPS values and a set of channels to be reconfigured. The request shall not decrease the MTU of any channel and shall not decrease the MPS of a channel if more than one channel is specified.

When an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet is received and all the parameters are valid, the recipient shall complete sending all existing PDUs that require an MPS larger than the new MPS value for the channel in the L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet and then send the L2CAP_CREDIT_BASED_RECONFIGURE_RSP packet, with a zero Result field, to the peer device.

After the L2CAP_CREDIT_BASED_RECONFIGURE_RSP packet is sent, the MTU and MPS values of the channels included in the L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet shall be set to the MTU and MPS fields from that packet, and new SDUs shall be sent using the new MTU and MPS values.

When an L2CAP_CREDIT_BASED_RECONFIGURE_REQ packet is received with invalid parameters, the recipient shall send an L2CAP_CREDIT_BASED_RECONFIGURE_RSP PDU with a non-zero Result field and not change any MTU and MPS values.

## 7.12 Operation with flushing on LE-U logical links

An upper layer may instruct the Host to mark L2CAP B-frames on the Attribute Protocol LE fixed channel as flushable data. The value of the ACL Flush Timeout defaults to infinite and may be negotiated by the upper layer to a non-infinite value. Flushable LE-U L2CAP packets are removed from the LE Link Layer transmission queue after their flush timers expire (see [Vol 6] Part B, Section 4.5.17.1).

# 8 PROCEDURES FOR FLOW CONTROL AND RETRANSMISSION

When Enhanced Retransmission mode, Streaming mode, Flow Control mode, or Retransmission mode is used, the procedures defined in this section shall be used, including the numbering of information frames, the handling of SDU segmentation and reassembly, and the detection and notification of frames with errors. Retransmission mode and Enhanced Retransmission mode also allow the sender to resend frames with errors on request from the receiver.

## 8.1 Information retrieval

Before attempting to configure Enhanced Retransmission mode, Streaming mode, Flow Control mode, or Retransmission mode on a channel, support for the suggested mode shall be verified by performing an information retrieval for the “Extended features supported” information type (0x0002). If the information retrieval is not successful or the “Extended features mask” bit is not set for the required mode, the mode shall not be suggested in a configuration request.

## 8.2 Function of PDU Types for Flow Control and Retransmission

Two frame formats are defined for Enhanced Retransmission mode, Streaming mode, Flow Control mode, and Retransmission mode (see Section 3.3). The I-frame is used to transport user information instead of the B-frame. The S-frame is used for supervision.

### 8.2.1 Information frame (I-frame)

I-frames are sequentially numbered frames containing information fields. I-frames also include some of the functionality of RR frames (see below).

### 8.2.2 Supervisory frame (S-frame)

The S-frame is used to control the transmission of I-frames. For Retransmission mode and Flow Control mode, the S-frame has two formats: Receiver Ready (RR) and Reject (REJ). A description of how S-frames are used in Enhanced Retransmission mode is given in Section 8.6.1. S-frames are not used in Streaming mode. The following description of S-frames only applies to Retransmission mode and Flow Control mode.

#### 8.2.2.1 Receiver Ready (RR)

The receiver ready (RR) S-frame is used to:

1. Acknowledge I-frames numbered up to and including ReqSeq - 1.
2. Enable or disable retransmission of I-frames by updating the receiver with the current status of the Retransmission Disable Bit.

The RR frame has no information field.

#### 8.2.2.2 Reject (REJ)

The reject (REJ) S-frame is used to request retransmission of all I-frames starting with the I-frame with TxSeq equal to ReqSeq specified in the REJ. The value of ReqSeq in the REJ frame acknowledges I-frames numbered up to and including ReqSeq - 1. I-frames that have not been transmitted, shall be transmitted following the retransmitted I-frames.

When a REJ is transmitted, it triggers a REJ Exception condition. A second REJ frame shall not be transmitted until the REJ Exception condition is cleared. The receipt of an I-frame with a TxSeq equal to the ReqSeq of the REJ frame clears the REJ Exception. The REJ Exception condition only applies to traffic in one direction.

Note: This means that only valid I-frames can be rejected.

## 8.3 Variables and sequence numbers

The sending peer uses the following variables and sequence numbers:

- TxSeq – the send sequence number used to sequentially number each new I-frame transmitted.
- NextTxSeq – the sequence number to be used in the next new I-frame transmitted.
- ExpectedAckSeq – the sequence number of the next I-frame expected to be acknowledged by the receiving peer.

The receiving peer uses the following variables and sequence numbers:

- ReqSeq – The sequence number sent in an acknowledgment frame to request transmission of I-frame with TxSeq = ReqSeq and acknowledge receipt of I-frames up to and including (ReqSeq-1).
- ExpectedTxSeq – the value of TxSeq expected in the next I-frame.
- BufferSeq – When segmented I-frames are buffered this is used to delay acknowledgment of received I-frame so that new I-frame transmissions do not cause buffer overflow.

All variables have the range 0 to 63. Arithmetic operations on state variables (NextTxSeq, ExpectedTxSeq, ExpectedAckSeq, BufferSeq) and sequence numbers (TxSeq, ReqSeq) contained in this document shall be taken *mod* 64.

### 8.3.1 Sending peer

#### 8.3.1.1 Send sequence number TxSeq

I-frames contain TxSeq, the send sequence number of the I-frame. When an I-frame is first transmitted, TxSeq is set to the value of the send state variable NextTxSeq. TxSeq is not changed if the I-frame is retransmitted.

#### 8.3.1.2 Send state variable NextTxSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. A send state variable NextTxSeq shall be maintained for each remote endpoint. NextTxSeq is the sequence number of the next in-sequence I-frame to be transmitted to that remote endpoint. When the link is created NextTxSeq shall be initialized to 0.

The value of NextTxSeq shall be incremented by 1 after each in-sequence I-frame transmission, and shall not exceed ExpectedAckSeq by more than the maximum number of outstanding I-frames (TxWindow). The value of TxWindow shall be in the range 1 to 32 for Retransmission mode and Flow Control mode. The value of TxWindow shall be in the range 1 to 63 for Enhanced Retransmission mode.

#### 8.3.1.3 Acknowledge state variable ExpectedAckSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. An acknowledge state variable ExpectedAckSeq shall be maintained for each remote endpoint. ExpectedAckSeq is the sequence number of the next in-sequence I-frame that the remote receiving peer is expected to acknowledge. (ExpectedAckSeq – 1 equals the TxSeq of the last acknowledged I-frame). When the link is created ExpectedAckSeq shall be initialized to 0.

Note: If the next acknowledgment acknowledges a single I-frame then its ReqSeq will be expectedAckSeq + 1.

If a valid ReqSeq is received from the peer then ExpectedAckSeq is set to ReqSeq. A valid ReqSeq value is one that is in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

Note: The comparison with NextTxSeq must be ≤ in order to handle the situations where there are no outstanding I-frames.

These inequalities shall be interpreted in the following way: ReqSeq is valid, if and only if (ReqSeq-ExpectedAckSeq) *mod* 64 ≤ (NextTxSeq-ExpectedAckSeq) *mod* 64.

Furthermore, from the description of NextTxSeq, it can be seen that (NextTxSeq-ExpectedAckSeq) *mod* 64 ≤ TxWindow.

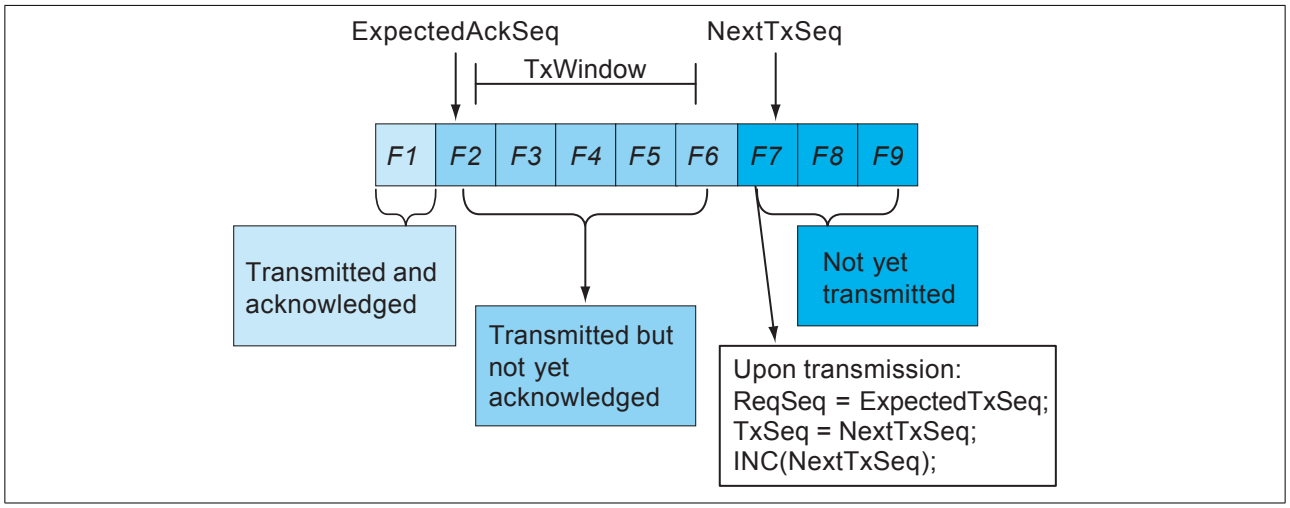



*Figure 8.1: Example of the transmitter side*

Figure 8.1 shows TxWindow=5, and three frames awaiting transmission. The frame with number *F7* may be transmitted when the frame with *F2* is acknowledged. When the frame with *F7* is transmitted, TxSeq is set to the value of NextTxSeq. After TxSeq has been set, NextTxSeq is incremented by one.

The sending peer expects to receive valid ReqSeq values, which are in the range ExpectedAckSeq to NextTxSeq. Upon receipt of a ReqSeq value equal to the current NextTxSeq all outstanding I-frames have been acknowledged by the receiver.

### 8.3.2 Receiving peer

#### 8.3.2.1 Receive sequence number ReqSeq

All I-frames and S-frames contain ReqSeq, the send sequence number (TxSeq) that the receiving peer requests in the next I-frame.

When an I-frame or an S-frame is transmitted, the value of ReqSeq shall be set to the current value of the receive state variable ExpectedTxSeq or the buffer state variable BufferSeq. The value of ReqSeq shall indicate that the data Link Layer entity transmitting the ReqSeq has correctly received all I-frames numbered up to and including ReqSeq – 1. If BufferSeq ≠ ExpectedTxSeq, then the receiver should also set the retransmission disable bit to 1 to prevent unnecessary retransmissions.

Note: The option to set ReqSeq to BufferSeq instead of ExpectedTxSeq allows the receiver to impose flow control for buffer management or other purposes.

#### 8.3.2.2 Receive state variable ExpectedTxSeq

Each channel shall have a receive state variable (ExpectedTxSeq). The receive state variable is the sequence number (TxSeq) of the next in-sequence I-frame expected.

The value of the receive state variable shall be the last in-sequence, valid I-frame received.

#### 8.3.2.3 Buffer state variable BufferSeq

Each channel may have an associated BufferSeq. BufferSeq is used to delay acknowledgment of frames until they have been pulled by the upper layers, thus preventing buffer overflow. BufferSeq and ExpectedTxSeq are equal when there is no extra segmentation performed and frames are pushed to the upper layer immediately on reception. When buffer space is scarce, for example when frames reside in the buffer for a period, the receiver may choose to set ReqSeq to BufferSeq instead of ExpectedTxSeq, incrementing BufferSeq as buffer space is released. The windowing mechanism will ensure that transmission is halted when ExpectedTxSeq - BufferSeq is equal to TxWindow.

Updates of BufferSeq may be based on changes in available buffer space instead of delivery of I-frame contents.

I-frames shall have sequence numbers in the range ExpectedTxSeq ≤ TxSeq < (BufferSeq + TxWindow).

On receipt of an I-frame with TxSeq equal to ExpectedTxSeq, ExpectedTxSeq shall be incremented by one regardless of how many I-frames with TxSeq greater than ExpectedTxSeq were previously received.

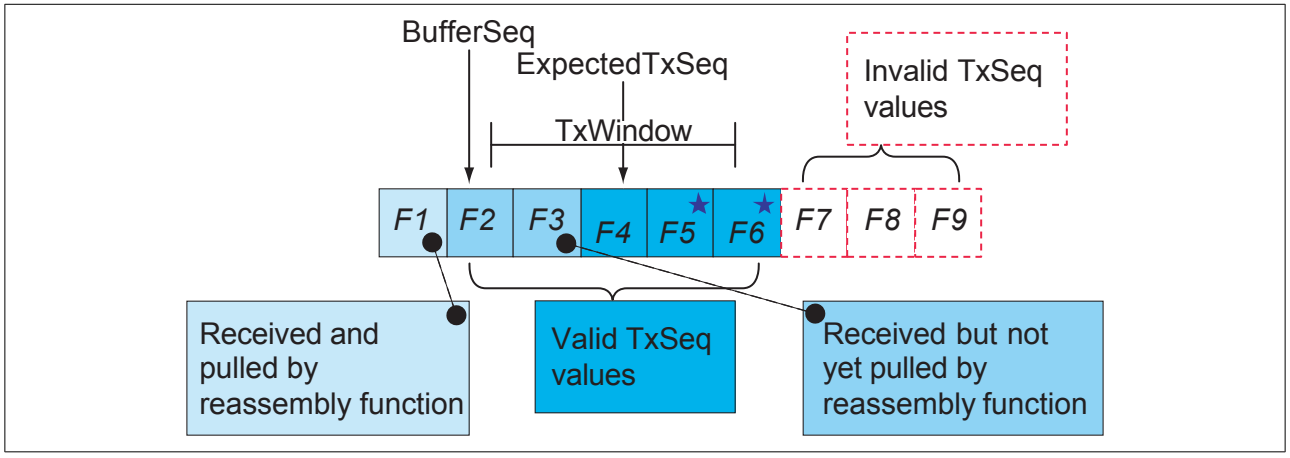



*Figure 8.2: Example of the receiver side*

Figure 8.2 shows TxWindow=5. *F1* is successfully received and pulled by the upper layer. BufferSeq shows that *F2* is the next I-frame to be pulled, and ExpectedTxSeq

points to the next I-frame expected from the peer. An I-frame with TxSeq equal to 5 has been received thus triggering an SREJ or REJ exception. The star indicates I-frames received but discarded owing to the SREJ or REJ exception. They will be resent as part of the error recovery procedure.

In Figure 8.2 there are several I-frames in a buffer awaiting the SDU reassembly function to pull them and the TxWindow is full. The receiver would usually disable retransmission in Retransmission mode or Flow Control mode by setting the Retransmission Disable Bit to 1 and send an RR back to the sending side. This tells the transmitting peer that there is no point in performing retransmissions. Both sides will send S-frames to make sure the peer entity knows the current value of the Retransmission Disable Bit.

## 8.4 Retransmission mode

### 8.4.1 Transmitting frames

A new I-frame shall only be transmitted when the TxWindow is not full. No I-frames shall be transmitted if the last RetransmissionDisableBit (R) received is set to one.

A previously transmitted I-frame may be retransmitted as a result of an error recovery procedure, even if the TxWindow is full. When an I-frame is retransmitted it shall always be sent with the same TxSeq value used in its initial transmission.

The state of the RetransmissionDisableBit (R) is stored and used along with the state of the RetransmissionTimer to decide the actions when transmitting I-frames. The RetransmissionTimer is running whenever I-frames have been sent but not acknowledged.

#### 8.4.1.1 Last received R was set to zero

If the last R received was set to zero, then I-frames may be transmitted. If there are any I-frames which have been sent and not acknowledged then they shall be retransmitted when the RetransmissionTimer elapses. If the retransmission timer has not elapsed then a retransmission shall not be sent and only new I-frames may be sent.

- If unacknowledged I-frames have been sent and the RetransmissionTimer has elapsed, then an unacknowledged I-frame shall be retransmitted. The RetransmissionTimer shall be restarted.
- If unacknowledged I-frames have been sent but the Retransmission timer has not elapsed, then a new I-frame shall be sent if one is waiting and no timer action shall be taken.
- If no unacknowledged I-frames have been sent and a new I-frame is waiting, then the new I-frame shall be sent, the RetransmissionTimer shall be started, and the Monitor Timer shall be stopped if it is running.

- If no unacknowledged I-frames have been sent, no new I-frames are waiting to be transmitted, and the RetransmissionTimer is running, then the retransmission timer shall be stopped and the monitor timer shall be started.

Table 8.1 summarizes actions when the RetransmissionTimer is in use and R=0.

| **Unacknowledged I-frames sent = Retransmission Timer is running** | **Retransmission Timer has elapsed** | **New I-frames are waiting** | **Transmit Action** | **Timer Action** |
|---|---|---|---|---|
| True | True | True or False | Retransmit unacknowledged I-frame | Restart Retransmission Timer |
| True | False | True | Transmit new I-frame | No timer action |
| True | False | False | No transmit action | No Timer action |
| False | False | True | Transmit new I-frame | Restart Retransmission Timer |
| False | False | False | No Transmit action | If MonitorTimer is not running then restart MonitorTimer |

*Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0*

If the RetransmissionTimer is not in use, no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted

- If the MonitorTimer is running and has not elapsed, then no transmit action shall be taken and no timer action shall be taken.
- If the MonitorTimer has elapsed, then an S-frame shall be sent and the MonitorTimer shall be restarted.

If any I-frames become available for transmission, then the MonitorTimer shall be stopped, the RetransmissionTimer shall be started, and the rules for when the RetransmissionTimer is in use shall be applied.

When an I-frame is sent ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTxSeq and NextTxSeq shall be incremented by one.

#### 8.4.1.2 Last received R was set to one

If the last R received was set to one, then I-frames shall not be transmitted. The only frames which may be sent are S-frames. An S-frame shall be sent according to the rules below:

- If the MonitorTimer is running and has not elapsed, then no transmit action shall be taken and no timer action shall be taken.
- If the MonitorTimer has elapsed, then an S-frame shall be sent and the MonitorTimer shall be restarted.

### 8.4.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be accepted for the SDU reassembly function. ExpectedTxSeq is used by the reassembly function.

The first valid I-frame received after an REJ was sent, with a TxSeq of the received I-frame equal to ReqSeq of the REJ, shall clear the REJ Exception condition.

The ReqSeq shall be processed according to Section 8.4.6.

If a valid I-frame with TxSeq ≠ ExpectedTxSeq is received then an exception condition shall be triggered which is handled according to Section 8.4.7.

### 8.4.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released or how many new incoming I-frames could be stored. If BufferSeq is incremented, an acknowledgment shall be sent to the peer entity.

The acknowledgment may either be an RR or an I-frame. The acknowledgment shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there are no I-frames buffered for pulling ExpectedTxSeq is equal to BufferSeq.

If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

### 8.4.4 Sending and receiving acknowledgments

Either the MonitorTimer or the RetransmissionTimer shall be active while in Retransmission mode. Both timers shall not be active concurrently.

#### 8.4.4.1 Sending acknowledgments

Whenever an L2CAP entity transmits an I-frame or an S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

#### 8.4.4.2 Receiving acknowledgments

On receipt of a valid S-frame or I-frame, the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.

The following rules shall be applied:

1. If the RetransmissionDisableBit changed value from 0 to 1 (stop retransmissions) then the receiving entity shall
    a. If the RetransmissionTimer is running then stop it and start the MonitorTimer.
    b. Store the state of the RetransmissionDisableBit received.
2. If the RetransmissionDisableBit changed value from 1 to 0 (start retransmissions) then the receiving entity shall
    a. Store the state of the RetransmissionDisableBit received.
    b. If there are any I-frames that have been sent but not acknowledged, then stop the MonitorTimer and start the RetransmissionTimer.
    c. Any buffered I-frames shall be transmitted according to Section 8.4.1.
3. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame, and the RetransmissionDisableBit equals 1 (retransmissions stopped), then the receiving entity shall
    a. Follow the rules in Section 8.4.1.
4. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the RetransmissionDisableBit equals 0 (retransmissions started) then the receiving entity shall
    a. If the RetransmissionTimer is running, then stop it.
    b. If any unacknowledged I-frames have been sent then the RetransmissionTimer shall be restarted.
    c. Follow the rules in Section 8.4.1.
    d. If the RetransmissionTimer is not running and the MonitorTimer is not running, then start the MonitorTimer.

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ExpectedAckSeq shall be set to ReqSeq

to indicate that the I-frames with TxSeq up to and including (ReqSeq - 1) have been acknowledged.

### 8.4.5 Receiving REJ frames

Upon receipt of a valid REJ frame, where ReqSeq identifies an I-frame not yet acknowledged, the ReqSeq acknowledges I-frames with TxSeq up to and including ReqSeq - 1. Therefore the REJ acknowledges all I-frames before the I-frame it is rejecting.

ExpectedAckSeq shall be set equal to ReqSeq to mark I-frames up to and including ReqSeq - 1 as received.

NextTxSeq shall be set to ReqSeq to cause transmissions of I-frames to resume from the point where TxSeq equals ReqSeq.

If ReqSeq equals ExpectedAckSeq then the REJ frame shall be ignored.

### 8.4.6 Waiting acknowledgments

A counter, TransmitCounter, counts the number of times an L2CAP PDU has been transmitted. This shall be set to 1 after the first transmission. If the RetransmissionTimer expires the following actions shall be taken:

1. If the TransmitCounter is less than MaxTransmit then:
   a. Increment the TransmitCounter by one.
   b. Retransmit the last unacknowledged I-frame, according to Section 8.4.1.
2. If the TransmitCounter is equal to MaxTransmit this channel to the peer entity shall be assumed lost. The channel shall move to the CLOSED state and appropriate action shall be taken to report this to the upper layers.

### 8.4.7 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in Retransmission mode are defined in this section.

#### 8.4.7.1 TxSeq sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1 (BufferSeq ≤TxSeq<ExpectedTxSeq). The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4. The Information field shall be discarded since it has already been received.

- *Out-of-sequence I-frame*

  The out-of-sequence I-frame is identified by a TxSeq within the valid range. The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4.

  A REJ exception is triggered, and an REJ frame with ReqSeq equal to ExpectedTxSeq shall be sent to initiate recovery. The received I-frame shall be discarded.

- *Invalid TxSeq*

  An invalid TxSeq value is a value that does not meet either of the above conditions. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

#### 8.4.7.2 ReqSeq sequence error

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

#### 8.4.7.3 Timer recovery error

If an L2CAP entity fails to receive an acknowledgment for the last I-frame sent, then it will not detect an out-of-sequence exception condition and therefore will not transmit an REJ frame.

The L2CAP entity that transmitted an unacknowledged I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.4.6.

#### 8.4.7.4 Invalid frame

Any frame received which is invalid (as defined in Section 3.3.6) shall be discarded, and no action shall be taken as a result of that frame.

## 8.5 Flow Control mode

When a link is configured to work in flow control mode, the flow control operation is similar to the procedures in retransmission mode, but all operations dealing with CRC errors in received packets are not used. Therefore

- REJ frames shall not be used in Flow Control mode.
- The RetransmissionDisableBit shall always be set to zero in the transmitter, and shall be ignored in the receiver.

The behavior of flow control mode is specified in this section.

Assuming that the TxWindow size is equal to the buffer space available in the receiver (counted in number of I-frames), in flow control mode the number of unacknowledged frames in the transmitter window is always less than or equal to the number of frames for which space is available in the receiver.

Note: A missing frame still occupies a place in the window.

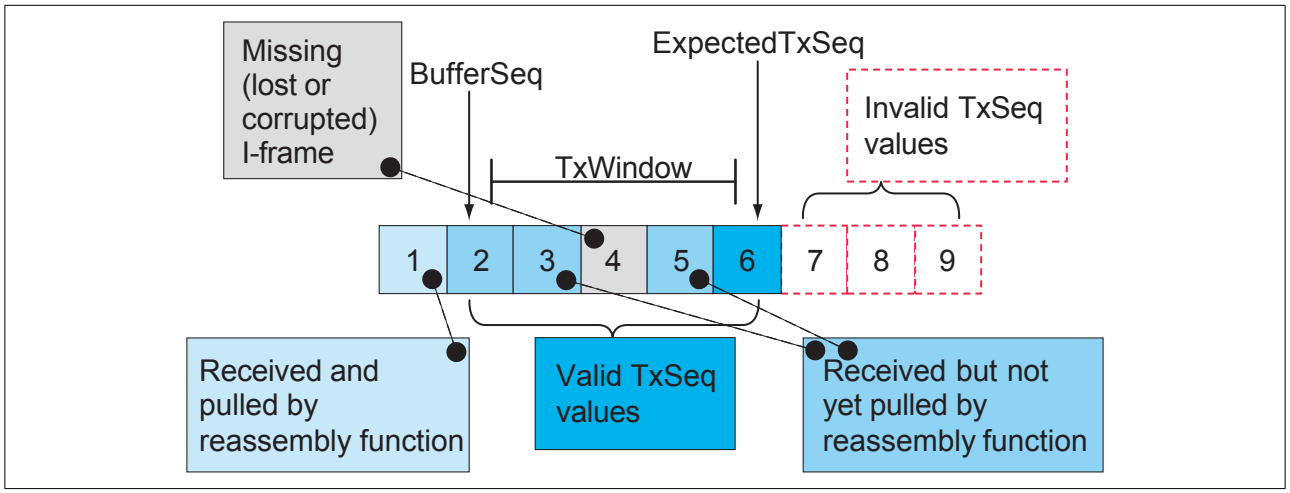



*Figure 8.3: Overview of the receiver side when operating in Flow Control mode*

### 8.5.1 Transmitting I-frames

A new I-frame shall only be transmitted when the TxWindow is not full.

Upon transmission of the I-frame the following actions shall be performed:

- If no unacknowledged I-frames have been sent then the MonitorTimer shall be stopped and the RetransmissionTimer shall be started.
- If any I-frames have been sent and not acknowledged then the RetransmissionTimer remains active and no timer operation is performed.

The control field parameter ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTxSeq, and NextTxSeq shall be incremented by one.

### 8.5.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be made available to the reassembly function. ExpectedTxSeq shall be incremented by one. An acknowledgment shall not be sent until the SDU reassembly function has pulled the I-frame.

Upon receipt of a valid I-frame with an out-of-sequence TxSeq (see Section 8.5.6) all frames with a sequence number less than TxSeq shall be assumed lost and marked as missing. The missing I-frames are in the range from ExpectedTxSeq (the frame that the device was expecting to receive) up to TxSeq-1, (the frame that the device actually received). ExpectedTxSeq shall be set to TxSeq +1. The received I-frame shall be made available for pulling by the reassembly function. The acknowledgment shall not occur until the SDU reassembly function has pulled the I-frame. The ReqSeq shall be processed according to Section 8.5.4.

### 8.5.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released or how many new incoming I-frames could be stored. If BufferSeq is incremented, an acknowledgment shall be sent to the peer entity. If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

The acknowledgment may be an RR or an I-frame. The acknowledgment shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there is no I-frame buffered for pulling, ExpectedTxSeq is equal to BufferSeq.

### 8.5.4 Sending and receiving acknowledgments

One of the timers MonitorTimer or RetransmissionTimer shall always be active while in Flow Control mode. Both timers shall never be active concurrently.

#### 8.5.4.1 Sending acknowledgments

Whenever a data Link Layer entity transmits an I-frame or a S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

#### 8.5.4.2 Receiving acknowledgments

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall be used to acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.

1. If any outstanding I-frames were acknowledged then
   a. Stop the RetransmissionTimer
   b. If there are still unacknowledged I-frames then restart the RetransmissionTimer, otherwise start the MonitorTimer.
   c. Transmit any I-frames awaiting transmission according to Section 8.5.1.

ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including ExpectedAckSeq have been acknowledged.

### 8.5.5 Waiting acknowledgments

If the RetransmissionTimer expires the following actions shall be taken:

The I-frame supervised by the RetransmissionTimer shall be considered lost, and ExpectedAckSeq shall be incremented by one.

1. If I-frames are waiting to be sent
   a. the RetransmissionTimer is restarted.
   b. I-frames awaiting transmission are transmitted according to Section 8.5.1.
2. If there are no I-frames waiting to be sent
   a. If there are still unacknowledged I-frames the RetransmissionTimer is restarted, otherwise the MonitorTimer is started.

### 8.5.6 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in flow control only mode are defined in this section.

#### 8.5.6.1 TxSeq sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1. The ReqSeq shall be processed according to Section 8.5.4. The Information field shall be discarded since it has already been received.

- *Out-of-sequence I-frame*

  The out-of-sequence I-frame is identified by a TxSeq such that ExpectedTxSeq < TxSeq < (BufferSeq + TxWindow). The ReqSeq shall be processed according to Section 8.5.4.

  The missing I-frame(s) are considered lost and ExpectedTxSeq is set equal to TxSeq+1 as specified in Section 8.5.2. The missing I-frame(s) are reported as lost to the SDU reassembly function.

- *Invalid TxSeq*

  An invalid TxSeq value is a value that does not meet either of the above conditions and TxSeq is not equal to ExpectedTxSeq. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

#### 8.5.6.2 ReqSeq sequence error

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

An L2CAP entity that fails to receive an acknowledgment for an I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.5.5.

#### 8.5.6.3 Invalid frame

Any frame received that is invalid (as defined in Section 3.3.6) shall be discarded, and no action shall be taken as a result of that frame, unless the receiving L2CAP entity is configured to deliver erroneous frames to the layer above L2CAP. In that case, the data contained in invalid frames may also be added to the receive buffer and made available for pulling from the SDU reassembly function.

## 8.6 Enhanced Retransmission mode

Enhanced Retransmission mode operates as an HDLC balanced data link operational mode. Either L2CAP entity may send frames at any time without receiving explicit permission from the other L2CAP entity. A transmission may contain single or multiple frames and shall be used for I-frame transfer and/or to indicate status change.

### 8.6.1 Function of PDU types

Enhanced Retransmission mode uses I-frames to transfer upper layer information and S-frames for supervision. There are four S-frames defined: Receiver Ready (RR),

Reject (REJ), Receiver Not Ready (RNR), and Selective Reject (SREJ). All frames formats in Enhanced Retransmission mode shall use the Enhanced Control Field.

#### 8.6.1.1 Receiver Ready (RR)

The RR frame shall be used by an L2CAP entity to

1. Indicate that it is ready to receive I-frames
2. Acknowledge previously received I-frames numbered up to and including ReqSeq - 1.

An RR with P-bit set to 1 (P=1) is used to indicate the clearance of any busy condition that was initiated by an earlier transmission of an RNR frame by the same L2CAP entity.

#### 8.6.1.2 Reject (REJ)

The REJ frame shall be used by an L2CAP entity to request retransmission of I-frames starting with the frame numbered ReqSeq. I-frames numbered ReqSeq - 1 and below shall be considered acknowledged. Additional I-frames awaiting initial transmission may be transmitted following the retransmitted I-frame(s) up to the TxWindow size of the receiver.

At most only one REJ exception from a given L2CAP entity to another L2CAP entity shall be established at any given time. A REJ frame shall not be transmitted until an earlier REJ exception condition or all earlier SREJ exception conditions have been cleared. The REJ exception condition shall be cleared upon the receipt of an I-frame with TxSeq equal to the ReqSeq of the REJ frame.

Two L2CAP entities may be in REJ exception conditions with each other at the same time.

#### 8.6.1.3 Receiver Not Ready (RNR)

The RNR frame shall be used by an L2CAP entity to indicate a busy condition (i.e. temporary inability to receive I-frames). I-frames numbered up to and including ReqSeq - 1 shall be considered acknowledged. The I-frame numbered ReqSeq and any subsequent I-frames sent shall not be considered acknowledged. The acceptance status of these I-frames shall be indicated in subsequent transfers.

#### 8.6.1.4 Selective Reject (SREJ)

The SREJ frame shall be used by an L2CAP entity to request retransmission of one I-frame. The ReqSeq shall indicate the TxSeq of the earliest I-frame to be retransmitted (not yet reported by a SREJ). If the P-bit is set to 1 then I-frames numbered up to and including ReqSeq -1 shall be considered acknowledged. If the P-bit is set to 0 then the ReqSeq field in the SREJ shall not indicate acknowledgment of I-frames.

Each SREJ exception condition shall be cleared upon receipt of an I-frame with TxSeq equal to the ReqSeq sent in the SREJ frame.

An L2CAP entity may transmit one or more SREJ frames with the P=0 before one or more earlier SREJ exception conditions initiated with SREJ(P=0) have been cleared. An L2CAP entity shall not transmit more than one SREJ with P=1 before all earlier SREJ exception conditions have been cleared. A SREJ frame shall not be transmitted if an earlier REJ exception condition has not been cleared. Likewise a REJ frame shall not be transmitted if one or more SREJ exception conditions have not been cleared. Only one I-frame shall be retransmitted in response to receiving a SREJ frame with P=0. Additional I-frames awaiting initial transmission may be transmitted following the retransmission of the specific I-frame requested by SREJ with P=1.

#### 8.6.1.5 Functions of the Poll (P) and Final (F) bits

P-bit set to 1 shall be used to solicit a response frame with the F-bit set to 1 from the remote L2CAP entity at the earliest respond opportunity. At most only one frame with a P=1 shall be outstanding in a given direction at a given time. Before an L2CAP entity issues another frame with P=1, it shall have received a response frame from the remote L2CAP entity with F=1. If no valid frame is received with F=1 within Monitor timeout period, the frame with P=1 may be retransmitted.

The Final bit shall be used to indicate the frame as a response to a soliciting poll (S-frame with P=1). The frame with F=1 shall not be retransmitted. The Monitor-timeout is not used to monitor lost frames with F=1. Additional frames with F=0 may be transmitted following the frame with F=1.

S-frames shall not be transmitted with both the F-bit and the P-bit set to 1 at the same time.

### 8.6.2 Rules for timers

Timers are started upon transmission of a packet. Timers should be started when the corresponding packet leaves the Controller (transmitted or flushed). If the timer is not started when the packet leaves the Controller then it shall be started when the packet is delivered to the Controller. The specific rules for BR/EDR and BR/EDR/LE Controllers are described in the following sections.

#### 8.6.2.1 Timer rules for ACL-U logical links

If a flush timeout does not exist on the ACL-U logical link for the channel using Enhanced Retransmission mode then the value for the Retransmission timeout shall be at least two seconds and the value for the Monitor timeout shall be at least 12 seconds.

If a flush timeout exists on the link for Enhanced Retransmission mode then the value for the Retransmission timeout shall be three times the value of flush timeout, subject to a minimum of 1 second and maximum of 2 seconds.

If a flush timeout exists on the link for Enhanced Retransmission mode and both sides of the link are configured to the same flush timeout value then the monitor timeout shall be set to a value at least as large as the Retransmission timeout otherwise the value of the Monitor timeout shall be six times the value of flush timeout, subject to a minimum of the retransmission timeout value and a maximum of 12 seconds.

If an L2CAP entity knows that a specific packet has been flushed instead of transmitted then it may execute proper error recovery procedures immediately.

When configuring a channel over an ACL-U logical link the values sent in an L2CAP_CONFIGURATION_REQ packet for Retransmission timeout and Monitor timeout shall be 0.

Note: If the link has a flush timeout and the Non-Flushable Packet Boundary Flag feature is used to mark the Enhanced Retransmission mode packets as non-flushable then the link does not have a flush timeout with regards to Enhanced Retransmission mode.

#### 8.6.2.2 [This section is no longer used]

#### 8.6.2.3 [This section is no longer used]

### 8.6.3 General rules for the state machine

Enhanced Retransmission mode is specified using a pair of state machines, a Transmitter state machine and a Receiver state machine. The following rules apply to the state machine pair.

1. The state machine pair specifies the behavior of the protocol. Designers and implementers may choose any design / implementation technique, but it shall behave in a manner identical to the external behavior of the specified state machines.
2. There is a single state machine pair for each active L2CAP channel configured to use Enhanced Retransmission mode.
3. Variables are used to limit the number of states by maintaining the state of particular conditions. The variables are defined in Section 8.6.5.2.
4. For some combinations of event and condition, the state tables provide an action that involves alternatives or alternative groups of actions. The alternatives are mutually exclusive; selection of an alternate is done based upon (i) local status, (ii) a layer management action, or (iii) an implementation decision. There is no

relationship between the order of the alternatives between events, nor is it implied that the same alternative must be selected every time the event occurs.

5. The state tables use timers. Any Start Timer action restarts the specified timer from its initial value, even if the timer is already running. When the timer reaches 0 the appropriate timer expired event is set and the timer stops. The Stop Timer action stops a timer if it is running.
6. Events not recognized in a particular state are assumed to remain pending until any masking flag is modified or a transition is made to a state where they can be recognized.
7. Some state transitions and actions are triggered by internal events (e.g. requests from the upper layer). It is implementation specific how these internal events are realized. They are used for clarity in specifying the state machine. All events including Internal events are described in Section 8.6.5.3.
8. The state machines specify the exact frames to be sent by transmitters but are relaxed on what receivers are allowed to accept as valid. For example there are cases where the transmitter is required to send a frame with P=1. The correct response is a frame with F=1 but in some cases the receiver is allowed to accept a frame with F=0 in addition to F=1.

### 8.6.4 State diagram

The state diagram in Figure 8.4 shows the states and the main transitions. Not all events are shown on the state diagram.

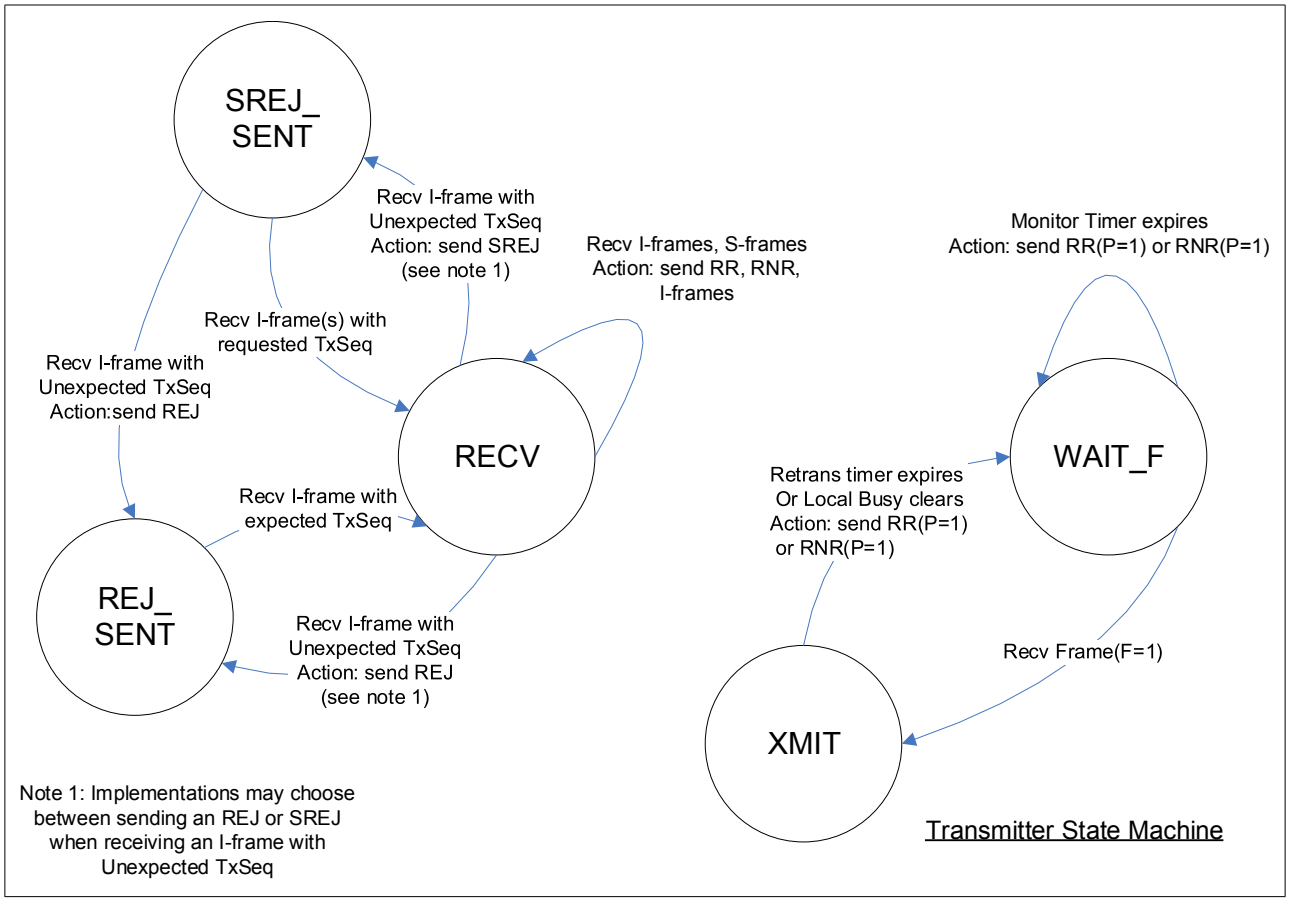



*Figure 8.4: Receiver state machine*

### 8.6.5 States tables

#### 8.6.5.1 State machines

Enhanced Retransmission mode is described as a pair of state machine. The Receiver state machine handles all received frames while the Transmitter State machine handles all asynchronous events including requests from the upper layer and the expiration of timers.

The Receiver state machine “calls” the Transmitter state machine using the PassToTx action. This shows up in the Transmitter state machine as an event. When the Transmitter state machine is called it runs to completion before returning to the Receiver state machine. Running to completion means that all actions are executed and the Transmitter state is changed to the new state.

The Receiver and Transmitter state machine share variables and timers.

#### 8.6.5.2 States

The following states have been defined to specify the protocol; the actual number of states and naming in a given implementation is outside the scope of the specification:

**RECV**—This is the main state of the Receiver state machine.

**REJ_SENT**—The L2CAP entity has sent a REJ frame to cause the remote L2CAP entity to resend I-frame(s). The L2CAP entity is waiting for the I-frame with a TxSeq that matches the ReqSeq sent in the REJ. Whether to send a REJ versus a SREJ is implementation dependent.

**SREJ_SENT**—The L2CAP entity has sent one or more SREJ frames to cause the remote L2CAP entity to resend missing I-frame(s). The local L2CAP entity is waiting for all requested I-frames to be received. If additional missing I-frames are detected while in SREJ_SENT then additional SREJ frames or a REJ frame can be sent to request those I-frames. Whether to send a SREJ versus a REJ is implementation dependent.

**XMIT**—This is the main state of the Transmitter state machine.

**WAIT_F**—Local busy has been cleared or the Retransmission timer has expired and an S-frame with P=1 has been sent. The local L2CAP entity is waiting for a frame with F=1. New I-frames cannot be sent while in the WAIT_F state to prevent the situation where retransmission of I-frames could result in the channel being disconnected.

#### 8.6.5.3 Variables and timers

Variables are used to limit the number of states and help clarify the state chart tables. Variables can be set to values, evaluated in conditions and compared in conditional statements. They are also used in the action descriptions. Below is a list of the operators, connectives and statements that can be used with variables.

| Operator, connective or statement | Description |
|---|---|
| := | Assignment operator. Used to set a variable to a value |
| = | Relational operator "equal" |
| > | Relational operator "greater than" |
| < | Relational operator "less than" |
| ≥ | Relational operator "greater than or equal" |
| ≤ | Relational operator "less than or equal" |
| + | Arithmetic operator "plus" |

| Operator, connective or statement | Description |
|---|---|
| **and** | logical connective "and." It returns TRUE if both operands are TRUE otherwise it returns FALSE. |
| **or** | logical connective "or." It returns TRUE if either of its operands are TRUE otherwise it returns FALSE. |
| **if** (expression) **then {**<br>statement<br>**}** | Conditional Statement. If expression is TRUE then the statement is executed otherwise the statement is not executed. The statement is composed of one or more actions. All the actions in the statement are indented under the **if** ... **then** clause and contained within braces “{ }”. |
| **if** (expression) **then {**<br>statement1<br>**}**<br>**else {**<br>statement2<br>**}** | Conditional Statement. If expression is TRUE then statement1 is executed otherwise statement2 is executed. A statement is composed of one or more actions. All the actions in the statement1 are indented under the **if** ... **then** clause and contained within braces “{ }”. All the actions of statement2 are indented under the else clause and contained within braces “{ }”. |

*Table 8.2: Operators, connectives, and statements used with variables*

Enhanced Retransmission mode uses the following variables and sequence numbers described in Section 8.3:

- TxSeq
- NextTxSeq
- ExpectedAckSeq
- ReqSeq
- ExpectedTxSeq
- BufferSeq

In addition to the variables above the following variables and timers are used:

**RemoteBusy**—when set to TRUE RemoteBusy indicates that the local L2CAP entity has received an RNR from the remote L2CAP entity and considers the remote L2CAP entity as busy. When the remote device is busy it will likely discard I-frames sent to it. The RemoteBusy flag is set to FALSE when the local L2CAP Entity receives an RR, REJ or SREJ. When set to FALSE the local L2CAP entity considers the remote L2CAP entity able to accept I-frames. When the channel is created RemoteBusy shall be set to FALSE.

**LocalBusy**—when set to TRUE, LocalBusy indicates the local L2CAP entity is busy and will discard received I-frames. When set to FALSE the local L2CAP entity is not

busy and is able to receive I-frames. When the channel is created LocalBusy shall be set to FALSE.

**UnackedFrames**—holds the number of unacknowledged I-frames. When the channel is created UnackedFrames shall be set to 0.

**UnackedList**—holds the unacknowledged I-frames so they can be retransmitted if necessary. I-frames in the list are accessed via their TxSeq number. For example UnackedList[5] accesses the I-frame with TxSeq 5.

**PendingFrames**—holds the number of pending I-frames. I-frames passed to L2CAP from the upper layer may be unable to be sent immediately because the remote L2CAP entity's TxWindow is full, is in a busy condition or the local L2CAP is in the incorrect state. When I-frames cannot be sent they are stored in a queue until conditions allow them to be sent. When the channel is created PendingFrames shall be set to 0.

**SrejList**—is a list of TxSeq values for I-frames that are missing and need to be retransmitted using SREJ. A SREJ has already been sent for each TxSeq on the list. When SrejList is empty it equals 0 (i.e. SrejList = 0). If SrejList is not empty it is greater than 0 (i.e. SrejList > 0).

**RetryCount**—holds the number of times an S-frame operation is retried. If an operation is tried MaxTransmit times without success the channel shall be closed.

**RetryIframes**[ ]—holds a retry counter for each I-frame that is sent within the receiving device's TxWindow. Each time an I-frame is retransmitted the corresponding counter within RetryIframes is incremented by one. When an attempt to retransmit the I-frame is made and the counter is equal to MaxTransmit then the channel shall be closed.

**RNRsent**—when set to TRUE it means that the local L2CAP entity has sent an RNR frame. It is used to determine if the L2CAP entity needs to send an RR to the remote L2CAP entity to clear the busy condition. When the channel is created RNRsent shall be set to FALSE.

**RejActioned**—is used to prohibit a frame with F=1 from causing I-frames already retransmitted in response to a REJ from being retransmitted again. RejActioned is set to TRUE if a received REJ is actioned when a frame sent with P=1 is unanswered. When the channel is created RejActioned shall be set to FALSE.

**SrejActioned**—is used in conjunction with SrejSaveReqSeq to prohibit a frame with F=1 from causing an I-frame already retransmitted in response to a SREJ from being retransmitted again. SrejActioned is set to TRUE if a received SREJ is actioned when a frame sent with P=1 is unanswered. When the channel is created SrejActioned shall be set to FALSE.

**SrejSaveReqSeq**—is used to save the ReqSeq of a SREJ frame that causes SrejActioned to be set to TRUE.

**SendRej**—when set to TRUE it indicates that the local L2CAP entity has determined that a REJ should be sent in the SREJ_SENT state while processing received I-frames. The sending of new SREJ frames is stopped. When the channel is created SendRej shall be set to FALSE.

**BufferSeqSrej**—is used while in the SREJ_SENT state to keep track of the value to which BufferSeq will be set upon exit of the SREJ_SENT state.

**FramesSent**—is used to keep track of the number I-frames sent by the Send-Data and Retransmit-I-frames actions.

**MaxTxWin**—contains the maximum window size plus 1. It is used in TxWindow *mod* operations as the divisor and in state table conditions. This value shall be set to 16384 (0x4000) if the Extended Window Size option is used; otherwise it shall be set to 64.

**RetransTimer**—The Retransmission Timer is used to detect lost I-frames. When the channel is created the RetransTimer shall be off.

**MonitorTimer**—The Monitor Timer is used to detect lost S-frames. When the channel is created the MonitorTimer shall be off.

#### 8.6.5.4 Events

**Data-Request**—The upper layer has requested that an SDU be sent. The SDU may need to be broken into multiple I-frames by L2CAP based on the MPS of the remote device and/or the maximum PDU allowed by the HCI or QoS requirements of the system.

**Local-Busy-Detected**—A local busy condition occurs when the local L2CAP entity is temporarily unable to receive, or unable to continue to receive, I-frames due to internal constraints. For example, the upper layer has not pulled received I-frames and the local L2CAP entity needs to send an acknowledgment to the remote L2CAP entity. The method for handling the detection of local busy is implementation specific. An implementation may wait to send an RNR to see if the busy condition will clear before the remote L2CAP entity's Retransmission timer expires. If the busy condition clears then frames can be acknowledged with an RR or I-frame. If the busy condition does not clear before the remote L2CAP entity's Retransmission timer expires then an RNR shall be sent in response to the RR or RNR poll sent by the remote L2CAP entity. An implementation may send an RNR as soon as the local busy condition is detected.

**Local-Busy-Clear**—The local busy condition clears when L2CAP has buffer space to receive more I-frames (i.e. SDU Reassembly function and/or upper layer has pulled I-frames) and if necessary the upper layer has cleared the busy condition.

**Recv ReqSeqAndFbit**—This is an event generated by the Receiver state machine. It contains the ReqSeq and F-bit value of a received frame. The value of the F-bit can be checked in a condition.

**Recv Fbit**—This is an event generated by the Receiver state machine. It contains the F-bit value of a received frame. The value of the F-bit can be checked in a condition.

**RetransTimer-Expires**—The Retransmission Timer has counted to down to 0 and stopped.

**MonitorTimer-Expires**—The Monitor Timer has counted down to 0 and stopped.

**Recv I-frame**—Receive an I-frame with any value for the F-bit.

**Recv RR, REJ, RNR, SREJ (P=x)** or **(F=x)**—Receive a specific S-frame (RR, REJ, etc.) with a specific value for the P and/or F bit. The F-bit and the P-bit shall not both be set to 1 in a transmitted S-frame so received S-frames with both P and F set to 1 should be ignored. If the P and/or F bit value is not specified in the event then either value is accepted.

**Recv RRorRNR**—Receive an RR or RNR with any value for the P-bit and F-bit.

**Recv REJorSREJ**—Receive an REJ or SREJ with any value for the P-bit and F-bit.

**Recv frame**—This is catch-all for all frames that are not explicitly declared as events in the state table.

#### 8.6.5.5 Conditions

**RemoteBusy = TRUE or FALSE**—TRUE indicates the remote L2CAP entity is in a busy condition and FALSE indicates the remote L2CAP entity is not busy.

**LocalBusy = TRUE or FALSE**—TRUE indicates the local L2CAP entity is in a busy condition and FALSE indicates the local L2CAP entity is not busy.

**RemWindow-Not-Full**—The number of unacknowledged I-frames sent by L2CAP has not yet reached the TxWindow size of the remote L2CAP entity.

**RemWindow-Full**—The number of unacknowledged I-frames sent by the L2CAP entity has reached the TxWindow size of the remote L2CAP entity. No more I-frames shall be sent until one or more I-frames have been acknowledged.

**RNRsent = TRUE or FALSE**—**TRUE** indicates an RNR has been sent while a local busy condition exists. It is set to FALSE when the local busy condition clears.

**F = 0 or 1**—the F-bit of a received frame is checked. The F-bit of the received frame is available as part of the Recv ReqSeqAndFbit and Recv Fbit events.

**RetryIframes[i] < or ≥ MaxTransmit**—Compare the appropriate counter in RetryIframes after processing the ReqSeq in the receive frame to determine if it has reached MaxTransmit or not.

**RetryCount < or ≥ MaxTransmit**—Compare RetryCount to determine if it has reached MaxTransmit or not.

**With-Expected-TxSeq**—The TxSeq of a received I-frame is equal to ExpectedTxSeq.

**With-Valid-ReqSeq**—The ReqSeq of the received frame is in the range ExpectedAckSeq ≤ ReqSeq < NextTxSeq.

**With-Valid-ReqSeq-Retrans**—The ReqSeq of the received frame is in the range ExpectedAckSeq ≤ ReqSeq < NextTxSeq.

**With-Valid-F-bit** —The F-bit of a received frame is valid if it is 0 or if it is 1 and a frame sent with P=1 by the local L2CAP entity is unanswered (i.e. the local L2CAP entity send a frame with P=1 and has not yet received a frame with F=1 until receiving this one). If the Transmitter state machine is in the WAIT_F state then a frame sent with P=1 is unanswered.

**With-unexpected-TxSeq**—The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the I-frame but has a TxSeq “greater” than ExpectedTxSeq where “greater” means later in sequence than ExpectedTxSeq.

**With-duplicate-TxSeq**—The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the I-frame but has a TxSeq “less” than ExpectedTxSeq where “less” means earlier in the sequence than ExpectedTxSeq. In other words this is a frame that has already been received.

**With-Invalid-TxSeq**—The TxSeq of the received I-frame is not within the TxWindow of the L2CAP entity receiving the frame.

**With-Invalid-ReqSeq**—The ReqSeq of the received frame is not in the range ExpectedAckSeq ≤ ReqSeq < NextTxSeq.

**With-Invalid-ReqSeq-Retrans**—The ReqSeq of the received frame is not in the range ExpectedAckSeq ≤ ReqSeq < NextTxSeq.

**Not-With-Expected-TxSeq**—The TxSeq of the received I-frame is within the TxWindow of the L2CAP entity receiving the frame but is not equal to ExpectedTxSeq. It is either unexpected or a duplicate.

**With-Expected-TxSeq-Srej**—The TxSeq of the received I-frame is equal to the TxSeq at the head of SrejList.

**SendRej = TRUE or FALSE**—**TRUE** indicates that a REJ will be sent after all frames requested using SREJ have been received.

**SrejList = or > 1**—Determine if the number of items in SrejList is equal to or greater than 1.

**With-Unexpected-TxSeq-Srej**—The TxSeq of the received I-frame is equal to one of the values stored in SrejList but is not the TxSeq at the head. This indicates that one or more I-frames requested using SREJ are missing either because the SREJ was lost or the requested I-frame(s) were lost. Either way the SREJ frames must be resent to retrieve the missing I-frames.

**With-duplicate-TxSeq-Srej**—The TxSeq of the received I-frame is equal to a TxSeq of one of the saved I-frames indicating it is a duplicate.

#### 8.6.5.6 Actions

**Send-Data**—This action is executed as a result of a Data-Request event. The number of I-frames sent without being acknowledged shall not exceed the TxWindow size of the receiving L2CAP entity (UnackedFrames is less than or equal to the remote L2CAP entity's TxWindow). Any I-frames that cannot be sent because they would exceed the TxWindow size are queued for later transmission. For each I-frame the following actions shall be carried out:

```
Send I-frame with TxSeq set to NextTxSeq and ReqSeq set to BufferSeq.
UnackedList[NextTxSeq] := I-frame
UnackedFrames := UnackedFrames + 1
FramesSent := FramesSent + 1
RetryIframes[NextTxSeq] := 1
NextTxSeq := (NextTxSeq + 1) mod MaxTxWin
Start-RetransTimer
```

**Pend-Data**—This action is executed as a result of a Data-Request when it is not possible to send I-frames because the window is full, the remote L2CAP entity is in a busy condition or the local L2CAP entity is not in a state where I-frames can be sent (e.g. WAIT_F). The I-frame(s) are queued for later transmission.

**Process-ReqSeq**—the ReqSeq contained in the received frame shall acknowledge previously transmitted I-frames. ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including (ReqSeq—1) have been acknowledged. The acknowledged I-frames shall be removed from UnackedList, the retry counters for each acknowledged frame shall be set to 0 and the number of acknowledged frames shall be subtracted from UnackedFrames so that UnackedFrames shall contain the number of the remaining unacknowledged I-frames. Pending I-frames are now available to be transmitted by the Send-Ack action. If UnackedFrames equals 0 then Stop-RetransTimer.

**Send RR, RNR (P=x) or (F=x)**—Send the specified S-frame with the specified value for the P-bit or F-bit. If a value for the P-bit or F-bit is not specified the value shall be 0. For example Send RR(P=1) means send an RR with the P-bit set to 1 and the F-bit set to 0. The ReqSeq field shall be set to BufferSeq. If an RNR is sent, RNRsent shall be set to TRUE.

**Send REJ (P =x) or (F=x)**—Send a REJ with the specified value for the P-bit or F-bit. The ReqSeq field shall be set to ExpectedTxSeq. If a value for the P-bit or F-bit is not specified the value shall be 0, which acknowledges previously received I-frames up to ExpectedTxSeq—1 and may allow the remote L2CAP entity to transmit new I-frames. If the local L2CAP entity is not in a position to acknowledge the previously received I-frames it may use SREJ(P=0) or RNR. It may also wait to send the REJ until it is able to acknowledge the I-frames.

**Send RRorRNR (P=x) or (F=1)**—Send an RR or RNR with the specified value for the P-bit or F-bit based on the value of LocalBusy. If a value for the P-bit or F-bit is not specified the value shall be 0. An RNR shall be sent if LocalBusy equals TRUE. If LocalBusy equals FALSE then an RR shall be sent.

**Send IorRRorRNR(F=1)**—Send I-frames, an RR or an RNR with the F-bit set to 1. The following algorithm shall be used:

```
FramesSent:=0
if LocalBusy = TRUE then {
    Send RNR(F=1)
}
if RemoteBusy = TRUE and UnackedFrames > 0 then {
    Start-RetransTimer
}
RemoteBusy := FALSE
Send-Pending-I-frames
if LocalBusy = FALSE and FramesSent = 0 then {
    Send RR(F=1)
}
```

The SendIorRRorRNR(F=1) sends frames by invoking other actions. During the execution of SendIorRRorRNR multiple actions may be invoked. The first action invoked shall send the first or only frame with the F-bit set to 1. All other frames sent shall have the F-bit set to 0.

**Send SREJ**—Send one or more SREJ frames with P=0. For each missing I-frame starting with ExpectedTxSeq up to but not including the TxSeq of the received I-frame, an SREJ frame is sent with ReqSeq set to the TxSeq of the missing frame. The TxSeq is inserted into the tail of SrejList. For example if ExpectedTxSeq is 3 and the received I-frame has a TxSeq of 5 there are two missing I-frames. An SREJ with ReqSeq 3 is sent followed by an SREJ with ReqSeq 4. TxSeq 3 is inserted first into SrejList followed

by TxSeq 4. After all SREJ frames have been sent ExpectedTxSeq shall be set to (the TxSeq of the received I-frame + 1) *mod* MaxTxWin.

**Send SREJ(SrejList)**—Send one or more SREJ frames with P=0. An I-frame was received that matches one of the TxSeq values in the SrejList but does not match the head of SrejList. This means I-frames requested via SREJ are still missing. For each TxSeq value starting with the head of SrejList and going backwards (i.e., from the head towards the tail) through the SrejList up to but not including the TxSeq of the received frame, an SREJ frame is sent with ReqSeq set to the TxSeq from SrejList. The TxSeq is removed from SrejList and reinserted into the tail of SrejList. Finally, remove the TxSeq of the received frame from the head of the list.

**Send SREJ(SrejList-tail)(F=1)**—Send a SREJ frame with F=1 and ReqSeq equal to the TxSeq at the tail of SrejList.

**Start-RetransTimer**—If the Monitor timer is not running then start the Retransmission Timer from its initial value (see Retransmission timeout in Section 5.4). If the Retransmission timer is already running it is restarted from its initial value. If the Monitor timer is running then the Retransmission timer is not started.

**Start-MonitorTimer**—Start the Monitor Timer from its initial value (see Monitor timeout in Section 5.4). If the timer is already running it is restarted from its initial value.

**PassToTx**—Pass the ReqSeq and F-bit value of a received frame to the Transmitter state machine. This will show up as a Recv ReqSeqAndFbit event in the Transmitter state machine.

**PassToTxFbit**—Pass the F-bit value of a received frame to the Transmitter state machine. This will show up as a Recv Fbit event in the Transmitter state machine.

**Data-Indication**—A received I-frame is passed to the SDU reassembly function. For the purpose of the state machine this operation is completed immediately so the Send_Ack action should be executed as one of the next actions. In some cases the SDU reassembly function cannot accept the I-frame so the I-frame will be stored within the L2CAP Entity consuming a portion of its TxWindow. When the I-frame is pulled by the SDU reassembly function the Send_Ack action should be executed. Before the Send_Ack action is executed BufferSeq is advanced as follows:

```
BufferSeq := (BufferSeq + 1) mod MaxTxWin
```

**Increment-ExpectedTxSeq**—ExpectedTxSeq is incremented as follows:

```
ExpectedTxSeq := (ExpectedTxSeq + 1) mod MaxTxWin
```

**Stop-RetransTimer**—the Retransmission timer is stopped.

**Stop-MonitorTimer** —the Monitor timer is stopped.

**Send-Ack (F=x)**—an acknowledgment with the specified value for the F-bit may be sent. This action may occur in an action block with other actions that also send frames. If a frame has already been sent then it is not necessary to send additional frames.

If the value for the F-bit is not specified it shall be set to 0. If the value specified is P then the F-bit shall be set equal to the value of the P-bit of the received frame being acknowledged. If more than one frame is sent in the acknowledgment only the first frame shall have an F-bit set to 1. An acknowledgment is an RR, RNR, or pending I-frame(s) (I-frames that have not been transmitted yet). If pending I-frames are available and are allowed to be sent then as many as allowed should be sent as an acknowledgment. Sending an RR or RNR as an acknowledgment for each received I-frame is not required. An implementation may wait to send an RR or RNR until a specific number of I-frames have been received, after a certain period of time has elapsed or some other algorithm. To keep data flowing it is recommended that an acknowledgment be sent before the TxWindow is full. It should also be noted that the maximum size of a remote L2CAP entity's unacknowledged I-frame list may be smaller than the local L2CAP entity's TxWindow. Therefore the local L2CAP entity should not expect the remote L2CAP entity to send enough frames to fill its TxWindow and should acknowledge I-frames accordingly.

The following algorithm shall be used when sending an acknowledgment.

```
if LocalBusy == TRUE then {
    Send_RNR(F=x)
}
else if (RemoteBusy == FALSE) and Pending I-frames Exist and RemWindow-
Not-Full then {
    Send-Pending-I-frames (F=x)
}
else {
    Send_RR (F=x)
}
```

**InitSrej**—Initialize the variables used for processing SREJ as follows:

```
Clear SrejList - (remove all values)
SendRej := FALSE
BufferSeqSrej := BufferSeq
```

**SaveIframeSrej**—Save the received I-frame. Missing I-frame(s) will be retransmitted in response to SREJ frames. Implementations may choose to save the I-frame in its proper sequence order by leaving room for the missing I-frames.

**StoreOrIgnore**—If the local L2CAP entity has room to store the received I-frame then it may store it otherwise it shall discard it. If the received I-frame is stored, ExpectedTxSeq is advanced as follows:

```
ExpectedTxSeq := (ExpectedTxSeq + 1) mod MaxTxWin
```

**PbitOutstanding**—If the Transmitter state machine of the local L2CAP entity is in the WAIT_F state then return TRUE otherwise return FALSE.

**Retransmit-I-frames**—All the unacknowledged I-frames starting with the I-frame with TxSeq equal to the ReqSeq field of the received S-frame (REJ or RR) is retransmitted. If the P-bit of the received S-frame is 1 then the F-bit of the first I-frame sent shall be 1. If the P-bit of the received S-frame is 0 then the F-bit of the first I-frame sent shall be 0. The F-bit of all other unacknowledged I-frames sent shall be 0. The retry counter in RetryIframes[ ] for each retransmitted I-frame is incremented by 1. If a retry counter in RetryIframes[ ] is equal to MaxTransmit then the channel shall be closed. FramesSent shall be incremented by 1 for each frame sent. If the RetransTimer is not already running then perform the Start-RetransTimer action.

**Retransmit-Requested-I-frame**—The unacknowledged I-frame with TxSeq equal to the ReqSeq field of the received S-frame (SREJ) is retransmitted. If the P-bit of the received S-frame is 1 then the F-bit of the retransmitted I-frame shall be 1. If the P-bit of the received S-frame is 0 then the F-bit of the retransmitted I-frame shall be 0. The retry counter in RetryIframes[ ] corresponding to the retransmitted I-frame is incremented by 1. If the RetransTimer is not already running then perform the Start-RetransTimer action.

**Send-Pending-I-frames (F=x)**—If PbitOutstanding equals FALSE then send all pending I-frames that can be sent without exceeding the receiver's TxWindow using the Send-Data action. If a value for the F-bit is specified then the F-bit of the first I-frame sent shall be set to the specified value and the F-bit of all other I-frames sent shall be set to 0. If no value for the F-bit is specified then all I-frames sent shall have the F-bit set to 0. Pending I-frames are I-frames that have been given to the L2CAP entity by the upper layer but have not yet been transmitted. If one or more I-frames are sent and the RetransTimer is not already running then perform the Start-RetransTimer action.

**Close Channel**—Close the L2CAP channel as described in Section 4.6.

**Ignore**—Silently discard the event.

**CloseChannelOrIgnore**—Either close the L2CAP channel as described in Section 4.6 (in which case the next state is ignored) or silently discard the event.

**PopSrejList**—Remove and discard the TxSeq from the head of SrejList.

**Data-IndicationSrej**—If the received I-frame fills a gap in a sequence of saved I-frames then all the saved I-frames in the sequence are passed to the SDU reassembly function. For the purpose of the state machine this operation is completed immediately. For example if the TxSeq of saved I-frames before receiving an I-frame is 2, 3, 5, 6, 9 and the received I-frame has a TxSeq of 4 then it fills the gap between 3 and 5 so the sequence 2, 3, 4, 5, 6 can be passed to the SDU reassembly function. When the I-frames are actually removed from the L2CAP entity receive buffers either by being processed immediately or when pulled by the SDU reassembly function, BufferSeqSrej is advanced as follows:

```
BufferSeqSrej := (BufferSeqSrej + 1) mod MaxTxWin
```

### 8.6.5.7 XMIT state table

| Event | Condition | Action | Next State |
|---|---|---|---|
| Data-Request | RemoteBusy = FALSE<br>**and**<br>RemWindow-Not-Full | Send-Data | XMIT |
| Data-Request | RemoteBusy = TRUE **or**<br>RemWindow-Full | Pend-Data | XMIT |
| Local-Busy-Detected | *none* | LocalBusy := TRUE<br>Optionally Send RNR | XMIT |
| Local-Busy-Clear | RNRsent = TRUE | LocalBusy := FALSE<br>RNRsent := FALSE<br>Send RR(P=1)<br>RetryCount := 1<br>Stop-RetransTimer<br>Start-MonitorTimer | WAIT_F |
| Local-Busy-Clear | RNRsent = FALSE | LocalBusy := FALSE<br>RNRsent := FALSE | XMIT |
| Recv ReqSeqAndFbit | *none* | Process-ReqSeq | XMIT |
| Recv Fbit | *none* | *none* | XMIT |
| RetransTimer-Expires | *none* | Send RRorRNR(P=1)<br>RetryCount := 1<br>Start-MonitorTimer | WAIT_F |

*Table 8.3: XMIT state table*

#### 8.6.5.8 WAIT_F state table

| Event | Condition | Action | Next State |
|---|---|---|---|
| Data-Request | *none* | Pend-Data | WAIT_F |
| Recv ReqSeqAndFbit | F = 1 | Process-ReqSeq<br>Stop-MonitorTimer<br>**If** UnackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}** | XMIT |
| Recv ReqSeqAndFbit | F = 0 | Process-ReqSeq | WAIT_F |
| Recv Fbit | F = 1 | Stop-MonitorTimer<br>**If** UnackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}** | XMIT |
| Recv Fbit | F = 0 | *none* | WAIT_F |
| MonitorTimer-Expires | RetryCount < MaxTransmit | RetryCount := RetryCount+1<br>Send RRorRNR(P=1)<br>Start-MonitorTimer | WAIT_F |
| MonitorTimer-Expires | RetryCount ≥ MaxTransmit | Close Channel | *none* |

*Table 8.4: WAIT_F state table*

#### 8.6.5.9 RECV state table

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame (F=0) | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = FALSE | Increment-ExpectedTxSeq<br>PassToTx<br>Data-Indication<br>Send-Ack(F=0) | RECV |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame (F=1) | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = FALSE | Increment-ExpectedTxSeq<br>PassToTx<br>Data-Indication<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>Send-Pending-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Ack(F=0) | RECV |
| Recv I-frame | With-duplicate-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = FALSE | PassToTx | RECV |
| Recv I-frame | With-unexpected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = FALSE<br>*Alternative 1* | PassToTx<br>SendREJ | REJ_SENT |
| Recv I-frame | With-unexpected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = FALSE<br>*Alternative 2* | PassToTx<br>InitSrej<br>SaveIframeSrej<br>SendSREJ | SREJ_SENT |
| Recv I-frame | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = TRUE | PassToTx<br>StoreOrIgnore | RECV |
| Recv I-frame | With-Valid-ReqSeq<br>**and** Not-With_Expected_TxSeq<br>**and** With-Valid-F-bit<br>**and** LocalBusy = TRUE | PassToTx | RECV |
| Recv RNR (P=0) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Stop-RetransTimer | RECV |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv RNR (P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Stop-RetransTimer<br>Send RRorRNR (F=1) | RECV |
| Recv RR(P=0) (F=0) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>**If** RemoteBusy = TRUE **and** Un-ackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}**<br>RemoteBusy := FALSE<br>Send-Pending-I-frames | RECV |
| Recv RR(F=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Pending-I-frames | RECV |
| Recv RR(P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>Send IorRRorRNR(F=1) | RECV |
| Recv REJ (F=0) | With-Valid-ReqSeq-Retrans **and**<br>RetryIframes[i] < MaxTransmit<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-I-frames<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then {**<br>RejActioned := TRUE<br>**}** | RECV |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv REJ (F=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned :=FALSE<br>**}**<br>Send-Pending-I-frames ] | RECV |
| Recv SREJ (P=0) (F=0) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>Retransmit-Requested-I-frame<br>**If** PbitOutstanding **then {**<br>SrejActioned := TRUE<br>SrejSaveReqSeq = ReqSeq<br>**}** | RECV |
| Recv SREJ (P=0) (F=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>**If** SrejActioned = TRUE **and** Srej-SaveReqSeq = ReqSeq **then {**<br>SrejActioned := FALSE<br>**} else {**<br>Retransmit-Requested-I-frame<br>**}** | RECV |
| Recv SREJ(P=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-Requested-I-frame<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then {**<br>SrejActioned = TRUE<br>SrejSaveReqSeq := ReqSeq<br>**}** | RECV |
| Recv REJ | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| RECV SREJ | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame | (With-Invalid-TxSeq **and** TxWindow > MaxTxWin÷2)<br>**or** With-Invalid-ReqSeq | Close Channel | *none* |
| Recv I-frame | With-Invalid-TxSeq<br>**and** TxWindow ≤ MaxTxWin÷2 | CloseChannelOrIgnore | RECV |
| Recv RRorRNR | With-Invalid-ReqSeq | Close Channel | *none* |
| Recv REJorS-REJ | With-Invalid-ReqSeq-Retrans | Close Channel | *none* |
| Recv frame | *none* | Ignore | RECV |

*Table 8.5: RECV state table*

#### 8.6.5.10 REJ_SENT state table

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame (F=0) | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | Increment-ExpectedTxSeq<br>PassToTx<br>Data-Indication<br>Send-Ack (F=0) | RECV |
| Recv I-frame (F=1) | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | Increment-ExpectedTxSeq<br>PassToTx<br>Data-Indication<br>**If** RejActioned = FALSE **then {**<br>Retransmit I-frames<br>Send-Pending-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Ack (F=0) | RECV |
| Recv I-frame | With-Unexpected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx | REJ_SENT |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv RR (F=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Pending-I-frames | REJ_SENT |
| Recv RR (P=0) (F=0) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>**If** RemoteBusy = TRUE **and** UnackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}**<br>RemoteBusy := FALSE<br>Send-Ack (F=0) | REJ_SENT |
| Recv RR (P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>**If** RemoteBusy = TRUE **and** UnackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}**<br>RemoteBusy := FALSE<br>Send RR (F=1) | REJ_SENT |
| Recv RNR (P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Send RR (F=1) | REJ_SENT |
| Recv RNR (P=0) | With-Valid-ReqSeq<br>and With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Send RR (F=0) | REJ_SENT |
| Recv REJ (F=0) | With-Valid-ReqSeq -Retrans **and** RetryIframes[i] < MaxTransmit<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-I-frames<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then {**<br>RejActioned := TRUE<br>**}** | REJ_SENT |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv REJ (F=1) | With-Valid-ReqSeq -Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Pending-I-frames | REJ_SENT |
| Recv SREJ (P=0) (F=0) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>Retransmit-Requested-I-frame<br>**If** PbitOutstanding **then** {<br>SrejActioned := TRUE<br>SrejSaveReqSeq := ReqSeq<br>**}** | REJ_SENT |
| Recv SREJ (P=0) (F=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>**If** SrejActioned = TRUE **and** SrejSaveReqSeq = ReqSeq **then** {<br>SrejActioned := FALSE<br>**} else** {<br>Retransmit-Requested-I-frame<br>**}** | REJ_SENT |
| Recv SREJ (P=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-Requested-I-frames<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then** {<br>SrejActioned := TRUE<br>SrejSaveReqSeq := ReqSeq<br>**}** | REJ_SENT |
| Recv REJ | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| Recv SREJ (P=0) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |

| Event | Condition | Action | Next State |
|---|---|---|---|
| RECV SREJ (P=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| Recv I-frame | (With-Invalid-TxSeq **and** TxWindow > MaxTxWin÷2) **or** With-Invalid-ReqSeq | Close Channel | *none* |
| Recv RRorRNR | With-Invalid-ReqSeq | Close Channel | *none* |
| RecvREJorS-REJ | With-Invalid-ReqSeq-Retrans | Close Channel | *none* |
| Recv I-frame | With-Invalid-TxSeq **and** TxWindow ≤ MaxTxWin÷2 **and** With-Valid-ReqSeq | CloseChannelOrIgnore | REJ_SENT |
| Recv frame | *none* | Ignore | REJ_SENT |

*Table 8.6: REJ_SENT state table*

#### 8.6.5.11 SREJ_SENT state table

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame | With-Expected-TxSeq-Srej **and** With-Valid-ReqSeq **and** With-Valid-F-bit **and** SendRej = FALSE **and** SrejList = 1 | SaveIframeSrej PopSrejList PassToTx Data-IndicatioSrej BufferSeq := BufferSeqSrej Send-Ack (F=0) | RECV |
| Recv I-frame | With-Expected-TxSeq-Srej **and** With-Valid-ReqSeq **and** With-Valid-F-bit **and** SendRej = TRUE **and** SrejList = 1 | SaveIframeSrej PopSrejList PassToTx Data-IndicationSrej BufferSeq := BufferSeqSrej Send REJ | REJ_SENT |
| Recv I-frame | With-Expected-TxSeq-Srej **and** With-Valid-ReqSeq **and** With-Valid-F-bit **and** SrejList > 1 | SaveIframeSrej PopSrejList PassToTx Data-IndicationSrej | SREJ_SENT |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame | With-Expected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | SaveIframeSrej<br>Increment-ExpectedTxSeq<br>PassToTx | SREJ_SENT |
| Recv I-frame | With-Unexpected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** SendRej = FALSE | SaveIframeSrej<br>PassToTx<br>Send SREJ | SREJ_SENT |
| | | PassToTx<br>SendRej := TRUE | SREJ_SENT |
| Recv I-frame | With-Unexpected-TxSeq<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit<br>**and** SendRej = TRUE | PassToTx | SREJ_SENT |
| Recv I-frame | With-Unexpected-TxSeq-Srej<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | SaveIframeSrej<br>PassToTx<br>Send SREJ(SrejList) | SREJ_SENT |
| Recv I-frame | With-duplicate-TxSeq-Srej<br>**and** With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx | SREJ_SENT |
| Recv RR(F=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Pending-I-frames | SREJ_SENT |
| Recv RR(P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>**If** RemoteBusy = TRUE **and** UnackedFrames > 0 **then {**<br>Start-RetransTimer<br>**}**<br>RemoteBusy := FALSE<br>Send SREJ(SrejList-tail)(F=1) | SREJ_SENT |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv RR(P=0) (F=0) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | PassToTx<br>**If** RemoteBusy = TRUE **and** Unacked-Frames > 0 **then {**<br>Start-RetransTimer<br>**}**<br>RemoteBusy := FALSE<br>Send-Ack(F=0) | SREJ_SENT |
| Recv RNR(P=1) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Send SREJ(SrejList-tail)(F=1) | SREJ_SENT |
| Recv RNR(P=0) | With-Valid-ReqSeq<br>**and** With-Valid-F-bit | RemoteBusy := TRUE<br>PassToTx<br>Send RR(F=0) | SREJ_SENT |
| Recv REJ (F=0) | With-Valid-ReqSeq-Retrans<br>**and**<br>RetryIframes[i] < MaxTransmit<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-I-frames<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then {**<br>RejActioned := TRUE<br>**}** | SREJ_SENT |
| Recv REJ (F=1) | With-Valid-ReqSeq-Retrans<br>**and**<br>RetryIframes[i] < MaxTransmit<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>**If** RejActioned = FALSE **then {**<br>Retransmit-I-frames<br>**} else {**<br>RejActioned := FALSE<br>**}**<br>Send-Pending-I-frames | SREJ_SENT |
| Recv SREJ(P=0) (F=0) | With-Valid-ReqSeq-Retrans<br>**and** RetryIframes[i] < MaxTransmit<br>**and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>Retransmit-Requested-I-frame<br>**If** PbitOutstanding **then {**<br>SrejActioned := TRUE<br>SrejSaveReqSeq = ReqSeq<br>**}** | SREJ_SENT |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv SREJ(P=0) (F=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTxFbit<br>**If** SrejActioned = TRUE **and** SrejSaveReqSeq = ReqSeq **then {**<br>SrejActioned := FALSE<br>**} else {**<br>Retransmit-Requested-I-frame<br>**}** | SREJ_SENT |
| Recv SREJ(P=1) | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] < MaxTransmit **and** With-Valid-F-bit | RemoteBusy := FALSE<br>PassToTx<br>Retransmit-Requested-I-frame<br>Send-Pending-I-frames<br>**If** PbitOutstanding **then {**<br>SrejActioned := TRUE<br>SrejSaveReqSeq = ReqSeq<br>**}** | SREJ_SENT |
| Recv REJ | With-Valid-ReqSeq-Retrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| Recv SREJ(P=0) | With-Valid-ReqSeqRetrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| Recv SREJ(P=1) | With-Valid-ReqSeqRetrans **and** RetryIframes[i] ≥ MaxTransmit | Close Channel | *none* |
| Recv I-frame | (With-Invalid-TxSeq **and** TxWindow > MaxTxWin÷2)<br>**or** With-Invalid-ReqSeq | Close Channel | *none* |
| Recv RRorRNR | With-Invalid-ReqSeq | Close Channel | *none* |
| Recv REJorSREJ | With-Invalid-ReqSeq-Retrans | Close Channel | *none* |

| Event | Condition | Action | Next State |
|---|---|---|---|
| Recv I-frame | With-Invalid-TxSeq<br>**and** TxWindow ≤ MaxTxWin÷2 | CloseChannelOrIgnore | SREJ_SENT |
| Recv frame | *none* | Ignore | SREJ_SENT |

*Table 8.7: SREJ_SENT state table*

## 8.7 Streaming mode

When a link is configured to work in Streaming mode, the frame format for outgoing data is the same as for Enhanced Retransmission mode but frames are not acknowledged. Therefore

- RR, REJ, RNR and SREJ frames shall not be used in Streaming mode.
- The F-bit shall always be set to zero in the transmitter, and shall be ignored in the receiver.
- the MonitorTimer and RetransmissionTimer shall not be used in Streaming mode.

A channel configured to work in Streaming mode shall be configured with a finite value for the Flush Timeout on the transmitter.

### 8.7.1 Transmitting I-frames

When transmitting a new I-frame the control field parameter ReqSeq shall be set to 0, TxSeq shall be set to NextTxSeq, and NextTxSeq shall be incremented by one.

### 8.7.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be made available to the reassembly function. ExpectedTxSeq shall be incremented by one.

Upon receipt of a valid I-frame with an out-of-sequence TxSeq (see Section 8.7.3.1) all frames with a sequence number less than TxSeq shall be assumed lost and marked as missing. The missing I-frames are in the range from ExpectedTxSeq (the frame that the device was expecting to receive) up to and including TxSeq - 1. ExpectedTxSeq shall be set to TxSeq +1. The received I-frame shall be made available for pulling by the reassembly function. The ReqSeq shall be ignored.

Note: It is possible for a complete window size of I-frames to be missing and thus, no missing I-frames are detected. For example, when a window size of 63 is used this situation occurs when 63 I-frames in a row are missing.

The Extended Window Size option should be used if the inability to detect missing I-frames will cause problems for an application.

If there is no buffer space for the received I-frame an existing I-frame (i.e. the oldest) shall be discarded (flushed) freeing up buffer space for the new I-frame. The discarded I-frame shall be marked as missing.

### 8.7.3 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in Streaming mode are defined in this section.

#### 8.7.3.1 TxSeq sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.

The out-of-sequence I-frame is identified by a TxSeq that is greater than ExpectedTxSeq (TxSeq > ExpectedTxSeq). The ReqSeq shall be ignored. The missing I-frame(s) are considered lost and ExpectedTxSeq is set equal to TxSeq+1 as specified in Section 8.7.2. The missing I-frame(s) are reported as lost to the SDU reassembly function.

# 9 [THIS SECTION IS NO LONGER USED]

# 10 PROCEDURES FOR CREDIT BASED FLOW CONTROL

There are two credit-based flow control modes: LE Credit Based Flow Control mode and Enhanced Credit Based Flow Control mode.

## 10.1 LE Credit Based Flow Control mode

LE Credit Based Flow Control mode is used for LE L2CAP connection-oriented channels with flow control using a credit based scheme for L2CAP data (i.e. not signaling packets).

The number of credits (K-frames) that can be received by a device on an L2CAP channel is determined during connection establishment. K-frames shall only be sent on an L2CAP channel if the device has a credit count greater than zero for that L2CAP channel. For each K-frame sent the device decreases the credit count for that L2CAP channel by one. The peer device may return credits for an L2CAP channel at any time by sending an L2CAP_FLOW_CONTROL_CREDIT_IND packet. When a credit packet is received by a device it shall increment the credit count for that L2CAP channel by the value of the Credits field in this packet. The number of credits returned for an L2CAP channel may exceed the initial credits provided in the L2CAP_LE_CREDIT_BASED_CONNECTION_REQ or L2CAP_LE_CREDIT_BASED_CONNECTION_RSP packet. The device sending the L2CAP_FLOW_CONTROL_CREDIT_IND packet shall ensure that the number of credits returned for an L2CAP channel does not cause the credit count to exceed 65535. The device receiving the credit packet shall disconnect the L2CAP channel if the credit count exceeds 65535. The device shall also disconnect the L2CAP channel if it receives a K-frame on an L2CAP channel from the peer device that has a credit count of zero. If a device receives an L2CAP_FLOW_CONTROL_CREDIT_IND packet with credit value set to zero, the packet shall be ignored. A device shall not send credit values of zero in L2CAP_FLOW_CONTROL_CREDIT_IND packets.

If an L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet is received and there is insufficient authentication between the two devices, the connection shall be rejected with a result value of “Connection refused - insufficient authentication”. If an L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet is received and there is insufficient authorization between the two devices, the connection shall be rejected with a result value of “Connection refused - insufficient authorization”. If an L2CAP_LE_CREDIT_BASED_CONNECTION_REQ packet is received and the

encryption key size is too short, the connection shall be rejected with a result value of “Connection refused – encryption key size too short[1]”.

Note: When encryption is not enabled, the result value “Connection refused – insufficient authentication” does not indicate that MITM protection is required.

## 10.2 Enhanced Credit Based Flow Control Mode

Enhanced Credit Based Flow Control mode is used for L2CAP connection-oriented channels on LE and BR/EDR with flow control using a credit-based scheme for L2CAP data (i.e. not signaling packets). The ACL logical transport shall have an infinite Automatic Flush Timeout.

The number of credits (K-frames) that can be received by a device on an L2CAP channel is determined during connection establishment. K-frames shall only be sent on an L2CAP channel if the device has a credit count greater than zero for that L2CAP channel. For each K-frame sent, the sending device shall decrease the credit count for that L2CAP channel by one. The peer device may return credits for an L2CAP channel at any time by sending an L2CAP_FLOW_CONTROL_CREDIT_IND packet. When a credit packet is received by a device, it shall increment the credit count for that L2CAP channel by the value of the Credits field in this packet. The number of credits returned for an L2CAP channel may exceed the initial credits provided in the L2CAP_CREDIT_BASED_CONNECTION_REQ or L2CAP_CREDIT_BASED_CONNECTION_RSP packet. The device sending the L2CAP_FLOW_CONTROL_CREDIT_IND packet shall ensure that the number of credits returned for an L2CAP channel does not cause the credit count to exceed 65535. The device receiving the credit packet shall disconnect the L2CAP channel if the credit count exceeds 65535. The device shall also disconnect the L2CAP channel if it receives a K-frame on an L2CAP channel from the peer device that has a credit count of zero. If a device receives an L2CAP_FLOW_CONTROL_CREDIT_IND packet with credit value set to zero, the packet shall be ignored. A device shall not send credit values of zero in L2CAP_FLOW_CONTROL_CREDIT_IND packets.

This paragraph applies if an L2CAP_CREDIT_BASED_CONNECTION_REQ packet is received on the LE transport. If there is insufficient authentication between the two devices, the connection shall be rejected with a result value of “All connections refused - insufficient authentication”. If there is insufficient authorization between the two devices, the connection shall be rejected with a result value of “All connections refused - insufficient authorization”. If the encryption key size is too short, the connection shall

[1]This was previously "Connection refused - insufficient encryption key size".

be rejected with a result value of “All connections refused - encryption key size too short[2]”.

Note: When encryption is not enabled, the result value “All connections refused - insufficient authentication” does not indicate that MITM protection is required.

See [Vol 3] Part C, Section 5.2.2 for the requirements for responding to a received L2CAP_CREDIT_BASED_CONNECTION_REQ packet.

[2]This was previously "All connections refused - insufficient encryption key size".

# Appendix A Configuration MSCs

The examples in this appendix describe a sample of the multiple possible configuration scenarios that might occur.

Figure A.1 illustrates the basic configuration process. In this example, the devices exchange MTU information. All other options are set to their default values.

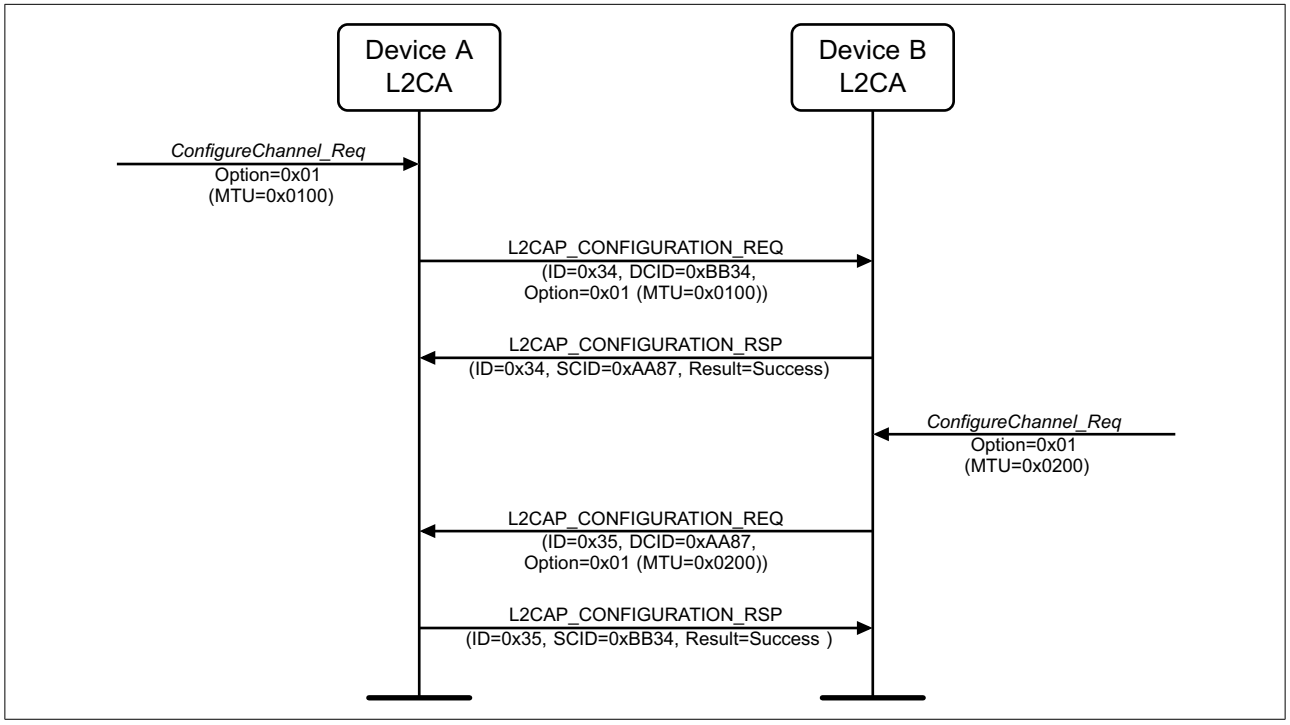



*Figure A.1: Basic MTU exchange*

Figure A.2 illustrates how two devices interoperate even though one device supports more options than the other does. Device A is an upgraded version. It uses a (hypothetical) option type 0x20. Device B rejects the command using the L2CAP_CONFIGURATION_RSP packet with result ‘unknown parameter’ informing Device A that option 0x20 is not understood. Device A then resends the request omitting option 0x20. As well as accepting this, Device B notices that it does not need such a large MTU and accepts the request but includes in the response the MTU option informing Device A that Device B will not send an L2CAP packet with a payload larger than 0x80 octets over this channel. On receipt of the response, Device A could reduce the buffer allocated to hold incoming traffic.

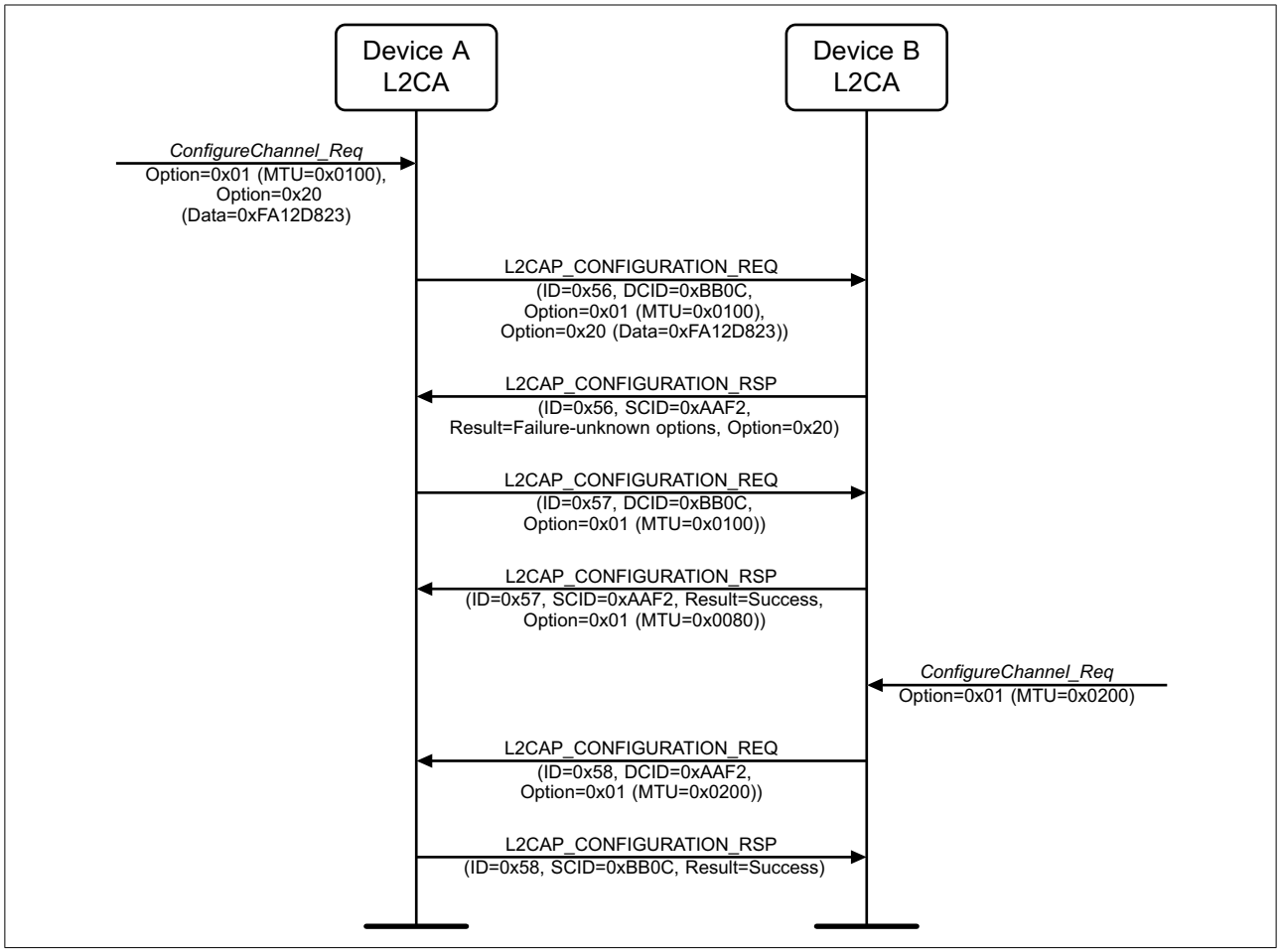



*Figure A.2: Dealing with unknown options*

Figure A.3 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is that Device A has placed the configuration request in an L2CAP packet that is too big to be accepted by Device B. Device B informs Device A of this problem using the L2CAP_COMMAND_REJECT_RSP message. Device A then resends the configuration options using two smaller L2CAP_CONFIGURATION_REQ messages. This exposes the second problem, which is that Device B has no open connection on CID 0x6A67. Again, Device B informs Device A of this problem using the L2CAP_COMMAND_REJECT_RSP message.

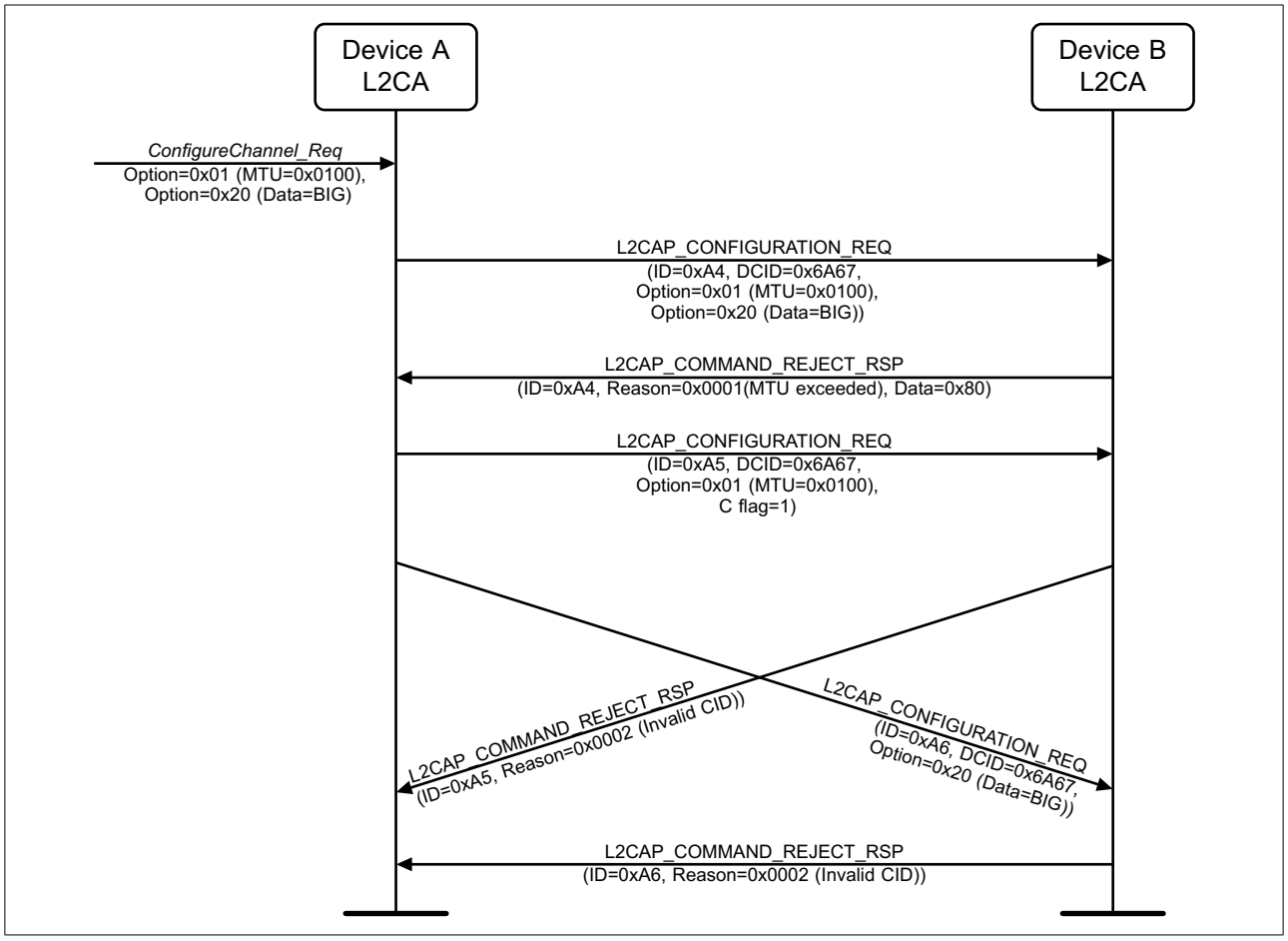



*Figure A.3: Unsuccessful configuration request*

# Appendix B Changes to signaling packet names

Previous versions of this specification used different names for the signalling packets defined in Section 4. Table B.1 shows the previous and current names of these packets.

| Previous name | Current name |
|---|---|
| Command reject | L2CAP_COMMAND_REJECT_RSP |
| Configuration request | L2CAP_CONFIGURATION_REQ |
| Configuration response | L2CAP_CONFIGURATION_RSP |
| Connection Parameter Update request | L2CAP_CONNECTION_PARAMETER_UPDATE_REQ |
| Connection Parameter Update response | L2CAP_CONNECTION_PARAMETER_UPDATE_RSP |
| Connection request | L2CAP_CONNECTION_REQ |
| Connection response | L2CAP_CONNECTION_RSP |
| Disconnection request | L2CAP_DISCONNECTION_REQ |
| Disconnection response | L2CAP_DISCONNECTION_RSP |
| Echo request | L2CAP_ECHO_REQ |
| Echo response | L2CAP_ECHO_RSP |
| Information request | L2CAP_INFORMATION_REQ |
| Information response | L2CAP_INFORMATION_RSP |
| LE Credit Based Connection request | L2CAP_LE_CREDIT_BASED_CONNECTION_REQ |
| LE Credit Based Connection response | L2CAP_LE_CREDIT_BASED_CONNECTION_RSP |
| LE Flow Control Credit | L2CAP_FLOW_CONTROL_CREDIT_IND |

*Table B.1: Changes to signaling packet names*

Host
Part B

# SERVICE DISCOVERY PROTOCOL (SDP) SPECIFICATION

*This Part defines a protocol for locating services provided by or available through a Bluetooth device.*

# CONTENTS

**1 Introduction ..... 1261**
1.1 General description ..... 1261
1.2 [This section is no longer used] ..... 1261
1.3 [This section is no longer used] ..... 1261
1.4 [This section is no longer used] ..... 1261
1.5 Conventions ..... 1261
1.5.1 Bit and byte ordering conventions ..... 1261

**2 Overview ..... 1262**
2.1 SDP Client-Server architecture ..... 1262
2.2 Service record ..... 1263
2.3 Service attribute ..... 1264
2.3.1 Attribute ID ..... 1265
2.3.2 Attribute value ..... 1265
2.4 Service class ..... 1265
2.4.1 A printer service class example ..... 1266
2.5 Searching for services ..... 1266
2.5.1 UUID ..... 1267
2.5.2 Service search patterns ..... 1267
2.6 Browsing for services ..... 1268
2.6.1 Example service browsing hierarchy ..... 1268

**3 Data representation ..... 1271**
3.1 Data element ..... 1271
3.2 Data element type descriptor ..... 1271
3.3 Data element size descriptor ..... 1272
3.4 Data element examples ..... 1273

**4 Protocol description ..... 1274**
4.1 Transfer byte order ..... 1274
4.2 Protocol Data Unit format ..... 1274
4.3 Partial responses and continuation state ..... 1275
4.4 Error handling ..... 1276
4.4.1 SDP_ERROR_RSP PDU ..... 1277
4.5 Service Search transaction ..... 1277
4.5.1 SDP_SERVICE_SEARCH_REQ PDU ..... 1278
4.5.2 SDP_SERVICE_SEARCH_RSP PDU ..... 1279
4.6 Service Attribute transaction ..... 1280
4.6.1 SDP_SERVICE_ATTR_REQ PDU ..... 1281
4.6.2 SDP_SERVICE_ATTR_RSP PDU ..... 1282

4.7 Service Search Attribute transaction ..... 1283
4.7.1 SDP_SERVICE_SEARCH_ATTR_REQ PDU ..... 1284
4.7.2 SDP_SERVICE_SEARCH_ATTR_RSP PDU ..... 1285

**5 Service attribute definitions ..... 1288**
5.1 Universal attribute definitions ..... 1288
5.1.1 ServiceRecordHandle attribute ..... 1288
5.1.2 ServiceClassIDList attribute ..... 1289
5.1.3 ServiceRecordState attribute ..... 1289
5.1.4 ServiceID attribute ..... 1289
5.1.5 ProtocolDescriptorList attribute ..... 1290
5.1.6 AdditionalProtocolDescriptorLists attribute ..... 1292
5.1.7 BrowseGroupList attribute ..... 1293
5.1.8 LanguageBaseAttributeIDList attribute ..... 1294
5.1.9 ServiceInfoTimeToLive attribute ..... 1295
5.1.10 ServiceAvailability attribute ..... 1295
5.1.11 BluetoothProfileDescriptorList attribute ..... 1296
5.1.12 DocumentationURL attribute ..... 1296
5.1.13 ClientExecutableURL attribute ..... 1296
5.1.14 IconURL attribute ..... 1297
5.1.15 ServiceName attribute ..... 1297
5.1.16 ServiceDescription attribute ..... 1298
5.1.17 ProviderName attribute ..... 1298
5.1.18 [This section is no longer used] ..... 1298
5.2 ServiceDiscoveryServer service class attribute definitions ..... 1298
5.2.1 ServiceRecordHandle attribute ..... 1298
5.2.2 ServiceClassIDList attribute ..... 1299
5.2.3 VersionNumberList attribute ..... 1299
5.2.4 ServiceDatabaseState attribute ..... 1299
5.2.5 [This section is no longer used] ..... 1300
5.3 BrowseGroupDescriptor service class attribute definitions ..... 1300
5.3.1 ServiceClassIDList attribute ..... 1300
5.3.2 GroupID attribute ..... 1300
5.3.3 [This section is no longer used] ..... 1300

**6 Security ..... 1301**

**Appendix A [This appendix is no longer used] ..... 1302**

**Appendix B Example SDP Transactions ..... 1303**
B.1 SDP example 1 – ServiceSearchRequest ..... 1303
B.2 SDP example 2 – ServiceAttributeTransaction ..... 1305
B.3 SDP example 3 – ServiceSearchAttributeTransaction ..... 1310

**Appendix C Changes to PDU names ..... 1321**

# 1 INTRODUCTION

## 1.1 General description

The Service Discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.

## 1.2 [This section is no longer used]

## 1.3 [This section is no longer used]

## 1.4 [This section is no longer used]

## 1.5 Conventions

### 1.5.1 Bit and byte ordering conventions

When multiple bit fields are contained in a single byte and represented in a drawing in this Part, the more significant (high-order) bits are shown toward the left and less significant (low-order) bits toward the right.

Multiple-byte fields are drawn with the more significant bytes toward the left and the less significant bytes toward the right. Multiple-byte fields are transferred in network byte order. See Section 4.1.

# 2 OVERVIEW

## 2.1 SDP Client-Server architecture

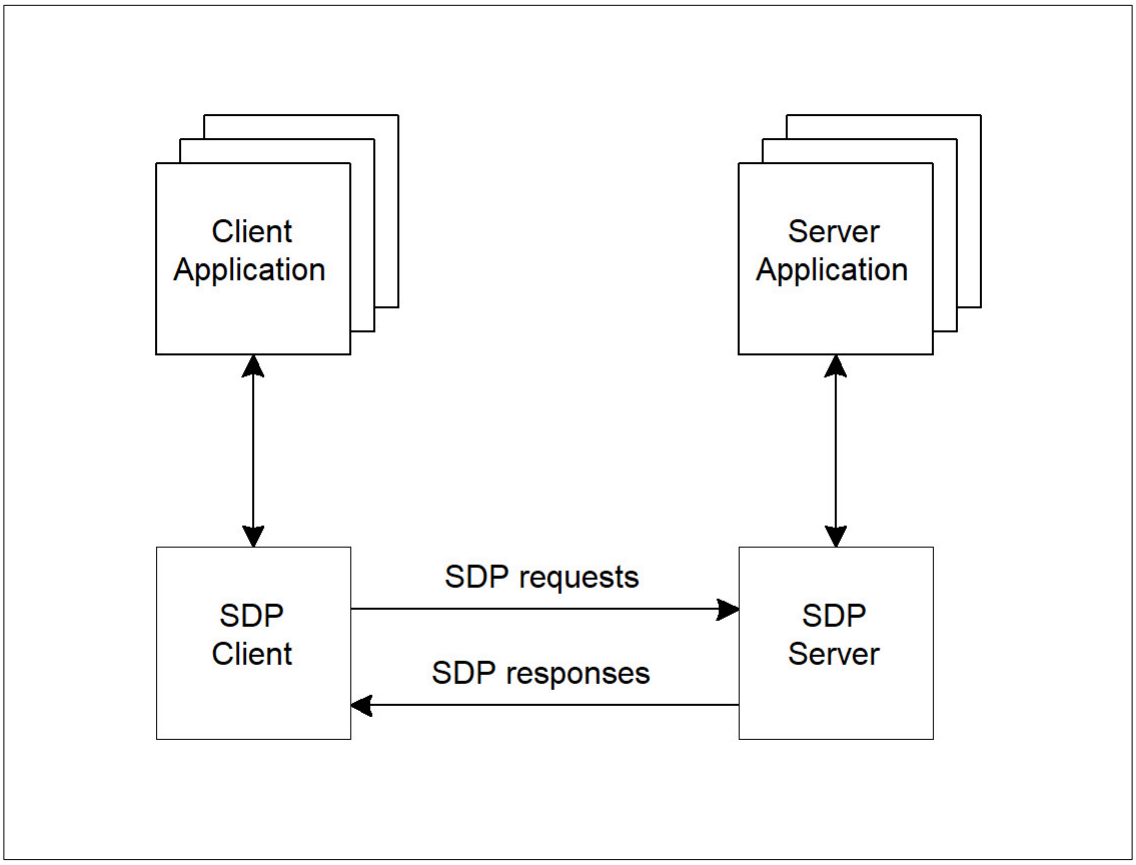



*Figure 2.1: SDP Client-Server interaction*

The service discovery mechanism provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to use the service.

From the perspective of the Service Discovery Protocol (SDP), the configuration shown in Figure 2.1 can be simplified to that shown in Figure 2.2.

*Figure 2.2: Simplified SDP Client-Server interaction*

SDP involves communication between an SDP Server and an SDP Client. The server maintains an SDP database which consists of a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client can retrieve information from a service record maintained by the SDP Server by issuing an SDP request.

If the client, or an application associated with the client, decides to use a service, it opens a separate connection to the service provider in order to use the service. SDP provides a mechanism for discovering services and their attributes (including associated service access protocols), but it does not provide a mechanism for making use of those services (such as delivering the service access protocols).

If multiple applications on a device provide services, an SDP Server can act on behalf of those service providers to handle requests for information about the services that they provide. Similarly, multiple client applications can use a single SDP Client to query servers on behalf of the client applications.

The set of SDP Servers that are available to an SDP Client will change dynamically based on the RF proximity of the servers to the client. When a server becomes available, a potential client must be notified by a means other than SDP so that the client can use SDP to query the server about its services. Similarly, when a server leaves proximity or becomes unavailable for any reason, there is no explicit notification via the Service Discovery protocol. However, the client can use SDP to poll the server and may infer that the server is not available if it no longer responds to requests.

Additional information regarding application interaction with SDP shall be contained in the Bluetooth Service Discovery Profile document.

## 2.2 Service record

A service is any entity that can provide information, perform an action, or control a resource on behalf of another entity. A service may be implemented as software, hardware, or a combination of hardware and software.

All of the information about a service that is maintained by an SDP Server is contained within a single service record. The service record shall only be a list of service attributes.

A service record handle is a 32-bit number that shall uniquely identify each service record within an SDP Server. In general, each handle is unique only within each SDP Server. If SDP Server S1 and SDP Server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will be meaningless if presented to S2.

The Service Discovery protocol does not provide a mechanism for notifying clients when service records are added to or removed from an SDP Server. While an L2CAP (Logical Link Control and Adaptation Protocol) connection is established to a server, a service record handle acquired from the server shall remain valid unless the service record it represents is removed. If a service is removed from the server, further requests to the server (during the L2CAP connection in which the service record handle was acquired) using the service's (now stale) record handle shall result in an error response indicating an invalid service record handle. An SDP Server shall ensure that no service record handle values are re-used while an L2CAP connection remains established. Service record handles shall remain valid across successive L2CAP connections while the ServiceDatabaseState attribute value remains unchanged. Further, service record handles should remain valid until such time that the corresponding service is permanently removed or changes in an incompatible way. See the ServiceRecordState and ServiceDatabaseState attributes in Section 5.

A device may have a service record with a service record handle of 0x00000000 representing the SDP Server itself. This service record contains attributes for the SDP Server and the protocol it supports. For example, one of its attributes is the list of SDP protocol versions supported by the server.

## 2.3 Service attribute

Each service attribute describes a single characteristic of a service. Some examples of service attributes are given in Table 2.1.

| | |
|---|---|
| ServiceClassIDList | Identifies the type of service represented by a service record. In other words, the list of classes of which the service is an instance |
| ServiceID | Uniquely identifies a specific instance of a service |
| ProtocolDescriptorList | Specifies the protocol stack(s) that may be used when using a service |
| ProviderName | The textual name of the individual or organization that provides a service |
| IconURL | Specifies a URL that refers to an icon image that may be used to represent a service |

| | |
|---|---|
| ServiceName | A text string containing a human-readable name for the service |
| ServiceDescription | A text string describing the service |

*Table 2.1: Example service attributes*

See Section 5.1 for attribute definitions that are common to all service records. Service providers can also define their own service attributes.

A service attribute consists of two components: an attribute ID and an attribute value.

### 2.3.1 Attribute ID

An attribute ID is a 16-bit unsigned integer that distinguishes each service attribute from other service attributes within a service record. The attribute ID also identifies the semantics of the associated attribute value.

A service class definition specifies each of the attribute IDs for a service class and assigns a meaning to the attribute value associated with each attribute ID. Each attribute ID is defined to be unique only within each service class.

All services belonging to a given service class assign the same meaning to each particular attribute ID. See Section 2.4.

In the Service Discovery protocol, an attribute ID is represented as a data element. See Section 3.

### 2.3.2 Attribute value

The attribute value is a variable length field whose meaning is determined by the attribute ID associated with it and by the service class of the service record in which the attribute is contained. In the Service Discovery protocol, an attribute value is represented as a data element. (See Section 3.) Generally, any type of data element is permitted as an attribute value, subject to the constraints specified in the service class definition that assigns an attribute ID to the attribute and assigns a meaning to the attribute value. See Section 5, for attribute value examples.

## 2.4 Service class

Each service is an instance of a service class. The service class definition provides the definitions of all attributes contained in service records that represent instances of that class. Each attribute definition specifies the numeric value of the attribute ID, the intended use of the attribute value, and the format of the attribute value. A service record contains attributes that are specific to a service class as well as universal attributes that are common to all services.

Each service class is also assigned a unique identifier. This service class identifier is contained in the attribute value for the ServiceClassIDList attribute, and is represented

as a UUID (see Section 2.5.1). Since the format and meanings of many attributes in a service record are dependent on the service class of the service record, the ServiceClassIDList attribute is very important. Its value shall be examined or verified before any class-specific attributes are used. Since all of the attributes in a service record must conform to all of the service's classes, the service class identifiers contained in the ServiceClassIDList attribute are related. Typically, each service class is a subclass of another class whose identifier is contained in the list. A service subclass definition differs from its superclass in that the subclass contains additional attribute definitions that are specific to the subclass.

When a new service class is defined that is a subclass of an existing service class, the new service class retains all of the attributes defined in its superclass. Additional attributes may be defined that are specific to the new service class. In other words, the mechanism for adding new attributes to some of the instances of an existing service class is to create a new service class that is a subclass of the existing service class.

If a Service Class UUID is exposed in the SDP database of a product, then the product containing the SDP record shall comply with the specification which defines the service corresponding to the UUID.

### 2.4.1 A printer service class example

A color postscript printer with duplex capability might conform to 4 ServiceClass definitions and have a ServiceClassIDList with UUIDs (See Section 2.5.1.) representing the following ServiceClasses:

DuplexColorPostscriptPrinterServiceClassID,
ColorPostscriptPrinterServiceClassID,
PostscriptPrinterServiceClassID,
PrinterServiceClassID

This example is only illustrative. This is not necessarily a practical printer class hierarchy.

## 2.5 Searching for services

The Service Search transaction allows a client to retrieve the service record handles for particular service records based on the values of attributes contained within those service records. Once an SDP Client has a service record handle, it can request the values of specific attributes.

The capability search for service records based on the values of arbitrary attributes is not provided. Rather, the capability is provided to search only for attributes whose

values are Universally Unique Identifiers[1] (UUIDs). Important attributes of services that can be used to search for a service are represented as UUIDs.

### 2.5.1 UUID

A UUID is a universally unique identifier that is expected to be unique across all space and all time (more precisely, the probability of independently-generated UUIDs being the same is negligible). UUIDs can be independently created in a distributed fashion. No central registry of assigned UUIDs is required. A UUID is a 128-bit value.

To reduce the burden of storing and transferring 128-bit UUID values, a range of UUID values has been pre-allocated for assignment to often-used, registered purposes. The first UUID in this pre-allocated range is known as the Bluetooth_Base_UUID and has the value 00000000-0000-1000-8000-00805F9B34FB. UUID values in the pre-allocated range have aliases that are represented as 16-bit or 32-bit values. These aliases are often called 16-bit and 32-bit UUIDs, but each actually represents a 128-bit UUID value.

The full 128-bit value of a 16-bit or 32-bit UUID may be computed by a simple arithmetic operation.

$128_bit_value = 16_bit_value \times 2^{96} + Bluetooth_Base_UUID$

$128_bit_value = 32_bit_value \times 2^{96} + Bluetooth_Base_UUID$

A 16-bit UUID may be converted to 32-bit UUID format by zero-extending the 16-bit value to 32-bits. An equivalent method is to add the 16-bit UUID value to a zero-valued 32-bit UUID.

Note: Two 16-bit UUIDs can be compared directly, as can two 32-bit UUIDs or two 128-bit UUIDs. If two UUIDs of differing sizes are to be compared, the shorter UUID must be converted to the longer UUID format before comparison.

### 2.5.2 Service search patterns

A service search pattern is a list of UUIDs used to locate matching service records. A service search pattern matches a service record if each and every UUID in the service search pattern is contained within any of the service record's attribute values. The UUIDs need not be contained within any specific attributes or in any particular order within the service record. The service search pattern matches if the UUIDs it contains constitute a subset of the UUIDs in the service record's attribute values. The only time a service search pattern does not match a service record is if the service search pattern contains at least one UUID that is not contained within the service record's attribute values. A valid service search pattern shall contain at least one UUID.

[1]The format of UUIDs is specified in ITU-T Rec. X.667(10/2012), alternatively known as ISO/IEC 9834-8:2014.

## 2.6 Browsing for services

Normally, a client searches for services based on some desired characteristic(s) (represented by a UUID) of the services. However, there are times when it is desirable to discover which types of services are described by an SDP Server's service records without any a priori information about the services. This process of looking for any offered services is termed browsing. In SDP, the mechanism for browsing for services is based on an attribute shared by all service classes. This attribute is called the BrowseGroupList attribute. The value of this attribute contains a list of UUIDs. Each UUID represents a browse group with which a service may be associated for the purpose of browsing.

When a client desires to browse an SDP Server's services, it creates a service search pattern containing the UUID that represents the root browse group. All services that may be browsed at the top level are made members of the root browse group by having the root browse group's UUID as a value within the BrowseGroupList attribute.

Normally, if an SDP Server has relatively few services, all of its services will be placed in the root browse group. However, the services offered by an SDP Server may be organized in a browse group hierarchy, by defining additional browse groups below the root browse group. Each of these additional browse groups is described by a service record with a service class of BrowseGroupDescriptor.

A browse group descriptor service record defines a new browse group by means of its Group ID attribute. In order for a service contained in one of these newly defined browse groups to be browseable, the browse group descriptor service record that defines the new browse group must in turn be browseable. The hierarchy of browseable services that is provided by the use of browse group descriptor service records allows the services contained in an SDP Server to be incrementally browsed and is particularly useful when the SDP Server contains many service records.

### 2.6.1 Example service browsing hierarchy

Figure 2.3 illustrates a fictitious service browsing hierarchy that illuminates the manner in which browse group descriptors are used. Browse group descriptor service records are identified with (G); other service records with (S).

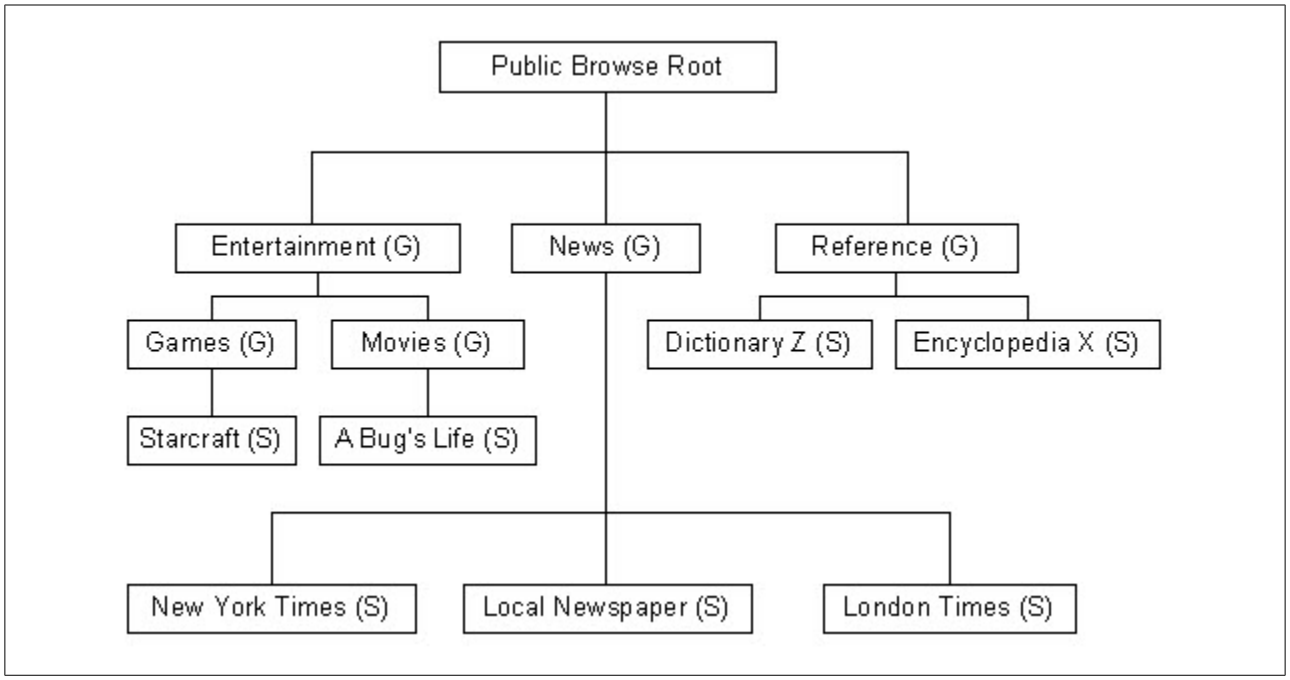



*Figure 2.3: Service browsing hierarchy*

Table 2.2 shows the services records and service attributes necessary to implement the browse hierarchy.

| Service Name | Service Class | Attribute Name | Attribute Value |
|---|---|---|---|
| Entertainment | BrowseGroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | EntertainmentID |
| News | BrowseGroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | NewsID |
| Reference | BrowseGroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | ReferenceID |
| Games | BrowseGroupDescriptor | BrowseGroupList | EntertainmentID |
| | | GroupID | GamesID |
| Movies | BrowseGroupDescriptor | BrowseGroupList | EntertainmentID |
| | | GroupID | MoviesID |
| Starcraft | Video Game Class ID | BrowseGroupList | GamesID |
| A Bug's Life | Movie Class ID | BrowseGroupList | MovieID |
| Dictionary Z | Dictionary Class ID | BrowseGroupList | ReferenceID |
| Encyclopedia X | Encyclopedia Class ID | BrowseGroupList | ReferenceID |
| New York Times | Newspaper ID | BrowseGroupList | NewspaperID |

| Service Name | Service Class | Attribute Name | Attribute Value |
|---|---|---|---|
| London Times | Newspaper ID | BrowseGroupList | NewspaperID |
| Local Newspaper | Newspaper ID | BrowseGroupList | NewspaperID |

*Table 2.2: Service browsing hierarchy*

# 3 DATA REPRESENTATION

Attribute values can contain information of various types with arbitrary complexity; thus enabling an attribute list to be generally useful across a wide variety of service classes and environments.

SDP defines a simple mechanism to describe the data contained within an attribute ID, attribute ID range, and attribute value. The primitive construct used is the data element.

## 3.1 Data element

A data element is a typed data representation. It consists of two fields: a header field and a data field. The header field, in turn, is composed of two parts: a type descriptor and a size descriptor. The data is a sequence of bytes whose length is specified in the size descriptor (described in Section 3.3) and whose meaning is (partially) specified by the type descriptor.

## 3.2 Data element type descriptor

A data element type is represented as a 5-bit type descriptor. The type descriptor is contained in the most significant (high-order) 5 bits of the first byte of the data element header. The following types have been defined.

| Type Descriptor Value | Valid Size Descriptor Values | Type Description |
|---|---|---|
| 0 | 0 | Nil, the null type |
| 1 | 0, 1, 2, 3, 4 | Unsigned integer |
| 2 | 0, 1, 2, 3, 4 | Signed integer |
| 3 | 1, 2, 4 | UUID, a universally unique identifier |
| 4 | 5, 6, 7 | Text string |
| 5 | 0 | Boolean (see below) |
| 6 | 5, 6, 7 | Data element sequence, a data element whose data field is a sequence of data elements |
| 7 | 5, 6, 7 | Data element alternative, data element whose data field is a sequence of data elements from which one data element is to be selected. |
| 8 | 5, 6, 7 | URL, a uniform resource locator |
| Other | | Reserved for future use |

*Table 3.1: Data element*

For the boolean type, false shall be represented by the value 0 and true shall be represented by the value 1. Any other value received shall be accepted as representing true.

## 3.3 Data element size descriptor

The data element size descriptor is represented as a 3-bit size index followed by 0, 8, 16, or 32 bits. The size index is contained in the least significant (low-order) 3 bits of the first byte of the data element header. The size index is encoded as follows.

| Size Index | Additional bits | Data Size |
|---|---|---|
| 0 | 0 | 1 byte. Exception: if the data element type is nil, the data size is 0 bytes. |
| 1 | 0 | 2 bytes |
| 2 | 0 | 4 bytes |
| 3 | 0 | 8 bytes |
| 4 | 0 | 16 bytes |
| 5 | 8 | The data size is contained in the additional 8 bits, which are interpreted as an unsigned integer. |
| 6 | 16 | The data size is contained in the additional 16 bits, which are interpreted as an unsigned integer. |
| 7 | 32 | The data size is contained in the additional 32 bits, which are interpreted as an unsigned integer. |

*Table 3.2: Data element size*

## 3.4 Data element examples

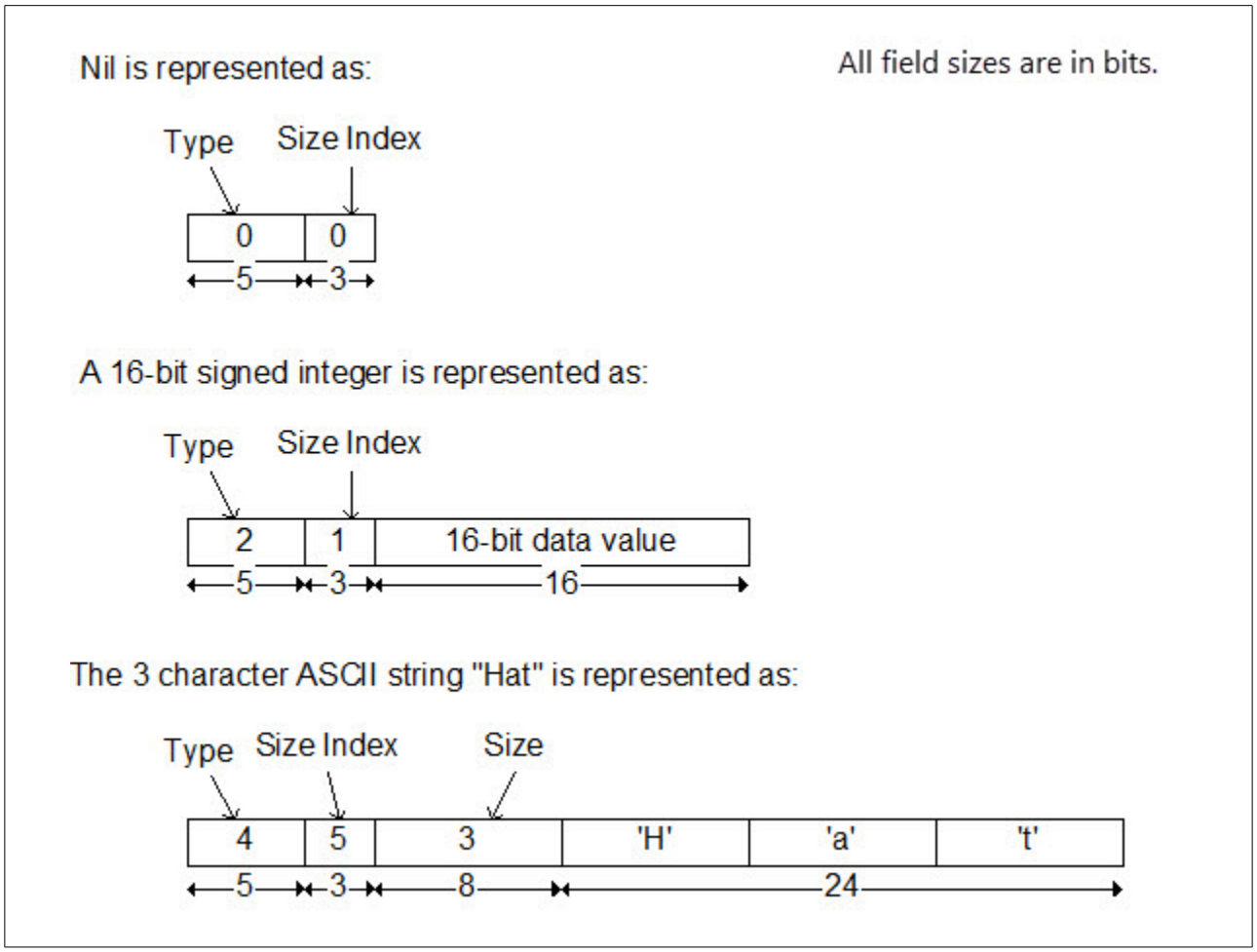



*Figure 3.1: Data element*

# 4 PROTOCOL DESCRIPTION

SDP is a simple protocol with minimal requirements on the underlying transport. It can function over a reliable packet transport (or even unreliable, if the client implements timeouts and repeats requests as necessary).

SDP uses a request/response model where each transaction consists of one request protocol data unit (PDU) and one response PDU. In the case where SDP is used with the Bluetooth L2CAP transport protocol, no more than one SDP request PDU per connection to a given SDP Server shall be outstanding at a given instant. In other words, a client shall wait for a response to its current request before issuing another request on the same L2CAP connection. Limiting SDP to sending one unacknowledged request PDU provides a simple form of flow control.

The protocol examples found in Appendix B could be helpful in understanding the protocol transactions.

## 4.1 Transfer byte order

The Service Discovery protocol shall transfer multiple-byte fields in standard network byte order (big-endian), with more significant (high-order) bytes being transferred before less-significant (low-order) bytes.

## 4.2 Protocol Data Unit format

Every SDP PDU consists of a PDU header followed by PDU-specific parameters. The header contains three fields: a PDU ID, a Transaction ID, and a ParameterLength. Each of these header fields is described here. Parameters may include a continuation state parameter, described below; PDU-specific parameters for each PDU type are described later in separate PDU descriptions.

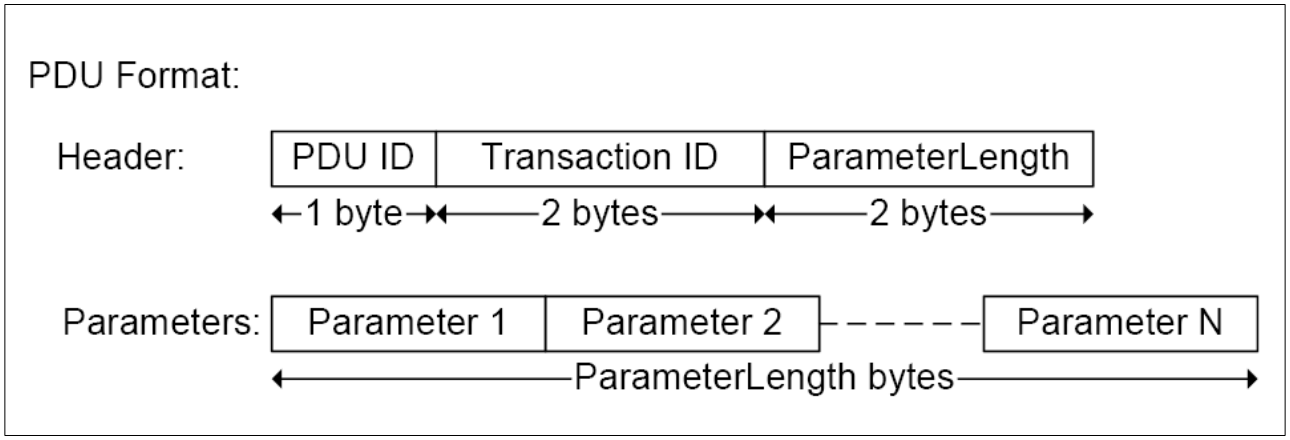



*Figure 4.1: Protocol Data Unit format*

*PDU ID:* *Size: 1 Byte*

| Value | Parameter Description |
|---|---|
| N | The PDU ID field identifies the type of PDU. I.e. its meaning and the specific parameters. |
| 0x01 | SDP_ERROR_RSP |
| 0x02 | SDP_SERVICE_SEARCH_REQ |
| 0x03 | SDP_SERVICE_SEARCH_RSP |
| 0x04 | SDP_SERVICE_ATTR_REQ |
| 0x05 | SDP_SERVICE_ATTR_RSP |
| 0x06 | SDP_SERVICE_SEARCH_ATTR_REQ |
| 0x07 | SDP_SERVICE_SEARCH_ATTR_RSP |
| All other values | Reserved for future use |

*TransactionID:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The TransactionID field uniquely identifies request PDUs and is used to match response PDUs to request PDUs. The SDP Client can choose any value for a request's TransactionID provided that it is different from all outstanding requests. The TransactionID value in response PDUs is required to be the same as the request that is being responded to.<br>Range: 0x0000 to 0xFFFF |

*ParameterLength:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The ParameterLength field specifies the length (in bytes) of all parameters contained in the PDU.<br>Range: 0x0000 to 0xFFFF |

## 4.3 Partial responses and continuation state

Some SDP requests may require responses that are larger than can fit in a single response PDU. In this case, the SDP Server shall generate a partial response along with a continuation state parameter. The continuation state parameter can be supplied by the client in a subsequent request to retrieve the next portion of the complete response. The continuation state parameter is a variable length field whose first byte contains the number of additional bytes of continuation information in the field (InfoLength).

The format of the continuation information is not standardized among SDP Servers. Each continuation state parameter is meaningful only to the SDP Server that generated

it. The SDP Server should not expose any sensitive information, such as internal data structures, in the continuation state parameter. The SDP Server shall validate a continuation state parameter supplied by the client before using it, including bounds checking.

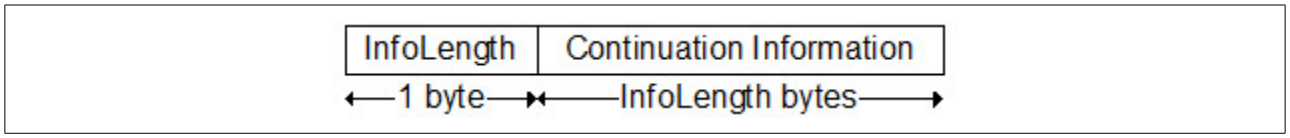



*Figure 4.2: Continuation state format*

After a client receives a partial response and the accompanying continuation state parameter, it can re-issue the original request (with a new transaction ID) and include the continuation state in the new request indicating to the server that the remainder of the original response is desired. The maximum allowable value of the InfoLength field is 16 (0x10).

Unless otherwise stated in a PDU response specification, an SDP Server may split a response at any arbitrary boundary when it generates a partial response. The SDP Server may select the boundary based on the contents of the reply, but is not required to do so. Therefore, length fields give the lengths as measured in the complete assembled record, not the length of the fields in the partial segment. When a service record is segmented into partial responses all attribute list values are relative to the complete record, not relevant to the partial record.

## 4.4 Error handling

Each transaction consists of a request and a response PDU. Generally, each type of request PDU has a corresponding type of response PDU. However, if the server determines that a request is improperly formatted or for any reason the server cannot respond with the appropriate PDU type, it shall respond with an SDP_ERROR_RSP PDU.

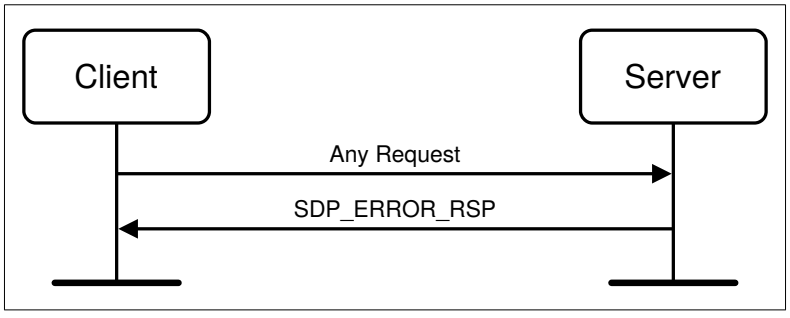



*Figure 4.3: Error handling*

### 4.4.1 SDP_ERROR_RSP PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ERROR_RSP | 0x01 | ErrorCode |

**Description:**

The SDP Server generates this PDU type in response to an improperly formatted request PDU or when the SDP Server, for whatever reason, cannot generate an appropriate response PDU.

**PDU parameters:**

*ErrorCode:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The ErrorCode identifies the reason that an SDP_ERROR_RSP PDU was generated. |
| 0x0001 | Invalid/unsupported SDP version |
| 0x0002 | Invalid Service Record Handle |
| 0x0003 | Invalid request syntax |
| 0x0004 | Invalid PDU Size |
| 0x0005 | Invalid Continuation State |
| 0x0006 | Insufficient Resources to satisfy Request |
| All other values | Reserved for future use |

Invalid PDU size should be used if an incoming request PDU's length is inconsistent with the specification of that request PDU or the length parameter of an incoming request PDU is inconsistent with that request PDU's actual contents.

## 4.5 Service Search transaction

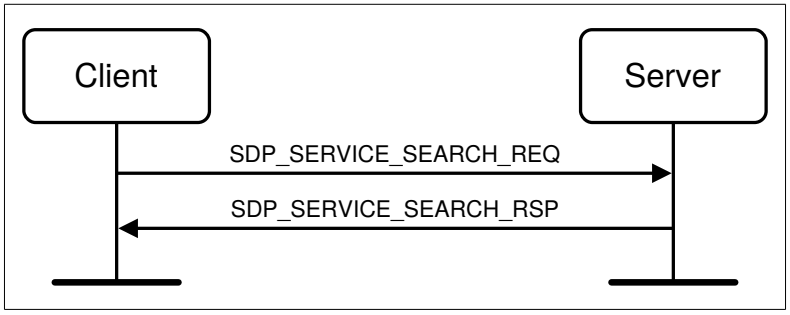



*Figure 4.4: Service Search transaction*

### 4.5.1 SDP_SERVICE_SEARCH_REQ PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_SEARCH_REQ | 0x02 | ServiceSearchPattern,<br>MaximumServiceRecordCount,<br>ContinuationState |

**Description:**

The SDP Client generates an SDP_SERVICE_SEARCH_REQ to locate service records that match the service search pattern given as the first parameter of the PDU. Upon receipt of this request, the SDP Server shall examine its service record data base and return an SDP_SERVICE_SEARCH_RSP containing the service record handles of service records within its SDP database that match the given service search pattern, or an appropriate error response.

No mechanism is provided to request information for all service records. However, see Section 2.6 for a description of a mechanism that permits browsing for non-specific services without a priori knowledge of the services.

**PDU parameters:**

*ServiceSearchPattern:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence shall contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern. |

[1]The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.

*MaximumServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumServiceRecordCount is a 16-bit count specifying the maximum number of service record handles to be returned in the response(s) to this request. The SDP Server shall not return more handles than this value specifies. If more than N service records match the request, the SDP Server determines which matching service record handles to return in the response(s).<br>Range: 0x0001 to 0xFFFF |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N is required to be less than or equal to 16. If no continuation state is to be provided in the request, N is set to 0. |

### 4.5.2 SDP_SERVICE_SEARCH_RSP PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_SEARCH_RSP | 0x03 | TotalServiceRecordCount,<br>CurrentServiceRecordCount,<br>ServiceRecordHandleList,<br>ContinuationState |

**Description:**

The SDP Server generates an SDP_SERVICE_SEARCH_RSP upon receipt of a valid SDP_SERVICE_SEARCH_REQ. The response contains a list of service record handles for service records that match the service search pattern given in the request. If a partial response is generated, it shall only contain complete service record handles; a service record handle value shall not be split across multiple PDUs.

**PDU parameters:**

*TotalServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The TotalServiceRecordCount is an integer containing the number of service records that match the requested service search pattern. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the MaximumServiceRecordCount value specified in the SDP_SERVICE_SEARCH_REQ. When multiple partial responses are used, each partial response contains the same value for TotalServiceRecordCount.<br>Range: 0x0000 to 0xFFFF |

*CurrentServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The CurrentServiceRecordCount is an integer indicating the number of service record handles that are contained in the next parameter. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the TotalServiceRecordCount value specified in the current response.<br>Range: 0x0000 to 0xFFFF |

*ServiceRecordHandleList:* *Size: (CurrentServiceRecordCount×4) Bytes*

| Value | Parameter Description |
|---|---|
| List of 32-bit handles | The ServiceRecordHandleList contains a list of service record handles. The number of handles in the list is given in the CurrentServiceRecordCount parameter. Each of the handles in the list refers to a service record that matches the requested service search pattern. This list of service record handles does not have the format of a data element. It contains no header fields, only the 32-bit service record handles. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is contained in the PDU, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response. |

## 4.6 Service Attribute transaction

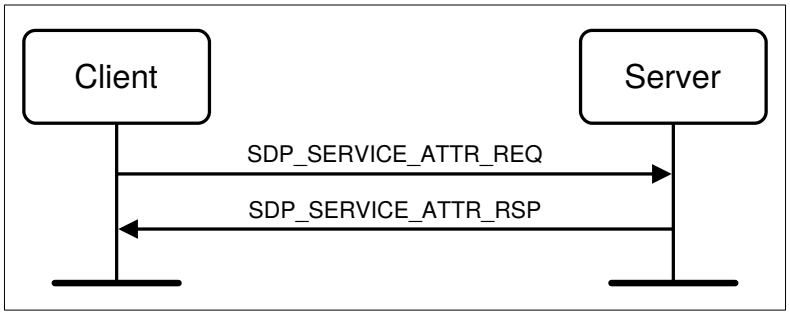



*Figure 4.5: Service Attribute transaction*

### 4.6.1 SDP_SERVICE_ATTR_REQ PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_ATTR_REQ | 0x04 | ServiceRecordHandle,<br>MaximumAttributeByteCount,<br>AttributeIDList,<br>ContinuationState |

**Description:**

The SDP Client generates an SDP_SERVICE_ATTR_REQ to retrieve specified attribute values from a specific service record. The service record handle of the desired service record and a list of desired attribute IDs to be retrieved from that service record are supplied as parameters.

**Command parameters:**

*ServiceRecordHandle:* *Size: 4 Bytes*

| Value | Parameter Description |
|---|---|
| 32-bit handle | The ServiceRecordHandle parameter specifies the service record from which attribute values are to be retrieved. The handle is obtained via a previous Service Search transaction. |

*MaximumAttributeByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP Server shall not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP Server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. In the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU.<br>Range: 0x0007 to 0xFFFF |

*AttributeIDList:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList shall be listed in ascending order without duplication of any attribute ID values. Note: All attributes can be requested by specifying the range 0x0000 to 0xFFFF. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N shall be less than or equal to 16. If no continuation state is to be provided in the request, N shall be set to 0. |

### 4.6.2 SDP_SERVICE_ATTR_RSP PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_ATTR_RSP | 0x05 | AttributeListByteCount,<br>AttributeList,<br>ContinuationState |

**Description:**

The SDP Server generates an SDP_SERVICE_ATTR_RSP upon receipt of a valid SDP_SERVICE_ATTR_REQ. The response contains a list of attributes (both attribute ID and attribute value) from the requested service record.

**PDU parameters:**

*AttributeListByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The AttributeListByteCount contains a count of the number of bytes in the AttributeList parameter. N shall never be larger than the MaximumAttributeByteCount value specified in the SDP_SERVICE_ATTR_REQ.<br>Range: 0x0002 to 0xFFFF |

*AttributeList:* *Size: AttributeListByteCount*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeList is a data element sequence containing attribute IDs and attribute values. The first element in the sequence contains the attribute ID of the first attribute to be returned. The second element in the sequence contains the corresponding attribute value. Successive pairs of elements in the list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_SERVICE_ATTR_REQ are contained in the AttributeList. Neither an attribute ID nor an attribute value is placed in the AttributeList for attributes in the service record that have no value. The attributes are listed in ascending order of attribute ID value. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response. |

If a partial response is generated, the response may be arbitrarily segmented into multiple PDUs (subject to the constraint imposed by the allowed range of values for the AttributeListByteCount parameter). Attributes in partial responses are not required to be completely within a single PDU.

## 4.7 Service Search Attribute transaction

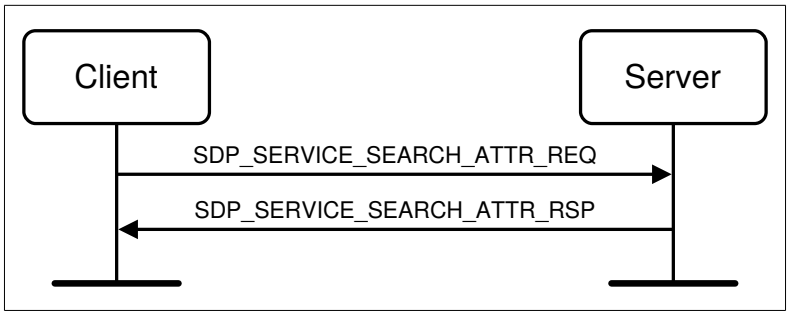



*Figure 4.6: Service Search Attribute transaction*

### 4.7.1 SDP_SERVICE_SEARCH_ATTR_REQ PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_SEARCH_ATTR_REQ | 0x06 | ServiceSearchPattern,<br>MaximumAttributeByteCount,<br>AttributeIDList,<br>ContinuationState |

**Description:**

The SDP_SERVICE_SEARCH_ATTR_REQ transaction combines the capabilities of the SDP_SERVICE_SEARCH_REQ and the SDP_SERVICE_ATTR_REQ into a single request. As parameters, it contains both a service search pattern and a list of attributes to be retrieved from service records that match the service search pattern. The SDP_SERVICE_SEARCH_ATTR_REQ and its response are more complex and can require more bytes than separate Service Search and Service Attribute transactions. However, using SDP_SERVICE_SEARCH_ATTR_REQ can reduce the total number of SDP transactions, particularly when retrieving multiple service records.

The service record handle for each service record is contained in the ServiceRecordHandle attribute of that service and may be requested along with other attributes.

**PDU parameters:**

*ServiceSearchPattern:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence shall contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern. |

[1]The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.

*MaximumAttributeByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP Server shall not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP Server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. In the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU.<br>Range: 0x0007 to 0xFFFF |

*AttributeIDList:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList shall be listed in ascending order without duplication of any attribute ID values. Note: All attributes may be requested by specifying the range 0x0000 to 0xFFFF. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N shall be less than or equal to 16. If no continuation state is to be provided in the request, N shall set to 0. |

### 4.7.2 SDP_SERVICE_SEARCH_ATTR_RSP PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_SERVICE_SEARCH_ATTR_RSP | 0x07 | AttributeListsByteCount,<br>AttributeLists,<br>ContinuationState |

**Description:**

The SDP Server generates an SDP_SERVICE_SEARCH_ATTR_RSP upon receipt of a valid SDP_SERVICE_SEARCH_ATTR_REQ. The response contains a list of attributes (both attribute ID and attribute value) from the service records that match the requested service search pattern.

**PDU parameters:**

*AttributeListsByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The AttributeListsByteCount contains a count of the number of bytes in the AttributeLists parameter. N shall never be larger than the MaximumAttribute-ByteCount value specified in the SDP_SERVICE_SEARCH_ATTR_REQ.<br>Range: 0x0002 to 0xFFFF |

*AttributeLists:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Se-quence | The AttributeLists is a data element sequence where each element in turn is a data element sequence representing an attribute list. Each attribute list contains attribute IDs and attribute values from one service record. The first element in each attribute list contains the attribute ID of the first attribute to be returned for that service record. The second element in each attribute list contains the corresponding attribute value. Successive pairs of elements in each attribute list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_SERVICE_SEARCH_ATTR_REQ are contained in the AttributeLists. Neither an attribute ID nor attribute value is placed in AttributeLists for attributes in the service record that have no value. Within each attribute list, the attributes are listed in ascending order of attribute ID value. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the Continuation-State parameter may be supplied in a subsequent request to retrieve the remainder of the response. |

If a partial response is generated, the response may be arbitrarily segmented into multiple PDUs (subject to the constraint imposed by the allowed range of values for the

AttributeListByteCount parameter). Attributes in partial responses are not required to be completely within a single PDU.

# 5 SERVICE ATTRIBUTE DEFINITIONS

The service classes and attributes contained in this document are necessarily a partial list of the service classes and attributes supported by SDP. Only service classes that directly support the SDP Server are included in this document. Additional service classes will be defined in other documents and possibly in future revisions of this document. Also, it is expected that additional attributes will be discovered that are applicable to a broad set of services; these may be added to the list of Universal attributes in future revisions of this document.

Where a range of values is specified for a class of attribute IDs, only numbers in that range are assigned numbers for that class and (except for universal attributes) are only assigned numbers for that class.

## 5.1 Universal attribute definitions

Universal attributes are those service attributes whose definitions are common to all service records. This does not mean that every service record contains values for all of these service attributes. However, if a service record has a service attribute with an attribute ID allocated to a universal attribute, the attribute value shall conform to the universal attribute's definition.

These attributes may be used with any service class.

Universal attributes use attribute IDs 0x0000 to 0x00FF. Specific IDs in this range are defined in Assigned Numbers.

Only two attributes are required to exist in every service record instance. They are the ServiceRecordHandle and the ServiceClassIDList. All other service attributes are optional within a service record.

### 5.1.1 ServiceRecordHandle attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceRecordHandle | 32-bit unsigned integer |

**Description:**

A service record handle is a 32-bit number that uniquely identifies each service record within an SDP Server. In general, each handle is unique only within each SDP Server. If SDP Server S1 and SDP Server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on

S1 will, in general, be meaningless if presented to S2. Service record handle values 0x00000001 to 0x0000FFFF are reserved for future use.

### 5.1.2 ServiceClassIDList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceClassIDList | Data Element Sequence |

**Description:**

The ServiceClassIDList attribute consists of a data element sequence in which each data element is a UUID representing the service classes that a given service record conforms to. The UUIDs should be listed in order from the most specific class to the most general class unless otherwise specified by the profile specifications defining the service classes. When profiles are enhanced, any new UUIDs should be added at the end of the ServiceClassIDList (after any existing UUIDs) to minimize interoperability issues with legacy implementations. The ServiceClassIDList shall contain at least one service class UUID.

### 5.1.3 ServiceRecordState attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceRecordState | 32-bit unsigned integer |

**Description:**

The ServiceRecordState is a 32-bit integer that is used to facilitate caching of Service Attributes. If this attribute is contained in a service record, its value shall be changed when any other attribute value is added to, deleted from or changed within the service record. This permits a client to check the value of this single attribute. If its value has not changed since it was last checked, the client knows that no other attribute values within the service record have changed.

### 5.1.4 ServiceID attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceID | UUID |

**Description:**

The ServiceID is a UUID that universally and uniquely identifies the service instance described by the service record. This service attribute is particularly useful if the same service is described by service records in more than one SDP Server.

### 5.1.5 ProtocolDescriptorList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ProtocolDescriptorList | Data Element Sequence or Data Element Alternative |

**Description:**

The ProtocolDescriptorList attribute describes one or more protocol stacks that can be used to gain access to the service described by the service record.

If the ProtocolDescriptorList describes a single stack, it takes the form of a data element sequence in which each element of the sequence is a protocol descriptor. Each protocol descriptor is, in turn, a data element sequence whose first element is a UUID identifying the protocol and whose successive elements are protocol-specific parameters. Potential protocol-specific parameters are a protocol version number and a connection-port number. The protocol descriptors are listed in order from the lowest layer protocol to the highest layer protocol used to gain access to the service.

If it is possible for more than one kind of protocol stack to be used to gain access to the service, the ProtocolDescriptorList takes the form of a data element alternative where each member is a data element sequence as described in the previous paragraph.

**Protocol descriptors**

A protocol descriptor identifies a communications protocol and provides protocol-specific parameters. A protocol descriptor is represented as a data element sequence. The first data element in the sequence shall be the UUID that identifies the protocol. Additional optional data elements provide protocol-specific information, such as the L2CAP protocol/service multiplexer (PSM) and the RFCOMM server channel number (CN) shown below.

**ProtocolDescriptorList examples**

These examples are intended to be illustrative. The parameter formats for each protocol are not defined within the specification.

In the examples in Table 5.1 and Table 5.2, it is assumed that a single RFCOMM instance exists on top of the L2CAP layer. In this case, the L2CAP protocol specific information (PSM) points to the single instance of RFCOMM.

*IrDA-like printer:*

| Item | Type | Value | Meaning |
|---|---|---|---|
| ProtocolDescriptorList | Data element sequence (3 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter 0 | uint8 | 0x03 | PSM = RFCOMM |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «RFCOMM» | |
| Parameter 0 | uint8 | 0x01 | CN = 1 |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UID | «PostscriptStream» | |

*Table 5.1: Example extract from an SDP record*

*IP Network Printing:*

| Item | Type | Value | Meaning |
|---|---|---|---|
| ProtocolDescriptorList | Data element sequence (6 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter | uint8 | 0x03 | PSM = RFCOMM |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «RFCOMM» | |
| Parameter | uint8 | 0x02 | CN = 2 |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «PPP» | |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «IP» | |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «TCP» | |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «IPP» | |

*Table 5.2: Example extract from an SDP record*

In the example in Table 5.3, two different and independent RFCOMM instances are available on top of the L2CAP layer. In this case, the L2CAP protocol specific

information (PSM) points to a distinct identifier that distinguishes each of the RFCOMM instances. See [Vol 3] Part A, Section 4.2 for the range of allowed PSM values.

| Item | Type | Value | Meaning |
|---|---|---|---|
| ProtocolDescriptorList | Data element alternative (2 items) | | |
| ProtocolDescriptorList | Data element sequence (4 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter | uint16 | 0x1001 | PSM |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «RFCOMM» | |
| Parameter | uint8 | 0x01 | CN = 1 |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «Obex» | |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «vCal» | |
| ProtocolDescriptorList | Data element sequence (4 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter | uint16 | 0x1003 | PSM |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «RFCOMM» | |
| Parameter | uint8 | 0x01 | CN = 1 |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «Obex» | |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «otherSynchronizationApplication» | |

*Table 5.3: Example extract from an SDP record with alternatives*

### 5.1.6 AdditionalProtocolDescriptorLists attribute

| Attribute Name | Attribute Value Type |
|---|---|
| AdditionalProtocolDescriptorLists | Data Element Sequence |

**Description:**

The AdditionalProtocolDescriptorLists attribute contains a sequence of ProtocolDescriptorList-elements. Each element having the same format as the

ProtocolDescriptorList described in Section 5.1.5. The ordering of the elements is significant and should be specified and fixed in Profiles that make use of this attribute.

The AdditionalProtocolDescriptorLists attribute supports services that require more channels in addition to the service described in Section 5.1.5. If the AdditionalProtocolDescriptorLists attribute is included in a service record, the ProtocolDescriptorList attribute shall be included.

**AdditionalProtocolDescriptorLists examples**

The following example in Table 5.4 is just illustrative and is not meant to refer to a real specified service or protocols.

| Item | Type | Value | Meaning |
|---|---|---|---|
| ProtocolDescriptorList | Data element sequence (2 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter | uint16 | 0x1233 | PSM = FooDataProtocol |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «FooDataProtocol» | |
| AdditionalProtocolDescriptorLists | Data element sequence (1 item) | | |
| ProtocolDescriptorList | Data element sequence (2 items) | | |
| ProtocolDescriptor | Data element sequence (2 items) | | |
| ProtocolID | UUID | «L2CAP» | |
| Parameter | uint16 | 0x24ab | PSM = FooControlProtocol |
| ProtocolDescriptor | Data element sequence (1 item) | | |
| ProtocolID | UUID | «FooControlProtocol» | |

*Table 5.4: Example extract from an SDP record containing additional protocol descriptor lists*

### 5.1.7 BrowseGroupList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| BrowseGroupList | Data Element Sequence |

**Description:**

The BrowseGroupList attribute consists of a data element sequence in which each element is a UUID that represents a browse group to which the service record belongs. The UUID for the top-level browse group ID, called PublicBrowseRoot and representing the root of the browsing hierarchy, is defined in Assigned Numbers.

### 5.1.8 LanguageBaseAttributeIDList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| LanguageBaseAttributeIDList | Data Element Sequence |

**Description:**

In order to support human-readable attributes for multiple natural languages in a single service record, a base attribute ID is assigned for each of the natural languages used in a service record. The human-readable universal attributes are then defined with an attribute ID offset from each of these base values, rather than with an absolute attribute ID.

The LanguageBaseAttributeIDList attribute is a list in which each member contains a language identifier, a character encoding identifier, and a base attribute ID for each of the natural languages used in the service record. The LanguageBaseAttributeIDList attribute consists of a data element sequence in which each element is a 16-bit unsigned integer. The elements are grouped as triplets (threes).

The first element of each triplet contains an identifier representing the natural language. The language is encoded according to ISO 639:1988 (E/F): “Code for the representation of names of languages”.

The second element of each triplet contains an identifier that specifies a character encoding used for the language. Values for character encoding can be found in IANA's database[1], and have the values that are referred to as MIBEnum values. The recommended character encoding is UTF-8.

The third element of each triplet contains an attribute ID that serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

The base attribute ID value shall be chosen so that all attributes that are specified as being offset from it (e.g. ServiceName, ServiceDescription, and ProviderName) shall have resulting Attribute IDs that are in the range 0x0100 to 0x01FF or a range specified in another specification as being used for this purpose. The range 0x0100 to 0x01FF shall only be used for this purpose.

To facilitate the retrieval of human-readable universal attributes in a principal language, the base attribute ID value for the primary language supported by a service record shall be 0x0100. Also, if a LanguageBaseAttributeIDList attribute is contained in a service record, the base attribute ID value contained in its first element shall be

[1]See https://www.iana.org/assignments/character-sets/character-sets.xhtml

0x0100. If one or more human readable attributes are contained in a service record, the LanguageBaseAttributeIDList attribute should be included in that service record.

### 5.1.9 ServiceInfoTimeToLive attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceInfoTimeToLive | 32-bit unsigned integer |

**Description:**

The ServiceTimeToLive attribute is a 32-bit integer that contains the number of seconds for which the information in a service record is expected to remain valid and unchanged. This time interval is measured from the time that the attribute value is retrieved from the SDP Server. This value does not imply a guarantee that the service record will remain available or unchanged. It is simply a hint that a client can use to determine a suitable polling interval to re-validate the service record contents.

### 5.1.10 ServiceAvailability attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceAvailability | 8-bit unsigned integer |

**Description:**

The ServiceAvailability attribute is an 8-bit unsigned integer that represents the relative ability of the service to accept additional clients. A value of 0xFF indicates that the service is not currently in use and is thus fully available, while a value of 0x00 means that the service is not accepting new clients. For services that support multiple simultaneous clients, intermediate values indicate the relative availability of the service on a linear scale.

For example, a service that can accept up to 3 clients should provide ServiceAvailability values of 0xFF, 0xAA, 0x55, and 0x00 when 0, 1, 2, and 3 clients, respectively, are using the service. The value 0xAA is $\frac{2}{3}$ × 0xFF and represents $\frac{2}{3}$ availability, while the value 0x55 is $\frac{1}{3}$ × 0xFF and represents $\frac{1}{3}$ availability. The availability value is be approximated as

( 1 - ( current_number_of_clients ÷ maximum_number_of_clients ) ) × 0xFF

When the maximum number of clients is large, this formula must be modified to ensure that ServiceAvailability values of 0x00 and 0xFF are reserved for their defined meanings of unavailability and full availability, respectively.

Note: The maximum number of clients a service can support can vary according to the resources being used by the service's current clients.

A non-zero value for ServiceAvailability does not guarantee that the service will be available for use. It should be treated as a hint or an approximation of availability status.

### 5.1.11 BluetoothProfileDescriptorList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| BluetoothProfileDescriptorList | Data Element Sequence |

**Description:**

The BluetoothProfileDescriptorList attribute consists of a data element sequence in which each element is a profile descriptor that contains information about a Bluetooth profile to which the service represented by this service record conforms. Each profile descriptor is a data element sequence whose first element is the UUID assigned to the profile and whose second element is a 16-bit profile version number.

Each version of a profile is assigned a 16-bit unsigned integer profile version number, which consists of two 8-bit fields. The higher-order 8 bits contain the major version number field and the lower-order 8 bits contain the minor version number field.

### 5.1.12 DocumentationURL attribute

| Attribute Name | Attribute Value Type |
|---|---|
| DocumentationURL | URL |

**Description:**

This attribute is a URL which points to documentation on the service described by a service record.

### 5.1.13 ClientExecutableURL attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ClientExecutableURL | URL |

**Description:**

This attribute contains a URL that refers to the location of an application that can make use of the service described by the service record. Since different operating environments require different executable formats, a mechanism has been defined to allow this single attribute to be used to locate an executable that is appropriate for the client device’s operating environment. In the attribute value URL, the first byte with the

value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired operating environment before the URL is to be used.

The list of standardized strings representing operating environments is contained in Assigned Numbers.

For example, assume that the value of the ClientExecutableURL attribute is http://my.fake/public/*/client.exe. On a device capable of executing SH3 WindowsCE files, this URL would be changed to http://my.fake/public/sh3-microsoft-wince/client.exe. On a device capable of executing Windows 98 binaries, this URL would be changed to http://my.fake/public/i86-microsoft-win98/client.exe.

### 5.1.14 IconURL attribute

| Attribute Name | Attribute Value Type |
|---|---|
| IconURL | URL |

**Description:**

This attribute contains a URL that refers to the location of an icon that can be used to represent the service described by the service record. Since different hardware devices require different icon formats, a mechanism has been defined to allow this single attribute to be used to locate an icon that is appropriate for the client device. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired icon format before the URL is to be used.

The list of standardized strings representing icon formats is contained in Assigned Numbers.

For example, assume that the value of the IconURL attribute is http://my.fake/public/icons/*. On a device that prefers 24 x 24 icons with 256 colors, this URL would be changed to http://my.fake/public/icons/24x24x8.png. On a device that prefers 10 x 10 monochrome icons, this URL would be changed to http://my.fake/public/icons/10x10x1.png.

### 5.1.15 ServiceName attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceName | String |

**Description:**

The ServiceName attribute is a string containing the name of the service represented by a service record. It should be brief and suitable for display with an Icon representing

the service. An offset specified in Assigned Numbers is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

### 5.1.16 ServiceDescription attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceDescription | String |

**Description:**

This attribute is a string containing a brief description of the service. It should be less than 200 characters in length. An offset specified in Assigned Numbers is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

### 5.1.17 ProviderName attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ProviderName | String |

**Description:**

This attribute is a string containing the name of the person or organization providing the service. An offset specified in Assigned Numbers is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

### 5.1.18 [This section is no longer used]

## 5.2 ServiceDiscoveryServer service class attribute definitions

This service class describes service record that contains attributes of the service discovery server itself. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the ServiceDiscoveryServerServiceClassID.

This service class uses attribute IDs 0x0200 to 0x02FF. Specific IDs in this range are defined in Assigned Numbers.

### 5.2.1 ServiceRecordHandle attribute

Described in the universal attribute definition for ServiceRecordHandle (Section 5.1.1).

**Value**

A 32-bit integer with the value 0x00000000.

### 5.2.2 ServiceClassIDList attribute

Described in the universal attribute definition for ServiceClassIDList (Section 5.1.2).

**Value**

A UUID representing the ServiceDiscoveryServerServiceClassID.

### 5.2.3 VersionNumberList attribute

| Attribute Name | Attribute Value Type |
|---|---|
| VersionNumberList | Data Element Sequence |

**Description:**

The VersionNumberList is a data element sequence in which each element of the sequence is a version number supported by the SDP Server.

A version number is a 16-bit unsigned integer consisting of two fields. The higher-order 8 bits contain the major version number field and the low-order 8 bits contain the minor version number field. The initial version of SDP has a major version of 1 and a minor version of 0.

### 5.2.4 ServiceDatabaseState attribute

| Attribute Name | Attribute Value Type |
|---|---|
| ServiceDatabaseState | 32-bit unsigned integer |

**Description:**

The ServiceDatabaseState is a 32-bit integer that is used to facilitate caching of service records. If this attribute exists, its value shall be changed when any of the other service records are added to or deleted from the server's SDP database. If this value has not changed since the last time a client queried its value, the client knows that a) none of the other service records maintained by the SDP Server have been added or deleted; and b) any service record handles acquired from the server are still valid. A client shall query this attribute's value when a connection to the server is established, prior to using any service record handles acquired during a previous connection.

The ServiceDatabaseState attribute does not change when existing service records are modified, including the addition, removal, or modification of service attributes. A service record's ServiceRecordState attribute indicates when that service record is modified.

### 5.2.5 [This section is no longer used]

## 5.3 BrowseGroupDescriptor service class attribute definitions

This service class describes the ServiceRecord provided for each BrowseGroupDescriptor service offered on a Bluetooth device. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the BrowseGroupDescriptorServiceClassID.

This service class uses attribute IDs 0x0200 to 0x02FF. Specific IDs in this range are defined in Assigned Numbers.

### 5.3.1 ServiceClassIDList attribute

Described in the universal attribute definition for ServiceClassIDList (Section 5.1.2).

**Value**

A UUID representing the BrowseGroupDescriptorServiceClassID.

### 5.3.2 GroupID attribute

| Attribute Name | Attribute Value Type |
|---|---|
| GroupID | UUID |

**Description:**

This attribute contains a UUID that can be used to locate services that are members of the browse group that this service record describes.

### 5.3.3 [This section is no longer used]

# 6 SECURITY

In Security Mode 4, SDP should use no security but may use security (an authenticated or unauthenticated link key with encryption). See [Vol 3] Part C, Section 5.2.2).

# Appendix A [This appendix is no longer used]

# Appendix B Example SDP Transactions

The following are simple examples of typical SDP transactions. These are meant to be illustrative of SDP flows. The examples do not consider:

- Caching (in a caching system, the SDP Client would make use of the ServiceRecordState and ServiceDatabaseState attributes);
- Service availability (if this is of interest, the SDP Client should use the ServiceAvailability and/or ServiceTimeToLive attributes);
- SDP versions (the VersionNumberList attribute could be used to determine compatible SDP versions);
- SDP Error Responses (an SDP error response is possible for any SDP request that is in error); and
- Communication connection (the examples assume that an L2CAP connection is established).

The examples are meant to be illustrative of the protocol. The format used is `ObjectName[ObjectSizeInBytes] {SubObjectDefinitions}`, but this is not meant to illustrate an interface. The `ObjectSizeInBytes` is the size of the object in decimal. The `SubObjectDefinitions` (inside of {} characters) are components of the immediately enclosing object. Hexadecimal values shown as lower-case letters, such as for transaction IDs and service handles, are variables (the particular value is not important for the illustration, but each such symbol always represents the same value). Comments are included in this manner: `/* comment text */`

## B.1 SDP example 1 – ServiceSearchRequest

The first example is that of an SDP Client searching for a generic printing service. The client does not specify a particular type of printing service. In the example, the SDP Server has two available printing services. The transaction illustrates:

1. SDP Client to SDP Server: SDP_SERVICE_SEARCH_REQ, specifying the PrinterServiceClassID (represented as a DataElement with a 32-bit UUID value of `ppp...ppp`) as the only element of the ServiceSearchPattern. The PrinterServiceClassID is assumed to be a 32-bit UUID and the data element type for it is illustrated. The TransactionID is illustrated as `tttt`.
2. SDP Server to SDP Client: SDP_SERVICE_SEARCH_RSP, returning handles to two printing services, represented as `qqqqqqqq` for the first printing service and `rrrrrrrr` for the second printing service. The Transaction ID is the same value as supplied by the SDP Client in the corresponding request (`tttt`).

```
  /* Sent from SDP Client to SDP Server */
SDP_SERVICE_SEARCH_REQ[15] {
  PDUID[1] {
    0x02
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000A
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* PrinterServiceClassID */
        0b00011 0b010 0xpppppppp
      }
    }
  }
  MaximumServiceRecordCount[2] {
    0x0003
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP Server to SDP Client */
SDP_SERVICE_SEARCH_RSP[18] {
  PDUID[1] {
    0x03
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000D
  }
```

```
  TotalServiceRecordCount[2] {
    0x0002
  }
  CurrentServiceRecordCount[2] {
    0x0002
  }
  ServiceRecordHandleList[8] {
    /* print service 1 handle */
    0xqqqqqqqq
    /* print service 2 handle */
    0xrrrrrrrr
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

## B.2 SDP example 2 – ServiceAttributeTransaction

The second example continues the first example. In Example 1, the SDP Client obtained handles to two printing services. In Example 2, the client uses one of those service handles to obtain the ProtocolDescriptorList attribute for that printing service. The transaction illustrates:

1. SDP Client to SDP Server: SDP_SERVICE_ATTR_REQ, presenting the previously obtained service handle (the one denoted as `qqqqqqqq`) and specifying the ProtocolDescriptorList attribute ID (AttributeID 0x0004) as the only attribute requested (other attributes could be retrieved in the same transaction if desired). The TransactionID is illustrated as `uuuu` to distinguish it from the TransactionID of Example 1.
2. SDP Server to SDP Client: SDP_SERVICE_ATTR_RSP, returning the ProtocolDescriptorList for the specified printing service. This protocol stack is assumed to be ( (L2CAP), (RFCOMM, 2), (PostscriptStream) ). The ProtocolDescriptorList is a data element sequence in which each element is, in turn, a data element sequence whose first element is a UUID representing the protocol, and whose subsequent elements are protocol-specific parameters. In this example, one such parameter is included for the RFCOMM protocol, an 8-bit value indicating RFCOMM server channel 2. The Transaction ID is the same value as supplied by the SDP Client in the corresponding request (`uuuu`). The Attributes returned are illustrated as a data element sequence where the protocol descriptors

are 32-bit UUIDs and the RFCOMM server channel is a data element with an 8-bit value of 2.

```
/* Sent from SDP Client to SDP Server */
SDP_SERVICE_ATTR_REQ[17] {
  PDUID[1] {
    0x04
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x000C
  }
  ServiceRecordHandle[4] {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2] {
    0x0080
  }
  AttributeIDList[5] {
    DataElementSequence[5] {
      0b00110 0b101 0x03
      AttributeID[3] {
        0b00001 0b001 0x0004
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP Server to SDP Client */
SDP_SERVICE_ATTR_RSP[38] {
  PDUID[1] {
    0x05
  }
  TransactionID[2] {
```

```
    0xuuuu
  }
  ParameterLength[2] {
    0x0021
  }
  AttributeListByteCount[2] {
    0x001E
  }
  AttributeList[30] {
    DataElementSequence[30] {
      0b00110 0b101 0x1C
      Attribute[28] {
        AttributeID[3] {
          0b00001 0b001 0x0004
        }
        AttributeValue[25] {
          /* ProtocolDescriptorList */
          DataElementSequence[25] {
            0b00110 0b101 0x17
            /* L2CAP protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* L2CAP Protocol UUID */
                0b00011 0b010 <32-bit L2CAP UUID>
              }
            }
            /* RFCOMM protocol descriptor */
            DataElementSequence[9] {
              0b00110 0b101 0x07
              UUID[5] {
                /* RFCOMM Protocol UUID */
                0b00011 0b010 <32-bit RFCOMM UUID>
              }
              /* parameter for server 2 */
              uint8[2] {
                0b00001 0b000 0x02
              }
            }
            /* PostscriptStream protocol descriptor */
            DataElementSequence[7] {
```

```
              0b00110 0b101 0x05
              UUID[5] {
                /* PostscriptStream Protocol UUID */
                0b00011 0b010 <32-bit PostscriptStream UUID>
              }
            }
          }
        }
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
SDP_SERVICE_ATTR_REQ[17] {
  PDUID[1] {
    0x04
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x000C
  }
  ServiceRecordHandle[4] {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2] {
    0x0080
  }
  AttributeIDList[5] {
    DataElementSequence[5] {
      0b00110 0b101 0x03
      AttributeID[3] {
        0b00001 0b001 0x0004
      }
    }
  }
  ContinuationState[1] {
```

```
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP Server to SDP Client */
SDP_SERVICE_ATTR_RSP[38] {
  PDUID[1] {
    0x05
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x0021
  }
  AttributeListByteCount[2] {
    0x001E
  }
  AttributeList[30] {
    DataElementSequence[30] {
      0b00110 0b101 0x1C
      Attribute[28] {
        AttributeID[3] {
          0b00001 0b001 0x0004
        }
        AttributeValue[25] {
          /* ProtocolDescriptorList */
          DataElementSequence[25] {
            0b00110 0b101 0x17
            /* L2CAP protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* L2CAP Protocol UUID */
                0b00011 0b010 <32-bit L2CAP UUID>
              }
            }
            /* RFCOMM protocol descriptor */
            DataElementSequence[9] {
              0b00110 0b101 0x07
```

```
            UUID[5] {
              /* RFCOMM Protocol UUID */
              0b00011 0b010 <32-bit RFCOMM UUID>
            }
            /* parameter for server 2 */
            uint8[2] {
              0b00001 0b000 0x02
            }
          }
          /* PostscriptStream protocol descriptor */
          DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
              /* PostscriptStream Protocol UUID */
              0b00011 0b010 <32-bit PostscriptStream UUID>
            }
          }
        }
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

## B.3 SDP example 3 – ServiceSearchAttributeTransaction

The third example is a form of service browsing, although it is not generic browsing in that it does not make use of SDP browse groups. Instead, an SDP Client is searching for available Synchronization services that can be presented to the user for selection. The SDP Client does not specify a particular type of synchronization service. In the example, the SDP Server has three available synchronization services: an address book synchronization service and a calendar synchronization service (both from the same provider), and a second calendar synchronization service from a different provider. The SDP Client is retrieving the same attributes for each of these services; namely, the data formats supported for the synchronization service (vCard, vCal, ICal, etc.) and those attributes that are relevant for presenting information to the user about the services. Also assume that the maximum size of a response is 400 bytes. Since the

result is larger than this, the SDP Client will repeat the request supplying a continuation state parameter to retrieve the remainder of the response. The transaction illustrates:

1. SDP Client to SDP Server: SDP_SERVICE_SEARCH_ATTR_REQ, specifying the generic SynchronisationServiceClassID (represented as a data element whose 32-bit UUID value is `sss...sss`) as the only element of the ServiceSearchPattern. The SynchronisationServiceClassID is assumed to be a 32-bit UUID. The requested attributes are the ServiceRecordHandle (Attribute ID 0x0000), ServiceClassIDList (AttributeID 0x0001), IconURL (AttributeID 0x000C), ServiceName (AttributeID 0x0100), ServiceDescription (AttributeID 0x0101), and ProviderName (AttributeID 0x0102) attributes; as well as the service-specific SupportedDataStores (AttributeID 0x0301). Since the first two attribute IDs (0x0000 and 0x0001) and three other attribute IDs (0x0100, 0x0101, and 0x0102) are consecutive, they are specified as attribute ranges. The TransactionID is illustrated as `vvvv` to distinguish it from the TransactionIDs of the other Examples.

   Values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

2. SDP Server to SDP Client: SDP_SERVICE_SEARCH_ATTR_RSP, returning the specified attributes for each of the three synchronization services. In the example, each ServiceClassIDList is assumed to contain a single element, the generic SynchronisationServiceClassID (a 32-bit UUID represented as sss...sss). Each of the other attributes contain illustrative data in the example (the strings have illustrative text; the icon URLs are illustrative, for each of the respective three synchronization services; and the SupportedDataStore attribute is represented as an unsigned 8-bit integer where 0x01 = vCard2.1, 0x02 = vCard3.0, 0x03 = vCal1.0 and 0x04 = iCal). One of the service records (the third for which data is returned) has no ServiceDescription attribute. The attributes are returned as a data element sequence, where each element is in turn a data element sequence representing a list of attributes. Within each attribute list, the ServiceClassIDList is a data element sequence while the remaining attributes are single data elements. The Transaction ID is the same value as supplied by the SDP Client in the corresponding request (0xvvvv). Since the entire result cannot be returned in a single response, a non-null continuation state is returned in this first response.

3. The total length of the initial data element sequence (487 in the example) is indicated in the first response, even though only a portion of this data element sequence (368 bytes in the example, as indicated in the AttributeLists byte count) is returned in the first response. The remainder of this data element sequence is returned in the second response (without an additional data element header).

4. SDP Client to SDP Server: SDP_SERVICE_SEARCH_ATTR_REQ, with the same parameters as in step 1, except that the continuation state received from the server

in step 2 is included as a request parameter. The TransactionID is changed to 0xwww to distinguish it from previous request.

5. SDP Server to SDP Client: SDP_SERVICE_SEARCH_ATTR_RSP, with the remainder of the result computed in step 2 above. Since all of the remaining result fits in this second response, a null continuation state is included.

```
/* Part 1 -- Sent from SDP Client to SDP Server */
SDP_SERVICE_SEARCH_ATTR_REQ[33] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x001C
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
      }
    }
  }
  MaximumAttributeByteCount[2] {
    0x0190
  }
AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
```

```
      AttributeID[3] {
        0b00001 0b001 0x0301
      }    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Part 2 -- Sent from SDP Server to SDP Client */
SDP_SERVICE_SEARCH_ATTR_RSP[384] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x017B
  }
  AttributeListByteCount[2] {
    0x0170
  }
  AttributeLists[368] {
    DataElementSequence[487] {
      0b00110 0b110 0x01E4
      DataElementSequence[178] {
        0b00110 0b101 0xB0
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
            /* service record handle */
            0b00001 0b010 0xhhhhhhhh
          }
        }
        Attribute[10] {
          AttributeID[3] {
            0b00001 0b001 0x0001
```

```
    }
    AttributeValue[7] {
      DataElementSequence[7] {
        0b00110 0b101 0x05
        UUID[5] {
          /* SynchronisationServiceClassID */
          0b00011 0b010 0xssssssss
        }
      }
    }
  }
  Attribute[35] {
    AttributeID[3] {
      0b00001 0b001 0x000C
    }
    AttributeValue[32] {
      /* IconURL; '*' replaced by client application */
      0b01000 0b101 0x1E
      "http://Synchronisation/icons/*"
    }
  }
  Attribute[22] {
    AttributeID[3] {
      0b00001 0b001 0x0100
    }
    AttributeValue[19] {
      /* service name */
      0b00100 0b101 0x11
      "Address Book Sync"
    }
  }
  Attribute[59] {
    AttributeID[3] {
      0b00001 0b001 0x0101
    }
    AttributeValue[56] {
      /* service description */
      0b00100 0b101 0x36
      "Synchronisation Service for vCard Address Book Entries"
    }
  }
```

```
    Attribute[37] {
      AttributeID[3] {
        0b00001 0b001 0x0102
      }
      AttributeValue[34] {
        /* service provider */
        0b00100 0b101 0x20
        "Synchronisation Specialists Inc."
      }
    }
    Attribute[5] {
      AttributeID[3] {
        0b00001 0b001 0x0301
      }
      AttributeValue[2] {
        /* Supported Data Store 'phonebook' */
        0b00001 0b000 0x01
      }
    }
  }
  DataElementSequence[175] {
    0b00110 0b101 0xAD
    Attribute[8] {
      AttributeID[3] {
        0b00001 0b001 0x0000
      }
      AttributeValue[5] {
        /* service record handle */
        0b00001 0b010 0xmmmmmmmm
      }
    }
    Attribute[10] {
      AttributeID[3] {
        0b00001 0b001 0x0001
      }
      AttributeValue[7] {
        DataElementSequence[7] {
          0b00110 0b101 0x05
          UUID[5] {
            /* SynchronisationServiceClassID */
            0b00011 0b010 0xssssssss
```

```
        }
      }
    }
  }
  Attribute[35] {
    AttributeID[3] {
      0b00001 0b001 0x000C
    }
    AttributeValue[32] {
      /* IconURL; '*' replaced by client application */
      0b01000 0b101 0x1E
      "http://Synchronisation/icons/*"
    }
  }
  Attribute[21] {
    AttributeID[3] {
      0b00001 0b001 0x0100
    }
    AttributeValue[18] {
      /* service name */
      0b00100 0b101 0x10
      "Appointment Sync"
    }
  }
  Attribute[57] {
    AttributeID[3] {
      0b00001 0b001 0x0101
    }
    AttributeValue[54] {
      /* service description */
      0b00100 0b101 0x34
      "Synchronisation Service for vCal Appointment Entries"
    }
  }
  Attribute[37] {
    AttributeID[3] {
      0b00001 0b001 0x0102
    }
    AttributeValue[34] {
      /* service provider */
      0b00100 0b101 0x20
```

```
            "Synchronisation Specialists Inc."
          }
        }
        Attribute[5] {
          AttributeID[3] {
            0b00001 0b001 0x0301
          }
          AttributeValue[2] {
            /* Supported Data Store 'calendar' */
            0b00001 0b000 0x03
          }
        }
      }
    /* } Data element sequence of attribute lists */
    /* is not completed in this PDU. */
  }
  ContinuationState[9] {
    /* 8 bytes of continuation state */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

/* Part 3 -- Sent from SDP Client to SDP Server */
SDP_SERVICE_SEARCH_ATTR_REQ[41] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x0024
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
      }
    }
```

```
  }
  MaximumAttributeByteCount[2] {
    0x0180
  }
    AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
      AttributeID[3] {
        0b00001 0b001 0x0301
      }
    }
  }
  ContinuationState[9] {
    /* same 8 bytes of continuation state */
    /* received in part 2 */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

Part 4 -- Sent from SDP Server to SDP Client

SDP_SERVICE_SEARCH_ATTR_RSP[115] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x006E
  }
  AttributeListByteCount[2] {
```

```
    0x006B
  }
  AttributeLists[107] {
    /* Continuing the data element sequence of */
    /* attribute lists begun in Part 2. */
      DataElementSequence[107] {
        0b00110 0b101 0x69
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
            /* service record handle */
            0b00001 0b010 0xFFFFFFFF
          }
        }
        Attribute[10] {
          AttributeID[3] {
            0b00001 0b001 0x0001
          }
          AttributeValue[7] {
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* SynchronisationServiceClassID */
                0b00011 0b010 0xssssssss
              }
            }
          }
        }
        Attribute[35] {
          AttributeID[3] {
            0b00001 0b001 0x000C
          }
          AttributeValue[32] {
            /* IconURL; '*' replaced by client application */
            0b01000 0b101 0x1E
            "http://DevManufacturer/icons/*"
          }
        }
        Attribute[18] {
```

```
          AttributeID[3] {
            0b00001 0b001 0x0100
          }
          AttributeValue[15] {
            /* service name */
            0b00100 0b101 0x0D
            "Calendar Sync"
          }
        }
        Attribute[29] {
          AttributeID[3] {
            0b00001 0b001 0x0102
          }
          AttributeValue[26] {
            /* service provider */
            0b00100 0b101 0x18
            "Device Manufacturer Inc."
          }
        }
        Attribute[5] {
          AttributeID[3] {
            0b00001 0b001 0x0301
          }
          AttributeValue[2] {
            /* Supported Data Store 'calendar' */
            0b00001 0b000 0x03
          }
        }
      }
    /* This completes the data element sequence */
    /* of attribute lists begun in Part 2. */
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

# Appendix C Changes to PDU names

Previous versions of this specification used different names for the PDUs defined in Section 4. Table C.1 shows the previous and current names of these PDUs.

| Previous name | Current name |
|---|---|
| SDP_ErrorResponse | SDP_ERROR_RSP |
| SDP_ServiceAttributeRequest | SDP_SERVICE_ATTR_REQ |
| SDP_ServiceAttributeResponse | SDP_SERVICE_ATTR_RSP |
| SDP_ServiceSearchAttributeRequest | SDP_SERVICE_SEARCH_ATTR_REQ |
| SDP_ServiceSearchAttributeResponse | SDP_SERVICE_SEARCH_ATTR_RSP |
| SDP_ServiceSearchRequest | SDP_SERVICE_SEARCH_REQ |
| SDP_ServiceSearchResponse | SDP_SERVICE_SEARCH_RSP |

*Table C.1: Changes to PDU names*

Host
Part C

# GENERIC ACCESS PROFILE

*This profile defines the generic procedures related to discovery of Bluetooth devices (idle mode procedures) and link management aspects of connecting to Bluetooth devices (connecting mode procedures). It also defines procedures related to use of different security levels. In addition, this profile includes common format requirements for parameters accessible on the user interface level.*

# CONTENTS

**1 Foreword ..... 1333**
1.1 Scope ..... 1333
1.2 Symbols and conventions ..... 1334
1.2.1 [This section is no longer used] ..... 1334
1.2.2 Signaling diagram conventions ..... 1334
1.2.3 Notation for timers and counters ..... 1335
1.2.4 Notation for UUIDs ..... 1335
1.3 GAP requirements ..... 1335

**2 Profile overview ..... 1336**
2.1 Profile stack ..... 1336
2.2 Profile roles ..... 1336
2.2.1 Roles when operating over BR/EDR Physical Transport ..... 1336
2.2.2 Roles when operating over an LE physical transport 1337
2.2.2.1 Broadcaster role ..... 1339
2.2.2.2 Observer role ..... 1339
2.2.2.3 Peripheral role ..... 1340
2.2.2.4 Central role ..... 1340
2.2.2.5 Concurrent operation in multiple GAP roles ..... 1340
2.3 User requirements and scenarios ..... 1340
2.4 Profile fundamentals ..... 1340
2.5 [This section is no longer used] ..... 1341

**3 User interface aspects ..... 1342**
3.1 The user interface level ..... 1342
3.2 Representation of Bluetooth parameters ..... 1342
3.2.1 Bluetooth Device Address (BD_ADDR) ..... 1342
3.2.1.1 Definition ..... 1342
3.2.1.2 Term on user interface level ..... 1342
3.2.1.3 Representation ..... 1342
3.2.2 Bluetooth Device Name (the user-friendly name) ..... 1342
3.2.2.1 Definition ..... 1342
3.2.2.2 Term on user interface level ..... 1343
3.2.2.3 Representation ..... 1343
3.2.3 Bluetooth Passkey (Bluetooth PIN) ..... 1343
3.2.3.1 Definition ..... 1343
3.2.3.2 Terms at user interface level ..... 1343
3.2.3.3 Representation ..... 1343

3.2.4 Class of Device ..... 1345
3.2.4.1 Definition ..... 1345
3.2.4.2 Term on user interface level ..... 1345
3.2.4.3 Representation ..... 1345
3.2.4.4 Usage ..... 1345
3.2.5 Appearance characteristic ..... 1346
3.2.5.1 Definition ..... 1346
3.2.5.2 Usage at user interface level ..... 1346
3.2.5.3 Representation ..... 1346
3.2.6 Broadcast Code ..... 1346
3.2.6.1 Definition ..... 1346
3.2.6.2 Terms at user interface level ..... 1346
3.2.6.3 Representation ..... 1346
3.3 Pairing ..... 1347

**4 Modes – BR/EDR physical transport ..... 1348**
4.1 Discoverability modes ..... 1348
4.1.1 Non-discoverable mode ..... 1349
4.1.1.1 Definition ..... 1349
4.1.1.2 Term on UI-level ..... 1349
4.1.2 Limited Discoverable mode ..... 1349
4.1.2.1 Definition ..... 1349
4.1.2.2 Conditions ..... 1350
4.1.2.3 Term on UI-level ..... 1350
4.1.3 General Discoverable mode ..... 1350
4.1.3.1 Definition ..... 1350
4.1.3.2 Conditions ..... 1350
4.1.3.3 Term on UI-level ..... 1351
4.2 Connectability modes ..... 1351
4.2.1 Non-connectable mode ..... 1351
4.2.1.1 Definition ..... 1351
4.2.1.2 Term on UI-level ..... 1351
4.2.2 Connectable mode ..... 1352
4.2.2.1 Definition ..... 1352
4.2.2.2 Term on UI-level ..... 1353
4.3 Bondable modes ..... 1353
4.3.1 Non-bondable mode ..... 1353
4.3.1.1 Definition ..... 1353
4.3.1.2 Term on UI-level ..... 1353
4.3.2 Bondable mode ..... 1353
4.3.2.1 Definition ..... 1353
4.3.2.2 Term on UI-level ..... 1354
4.4 Synchronizability modes ..... 1354

4.4.1 Non-synchronizable mode ..... 1354
4.4.1.1 Definition ..... 1354
4.4.1.2 Term on UI-level ..... 1354
4.4.2 Synchronizable mode ..... 1354
4.4.2.1 Definition ..... 1354
4.4.2.2 Term on UI-level ..... 1354

**5 Security aspects – BR/EDR physical transport ..... 1355**
5.1 Authentication ..... 1355
5.1.1 Purpose ..... 1355
5.1.2 Term on UI level ..... 1355
5.1.3 Procedure ..... 1356
5.1.4 Conditions ..... 1357
5.2 Security modes ..... 1357
5.2.1 Legacy security modes ..... 1358
5.2.1.1 Security mode 1 (non-secure) ..... 1358
5.2.1.2 Security mode 2 (service level enforced security) ..... 1358
5.2.1.3 Security mode 3 (link level enforced security) ..... 1358
5.2.2 Security mode 4 (service level enforced security) ..... 1359
5.2.2.1 Security for access to remote service (initiating side) ..... 1360
5.2.2.2 Security for access to local service by remote device (responding side) ..... 1363
5.2.2.3 Secure Simple Pairing after authentication failure ..... 1366
5.2.2.4 IO capabilities ..... 1367
5.2.2.5 Mapping of input / output capabilities to IO capability ..... 1368
5.2.2.6 IO and OOB capability mapping to authentication stage 1 method ..... 1368
5.2.2.7 Out of Band (OOB) ..... 1370
5.2.2.8 Security database ..... 1371

**6 Idle mode procedures – BR/EDR physical transport ..... 1376**
6.1 General Inquiry ..... 1376
6.1.1 Purpose ..... 1376
6.1.2 Term on UI level ..... 1377
6.1.3 Description ..... 1377
6.1.4 Conditions ..... 1377
6.2 Limited Inquiry ..... 1377
6.2.1 Purpose ..... 1377

6.2.2 Term on UI level ..... 1378
6.2.3 Description ..... 1378
6.2.4 Conditions ..... 1378
6.3 Name Discovery ..... 1378
6.3.1 Purpose ..... 1378
6.3.2 Term on UI level ..... 1379
6.3.3 Description ..... 1379
6.3.3.1 Name Request ..... 1379
6.3.3.2 Name Discovery ..... 1379
6.3.4 Conditions ..... 1380
6.4 Device Discovery ..... 1380
6.4.1 Purpose ..... 1380
6.4.2 Term on UI level ..... 1380
6.4.3 Description ..... 1380
6.4.4 Conditions ..... 1381
6.5 Bonding ..... 1381
6.5.1 Purpose ..... 1381
6.5.2 Term on UI level ..... 1381
6.5.3 Description ..... 1382
6.5.3.1 General Bonding ..... 1382
6.5.3.2 Dedicated Bonding ..... 1382
6.5.4 Conditions ..... 1384

**7 Establishment procedures – BR/EDR physical transport ..... 1385**
7.1 Link Establishment ..... 1385
7.1.1 Purpose ..... 1385
7.1.2 Term on UI level ..... 1385
7.1.3 Description ..... 1386
7.1.3.1 B in security mode 1, 2, or 4 ..... 1386
7.1.3.2 B in security mode 3 ..... 1387
7.1.4 Conditions ..... 1387
7.2 Channel Establishment ..... 1388
7.2.1 Purpose ..... 1388
7.2.2 Term on UI level ..... 1388
7.2.3 Description ..... 1388
7.2.3.1 B in security mode 2 or 4 ..... 1389
7.2.3.2 B in security mode 1 or 3 ..... 1389
7.2.4 Conditions ..... 1389
7.3 Connection Establishment ..... 1390
7.3.1 Purpose ..... 1390
7.3.2 Term on UI level ..... 1390
7.3.3 Description ..... 1390
7.3.3.1 B in security mode 2 or 4 ..... 1390

7.3.3.2 B in security mode 1 or 3 ..... 1391
7.3.4 Conditions ..... 1391
7.4 Establishment of additional connection ..... 1391
7.5 Synchronization Establishment ..... 1392
7.5.1 Purpose ..... 1392
7.5.2 Term on UI Level ..... 1392
7.5.3 Description ..... 1392
7.5.4 Conditions ..... 1392

**8 Extended inquiry response data format ..... 1394**

**9 Operational modes and procedures – LE physical transport ..... 1396**
9.1 Broadcast mode and Observation procedure ..... 1396
9.1.1 Broadcast mode ..... 1397
9.1.1.1 Definition ..... 1397
9.1.1.2 Conditions ..... 1397
9.1.2 Observation procedure ..... 1397
9.1.2.1 Definition ..... 1397
9.1.2.2 Conditions ..... 1397
9.2 Discovery modes and procedures ..... 1398
9.2.1 Requirements ..... 1398
9.2.2 Non-discoverable mode ..... 1398
9.2.2.1 Description ..... 1398
9.2.2.2 Conditions ..... 1399
9.2.3 Limited Discoverable mode ..... 1399
9.2.3.1 Description ..... 1399
9.2.3.2 Conditions ..... 1399
9.2.4 General Discoverable mode ..... 1400
9.2.4.1 Description ..... 1400
9.2.4.2 Conditions ..... 1401
9.2.5 Limited Discovery procedure ..... 1402
9.2.5.1 Description ..... 1402
9.2.5.2 Conditions ..... 1402
9.2.6 General Discovery procedure ..... 1403
9.2.6.1 Description ..... 1403
9.2.6.2 Conditions ..... 1404
9.2.7 Name Discovery procedure ..... 1405
9.2.7.1 Description ..... 1405
9.2.7.2 Conditions ..... 1405
9.3 Connection modes and procedures ..... 1405
9.3.1 Requirements ..... 1406
9.3.2 Non-connectable mode ..... 1407
9.3.2.1 Description ..... 1407

9.3.2.2 Conditions ..... 1407
9.3.3 Directed Connectable mode ..... 1407
9.3.3.1 Description ..... 1407
9.3.3.2 Conditions ..... 1407
9.3.4 Undirected Connectable mode ..... 1407
9.3.4.1 Description ..... 1407
9.3.4.2 Conditions ..... 1408
9.3.5 Auto Connection Establishment procedure ..... 1408
9.3.5.1 Description ..... 1408
9.3.5.2 Conditions ..... 1408
9.3.6 General Connection Establishment procedure ..... 1410
9.3.6.1 Description ..... 1410
9.3.6.2 Conditions ..... 1411
9.3.7 Selective Connection Establishment procedure ..... 1412
9.3.7.1 Description ..... 1412
9.3.7.2 Conditions ..... 1412
9.3.8 Direct Connection Establishment procedure ..... 1415
9.3.8.1 Description ..... 1415
9.3.8.2 Conditions ..... 1415
9.3.9 Connection Parameter Update procedure ..... 1416
9.3.9.1 Description ..... 1416
9.3.9.2 Conditions ..... 1416
9.3.10 Terminate Connection procedure ..... 1417
9.3.10.1 Description ..... 1417
9.3.10.2 Conditions ..... 1417
9.3.11 Connection Establishment Timing parameters ..... 1417
9.3.11.1 Description ..... 1417
9.3.11.2 Conditions ..... 1417
9.3.12 Connection interval timing parameters ..... 1418
9.3.12.1 Description ..... 1418
9.3.12.2 Conditions ..... 1419
9.3.13 Connected Isochronous Stream Central Establishment procedure ..... 1420
9.3.13.1 Description ..... 1420
9.3.13.2 Conditions ..... 1420
9.3.14 Connected Isochronous Stream Peripheral Establishment procedure ..... 1420
9.3.14.1 Description ..... 1420
9.3.14.2 Conditions ..... 1420
9.3.15 Connected Isochronous Stream Terminate procedure ..... 1420
9.3.15.1 Description ..... 1420
9.3.15.2 Conditions ..... 1420

9.3.16 Connection Subrate procedure ..... 1421
9.3.16.1 Description ..... 1421
9.3.16.2 Conditions ..... 1421
9.3.17 Periodic Advertising Connection procedure ..... 1421
9.3.17.1 Definition ..... 1421
9.3.17.2 Conditions ..... 1421
9.4 Bonding modes and procedures ..... 1421
9.4.1 Requirements ..... 1422
9.4.2 Non-bondable mode ..... 1422
9.4.2.1 Description ..... 1422
9.4.2.2 Conditions ..... 1422
9.4.3 Bondable mode ..... 1422
9.4.3.1 Description ..... 1422
9.4.3.2 Conditions ..... 1422
9.4.4 Bonding procedure ..... 1423
9.4.4.1 Description ..... 1423
9.4.4.2 Conditions ..... 1423
9.5 Periodic advertising modes and procedure ..... 1423
9.5.1 Periodic Advertising Synchronizability mode ..... 1424
9.5.1.1 Definition ..... 1424
9.5.1.2 Conditions ..... 1424
9.5.2 Periodic Advertising mode ..... 1424
9.5.2.1 Definition ..... 1424
9.5.2.2 Conditions ..... 1425
9.5.3 Periodic Advertising Synchronization Establishment procedure ..... 1425
9.5.3.1 Definition ..... 1425
9.5.3.2 Conditions ..... 1425
9.5.4 Periodic Advertising Synchronization Transfer procedure ..... 1425
9.5.4.1 Definition ..... 1425
9.5.4.2 Conditions ..... 1425
9.5.5 [This section is no longer used] ..... 1426
9.6 Isochronous Broadcast modes and procedures ..... 1426
9.6.1 Broadcast Isochronous Synchronizability mode ..... 1426
9.6.1.1 Definition ..... 1426
9.6.1.2 Conditions ..... 1426
9.6.2 Broadcast Isochronous Broadcasting mode ..... 1427
9.6.2.1 Definition ..... 1427
9.6.2.2 Conditions ..... 1427
9.6.3 Broadcast Isochronous Synchronization Establishment procedure ..... 1427
9.6.3.1 Definition ..... 1427

9.6.3.2 Conditions ..... 1427
9.6.4 Broadcast Isochronous Channel Map Update procedure ..... 1427
9.6.4.1 Definition ..... 1427
9.6.4.2 Conditions ..... 1427
9.6.5 Broadcast Isochronous Terminate procedure ..... 1427
9.6.5.1 Definition ..... 1427
9.6.5.2 Conditions ..... 1428
9.7 Channel Sounding procedures ..... 1428
9.7.1 Channel Sounding initiator procedure ..... 1428
9.7.1.1 Description ..... 1428
9.7.1.2 Conditions ..... 1428
9.7.2 Channel Sounding reflector procedure ..... 1428
9.7.2.1 Description ..... 1428
9.7.2.2 Conditions ..... 1428

**10 Security aspects – LE physical transport ..... 1430**
10.1 Requirements ..... 1430
10.2 LE security modes ..... 1430
10.2.1 LE security mode 1 ..... 1431
10.2.2 [This section is no longer used] ..... 1431
10.2.3 [This section is no longer used] ..... 1431
10.2.4 Secure Connections Only mode ..... 1431
10.2.5 LE security mode 3 ..... 1431
10.3 Authentication procedure ..... 1432
10.3.1 Responding to a service request ..... 1432
10.3.1.1 Handling of GATT indications and notifications ..... 1436
10.3.1.2 Cross-transport key generation ..... 1436
10.3.2 Initiating a service request ..... 1436
10.3.2.1 Cross-transport key generation ..... 1440
10.3.2.2 Handling of GATT indications and notifications ..... 1440
10.4 [This section is no longer used] ..... 1440
10.5 Authorization procedure ..... 1440
10.6 Encryption procedure ..... 1440
10.7 Privacy feature ..... 1441
10.7.1 Privacy feature in a Peripheral ..... 1442
10.7.1.1 Privacy feature in a Peripheral with Controller-based privacy ..... 1443
10.7.1.2 Privacy feature in a Peripheral with Host-based privacy ..... 1443
10.7.2 Privacy feature in a Central ..... 1443

10.7.2.1 Privacy feature in a Central with Controller-based privacy ..... 1444
10.7.2.2 Privacy feature in a Central with Host-based privacy ..... 1444
10.7.3 Privacy feature in a Broadcaster ..... 1444
10.7.4 Privacy feature in an Observer ..... 1445
10.8 Random Device address ..... 1445
10.8.1 Static address ..... 1446
10.8.2 Private address ..... 1446
10.8.2.1 Non-Resolvable Private Address Generation procedure ..... 1446
10.8.2.2 Resolvable Private Address Generation procedure ..... 1446
10.8.2.3 Resolvable Private Address Resolution procedure ..... 1446
10.9 Encrypted Broadcast Isochronous Group ..... 1446
10.10 Encrypted Advertising Data procedure ..... 1447
10.11 LE Channel Sounding ..... 1447
10.11.1 Channel Sounding security ..... 1447

**11 Advertising and Scan Response data format ..... 1449**

**12 GAP Service and characteristics for GATT Server ..... 1451**
12.1 Device Name characteristic ..... 1452
12.2 Appearance characteristic ..... 1452
12.3 Peripheral Preferred Connection Parameters characteristic ..... 1453
12.4 Central Address Resolution characteristic ..... 1454
12.5 Resolvable Private Address Only characteristic ..... 1454
12.6 Encrypted Data Key Material ..... 1455
12.7 LE GATT Security Levels Characteristic ..... 1456

**13 BR/EDR/LE operation ..... 1458**
13.1 Modes, procedures, and security aspects ..... 1458
13.1.1 Discoverable mode requirements ..... 1458
13.2 Bonding for BR/EDR/LE implementations ..... 1458
13.3 Relationship between physical transports ..... 1459

**14 BR/EDR/LE security aspects ..... 1460**
14.1 Cross-transport key derivation ..... 1460
14.2 Collision handling ..... 1461
14.3 Secure Connections Only Mode ..... 1461

**15 Bluetooth Device requirements ..... 1462**
15.1 Bluetooth Device address ..... 1462

15.1.1 Bluetooth Device Address types ..... 1462
15.1.1.1 Public Bluetooth address ..... 1462
15.1.1.2 Random Bluetooth address ..... 1462
15.2 GATT Profile requirements ..... 1462
15.3 SDP requirements ..... 1462
15.4 SDP service record requirement ..... 1463

**16 Definitions ..... 1465**
16.1 General definitions ..... 1465
16.2 Connection-related definitions ..... 1465
16.3 Device-related definitions ..... 1466
16.4 Procedure-related definitions ..... 1467
16.5 Security-related definitions ..... 1467

**17 References ..... 1469**

**Appendix A Timers and Constants ..... 1470**

**Appendix B Information Flows of Related Procedures ..... 1475**
B.1 LMP – authentication ..... 1475
B.2 LMP – pairing ..... 1476
B.3 Service Discovery ..... 1476
B.4 Generating a resolvable private address ..... 1477
B.5 Resolving a resolvable private address ..... 1477

# 1 FOREWORD

Interoperability between devices from different manufacturers is provided for a specific service and use case, if the devices conform to a Bluetooth SIG- defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives a description of the air interface for specified service(s) and use case(s).

Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.

## 1.1 Scope

The purpose of the Generic Access Profile is:

To introduce definitions, recommendations and common requirements related to modes and access procedures that are to be used by transport and application profiles.

To describe how devices are to behave in standby and connecting states in order to avoid situations where links and channels cannot be established between Bluetooth devices or that prevent multi-profile operation. Special focus is put on discovery, link establishment and security procedures.

To state requirements on user interface aspects, mainly coding schemes and names of procedures and parameters, that are needed to provide a satisfactory user experience.

This profile defines three implementation transport types based on the supported Core Configurations as defined in [Vol 0] Part D, Section 2. These implementation transport types are defined in Table 1.1:

| Implementation Transport Type | Description |
|---|---|
| BR/EDR-only | Implementations of a BR/EDR Core Configuration (see [Vol 0] Part D, Section 2.1.1, [Vol 0] Part D, Section 2.2.1, and [Vol 0] Part D, Section 2.3.1) |
| LE only | Implementations of an LE Core Configuration (see [Vol 0] Part D, Section 2.1.2, [Vol 0] Part D, Section 2.2.2, and [Vol 0] Part D, Section 2.3.2) |
| BR/EDR/LE | Implementations of a BR/EDR/LE Core Configuration (see [Vol 0] Part D, Section 2.1.3, [Vol 0] Part D, Section 2.2.3, and [Vol 0] Part D, Section 2.3.3) |

*Table 1.1: Implementation transport types*

The terms physical transport, physical link and physical channel as defined in [Vol 1] Part A, Section 3 are used in the specification.

## 1.2 Symbols and conventions

### 1.2.1 [This section is no longer used]

### 1.2.2 Signaling diagram conventions

The arrows shown in Figure 1.1 are used in diagrams describing procedures:

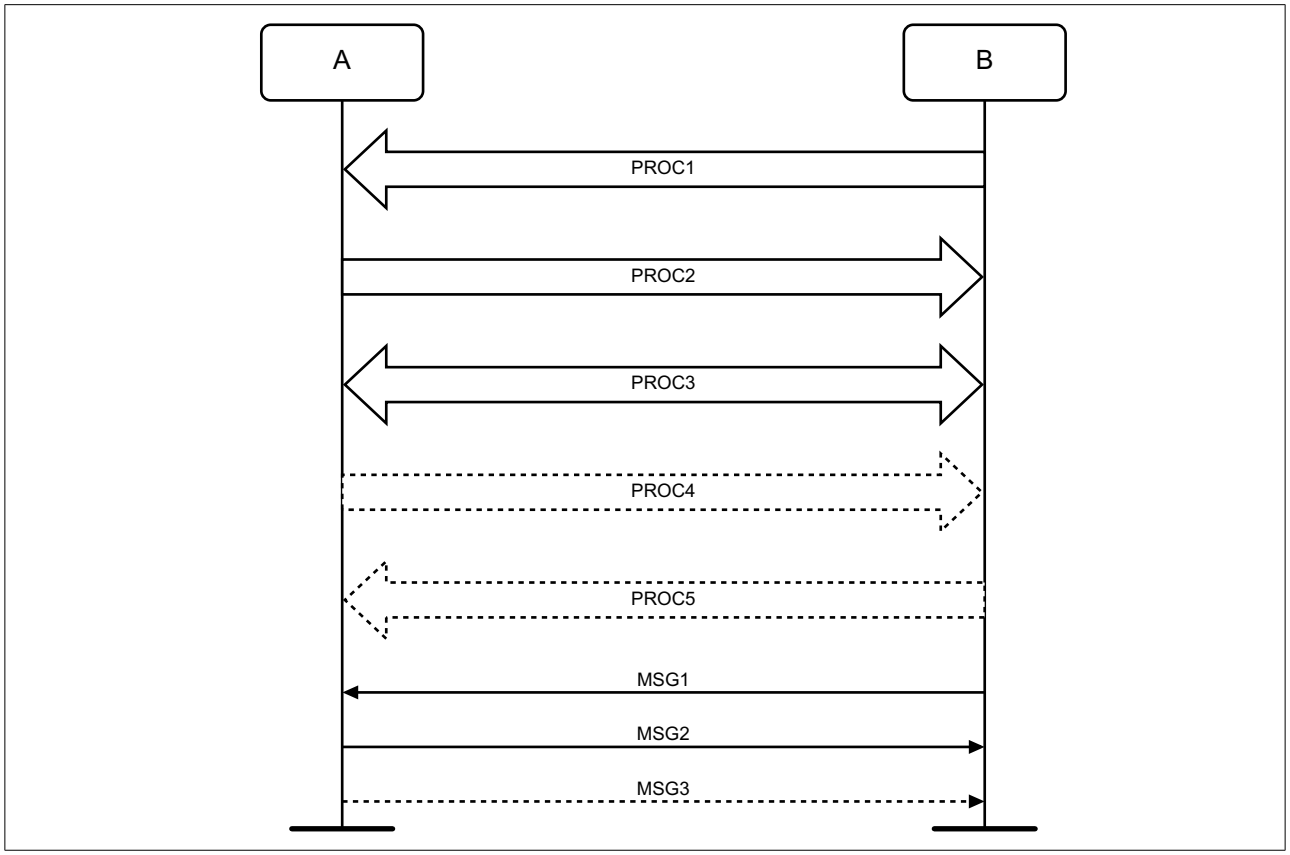



*Figure 1.1: Arrows used in signaling diagrams*

In Figure 1.1, the following cases are shown: PROC1 is a sub-procedure initiated by B. PROC2 is a sub-procedure initiated by A. PROC3 is a sub-procedure where the initiating side is undefined (may be both A or B). Dashed arrows denote optional steps. PROC4 indicates an optional sub-procedure initiated by A, and PROC5 indicates an optional sub-procedure initiated by B.

MSG1 is a message sent from B to A. MSG2 is a message sent from A to B. MSG3 indicates an optional message from A to B.

### 1.2.3 Notation for timers and counters

Timers are introduced specific to this profile. To distinguish them from timers used in the Bluetooth protocol specifications and other profiles, these timers are named in the following format: '$T_{GAP}(nnn)$'.

### 1.2.4 Notation for UUIDs

The use of « » (e.g. «Device Name») indicates a Bluetooth SIG-defined UUID.

## 1.3 GAP requirements

The sections of GAP that apply to an implementation depend on which Core-Host Configuration or Core-Complete Configuration (see [Vol 0] Part D, Section 2) it implements, as shown in Table 1.2.

| Core Configuration | Applicable sections |
|---|---|
| BR/EDR | 1 to 8, 12, 15 to 17, Appendices A and B |
| LE | 1 to 3, 9 to 12, 15 to 17, Appendices A and B |
| BR/EDR/LE | All |

*Table 1.2: Required sections for each Core Configuration*

# 2 PROFILE OVERVIEW

## 2.1 Profile stack

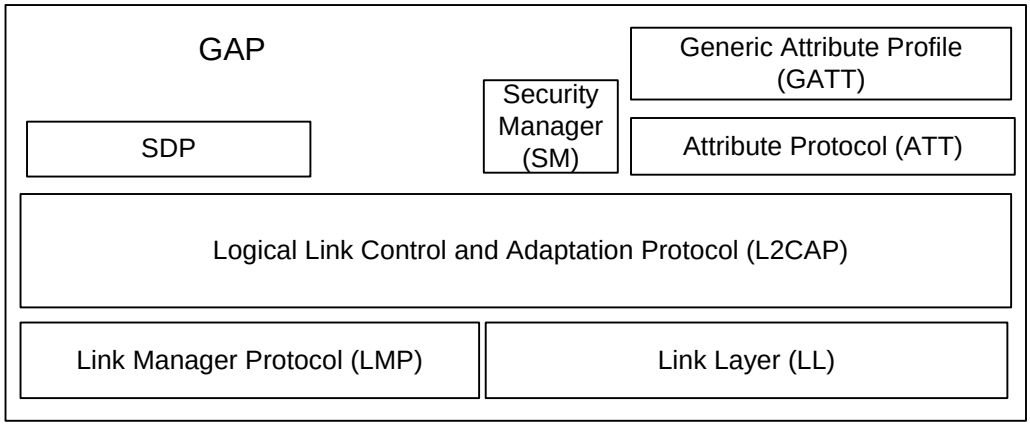



*Figure 2.1: Relationship of GAP with lower layers of the Bluetooth architecture*

The purpose of this profile is to describe:

- Profile roles
- Discoverability modes and procedures
- Connection modes and procedures
- Security modes and procedures

## 2.2 Profile roles

### 2.2.1 Roles when operating over BR/EDR Physical Transport

In GAP, for describing the Bluetooth communication that occurs between two devices in the BR/EDR GAP role, the generic notation of the A-party (the *paging device* in case of link establishment, or *initiator* in case of another procedure on an established link) and the B-party (*paged device* or *acceptor*) is used. The A-party is the one that, for a given procedure, initiates device discovery, initiates the establishment of a physical link or initiates a transaction on an existing link.

This profile handles the procedures between two devices related to discovery and connecting (link and connection establishment) for the case where none of the two devices has any link established as well as the case where (at least) one device has a link established (possibly to a third device) before starting the described procedure.

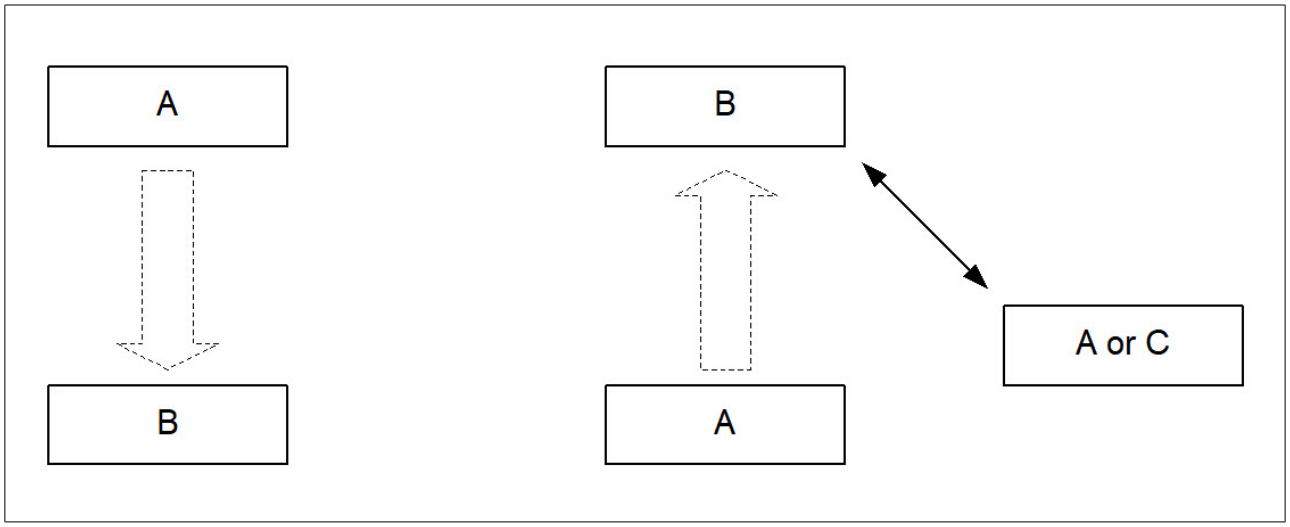



*Figure 2.2: This profile covers procedures initiated by one device (A) towards another device (B) whether or not they have an existing Bluetooth link active*

The initiator and the acceptor generally operate the generic procedures according to this profile or another profile referring to this profile. If the acceptor operates according to several profiles simultaneously, this profile describes generic mechanisms for how to handle this.

### 2.2.2 Roles when operating over an LE physical transport

There are four GAP roles defined for devices operating over an LE physical transport:

- Broadcaster
- Observer
- Peripheral
- Central

The GAP roles "Central" and "Peripheral" are related to, but not the same as, the Link Layer roles with the same names and therefore can have different requirements. If a connection is established successfully, the device with one of these GAP roles will also have the corresponding Link Layer role. However, the device can be in the GAP role without a connection being established.

Table 2.1 defines requirements for physical layer and Link Layer functionalities for each GAP role when operating over an LE physical transport.

| | | GAP Roles When Operating Over an LE Physical Transport | | | |
|---|---|---|---|---|---|
| | | **Broadcaster** | **Observer** | **Peripheral** | **Central** |
| **Radio Physical Layer functionality:** | | | | | |
| • Transmitter characteristics | | M | O | M | M |
| • Receiver characteristics | | O | M | M | M |
| **Link Layer functionality:** | | | | | |
| *States:* | | | | | |
| • Standby state | | M | M | M | M |
| • Advertising state | | M | E | M | E |
| • Scanning state | | E | M | E | M |
| • Initiating state | | E | E | E | M |
| • Synchronization State | | E | O | E | E |
| • Isochronous Broadcasting State | | O | E | E | E |
| • Connection state | Peripheral role | E | E | M | E |
| | Central role | E | E | E | M |
| *Advertising event types:* | | | | | |
| • Connectable and scannable undirected event | | E | E | M | E |
| • Connectable undirected event[1] | | E | E | O | E |
| • Connectable directed event | | E | E | O | E |
| • Non-connectable and non-scannable undirected event | | M | E | O | E |
| • Non-connectable and non-scannable directed event[1] | | O | E | O | E |
| • Scannable undirected event | | O | E | O | E |
| • Scannable directed event[1] | | O | E | O | E |
| *Scanning type:* | | | | | |
| • Passive scanning | | E | M | E | O |
| • Active scanning | | E | O | E | C.1 |
| *Link Layer control procedures:* | | | | | |
| • Connection Update procedure | | E | E | M | M |
| • Channel Map Update procedure | | E | E | M | M |
| • Encryption procedure | | E | E | O | O |
| • Central-initiated Feature Exchange procedure | | E | E | M | M |

| | GAP Roles When Operating Over an LE Physical Transport | | | |
|---|---|---|---|---|
| | **Broadcaster** | **Observer** | **Peripheral** | **Central** |
| • Peripheral-initiated Feature Exchange procedure | E | E | C.2 | C.2 |
| • Connection Parameters Request procedure | E | E | O | O |
| • Version Exchange procedure | E | E | M | M |
| • ACL Termination procedure | E | E | M | M |
| • LE Ping procedure | E | E | O | O |
| • Data Length Update procedure | E | E | O | O |
| • PHY Update procedure | E | E | O | O |
| • Minimum Number Of Used Channels procedure | E | E | O | O |
| • Periodic Advertising Sync Transfer procedure | E | E | O | O |
| • Connected Isochronous Stream Creation procedure | E | E | O | O |
| • Connected Isochronous Stream Termination procedure | E | E | O | O |
| *Broadcast control procedures:* | | | | |
| • Broadcast Isochronous Channel Map Update procedure | O | O | E | E |
| • Broadcast Isochronous Termination procedure | O | O | E | E |
| C.1: Optional if passive scanning is supported, otherwise mandatory. | | | | |
| C.2: Mandatory if Connection Parameters Request procedure is supported, otherwise optional. | | | | |

*Table 2.1: GAP requirements for Radio Physical Layer and Link Layer functionalities for each GAP role when operating over an LE physical transport*

[1]These advertising event types are excluded if the device does not support LE Extended Advertising.

#### 2.2.2.1 Broadcaster role

A device operating in the Broadcaster role is a device that sends advertising events or periodic advertising events as described in [Vol 6] Part B, Section 4.4.2, and may also send Broadcast Isochronous Stream (BIS) events as described in [Vol 6] Part B, Section 4.4.6. A device operating in the Broadcaster role is referred to as a Broadcaster and shall have a transmitter and may have a receiver.

#### 2.2.2.2 Observer role

A device operating in the Observer role is a device that receives advertising events or periodic advertising events as described in [Vol 6] Part B, Section 4.4.3, and may also

receive BIS events as described in [Vol 6] Part B, Section 4.4.6. A device operating in the Observer role is referred to as an Observer and shall have a receiver and may have a transmitter.

#### 2.2.2.3 Peripheral role

Any device that accepts the establishment of an LE active physical link using any of the connection establishment procedures as defined in Section 9 is referred to as being in the Peripheral role. A device operating in the Peripheral role will be in the Peripheral role in the Link Layer Connection state as described in [Vol 6] Part B, Section 4.5. A device operating in the Peripheral role is referred to as a Peripheral. A Peripheral shall have both a transmitter and a receiver.

#### 2.2.2.4 Central role

A device that supports the Central role initiates the establishment of an LE active physical link. A device operating in the Central role will be in the Central role in the Link Layer Connection state as described in [Vol 6] Part B, Section 4.5. A device operating in the Central role is referred to as a Central. A Central shall have both a transmitter and a receiver.

#### 2.2.2.5 Concurrent operation in multiple GAP roles

A device may operate in multiple GAP roles concurrently if supported by the Controller. The Host should read the supported Link Layer states and state combinations from the Controller before any procedures or modes are used.

## 2.3 User requirements and scenarios

The Bluetooth user should, where expected, be able to connect a Bluetooth device to any other Bluetooth device. Even if the two connected devices don’t share any common application, it should be possible for the user to find this out using basic Bluetooth capabilities. When the two devices do share the same application but are from different manufacturers, the ability to connect them should not be blocked just because manufacturers choose to call basic Bluetooth capabilities by different names on the user interface level or implement basic procedures to be executed in different orders.

## 2.4 Profile fundamentals

This profile states the requirements on names, values and coding schemes used for names of parameters and procedures experienced on the user interface level.

This profile defines modes of operation that are not service- or profile-specific, but that are generic and can be used by profiles referring to this profile, and by devices implementing multiple profiles.

This profile defines the general procedures that can be used for discovering identities, names and basic capabilities of other Bluetooth devices that are in a mode where they can be discovered. Only procedures where no channel or connection establishment is used are specified.

This profile defines the general procedure for how to create bonds (i.e., dedicated exchange of link keys) between Bluetooth devices.

This profile describes the general procedures that can be used for establishing connections to other Bluetooth devices that are in a mode that allows them to accept connections and service requests.

## 2.5 [This section is no longer used]

# 3 USER INTERFACE ASPECTS

## 3.1 The user interface level

In the context of the specification, the user interface level refers to places (such as displays, dialog boxes, manuals, packaging, advertising, etc.) where users of Bluetooth devices encounter names, values, and numeric representations of Bluetooth terminology and parameters.

This profile specifies the generic terms that should be used on the user interface level.

## 3.2 Representation of Bluetooth parameters

### 3.2.1 Bluetooth Device Address (BD_ADDR)

#### 3.2.1.1 Definition

A BD_ADDR is the address used by a Bluetooth device as defined in Section 15.1. It is received from a remote device during the device discovery procedure.

#### 3.2.1.2 Term on user interface level

When the Bluetooth address is referred to on the UI level, the term ‘Bluetooth Device Address’ should be used.

#### 3.2.1.3 Representation

On the Baseband level the BD_ADDR is represented as 48 bits (see [Vol 2] Part B, Section 1.2). On the Link Layer the public and random device address are represented as 48-bit addresses.

On the UI level the Bluetooth address shall be represented as 12 hexadecimal characters, possibly divided into sub-parts separated by ‘:’ (e.g., ‘000C3E3A4B69’ or ‘00:0C:3E:3A:4B:69’). On the UI level any number shall have the MSB → LSB (from left to right) ‘natural’ ordering.

### 3.2.2 Bluetooth Device Name (the user-friendly name)

#### 3.2.2.1 Definition

The Bluetooth Device Name is the user-friendly name that a Bluetooth device exposes to remote devices. For a BR/EDR-only implementation, the name is a character string returned in the LMP_NAME_RES in response to an LMP_NAME_REQ. For an LE-only implementation, the name is a character string held in the Device Name characteristic as defined in Section 12.1.

#### 3.2.2.1.1 Bluetooth Device Name in a BR/EDR/LE implementation

A BR/EDR/LE implementation shall have a single Bluetooth Device Name which shall be identical irrespective of the physical channel used to perform the name discovery procedure.

For the BR/EDR physical channel the name is received in the LMP_NAME_RES. For the LE physical channel the name can be read from the Device Name characteristic as defined in Section 12.1.

Note: The Device Name Characteristic of the local device can be read by a remote device using ATT over BR/EDR if the local device supports ATT over BR/EDR.

### 3.2.2.2 Term on user interface level

When the Bluetooth Device Name is referred to on the UI level, the term ‘Bluetooth Device Name’ should be used.

### 3.2.2.3 Representation

The Bluetooth Device Name can be up to 248 bytes (see [Vol 2] Part C, Section 4.3.5). It shall be encoded according to UTF-8 (therefore the name entered on the UI level may be restricted to as few as 62 characters if codepoints outside the range U+0000 to U+007F are used).

A device cannot expect that a general remote device is able to handle more than the first 40 characters of the Bluetooth Device Name. If a remote device has limited display capabilities, it may use only the first 20 characters.

## 3.2.3 Bluetooth Passkey (Bluetooth PIN)

### 3.2.3.1 Definition

The Bluetooth Passkey may be used to authenticate two Bluetooth devices with each other during the creation of a mutual link key via the pairing procedure. The passkey may be used in the pairing procedures to generate the initial link key.

The PIN may be entered on the UI level but may also be stored in the device; e.g., in the case of a device without an interface for entering and displaying digits.

### 3.2.3.2 Terms at user interface level

When the Bluetooth PIN is referred to on the UI level, the term ‘Bluetooth Passkey’ should be used.

### 3.2.3.3 Representation

There are a number of different representations of the Bluetooth Passkey. At a high level there are two distinct representations: one used with the Secure Simple Pairing

and Security Manager, and another used with legacy pairing (where it is generally referred to as the Bluetooth PIN).

For Secure Simple Pairing and Security Manager, the Bluetooth Passkey is a 6-digit numeric value. It is represented as integer value in the range 0x00000000 – 0x000F423F (000000 to 999999). The numeric value may be used as the input to the Authentication stage 1 for Secure Simple Pairing Passkey Entry (see [Vol 2] Part H, Section 7.2.3), or as the TK value in the Security Manager for the process defined in [Vol 3] Part H, Section 2.3.5.

For legacy pairing (see Section B.2), the Bluetooth PIN has different representations on different levels. PINBB is used on the Baseband level, and PINUI is used on the user interface level. PINBB is the PIN used in [Vol 2] Part H, Section 3.2.1 for calculating the initialization key during the Pairing procedure.

PINUI is the character representation of the PIN that is entered on the UI level. The transformation from PINUI to PINBB shall be according to UTF-8. PINBB may be 128 bits (16 bytes).

PIN codes may be up to 16 characters. In order to take advantage of the full level of security all PINs should be 16 characters long. Variable PINs should be composed of alphanumeric characters chosen from within the Unicode range U+0000 to U+007F. If the PIN contains any decimal digits these shall be encoded using the Unicode Basic Latin characters (i.e., U+0030 to U+0039) (Note 1).

For compatibility with devices with numeric keypads, fixed PINs shall be composed of only decimal digits, and variable PINs should be composed of only decimal digits.

If a device supports entry of characters outside the Unicode range U+0000 to U+007F, other Unicode code points may be used (Note 2), except the Halfwidth and Fullwidth Forms (i.e., U+FF00 to U+FFEF) shall not be used (Note 3).

**Examples:**

| **User-entered code** | **Corresponding $PIN_{BB}$[0..length-1]**<br>**(value as a sequence of octets in hexadecimal notation)** |
|---|---|
| ’0196554200906493’ | length = 16, value = 0x30 0x31 0x39 0x36 0x35 0x35 0x34 0x32 0x30 0x30 0x39 0x30 0x36 0x34 0x39 0x33 |
| ’Børnelitteratur’ | length = 16, value = 0x42 0xC3 0xB8 0x72 0x6E 0x65 0x6C 0x69 0x74 0x74 0x65 0x72 0x61 0x74 0x75 0x72 |

Note 1: This is to prevent interoperability problems since there are decimal digits at other code points (e.g., the Fullwidth digits at code points U+FF10 to U+FF19).

Note 2: Unicode characters outside the Basic Latin range (U+0000 to U+007F) encode to multiple bytes; therefore, when characters outside the Basic Latin range are used the maximum number of characters in the PINUI will be less than 16. The second example illustrates a case where a 15 character string encodes to 16 bytes because the character ø is outside the Basic Latin range and encodes to two bytes (0xC3 0xB8).

Note 3: This is to prevent interoperability problems since the Halfwidth and Fullwidth forms contain alternative variants of ASCII, Katakana, Hangul, punctuation and symbols. All of the characters in the Halfwidth and Fullwidth forms have other related Unicode characters; for example, U+3150 (Hangul Letter AE) can be used instead of U+FFC3 (Halfwidth Hangul Letter AE).

### 3.2.4 Class of Device

#### 3.2.4.1 Definition

Class of Device is a parameter received during the device discovery procedure on the BR/EDR physical transport, indicating the type of device. The Class of Device parameter is only used on BR/EDR and BR/EDR/LE devices using the BR/EDR physical transport.

#### 3.2.4.2 Term on user interface level

The information within the Class of Device parameter should be referred to as ‘Bluetooth Device Class’ (i.e., the major and minor device class fields) and ‘Bluetooth Service Type’ (i.e., the service class field). The terms for the defined Bluetooth Device Classes and Bluetooth Service Types are defined in [3].

When using a mix of information found in the Bluetooth Device Class and the Bluetooth Service Type, the term ‘Bluetooth Device Type’ should be used.

#### 3.2.4.3 Representation

The Class of Device is a bit field and is defined in [3]. The UI-level representation of the information in the Class of Device is implementation specific.

#### 3.2.4.4 Usage

Some devices provide more than one service and a given service may be provided by different device types. Therefore, the device type does not have a one-to-one relationship with services supported. The major and minor device class field should not be used to determine whether a device supports any specific service(s). It may be used as an indication of devices that are most likely to support desired services before service discovery requests are made, and it may be used to guide the user when selecting among several devices that support the same service.

### 3.2.5 Appearance characteristic

#### 3.2.5.1 Definition

The Appearance characteristic contains a 16-bit number that can be mapped to an icon or string that describes the physical representation of the device during the device discovery procedure. It is a characteristic of the GAP Service located on the device's GATT Server. See Section 12.2.

#### 3.2.5.2 Usage at user interface level

The Appearance characteristic value should be mapped to an icon or string or something similar that conveys to the user a visual description of the device. This allows the user to determine which device is being discovered purely by visual appearance. If a string is displayed, this string should be translated into the language selected by the user for the device.

#### 3.2.5.3 Representation

The Appearance characteristic value shall be set to one of the 16-bit numbers assigned by the Bluetooth SIG and defined in Section 1.12 of [4]. The UI-level representation of the Appearance characteristic value is implementation specific.

### 3.2.6 Broadcast Code

#### 3.2.6.1 Definition

The Broadcast_Code parameter is used to support an encrypted BIG. It is used in the process of encrypting the data in the transmission of an encrypted BIG and in the process of decrypting the data in the reception of a BIS within that BIG.

#### 3.2.6.2 Terms at user interface level

When the Broadcast_Code parameter is referred to on the UI level, the term 'Bluetooth Privacy Code' should be used.

#### 3.2.6.3 Representation

The Broadcast_Code parameter has different representations on different levels.

On the UI level, the Broadcast Code parameter shall be represented as a string of at least 4 octets that meets the requirements in Section 3.2.3.3 for a PINUI (e.g., it is not more than 16 octets when represented in UTF-8). 16 octets should be used for higher level of security.

On all levels other than UI, the Broadcast Code parameter shall be represented as a 128-bit value. The transformation from string to number shall be by representing

the string in UTF-8, placing the resulting bytes in 8-bit fields of the value starting at the least significant bit, and then padding with zeros in the most significant bits if necessary. For example, the string “Børne House” is represented as the value 0x00000000_6573756F_4820656E_72B8C342.

Note: If the Broadcast_Code is not intended to be displayed or entered in a UI, then it does not need to be a UTF-8 string.

## 3.3 Pairing

Pairing over a BR/EDR physical link is defined on LMP level (LMP pairing, see Section B.2). Pairing over an LE physical link is defined by the Security Manager specification ([Vol 3] Part H, Section 2.3). Either the user initiates the bonding procedure and enters the passkey with the explicit purpose of creating a bond (and maybe also a secure relationship) between two Bluetooth devices, or the user is requested to enter the passkey during the establishment procedure since the devices did not share a common link key beforehand. In the first case, the user is said to perform “bonding (with entering of passkey)” and in the second case the user is said to “authenticate using the passkey.”

# 4 MODES – BR/EDR PHYSICAL TRANSPORT

| Group | Ref. | Mode | Support |
|---|---|---|---|
| Discoverability modes: | 4.1 | Non-discoverable | C.1 |
| | | Limited discoverable | O |
| | | General discoverable | O |
| Connectability modes: | 4.2 | Non-connectable | O |
| | | Connectable | M |
| Bondable modes: | 4.3 | Non-bondable | C.4 |
| | | Bondable | C.2 |
| Synchronizability modes: | 4.4 | Non-synchronizable | M |
| | | Synchronizable | C.5 |
| C.1: Mandatory if limited discoverable mode is supported or general discoverable mode is not supported, otherwise optional. | | | |
| C.2: Mandatory if the bonding procedure is supported, otherwise optional. | | | |
| C.4: Optional if Bondable mode is supported, otherwise mandatory. | | | |
| C.5: Optional if the Synchronization Train procedure is supported, otherwise excluded. | | | |

*Table 4.1: Conformance requirements related to modes defined in this section*

## 4.1 Discoverability modes

With respect to inquiry, a Bluetooth device shall be either in non-discoverable mode or in a discoverable mode. (The device shall be in one, and only one, discoverability mode at a time.) The two discoverable modes defined here are called limited discoverable mode and general discoverable mode. Inquiry is defined in [Vol 2] Part B, Section 8.4.

When a Bluetooth device is in non-discoverable mode it does not respond to inquiry.

A Bluetooth device is said to be made discoverable, or set into a discoverable mode, when it is in limited discoverable mode or in general discoverable mode. Even when a Bluetooth device is made discoverable, it may be unable to respond to inquiry due to other Baseband activity (for example, reserved synchronous slots should have priority over response packets, so that synchronous links may prevent a response from being returned). A Bluetooth device that does not respond to inquiry is called a silent device.

After being made discoverable, the Bluetooth device shall be discoverable for at least $T_{GAP}(103)$.

The speed of discovery is dependent on the configuration of the inquiry scan interval and inquiry scan type of the Bluetooth device. The Host is able to configure these

parameters based on trade-offs between power consumption, bandwidth and the desired speed of discovery.

### 4.1.1 Non-discoverable mode

#### 4.1.1.1 Definition

When a Bluetooth device is in non-discoverable mode, it shall never enter the INQUIRY_SCAN state.

#### 4.1.1.2 Term on UI-level

Bluetooth device is ‘non-discoverable’ or in ’non-discoverable mode’.

### 4.1.2 Limited Discoverable mode

#### 4.1.2.1 Definition

The limited discoverable mode should be used by devices that need to be discoverable only for a limited period of time, during temporary conditions, or for a specific event.

There are two common reasons to use limited discoverable mode:

- Limited discoverable mode can be used to allow remote devices using the general inquiry procedure to prioritize or otherwise identify devices in limited discoverable mode when presenting discovered devices to the end user because, typically, the user is interacting with them.
- Limited discoverable mode can also be used to allow remote devices using the limited inquiry procedure to filter out devices using the general discoverable mode.

A Bluetooth device should not be in limited discoverable mode for more than $T_{GAP}(104)$. The scanning for the limited inquiry access code can be done either in parallel or in sequence with the scanning of the general inquiry access code. When in limited discoverable mode, one of the following options shall be used.

- *Parallel scanning*

  When a Bluetooth device is in limited discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

  If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Non-interlaced Inquiry scan is used.

In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC and the LIAC for at least $T_{GAP}(101)$.

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

- *Sequential scanning*

  When a Bluetooth device is in limited discoverable mode, it shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$ and enter the INQUIRY_SCAN state more often than once in $T_{GAP}(102)$ and scan for the LIAC for at least $T_{GAP}(101)$.

  If an inquiry message is received when in limited discoverable mode, the entry into the INQUIRY_RESPONSE state takes precedence over the next entries into INQUIRY_SCAN state until the inquiry response is completed.

#### 4.1.2.2 Conditions

When a device is in limited discoverable mode it shall set bit number 13 in the Major Service Class part of the Class of Device/Service field [3].

#### 4.1.2.3 Term on UI-level

Bluetooth device is ‘discoverable’ or in ‘discoverable mode’.

### 4.1.3 General Discoverable mode

#### 4.1.3.1 Definition

The general discoverable mode shall be used by devices that need to be discoverable continuously or for no specific condition.

Devices in the general discoverable mode will not be discovered by devices performing the limited inquiry procedure. General discoverable mode should not be used if it is known that the device performing discovery will be using the limited inquiry procedure (see Section 6.1).

#### 4.1.3.2 Conditions

When a Bluetooth device is in general discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Non-interlaced Inquiry scan is used.

In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$.

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

A device in general discoverable mode shall not respond to a LIAC inquiry.

#### 4.1.3.3 Term on UI-level

Bluetooth device is ‘discoverable’ or in ‘discoverable mode’.

## 4.2 Connectability modes

With respect to paging, a Bluetooth device shall be either in non-connectable mode or connectable mode. Paging is defined in [Vol 2] Part B, Section 8.3.

When a Bluetooth device is in non-connectable mode it does not respond to paging. When a Bluetooth device is in connectable mode it responds to paging.

The speed of connections is dependent on the configuration of the page scan interval and page scan type of the Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of connection.

### 4.2.1 Non-connectable mode

#### 4.2.1.1 Definition

When a Bluetooth device is in non-connectable mode it shall never enter the PAGE_SCAN state.

#### 4.2.1.2 Term on UI-level

Bluetooth device is ‘non-connectable’ or in ‘non-connectable mode’.

## 4.2.2 Connectable mode

### 4.2.2.1 Definition

When a Bluetooth device is in connectable mode it shall periodically enter the PAGE_SCAN state. The device performs page scan using the Bluetooth Device Address, BD_ADDR. Connection speed is a trade-off between power consumption / available bandwidth and speed. The Bluetooth Host is able to make these trade-offs using the Page Scan interval, Page Scan window, and Interlaced Scan parameters.

R0 page scanning should be used when connection speeds are critically important and when the paging device has a very good estimate of the Bluetooth clock. Under these conditions it is possible for paging to complete within two times the page scan window. Because the page scan interval is equal to the page scan window it is not possible for any other traffic to go over the Bluetooth link when using R0 page scanning. In R0 page scanning it is not possible to use interlaced scan. R0 page scanning is the highest power consumption mode of operation.

When connection times are critical but the other device either does not have an estimate of the Bluetooth clock or when the estimate is possibly out of date, it is better to use R1 page scanning with a very short page scan interval, $T_{GAP}(106)$, and Interlaced scan. This configuration is also useful to achieve nearly the same connection speeds as R0 page scanning but using less power and leaving bandwidth available for other connections. Under these circumstances it is possible for paging to complete within $T_{GAP}(106)$. In this case the Bluetooth device shall page scan for at least $T_{GAP}(101)$.

When connection times are important but not critical enough to sacrifice significant bandwidth and/or power consumption it is recommended to use either $T_{GAP}(107)$ or $T_{GAP}(108)$ for the scanning interval. Using Interlaced scan will reduce the connection time by half but may use twice the power consumption. Under these circumstances it is possible for paging to complete within one or two times the page scanning interval depending on whether Interlaced Scan is used. In this case the Bluetooth device shall page scan for at least $T_{GAP}(101)$.

In all cases the Bluetooth device shall enter the PAGE_SCAN state at least once in $T_{GAP}(102)$ and scan for at least $T_{GAP}(101)$.

The page scan interval, page scan window size, and scan type for the six scenarios are described in Table 4.2:

| Scenario | Page Scan Interval | Page Scan Window | Scan Type |
|---|---|---|---|
| R0 (1.28 s) | $T_{GAP}(107)$ | $T_{GAP}(107)$ | Normal scan |
| Fast R1 (100 ms) | $T_{GAP}(106)$ | $T_{GAP}(101)$ | Interlaced scan |

| Scenario | Page Scan Interval | Page Scan Window | Scan Type |
|---|---|---|---|
| Medium R1 (1.28 s) | $T_{GAP}(107)$ | $T_{GAP}(101)$ | Interlaced scan |
| Slow R1 (1.28 s) | $T_{GAP}(107)$ | $T_{GAP}(101)$ | Normal scan |
| Fast R2 (2.56 s) | $T_{GAP}(108)$ | $T_{GAP}(101)$ | Interlaced scan |
| Slow R2 (2.56 s) | $T_{GAP}(108)$ | $T_{GAP}(101)$ | Normal scan |

*Table 4.2: Page scan parameters for connection speed scenarios*

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the connection time.

#### 4.2.2.2 Term on UI-level

Bluetooth device is ‘connectable’ or in ‘connectable mode’.

## 4.3 Bondable modes

With respect to bonding, a Bluetooth device shall be either in non-bondable mode or in bondable mode. In bondable mode the Bluetooth device accepts bonding initiated by the remote device, and in non-bondable mode it does not.

### 4.3.1 Non-bondable mode

#### 4.3.1.1 Definition

When a Bluetooth device is in non-bondable mode it shall not accept a pairing request that results in bonding. Devices in non-bondable mode may accept connections that do not request or require bonding.

A device in non-bondable mode shall respond to a received LMP_IN_RAND with LMP_NOT_ACCEPTED with the Error_Code *Pairing Not Allowed* (0x18).

When both devices support Secure Simple Pairing and the local device is in non-bondable mode, the local Host shall respond to an IO capability request where the Authentication_Requirements parameter requests dedicated bonding or general bonding with a negative response.

#### 4.3.1.2 Term on UI-level

Bluetooth device is ‘non-bondable’ or in ‘non-bondable mode’ or “does not accept bonding”.

### 4.3.2 Bondable mode

#### 4.3.2.1 Definition

When a Bluetooth device is in bondable mode, and Secure Simple Pairing is not supported by either the local or remote device, the local device shall respond to a

received LMP_IN_RAND with LMP_ACCEPTED (or with LMP_IN_RAND if it has a fixed PIN).

When both devices support Secure Simple Pairing, the local Host shall respond to a user confirmation request with a positive response.

#### 4.3.2.2 Term on UI-level

Bluetooth device is ‘bondable’ or in ‘bondable mode’ or “accepts bonding”.

## 4.4 Synchronizability modes

A Bluetooth device shall be either in non-synchronizable mode or synchronizable mode. The synchronization train procedure is defined in [Vol 2] Part B, Section 2.7.2.

When a Bluetooth device is in synchronizable mode, it transmits timing and frequency information for its active Connectionless Peripheral Broadcast packets. When a Bluetooth device is non-synchronizable, timing and frequency information is not transmitted.

The Host is able to configure the Synchronization Train interval based on trade-offs between bandwidth, potential interference to other devices, power consumption, and the desired time for a Peripheral to receive a synchronization train packet.

### 4.4.1 Non-synchronizable mode

#### 4.4.1.1 Definition

When a Bluetooth device is in non-synchronizable mode it shall never enter the Synchronization Train substate.

#### 4.4.1.2 Term on UI-level

Bluetooth device is ‘non-synchronizable’ or in ‘non-synchronizable mode’.

### 4.4.2 Synchronizable mode

#### 4.4.2.1 Definition

When a Bluetooth device is in synchronizable mode, it shall enter the Synchronization Train substate using a synchronization train interval of $T_{GAP}$(Sync_Train_Interval).

After being made synchronizable, the Bluetooth device shall be synchronizable for at least $T_{GAP}$(Sync_Train_Duration).

#### 4.4.2.2 Term on UI-level

Bluetooth device is 'synchronizable' or in 'synchronizable mode'.

# 5 SECURITY ASPECTS – BR/EDR PHYSICAL TRANSPORT

| Procedure | Ref. | Support |
|---|---|---|
| Authentication | 5.1 | M |
| Security mode 1 | 5.2.1.1 | E |
| Security mode 2 | 5.2.1.2 | C.1 |
| Security mode 3 | 5.2.1.3 | E |
| Security mode 4 | 5.2.2 | M |
| C.1: Security mode 2 may only be used for backwards compatibility when the remote device does not support Secure Simple Pairing. | | |

*Table 5.1: Conformance requirements related to the generic authentication procedure and the security modes defined in this section*

## 5.1 Authentication

### 5.1.1 Purpose

The generic authentication procedure describes how the LMP-authentication and LMP-pairing procedures are used when authentication is initiated by one Bluetooth device towards another, depending on if a link key exists or not and if pairing is allowed or not.

### 5.1.2 Term on UI level

‘Bluetooth authentication’.

### 5.1.3 Procedure

Figure 5.1 defines the generic authentication procedure.

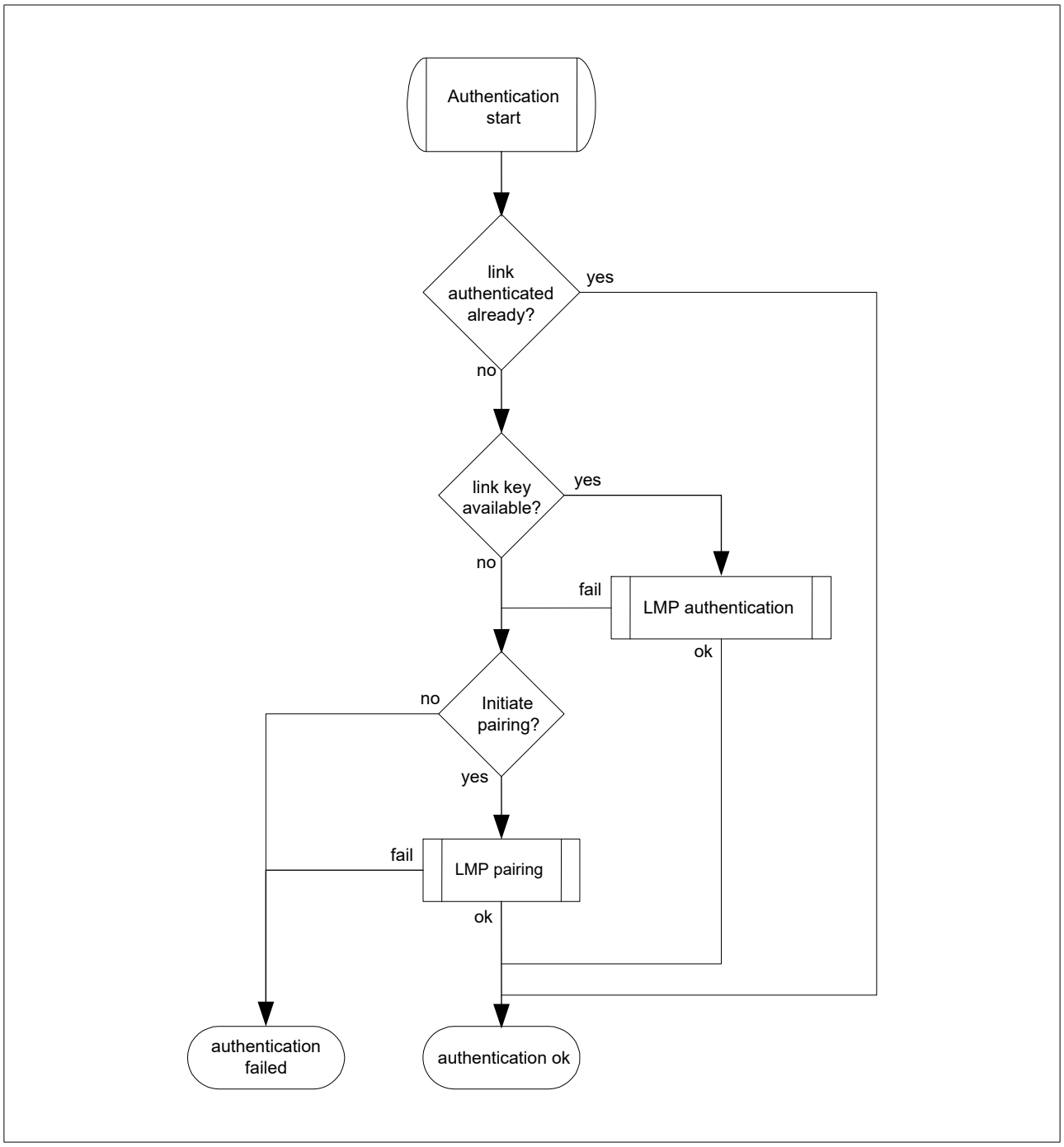



*Figure 5.1: Definition of the generic authentication procedure*

### 5.1.4 Conditions

The local device shall initiate authentication after link establishment. The remote device may initiate security during or after link establishment.

## 5.2 Security modes

The flow chart in Figure 5.2, including the steps in Figure 5.3, Figure 5.4, and Figure 5.5, specifies the overall channel establishment procedures. The following subsections give more details of these procedures.

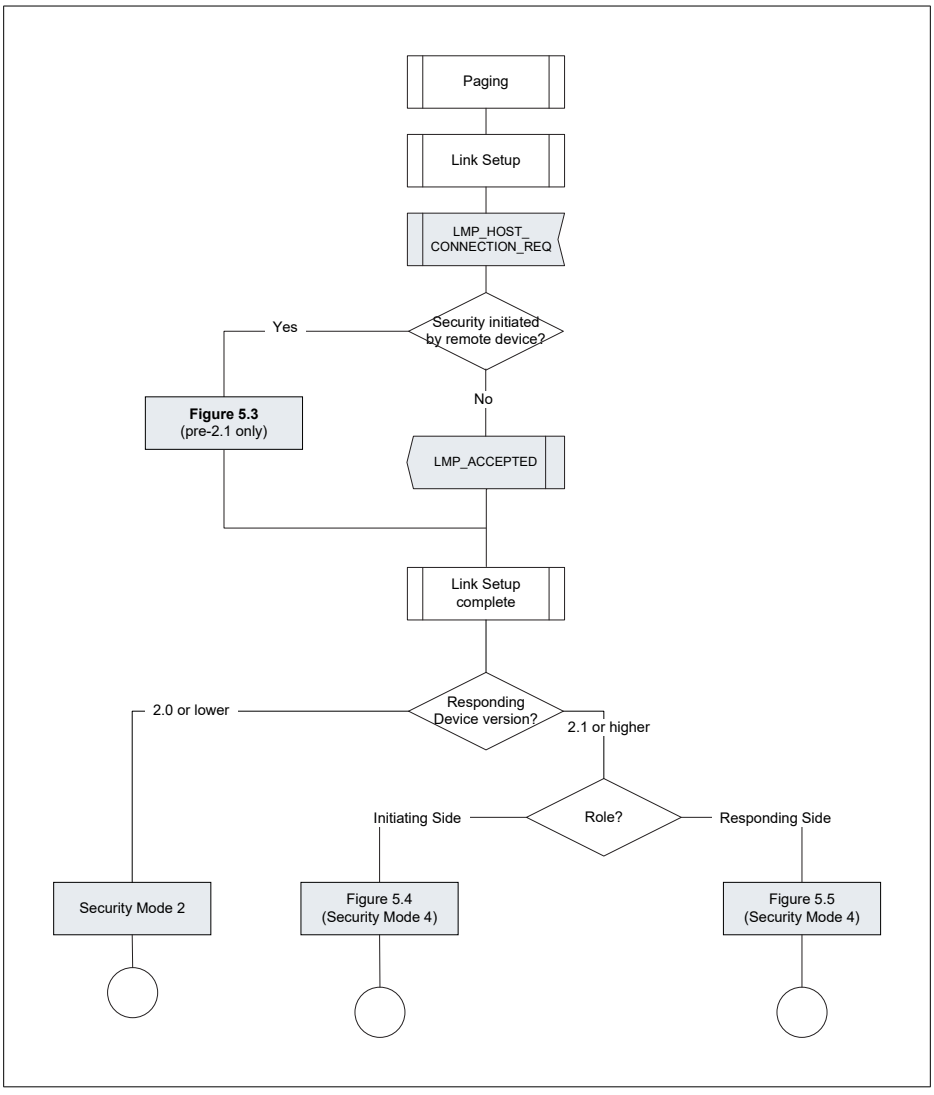



*Figure 5.2: Channel establishment with security*

A device may support two security modes simultaneously: security mode 2 for backwards compatibility with remote devices that do not support Secure Simple Pairing and security mode 4 for devices that support Secure Simple Pairing.

### 5.2.1 Legacy security modes

Legacy security modes apply to those devices with a Controller or a Host that does not support SSP.

#### 5.2.1.1 Security mode 1 (non-secure)

When a remote Bluetooth device is in security mode 1 it will never initiate any security procedure (i.e., it will never send LMP_AU_RAND, LMP_IN_RAND or LMP_ENCRYPTION_MODE_REQ).

#### 5.2.1.2 Security mode 2 (service level enforced security)

When a remote Bluetooth device is in security mode 2 it will not initiate any security procedure before a channel establishment request (L2CAP_CONNECTION_REQ) has been received or a channel establishment procedure has been initiated by itself. Whether a security procedure is initiated or not depends on the security requirements of the requested channel or service.

A Bluetooth device in security mode 2 should classify the security requirements of its services using at least the following attributes:

- Authorization required
- Authentication required
- Encryption required

Note: Security mode 1 can be considered (at least from a remote device point of view) as a special case of security mode 2 where no service has registered any security requirements.

#### 5.2.1.3 Security mode 3 (link level enforced security)

When a remote Bluetooth device is in security mode 3 it will initiate security procedures before it sends LMP_SETUP_COMPLETE.

A Bluetooth device in security mode 3 may reject the Host connection request (respond with LMP_NOT_ACCEPTED to the LMP_HOST_CONNECTION_REQ) based on settings in the Host (e.g., only communication with pre-paired devices allowed).

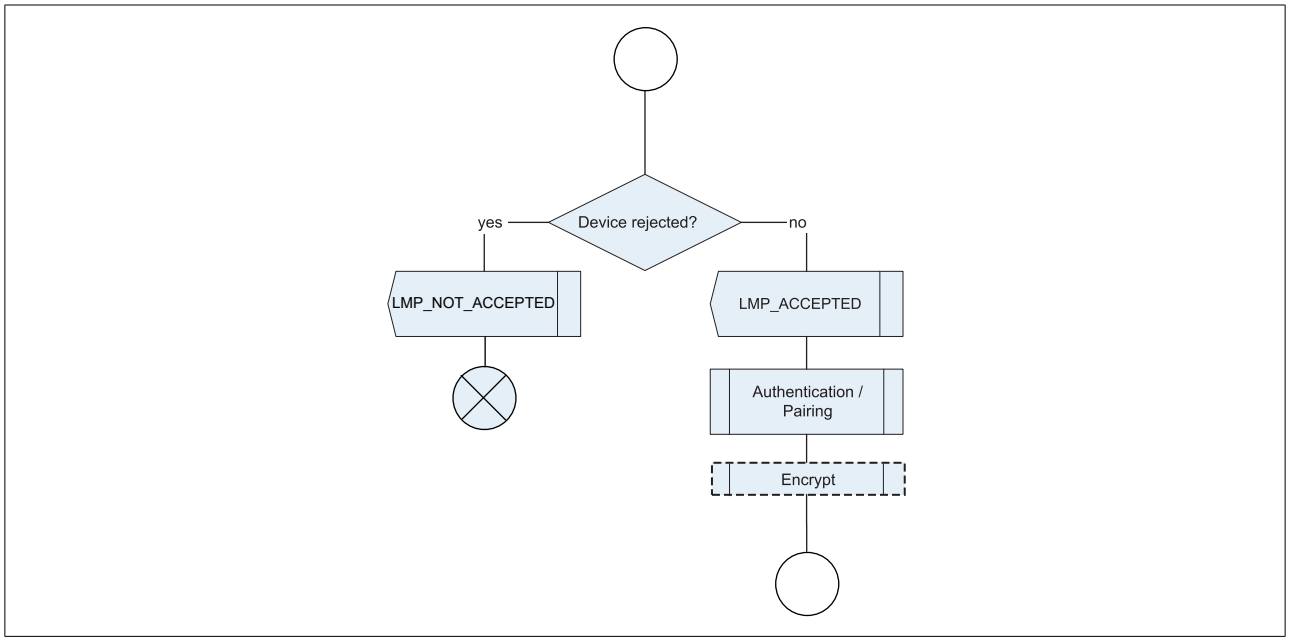



*Figure 5.3: Security mode 3 with a legacy remote device*

### 5.2.2 Security mode 4 (service level enforced security)

A Bluetooth device in security mode 4 shall classify the security requirements of its services using at least the following attributes (in order of decreasing security):

- Authenticated link key required
- Unauthenticated link key required
- Security optional; limited to specific services (see Section 5.2.2.8).

When both devices support Secure Simple Pairing, GAP shall require at least an unauthenticated link key and enabling encryption for all connections except those allowed to have security level 0 (see Section 5.2.2.8). A profile or protocol may define services that require more security (e.g., an authenticated link key) or no security (although unencrypted connections are only allowed when connecting to a service allowed to have security level 0). To allow an unauthenticated link key to be created during the Secure Simple Pairing procedure, the Authentication_Requirements parameter may be set to one of the MITM Protection Not Required options.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.

A Bluetooth device in security mode 4 shall respond to authentication requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons.

A Bluetooth device in security mode 4 enforces its security requirements before it attempts to access services offered by a remote device and before it grants access to services it offers to remote devices. Service access may occur via L2CAP channels or via channels established by protocols above L2CAP such as RFCOMM.

For services transmitting unicast data over the connectionless L2CAP channel, the transmitting device shall enforce its security requirements prior to sending data. There is no mechanism for a device receiving data via the L2CAP connectionless channel to prevent unencrypted data from being received. Hence, Section 5.2.2.1 addresses unicast connectionless data transmission together with devices initiating connection-oriented channels while Section 5.2.2.2 covers only devices responding to requests for connection-oriented channel establishment but does not cover unicast connectionless data reception.

A device may be in a Secure Connections Only mode. When in Secure Connections Only mode, all services (except those allowed to have Security Mode 4, Level 0) available on the BR/EDR physical transport require Security Mode 4, Level 4. The device shall reject both new outgoing and incoming service level connections when the service requires Security Mode 4, Level 4 and either the physical transport does not support Secure Connections or unauthenticated pairing is being requested.

A device operating with a physical transport operating in Secure Connections Only mode may disconnect the ACL connection using error code 0x05 (Authentication Failure) when the physical transport that does not support Secure Connections, tries to access a service that requires Security Mode 4, Level 4.

A device may operate several physical transports simultaneously - in this case all physical transports shall enable Secure Connections Only Mode simultaneously.

#### 5.2.2.1 Security for access to remote service (initiating side)

When the responding device does not support Secure Simple Pairing, it may disconnect the link while the initiating device is requesting the PIN to be entered by the user. This may occur due to the lack of an L2CAP channel being present for longer than an implementation-specific amount of time (e.g., a few seconds). When this occurs, the initiating device shall allow the user to complete entering the PIN and shall then re-page the remote device and restart the pairing procedure (see [Vol 2] Part C, Section 4.2.2) using the PIN entered.

##### 5.2.2.1.1 Pairing required for access to remote service

When a Bluetooth device in security mode 4 initiates access to a remote service via a connection-oriented L2CAP channel, the service requires security, and a sufficient link-key is not available, the local device shall perform pairing procedures and enable

encryption before sending a channel establishment request (an L2CAP connection request or a higher-layer channel establishment request such as that of RFCOMM).

When a Bluetooth device in security mode 4 transmits data to a remote service via the unicast connectionless L2CAP channel and a sufficient link-key is not available, the local device shall perform pairing procedures and enable encryption before transmitting unicast data on the connectionless L2CAP channel.

See Section 5.2.2.8 for details on determining whether or not a link key is sufficient.

If pairing does not succeed, the local device shall not send a channel establishment request. The local device may re-try pairing up to three (3) times. If pairing fails three consecutive times, the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link key generated is not at least as good as the expected or required type, the local device shall fail the establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.1.2 Authentication required for access to remote service

When a Bluetooth device in security mode 4 initiates access to a remote service via a connection-oriented L2CAP channel and a sufficient link key is available for the remote device, it shall authenticate the remote device and enable encryption before sending a channel establishment request (an L2CAP connection request or a higher layer-channel establishment request such as that of RFCOMM).

When a Bluetooth device in security mode 4 transmits unicast data to a remote service via the connectionless L2CAP channel and security is required for the application and a sufficient link-key is available then the local device shall authenticate the remote device and enable encryption before transmitting unicast data on the L2CAP connectionless channel.

See Section 5.2.2.8 for details on determining whether or not a link key is sufficient.

If authentication is required by the service but does not succeed, or if a sufficient link-key is not available, then the local device shall not enable encryption. If encryption is not enabled or if enabling encryption does not result in the correct encryption type (i.e. AES-CCM or E0), the local device shall not send a channel establishment request and shall not send any unicast data via the L2CAP connectionless channel for that application. The Host may then notify the user and offer to perform pairing.

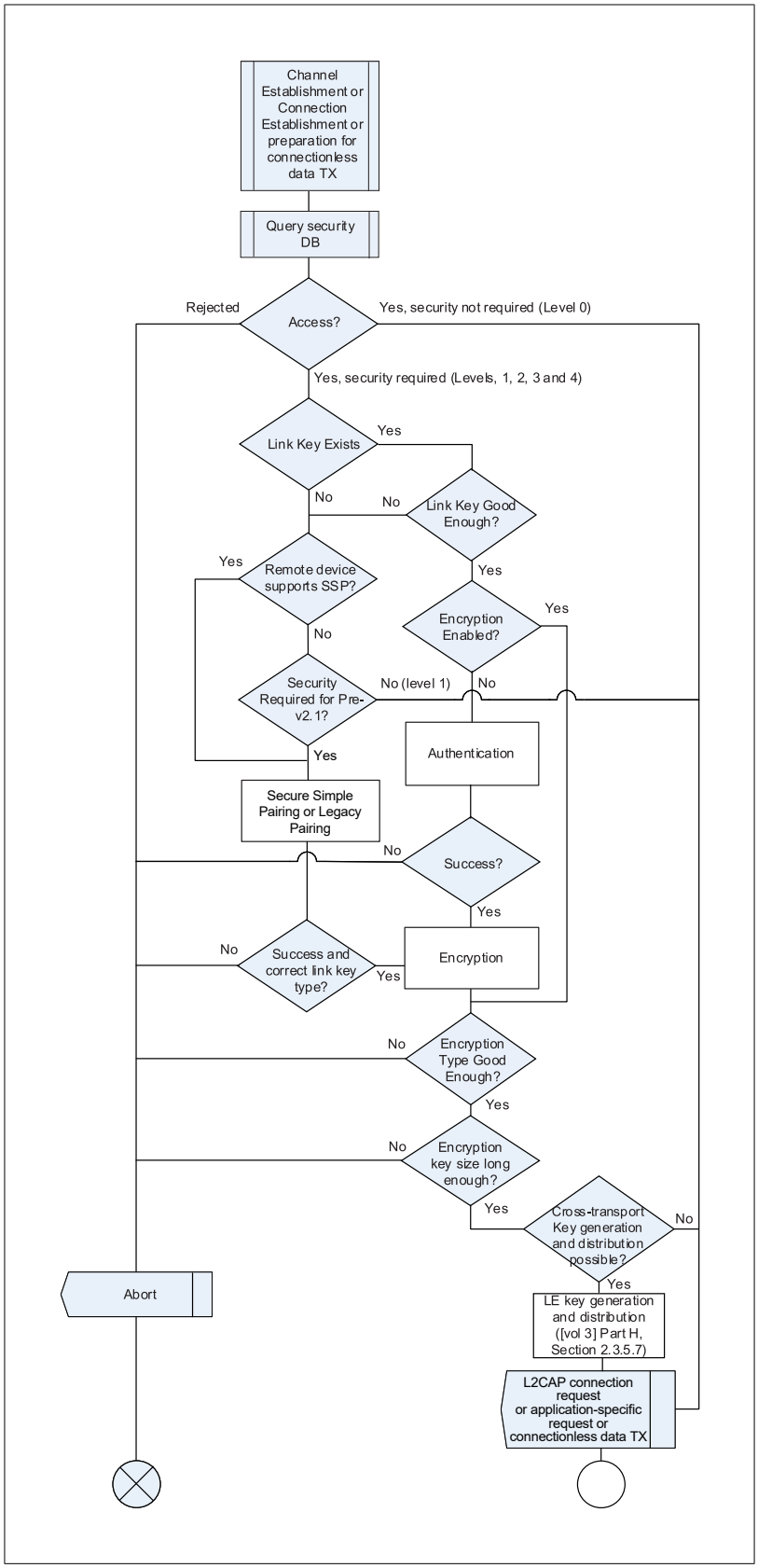



*Figure 5.4: Channel establishment using security mode 4 for initiating side*

#### 5.2.2.1.3 Cross-transport key generation and distribution

After encryption is enabled and both devices support cross-transport key generation, the Central of the BR/EDR transport may initiate LE key generation and distribution ([Vol 3] Part H, Section 2.3.5.7). The Peripheral shall not initiate LE key generation and distribution.

If a role switch gets performed after enabling encryption but before the LE keys can be generated and distributed, the new Central may initiate LE key generation and distribution once the role switch has completed. The new Peripheral shall not initiate LE key generation and distribution once the role switch has completed.

### 5.2.2.2 Security for access to local service by remote device (responding side)

When a remote device attempts to access a service offered by a Bluetooth device that is in security mode 4 and the service requires security, a sufficient link key exists, and authentication has not been performed, the local device shall authenticate the remote device and enable encryption after the channel establishment request is received but before a channel establishment confirmation (L2CAP connection response with result code of 0x0000 or a higher-level channel establishment confirmation such as that of RFCOMM) is sent.

When L2CAP is the channel establishment protocol being used for the requested service, an L2CAP connection response signaling packet shall be sent by the responding device containing a pending result code following receipt of an L2CAP connection request and prior to initiating security procedures which can result in prompting the local user for input (e.g., pairing using a PIN or Secure Simple Pairing using either the Passkey entry or Numeric Comparison association models). This will stop the L2CAP RTX timer on the remote device (which may be as short as 1 second) and will invoke the ERTX timer on the remote device, which is a minimum duration of 60 seconds.

See [Vol 3] Part A, Section 6.2 for additional information on L2CAP RTX and ERTX timers. See also [Vol 3] Part A, Section 4.3 and [Vol 3] Part A, Section 4.26 for additional information on the L2CAP connection response signaling packets and the defined result codes.

Higher layer channel establishment protocols should be designed to restrict timeouts to be 30 seconds or longer to allow for user input, or provide mechanisms to dynamically extend timeouts when user input may be required.

If authentication or pairing fails when a remote device is attempting to access a local service, the local device shall send a negative response to the channel establishment request (L2CAP connection request or application-specific channel establishment request) indicating a security issue if possible. If the channel

establishment protocol is L2CAP, then the result code in the L2CAP connection response shall be set to indicate that the connection was refused due to security reasons. If an L2CAP_CONNECTION_RSP is being sent, then the result code shall be set to 0x0003 (connection refused - security block). If an L2CAP_CREDIT_BASED_CONNECTION_RSP is being sent, then the result code shall be set to 0x0005 (All connections refused - insufficient authentication), 0x0006 (All connections refused - insufficient authorization), 0x0007 (All connections refused - encryption key size too short), or 0x0008 (All connections refused - insufficient encryption).

If the remote device has indicated support for Secure Simple Pairing, a channel establishment request is received for a service other than SDP, and encryption has not yet been enabled, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.2.1 Pairing required for access to local service by remote device

When a remote device attempts to access a service offered by a Bluetooth device that is in security mode 4 and pairing is required due to the link key being absent or insufficient, the local device shall perform pairing procedures and enable encryption after the channel establishment request is received and before a channel establishment confirmation (L2CAP connection response with result code of 0x0000 or a higher-level channel establishment response such as that of RFCOMM) is sent.

See Section 5.2.2.6 for details on determining whether or not a link key is sufficient.

If pairing does not succeed, then the local device shall not send a channel establishment confirmation. The local device may retry pairing up to three (3) times. If pairing fails three consecutive times, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link-key generated is not at least as good as the expected or required type or if enabling encryption does not result in the correct encryption type (i.e. AES-CCM or E0), then the local device shall fail the channel establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

#### 5.2.2.2.2 Authentication required for access to local service by remote device

See Section 5.2.2.6 for details on determining whether or not a link key is sufficient.

If authentication does not succeed, then the local device shall not send a channel establishment confirmation. The Host may at this point notify the user and offer to perform pairing.

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for

backwards compatibility reasons. However, authentication of the remote device shall be performed after the receipt of the channel establishment request is received, and before the channel establishment response is sent.

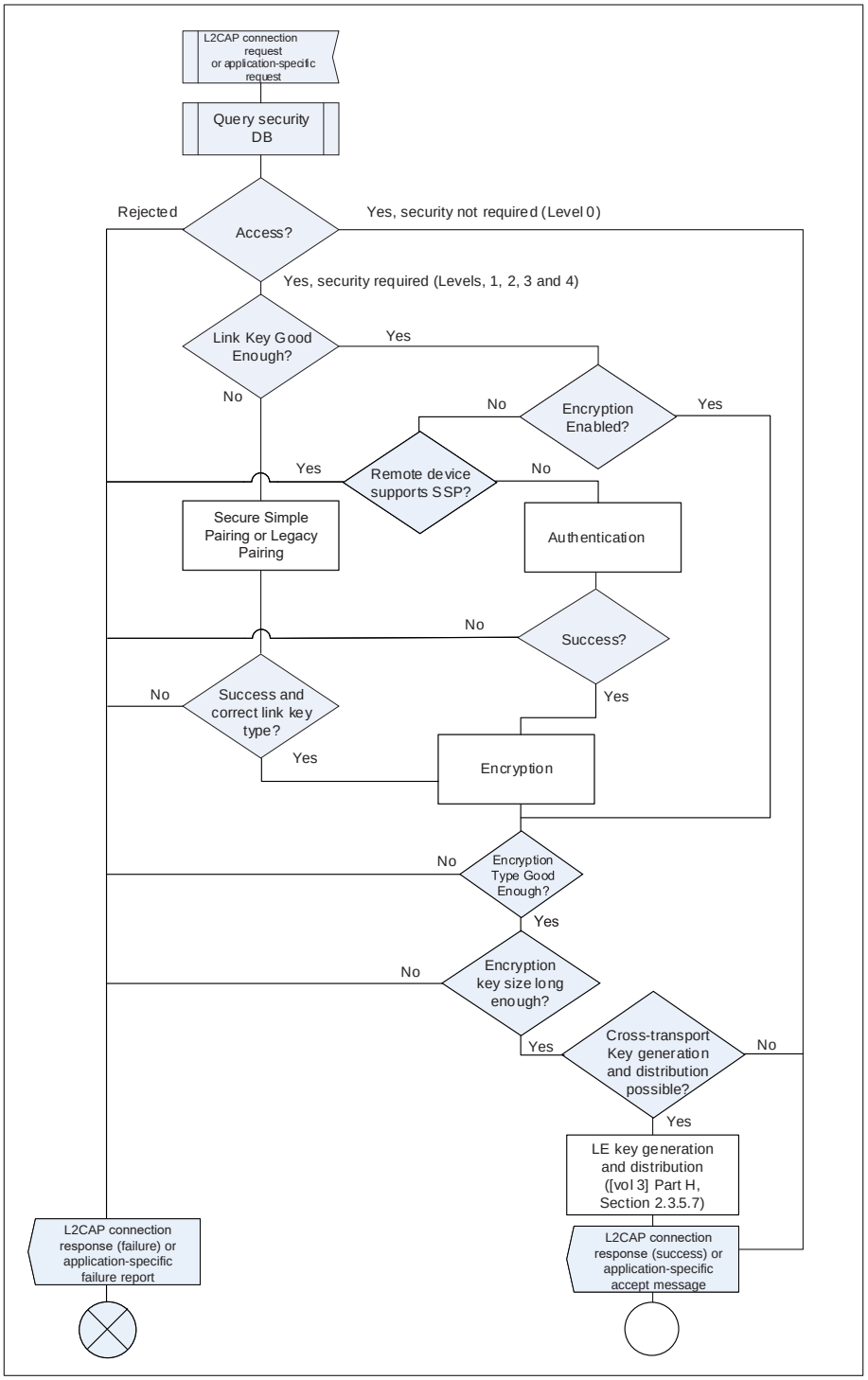



*Figure 5.5: Channel establishment using security mode 4 for the responding side*

#### 5.2.2.2.3 Cross-transport key generation and distribution

After encryption is enabled and both devices support cross-transport key generation, the Central of the BR/EDR transport may initiate LE key generation and distribution ([Vol 3] Part H, Section 2.3.5.7). The Peripheral shall not initiate LE key generation and distribution.

If a role switch gets performed after enabling encryption but before the LE keys can be generated and distributed, the new Central may initiate LE key generation and distribution once the role switch has completed. The new Peripheral shall not initiate LE key generation and distribution once the role switch has completed.

### 5.2.2.3 Secure Simple Pairing after authentication failure

When both devices support Secure Simple Pairing all non-SDP connections are encrypted regardless of whether security was required or whether the devices are bonded or not. The initial connection between the two devices will result in a link key through Secure Simple Pairing. Depending on whether or not bonding was performed and the security policy of the initiating device, the link key may be stored. When the link key is stored, subsequent connections to the same device will use authentication but this may fail if the remote device has deleted the link key. Table 5.2 defines what shall be done depending on the type of the link key and whether bonding was performed or not.

| Link Key Type | Devices Bonded? | Action to take when Authentication Fails |
|---|---|---|
| Combination | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate pairing<br>• Option 2: Notify user and ask if pairing is ok<br><br>Option 2 is recommended. |
| Combination | Yes | Notify user of security failure |
| Unauthenticated | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate secure simple pairing<br>• Option 2: Notify user and ask if secure simple pairing is ok.<br><br>Option 1 is recommended. |
| Unauthenticated | Yes | Notify user of security failure |

| Link Key Type | Devices Bonded? | Action to take when Authentication Fails |
|---|---|---|
| Authenticated | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate secure simple pairing<br>• Option 2: Notify user and ask if secure simple pairing is ok<br><br>Option 2 is recommended. |
| Authenticated | Yes | Notify user of security failure |

*Table 5.2: Secure Simple Pairing after authentication failure*

Non-bonded authenticated or unauthenticated link keys may be considered disposable by either device and may be deleted at any time.

#### 5.2.2.4 IO capabilities

Once a connection is established, if the Host determines that security is necessary and both devices support Secure Simple Pairing, the devices perform an IO capability exchange. The purpose of the IO capability exchange is to determine the authentication algorithm used in the Authentication stage 1 phase of Secure Simple Pairing.

The input and output capabilities are described in Table 5.3:

| Capability | Description |
|---|---|
| No input | Device does not have the ability to indicate 'yes' or 'no' |
| Yes / No | Device has at least two buttons that are mapped easily to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note below). |
| Keyboard | Device has a numeric keyboard that can input the numbers '0' to '9' and a confirmation. Device also has two buttons that can be easily mapped to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see Note below). |

*Table 5.3: User input capabilities*

Note: 'yes' could be indicated by pressing a button within a certain time limit otherwise 'no' would be assumed.

| Capability | Description |
|---|---|
| No output | Device does not have the ability to display or communicate a 6 digit decimal number |
| Numeric output | Device has the ability to display or communicate a 6 digit decimal number |

*Table 5.4: User output capabilities*

### 5.2.2.5 Mapping of input / output capabilities to IO capability

The individual input and output capabilities are mapped to a single IO capability which is later used to determine which authentication algorithm will be used.

| | | Local Output Capability | |
|---|---|---|---|
| | | **No Output** | **Numeric Output** |
| **Local Input Capability** | **No input** | NoInputNoOutput | DisplayOnly |
| | **Yes / No** | NoInputNoOutput | DisplayYesNo |
| | **Keyboard** | KeyboardOnly | DisplayYesNo |

*Table 5.5: IO capability mapping*

When a device has OOB authentication information from the remote device, it will indicate it in the LMP_IO_CAPABILITY_RES PDU. When either device has OOB information, the OOB association model will be used.

The Host may allow the Link Manager to ignore the IO capabilities and use the Numeric Comparison protocol with automatic accept by setting the Authentication_Requirements parameter to one of the MITM Protection *Not Required* options.

### 5.2.2.6 IO and OOB capability mapping to authentication stage 1 method

Determining which association model to use in Authentication stage 1 is performed in three steps. First, the devices look at the OOB Authentication Data Present parameter received in the remote IO capabilities. If either device has received OOB authentication data then the OOB association model is used. The event of receiving the OOB information is indicated by a device to its peer in the IO Capability Exchange step of Secure Simple Pairing.

| | | Device A | |
|---|---|---|---|
| | | **Has not received remote OOB authentication data** | **Has received remote OOB authentication data** |
| **Device B** | **Has not received remote OOB authentication data** | Use the IO capability mapping table | Use OOB association with<br>ra = 0<br>rb from OOB |
| | **Has received remote OOB authentication data** | Use OOB association with<br>ra from OOB<br>rb = 0 | Use OOB association with<br>ra from OOB<br>rb from OOB |

*Table 5.6: IO and OOB capability mapping*

Second, if neither device has received OOB authentication data and if both devices have set the Authentication_Requirements parameter to one of the MITM Protection Not

Required options, authentication stage 1 shall function as if both devices set their IO capabilities to DisplayOnly (e.g., Numeric comparison with automatic confirmation on both devices).

Finally, if neither device has received OOB authentication data and if one or both devices have set the Authentication_Requirements parameter to one of the *MITM Protection Required* options, the IO and OOB capabilities are mapped to the authentication stage 1 method as defined in Table 5.7. A Host that has set the Authentication_Requirements parameter to one of the *MITM Protection Required* options shall verify that the resulting Link Key is an Authenticated Combination Key (see [Vol 4] Part E, Section 7.7.24). Table 5.7 also lists whether the combination key results in an authenticated or unauthenticated link key.

| | | **Device A (Initiator)** | | | |
|---|---|---|---|---|---|
| | | **Display Only** | **DisplayYesNo** | **KeyboardOnly** | **NoInputNoOutput** |
| **Device B (Responder)** | **DisplayOnly** | Numeric Comparison with automatic confirmation on both devices. | Numeric Comparison with automatic confirmation on device B only. | Passkey Entry: Responder Display, Initiator Input. | Numeric Comparison with automatic confirmation on both devices. |
| | | Unauthenticated | Unauthenticated | Authenticated | Unauthenticated |
| | **DisplayYesNo** | Numeric Comparison with automatic confirmation on device A only. | Numeric Comparison: Both Display, Both Confirm. | Passkey Entry: Responder Display, Initiator Input. | Numeric Comparison with automatic confirmation on device A only and Yes/No confirmation whether to pair on device B. Device B does not show the confirmation value. |
| | | Unauthenticated | Authenticated | Authenticated | Unauthenticated |
| | **Keyboard Only** | Passkey Entry: Initiator Display, Responder Input. | Passkey Entry: Initiator Display, Responder Input. | Passkey Entry: Initiator and Responder Input. | Numeric Comparison with automatic confirmation on both devices. |
| | | Authenticated | Authenticated | Authenticated | Unauthenticated |

| | | Device A (Initiator) | | | |
|---|---|---|---|---|---|
| | | **Display Only** | **DisplayYesNo** | **KeyboardOnly** | **NoInputNoOutput** |
| | **NoInputNoOutput** | Numeric Comparison with automatic confirmation on both devices. | Numeric Comparison with automatic confirmation on device B only and Yes/No confirmation on whether to pair on device A. Device A does not show the confirmation value. | Numeric Comparison with automatic confirmation on both devices. | Numeric Comparison with automatic confirmation on both devices. |
| | | Unauthenticated | Unauthenticated | Unauthenticated | Unauthenticated |

*Table 5.7: IO capability mapping to authentication stage 1*

Note: The "DisplayOnly" IO capability only provides unidirectional authentication.

#### 5.2.2.7 Out of Band (OOB)

An out of band mechanism may also be used to communicate discovery information as well as other information related to the pairing process.

The contents of the OOB data block are:

Mandatory contents:

- Length (2 bytes)
- BD_ADDR (6 bytes)

Optional contents:

- Class of Device (3 bytes)
- Secure Simple Pairing Hash C (16 bytes)
- Secure Simple Pairing Randomizer R (16 bytes)
- Local name (variable length)
- Other information

The length field includes all bytes in the OOB data block including the length field itself. The BD_ADDR will be a fixed field in the beginning of the OOB data block. Following the BD_ADDR will be zero or more EIR tag fields containing optional contents. The EIR tag format will be used for the optional contents. See Figure 5.6.

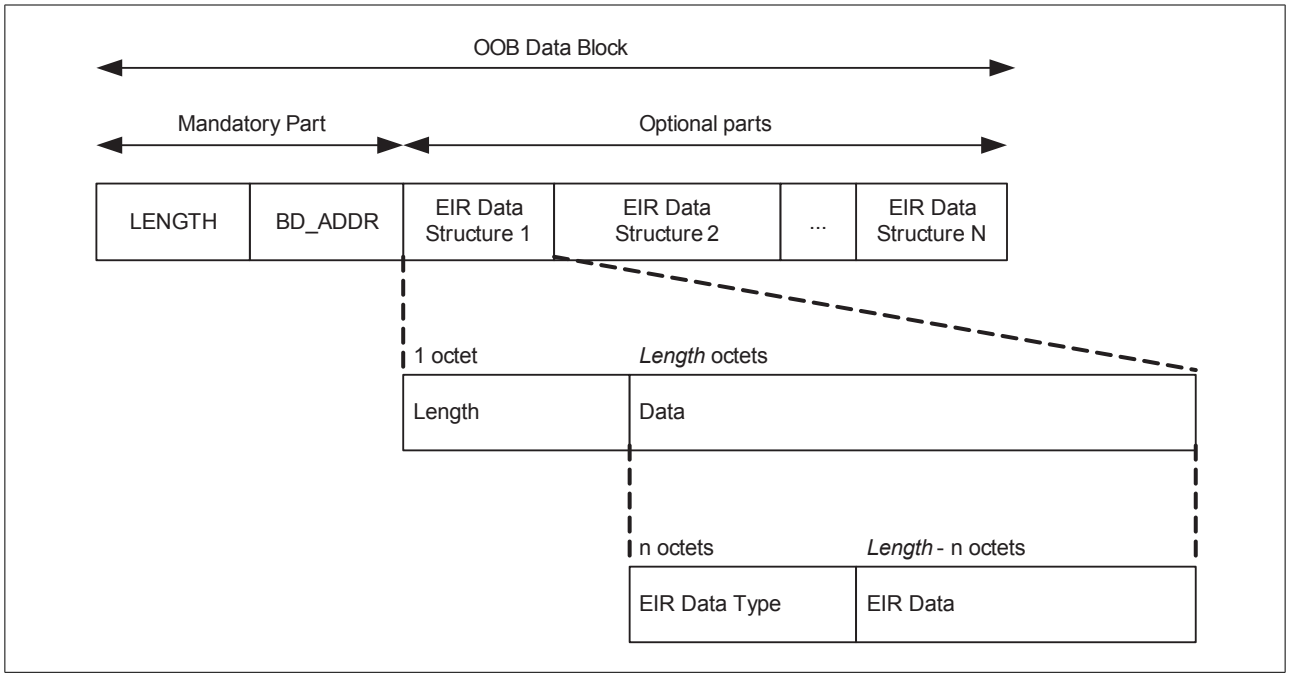



*Figure 5.6: OOB data block format*

If Secure Simple Pairing fails when one or both devices have OOB Authentication Data present, both devices shall discard the OOB Authentication Data and the device that originally initiated authentication shall re-initiate authentication. Although the user may get involved in authentication as defined by the IO capabilities of the two devices, falling back to the in-band association model will prevent deadlock conditions when one or both devices has stale OOB Authentication Data.

There is a MIME type defined for use with the OOB data format. The MIME type can be found from the following link: http://www.iana.org/assignments/media-types/application/vnd.bluetooth.ep.oob

#### 5.2.2.8 Security database

A Bluetooth device in security mode 4 shall classify and enforce the security requirements of its services using the following levels attributes (in order of decreasing security) for use when pairing with remote devices supporting Secure Simple Pairing:

- Level 4, for services with the following attributes:

  Authentication of the remote device required

  MITM protection required

  128-bit equivalent strength for link and encryption keys required using FIPS approved algorithms (E0 not allowed, SAFER+ not allowed, and P-192 not allowed; encryption key not shortened)

  User interaction acceptable

- Level 3, for services with the following attributes:

  Authentication of the remote device required

  MITM protection required

  Encryption required

  At least 56-bit equivalent strength for encryption key shall be used

  At least 128-bit equivalent strength for encryption key should be used

  User interaction acceptable
- Level 2, for services with the following attributes:

  Authentication of the remote device required

  MITM protection not required

  Encryption required

  At least 56-bit equivalent strength for encryption key shall be used

  At least 128-bit equivalent strength for encryption key should be used
- Level 1, for services with the following attributes:

  Authentication of the remote device required when encryption is enabled

  MITM protection not required

  At least 56-bit equivalent strength for encryption key when encryption is enabled shall be used

  At least 128-bit equivalent strength for encryption key should be used

  Minimal user interaction desired
- Level 0: Service requires the following:

  Authentication of the remote device not required

  MITM protection not required

  No encryption required

  No user interaction required

  Security Mode 4 Level 0 shall only be used for:

  a. L2CAP fixed signaling channels with CIDs 0x0001, 0x0003, and 0x003F

  b. SDP

  c. broadcast data sent on the connectionless L2CAP channel (CID 0x0002)

  d. services with the combinations of Service Class UUIDs and L2CAP traffic types listed in Section 1 of [5].

The security level required for each service offered should be stored in a security database that is accessed to determine the type of link key and the encryption key size that is required for access to the respective service. The security level required for service data transmitted on an L2CAP connection-oriented channel may differ from the security level required for service data transmitted on another L2CAP connection-oriented channel or on the connectionless L2CAP channel. Table 5.8 shows the type of link key required for each security level for both remote devices that support Secure Simple Pairing and for those that do not.

| **Security Level Required for Service** | **Link Key type required for remote devices that support SSP** | **Link Key type required for remote devices that do not support SSP** | **Comments** |
|---|---|---|---|
| Level 4<br>• Authentication of the remote device required<br>• MITM protection required<br>• Encryption required<br>• User interaction acceptable | Authenticated<br>(P-256 based Secure Simple Pairing and Secure Authentication) | NA | Highest Security<br>Only possible when both devices support Secure Connections |
| Level 3<br>• Authentication of the remote device required<br>• MITM protection required<br>• Encryption required<br>• User interaction acceptable | Authenticated | Combination<br>(16 digit PIN recommended) | High Security |
| Level 2<br>• Authentication of the remote device required<br>• MITM protection not necessary<br>• Encryption desired | Unauthenticated | Combination | Medium Security |
| Level 1<br>• Authentication of the remote device required when encryption is enabled<br>• MITM protection not necessary<br>• Encryption not necessary[1]<br>• Minimal user interaction desired | Unauthenticated | None | Low Security |

| Security Level Required for Service | Link Key type required for remote devices that support SSP | Link Key type required for remote devices that do not support SSP | Comments |
|---|---|---|---|
| Level 0<br>• Authentication of the remote device not required<br>• MITM protection not necessary<br>• Encryption not necessary<br>• No user interaction desired | None | None | Permitted only for SDP and service data sent via either L2CAP fixed signaling channels or the L2CAP connectionless channel to PSMs that correspond to service class UUIDs which are allowed to use Level 0. |

*Table 5.8: Security level mapping to link key requirements*

[1]Though encryption is not necessary for the service for Level 1, the specification mandates the use of encryption for all services other than SDP when the remote device supports SSP.

An *authenticated* link key is a link key where either the numeric comparison, out-of-band, or passkey entry Secure Simple Pairing association models were used. An authenticated link key has protection against MITM attacks. To ensure that an authenticated link key is created during the Secure Simple Pairing procedure, the Authentication_Requirements parameter should be set to one of the *MITM Protection Required* options.

An *unauthenticated* link key is a link key where the “Just Works” Secure Simple Pairing association model was used (see [Vol 1] Part A, Section 5.2.4). An unauthenticated link key does not have protection against MITM attacks. To allow an unauthenticated link key to be created during the Secure Simple Pairing procedure, the Authentication_Requirements parameter may be set to one of the *MITM Protection Not Required* options.

When both devices support Secure Simple Pairing and at least one device does not support Secure Connections, the strength of the link key is 96 effective bits. When both devices support Secure Connections, the strength of the link key is 128 effective bits. Secure Connections does not change the protection against MITM attacks.

A combination link key is a link key where BR/EDR Legacy Pairing was used to generate the link-key (see [Vol 2] Part C, Section 4.2.2.4).

When both devices support Secure Simple Pairing, GAP shall require at least an unauthenticated link key and enable encryption for service traffic sent or received via connection-oriented L2CAP channels. A profile or protocol may define services that require more security (for example, an authenticated link key) or no security in the case

of SDP or service traffic sent via the L2CAP connectionless channel for services that do not require security.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons. See Section 5.2.1.3 for more information.

The remote Controller's and remote Host's support for Secure Simple Pairing shall be determined by the Link Manager Secure Simple Pairing (Host Support) feature bit.

A previously generated link key is considered “sufficient” if the link key type is of the type required for the service, or of a higher strength. Authenticated link keys are considered higher strength than Unauthenticated or Combination keys. Unauthenticated link keys are considered higher strength than Combination keys.

A device shall enforce an encryption key with at least 128-bit equivalent strength for all services that require Security Mode 4, Level 4. For all other services that require encryption, a device shall enforce an encryption key with at least 56-bit equivalent strength and should enforce an encryption key with at least 128-bit equivalent strength, irrespective of whether the remote device supports Secure Simple Pairing.

After encryption has been enabled, the Host should check the encryption key size using the HCI_Read_Encryption_Key_Size command (see [Vol 4] Part E, Section 7.5.7), the HCI_Encryption_Change [v2] event (see [Vol 4] Part E, Section 7.7.8), or a vendor-specific method.

# 6 IDLE MODE PROCEDURES – BR/EDR PHYSICAL TRANSPORT

The requirements for devices initiating the inquiry and discovery procedures (device A) are specified in Table 6.1. The requirements on the behavior of the responding device (device B) are specified in Table 4.1.

| Procedure | Ref. | Support |
|---|---|---|
| General inquiry | 6.1 | C.1 |
| Limited inquiry | 6.2 | C.1 |
| Name discovery | 6.3 | O |
| Device discovery | 6.4 | O |
| Bonding | 6.5 | O |
| C.1: If initiation of bonding is supported, support for at least one inquiry procedure is mandatory, otherwise optional.<br><br>Note: Devices must support LMP-pairing (see [Vol 2] Part C, Section 4.2.2). | | |

*Table 6.1: Requirements for initiating inquiry and discovery procedures*

## 6.1 General Inquiry

### 6.1.1 Purpose

The purpose of the general inquiry procedure is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, and extended inquiry response information of devices in either general discoverable mode or limited discoverable mode.

The general inquiry procedure should be used for general purpose discovery, i.e. to discover all discoverable devices regardless of whether they are in general discoverable mode or limited discoverable mode. A device which discovers devices using the general inquiry procedure and presents them to users in some fashion should distinguish devices in the limited discoverable mode from those in the general discoverable mode, e.g., by sorting them to the top of a list of discovered devices or highlighting them in some way.

Note: The rationale for distinguishing the devices in limited discoverable mode to the end user is that devices typically enter limited discoverable mode only after explicit action by the end user, indicating that the user's immediate goal is to discover and interact with that specific device.

### 6.1.2 Term on UI level

'Bluetooth Device Inquiry'.

### 6.1.3 Description

Figure 6.1 specifies the procedure.

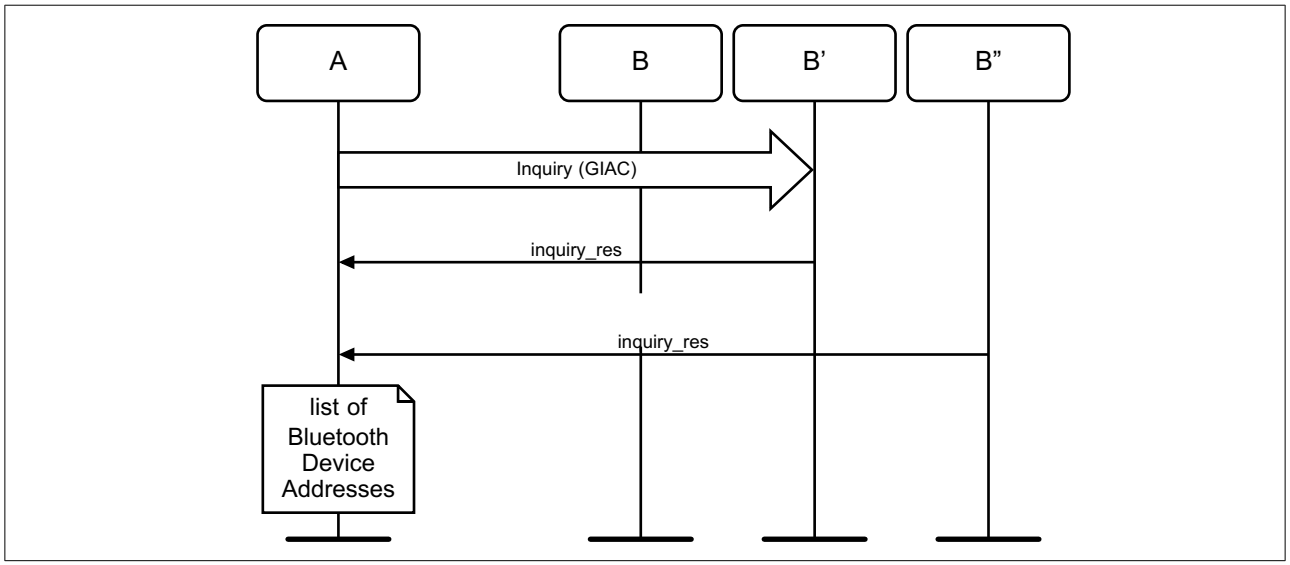



*Figure 6.1: General inquiry, where B is a device in non-discoverable mode, B´ is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note: All discoverable devices are discovered using general inquiry, independent of which discoverable mode they are in.)*

### 6.1.4 Conditions

When general inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the GIAC.

In order for Device A to receive inquiry responses, the remote devices in range have to be made discoverable (limited or general).

## 6.2 Limited Inquiry

### 6.2.1 Purpose

The purpose of the limited inquiry procedure is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, and extended inquiry response information of limited discoverable devices. The latter devices are devices that are in range with regard to the initiator, and set to scan for inquiry messages with the Limited Inquiry Access Code.

The limited inquiry procedure should only be used when it is known that the devices to be discovered are using limited discoverable mode. The general inquiry procedure (see Section 6.1) should be used for general purpose discovery when it is desired to discover all devices regardless of whether they are using limited discoverable mode or general discoverable mode.

### 6.2.2 Term on UI level

'Bluetooth Device Inquiry'.

### 6.2.3 Description

Figure 6.2 specifies the procedure.

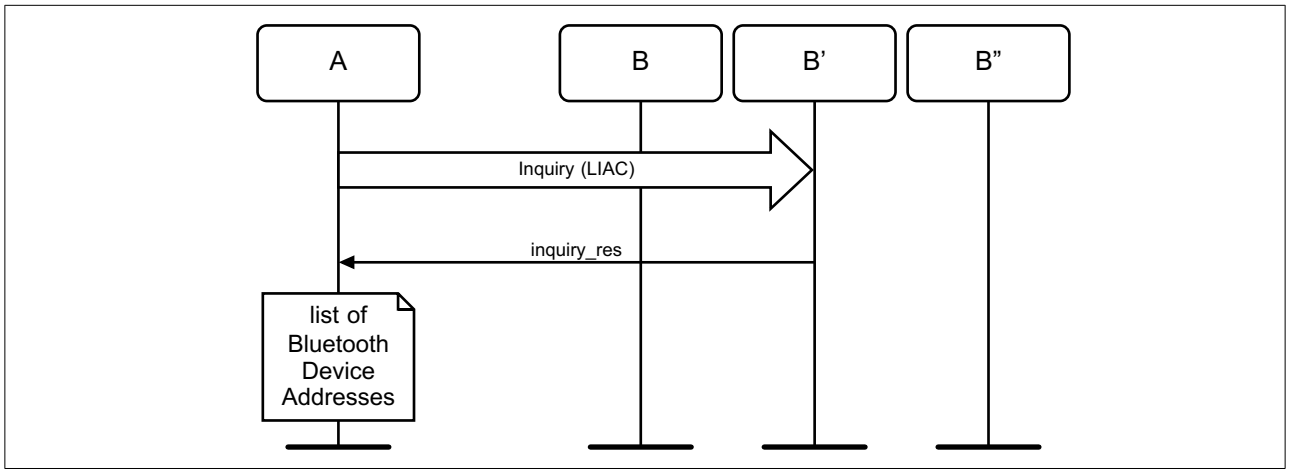



*Figure 6.2: Limited inquiry where B is a device in non-discoverable mode, B' is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note: Only limited discoverable devices can be discovered using limited inquiry.)*

### 6.2.4 Conditions

When limited inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the LIAC.

In order for Device A to receive inquiry responses, the remote devices in range has to be made limited discoverable.

## 6.3 Name Discovery

### 6.3.1 Purpose

The purpose of name discovery is to provide the initiator with the Bluetooth Device Name of connectable devices (i.e., devices in range that will respond to paging).

### 6.3.2 Term on UI level

'Bluetooth Device Name Discovery'

### 6.3.3 Description

#### 6.3.3.1 Name Request

Name request is the procedure for retrieving the Bluetooth Device Name from a connectable Bluetooth device. It is not necessary to perform the full link establishment procedure (see Section 7.1) in order to just to get the name of another device. Figure 6.3 specifies the procedure.

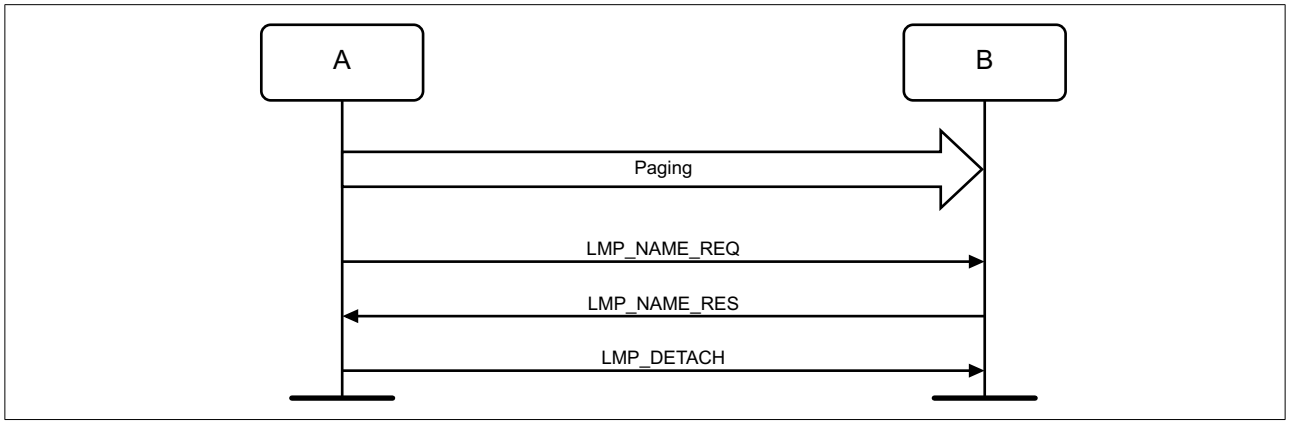



*Figure 6.3: Name Request procedure*

#### 6.3.3.2 Name Discovery

Name discovery is the procedure for retrieving the Bluetooth Device Name from connectable Bluetooth devices by performing name request towards known devices (i.e., Bluetooth devices for which the Bluetooth Device Addresses are available). Figure 6.4 specifies the procedure.

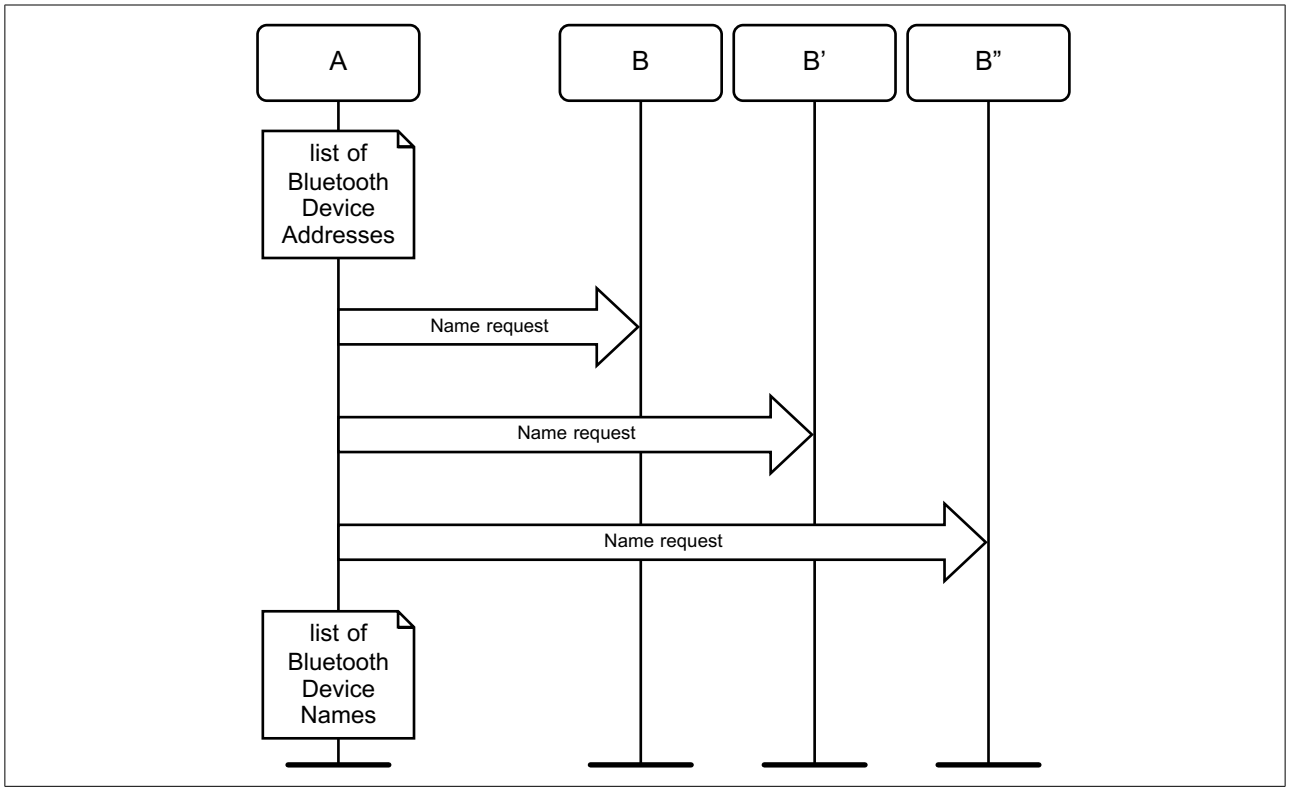



*Figure 6.4: Name discovery procedure*

### 6.3.4 Conditions

In the name request procedure, the initiator will use the Device Access Code of the remote device as retrieved immediately beforehand – normally through an inquiry procedure.

## 6.4 Device Discovery

This section only applies to BR/EDR-only and BR/EDR/LE implementations.

### 6.4.1 Purpose

The purpose of device discovery is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, Bluetooth Device Name, and extended inquiry response information of discoverable devices.

### 6.4.2 Term on UI level

'Bluetooth Device Discovery'

### 6.4.3 Description

During the Device Discovery procedure, first an inquiry (either general or limited) is performed, and then name discovery is done towards some or all of the devices that

responded to the inquiry. If the initiator of the device discovery receives a complete local name or a shortened local name that is considered long enough, via an extended inquiry response from a remote device, the initiator should not do a separate name discovery for that device.

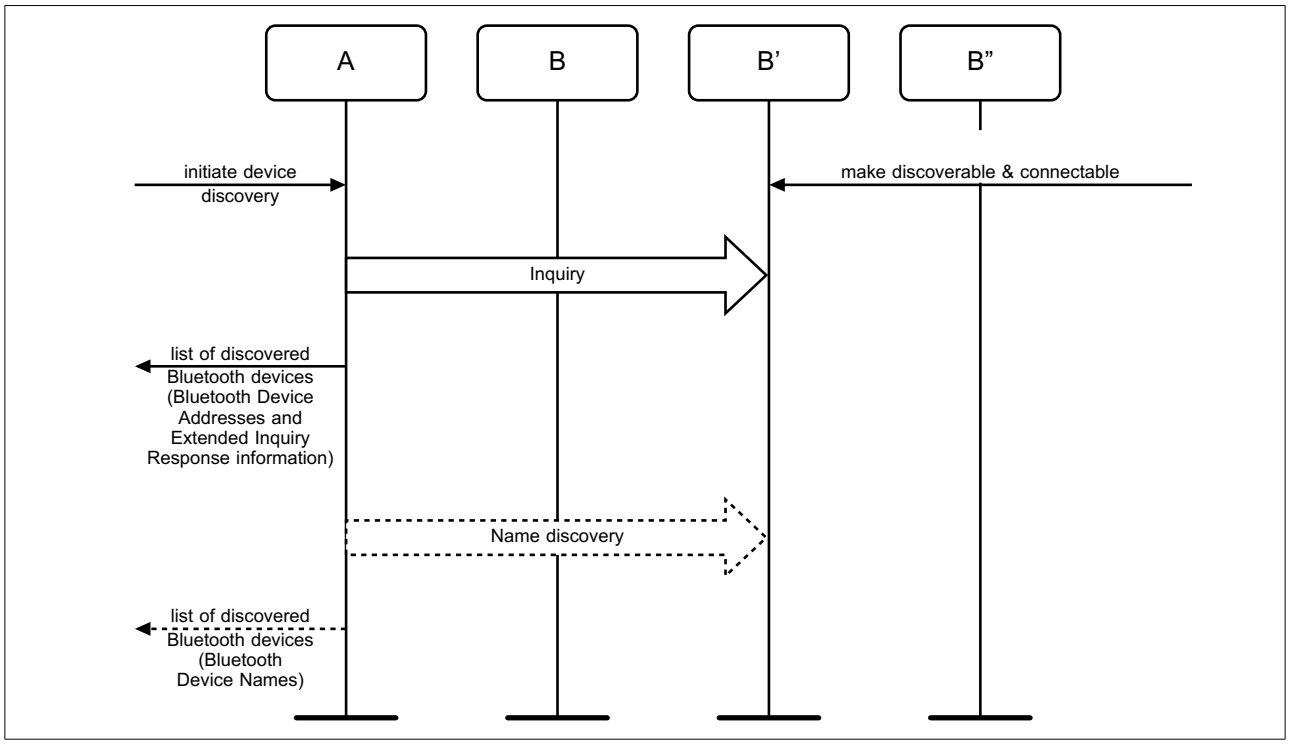



*Figure 6.5: Device Discovery procedure*

### 6.4.4 Conditions

Conditions for both inquiry (general or limited) and name discovery must be fulfilled (i.e., devices discovered during device discovery must be both discoverable and connectable).

## 6.5 Bonding

### 6.5.1 Purpose

The purpose of bonding is to create a relation between two Bluetooth devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both Bluetooth devices, to be used for future authentication. In addition to pairing, the bonding procedure can involve higher-layer initialization procedures.

### 6.5.2 Term on UI level

'Bluetooth Bonding'

### 6.5.3 Description

Two forms of bonding are described in the following sections: General Bonding and Dedicated Bonding.

#### 6.5.3.1 General Bonding

General Bonding refers to the process of performing bonding during connection setup or channel establishment procedures as a precursor to accessing a service. Figure 6.6 specifies General Bonding.

When the devices that are performing General Bonding both support Secure Simple Pairing, the Authentication_Requirements parameter should be set to MITM Protection Not Required – General Bonding unless the security policy of an available local service requires MITM Protection in which case the Authentication_Requirements parameter shall be set to MITM Protection Required – General Bonding. 'No bonding' is used when the device is performing a Secure Simple Pairing procedure, but does not intend to retain the link key after the physical link is disconnected.

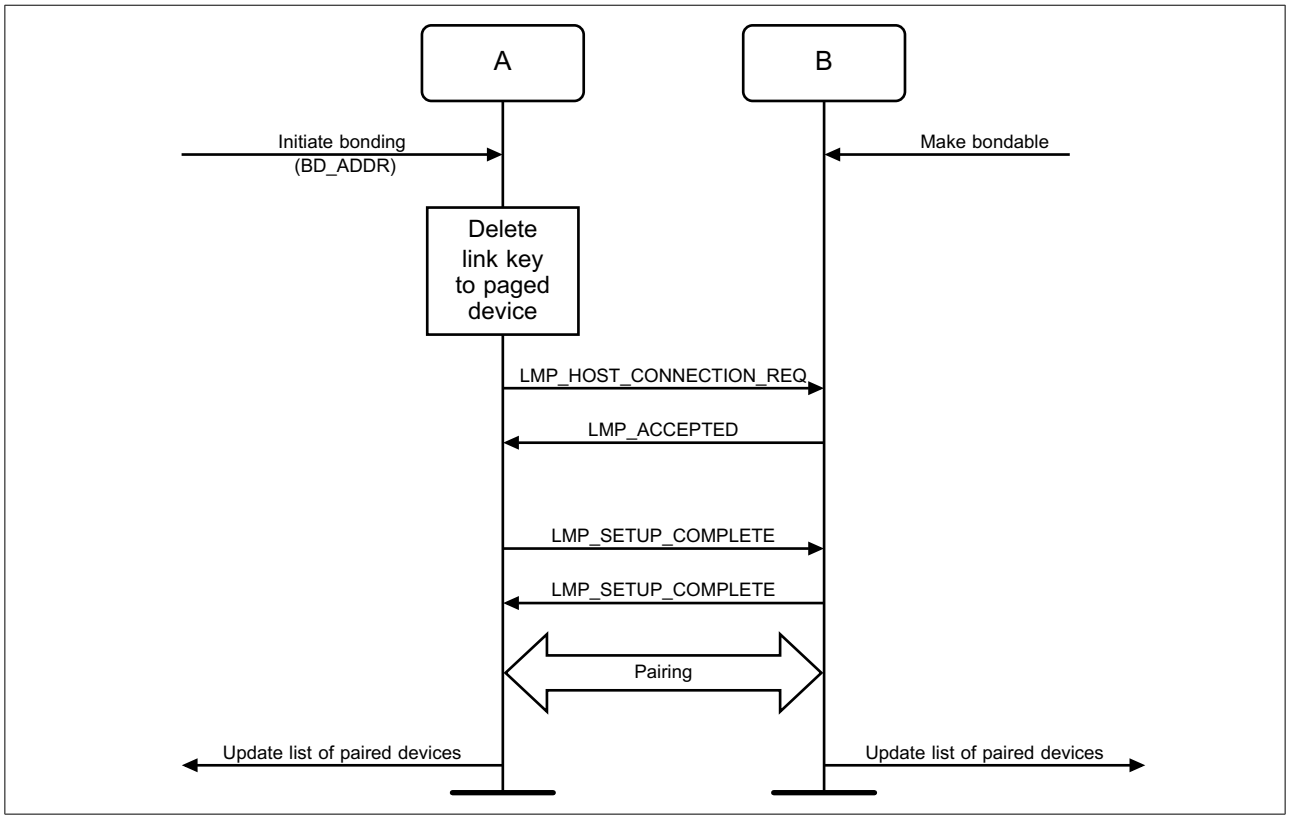



*Figure 6.6: General Bonding procedure*

#### 6.5.3.2 Dedicated Bonding

Dedicated Bonding refers to a procedure wherein one device connects to another only for the purpose of pairing without accessing a particular service. Figure 6.7 specifies

Dedicated Bonding. The main difference with dedicated bonding, as compared to a pairing done during link or channel establishment, is that for bonding it is the paging device (A) that initiates the authentication.

When the devices that are performing Dedicated Bonding both support Secure Simple Pairing, the Authentication_Requirements parameter should be set to *MITM Protection Not Required – Dedicated Bonding* unless the security policy of an available local service requires MITM Protection in which case the Authentication Required parameter shall be set to *MITM Protection Required – Dedicated Bonding*. 'No bonding' is used when the device is performing a Secure Simple Pairing procedure, but does not intend to retain the link key after the physical link is disconnected.

As an exception to the normal process, device B shall also accept pairing before the exchange of LMP_SETUP_COMPLETE PDUs (this can only happen if device A conforms to a lower version of the specification).

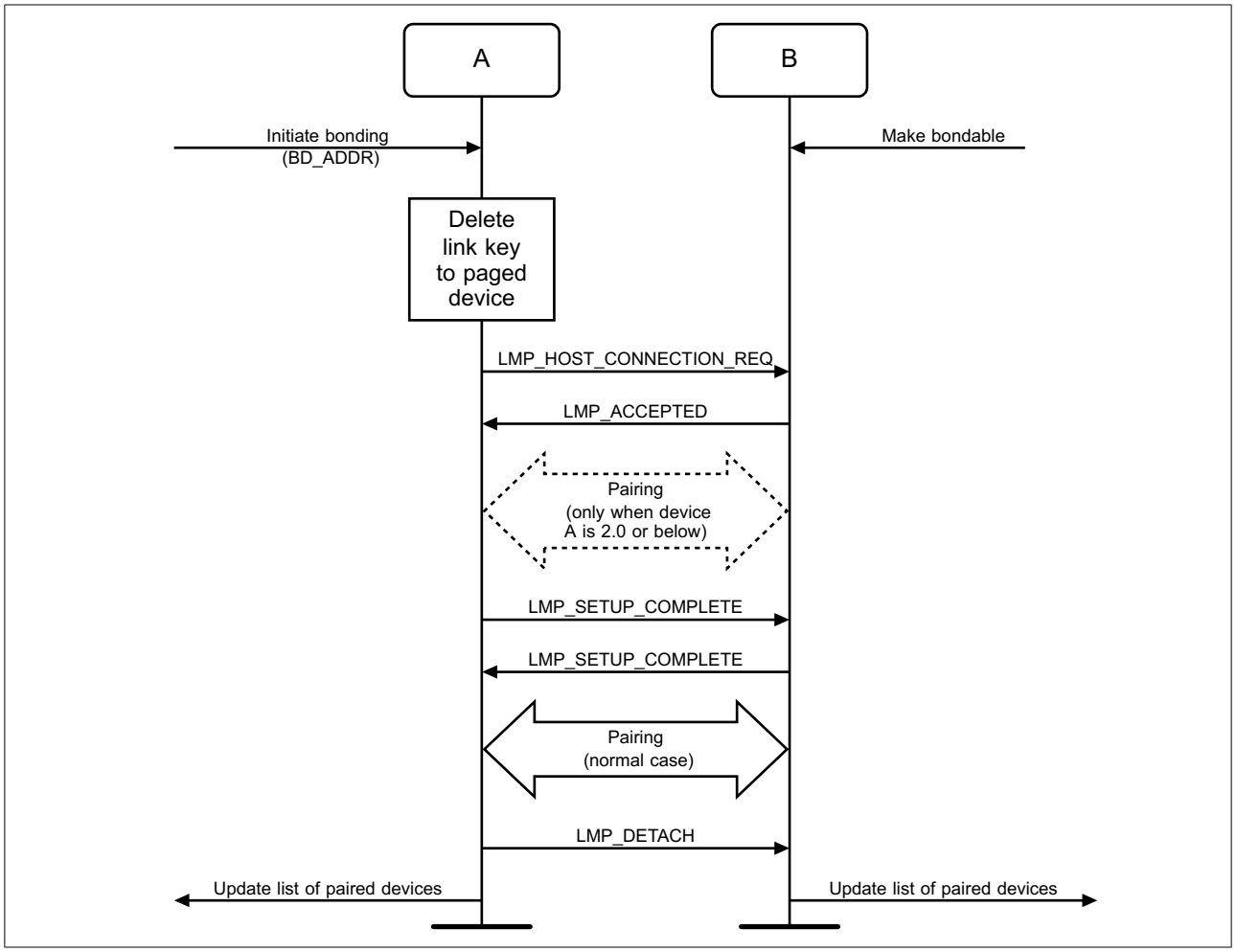



*Figure 6.7: Dedicated Bonding procedure*

### 6.5.4 Conditions

Before bonding can be initiated, the initiating device (A) must know the Device Access Code of the device to pair with. This is normally done by first performing device discovery. A Bluetooth Device that can initiate bonding (A) should use limited inquiry, and a Bluetooth Device that accepts bonding (B) should support the limited discoverable mode.

Bonding is in principle the same as link establishment with the conditions:

- The paged device (B) shall be set into bondable mode. The paging device (A) is assumed to allow pairing since it has initiated the bonding procedure.
- The paging device (the initiator of the bonding procedure, A) shall initiate authentication.
- Before initiating the authentication part of the bonding procedure, the paging device should delete any link key corresponding to a previous bonding with the paged device.

# 7 ESTABLISHMENT PROCEDURES – BR/EDR PHYSICAL TRANSPORT

| Procedure | Ref. | Support in A | Support in B |
|---|---|---|---|
| Link Establishment | 7.1 | M | M |
| Channel Establishment | 7.2 | O | M |
| Connection Establishment | 7.3 | O | O |
| Synchronization Establishment | 7.5 | O | O |

*Table 7.1: Establishment procedures*

The establishment procedures defined here do not include any discovery part. Before establishment procedures are initiated, the information provided during device discovery (in the FHS packet or the extended inquiry response packet of the inquiry response or in the response to a name request or in the synchronization train packet) must be available in the initiating device.

This information is:

- The Bluetooth Device Address (BD_ADDR) from which the Device Access Code is generated
- The system clock of the remote device

Additional information provided during device discovery that may be useful for making the decision to initiate an establishment procedure is:

- The Class of Device
- The Device name
- The supported Service Classes.

## 7.1 Link Establishment

### 7.1.1 Purpose

The purpose of the Link Establishment procedure is to establish a logical transport (of ACL type) between two Bluetooth devices using procedures from the Baseband Specification and Link Manager Protocol.

### 7.1.2 Term on UI level

‘Bluetooth link establishment’

### 7.1.3 Description

In this subsection, the paging device (A) is in security mode 3. During link establishment, the paging device cannot distinguish if the paged device (B) is in security mode 1, 2 or 4.

#### 7.1.3.1 B in security mode 1, 2, or 4

Figure 7.1 specifies the procedure.

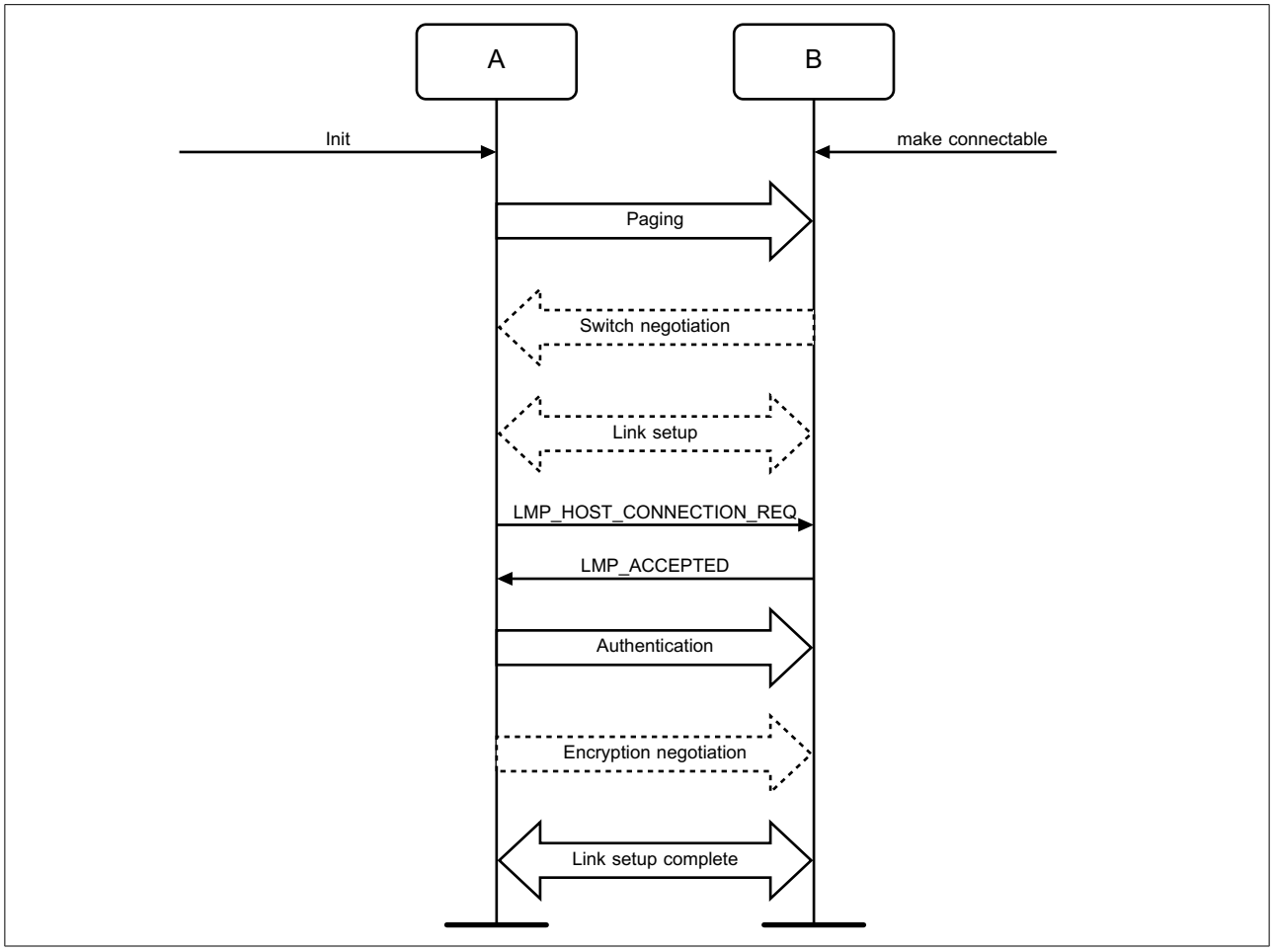



*Figure 7.1: Link Establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1, 2, or 4*

#### 7.1.3.2 B in security mode 3

Figure 7.2 specifies the procedure.

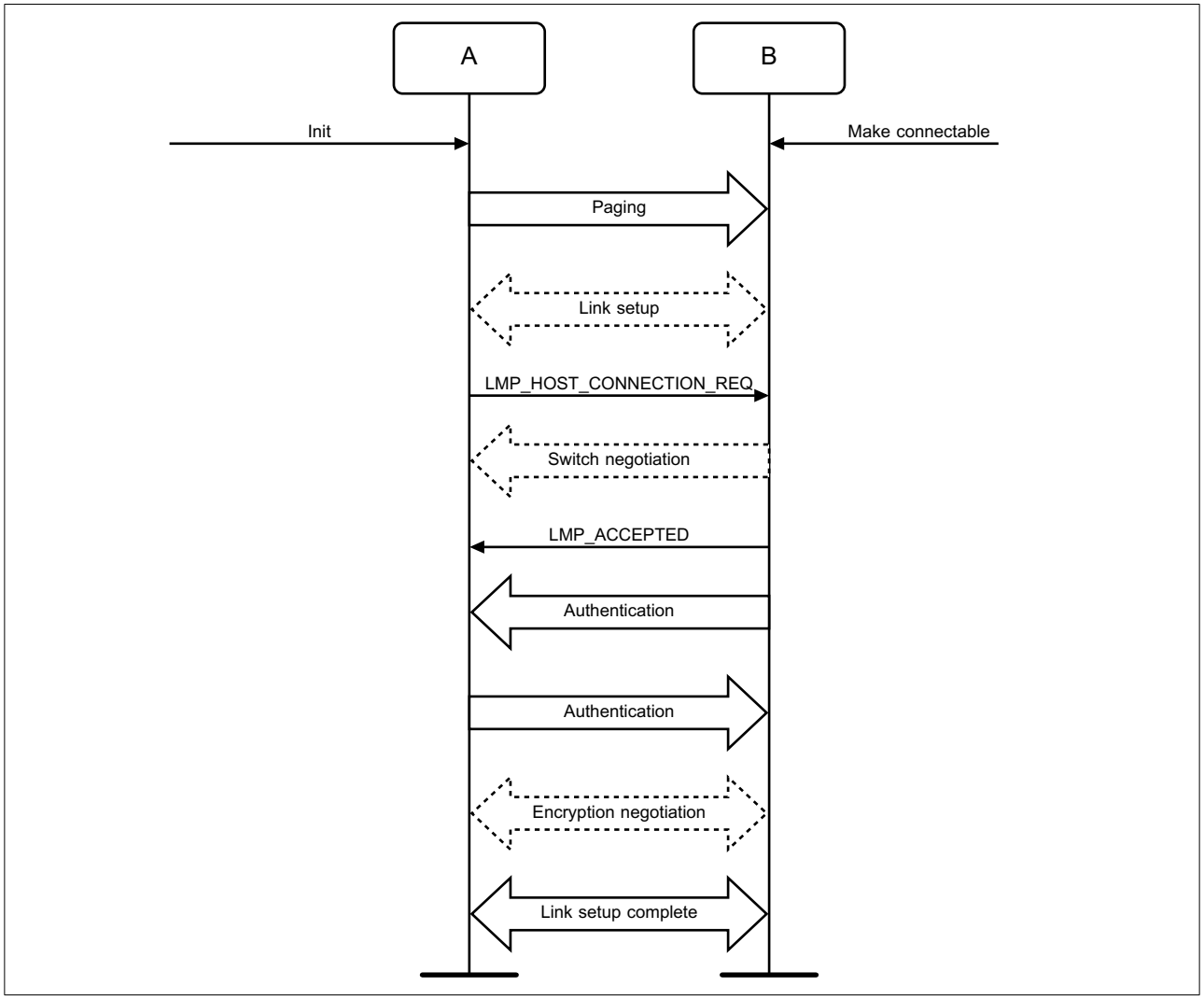



*Figure 7.2: Link Establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3*

### 7.1.4 Conditions

The paging procedure shall be according to [Vol 2] Part B, Section 8.3 and the paging device should use the Device Access Code and page mode received through a previous inquiry. When paging is completed, a physical link between the two Bluetooth devices is established.

If role switching is needed it should be done as early as possible after the physical link is established. If the paging device does not accept the switch, the paged device has to consider whether to keep the physical link or not.

Both devices may perform link setup (using LMP procedures that require no interaction with the Host on the remote side). Optional LMP features can be used after having confirmed (using LMP_FEATURES_REQ) that the other device supports the feature.

When the paging device needs to go beyond the link setup phase, it issues a request to be connected to the Host of the remote device. If the paged device is in security mode 3, this is the trigger for initiating authentication.

The paging device shall send LMP_HOST_CONNECTION_REQ during link establishment (i.e., before channel establishment) and may initiate authentication only after having sent LMP_HOST_CONNECTION_REQ.

After an authentication has been performed, any of the devices can initiate encryption.

Further link configuration may take place after the LMP_HOST_CONNECTION_REQ. When both devices are satisfied, they send LMP_SETUP_COMPLETE.

Link establishment is completed when both devices have sent LMP_SETUP_COMPLETE.

## 7.2 Channel Establishment

### 7.2.1 Purpose

The purpose of the Channel Establishment procedure is to establish a Bluetooth channel (L2CAP channel) between two Bluetooth devices as described in [Vol 3] Part A, Section 4.2.

### 7.2.2 Term on UI level

'Bluetooth channel establishment'

### 7.2.3 Description

In this subsection, the initiator (A) is in security mode 3. During channel establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

#### 7.2.3.1 B in security mode 2 or 4

Figure 7.3 specifies the procedure.

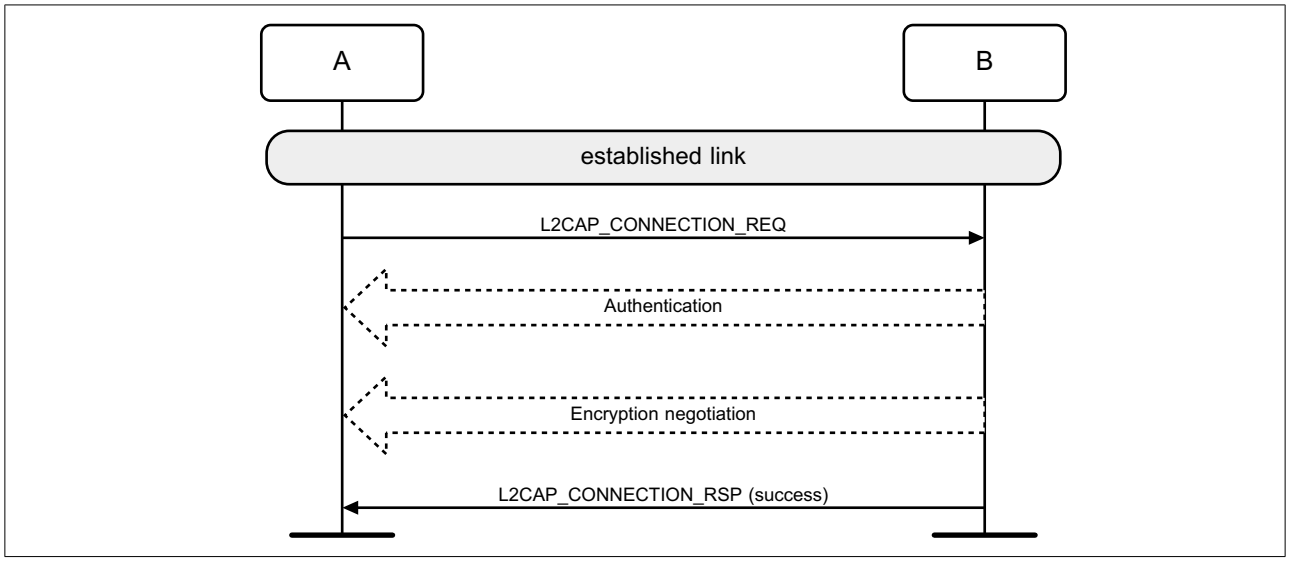



*Figure 7.3: Channel Establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2 or 4*

#### 7.2.3.2 B in security mode 1 or 3

Figure 7.4 specifies the procedure.

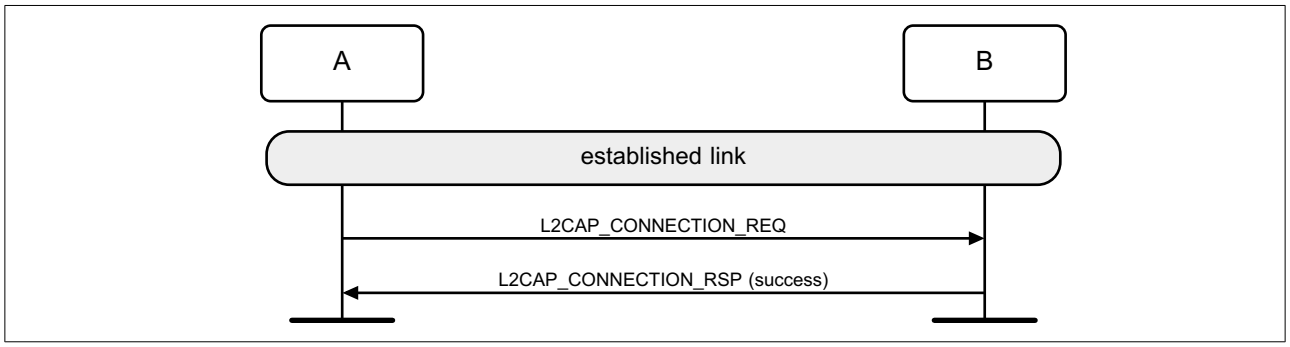



*Figure 7.4: Channel Establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3*

### 7.2.4 Conditions

Channel establishment starts after link establishment is completed when the initiator sends a channel establishment request (L2CAP_CONNECTION_REQ).

Depending on security mode, security procedures may take place after the channel establishment has been initiated.

Channel establishment is completed when the acceptor responds to the channel establishment request (with a positive L2CAP_CONNECTION_RSP).

## 7.3 Connection Establishment

### 7.3.1 Purpose

The purpose of the Connection Establishment procedure is to establish a connection between applications on two Bluetooth devices.

### 7.3.2 Term on UI level

'Bluetooth connection establishment'

### 7.3.3 Description

In this subsection, the initiator (A) is in security mode 3. During connection establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3. The connection establishment request and accept messages are defined by the application protocol.

#### 7.3.3.1 B in security mode 2 or 4

Figure 7.5 specifies the procedure.

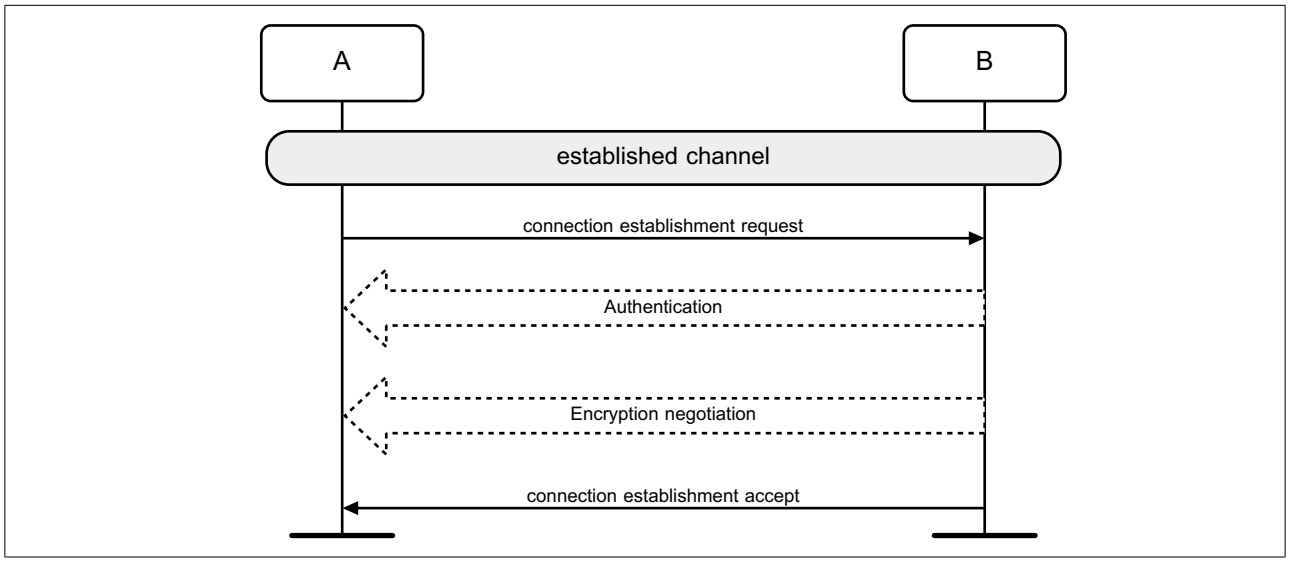



*Figure 7.5: Connection Establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2 or 4*

#### 7.3.3.2 B in security mode 1 or 3

Figure 7.6 specifies the procedure.

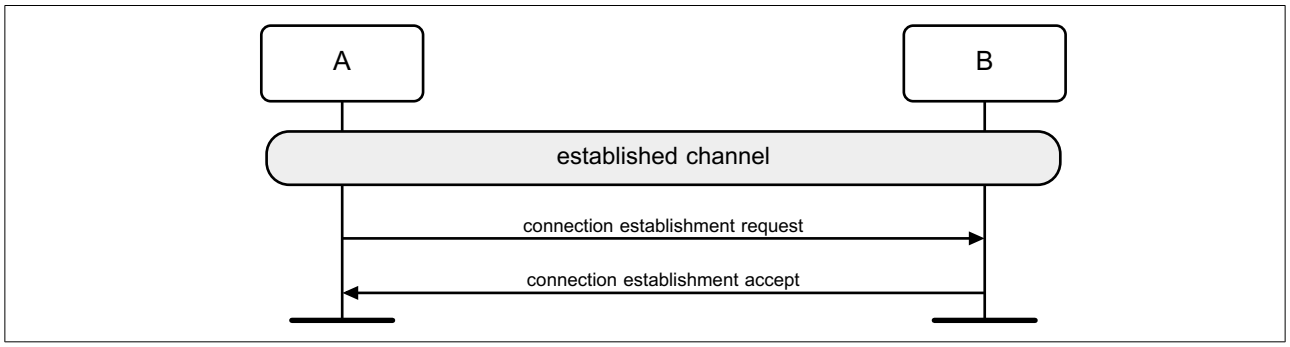



*Figure 7.6: Connection Establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3*

### 7.3.4 Conditions

Connection establishment starts after channel establishment is completed, when the initiator sends a connection establishment request. This request may be a TCS SETUP message [2] in the case of a Bluetooth telephony application Cordless Telephony Profile, or initialization of RFCOMM and establishment of DLC [1] in the case of a serial port-based application Serial Port Profile (although neither TCS or RFCOMM use the term ‘connection’ for this).

Connection establishment is completed when the acceptor accepts the connection establishment request.

## 7.4 Establishment of additional connection

When a Bluetooth device has established one connection with another Bluetooth device, it may be available for establishment of:

- A second connection on the same channel, and/or
- A second channel on the same logical link, and/or
- A second physical link.

If the new establishment procedure is to be towards the same device, the security part of the establishment depends on the security modes used. If the new establishment procedure is to be towards a new remote device, the device should behave according to Active modes independent of the fact that it already has another physical link established (unless allowed co-incident radio and Baseband events have to be handled).

## 7.5 Synchronization Establishment

### 7.5.1 Purpose

The purpose of the Synchronization Establishment procedure is for a device to receive synchronization train packets using the procedures in [Vol 2] Part B, Section 2.7.3

### 7.5.2 Term on UI Level

'Bluetooth synchronization establishment'

### 7.5.3 Description

In Figure 7.7 the receiving device (B) is attempting to receive synchronization train packets from device (A).

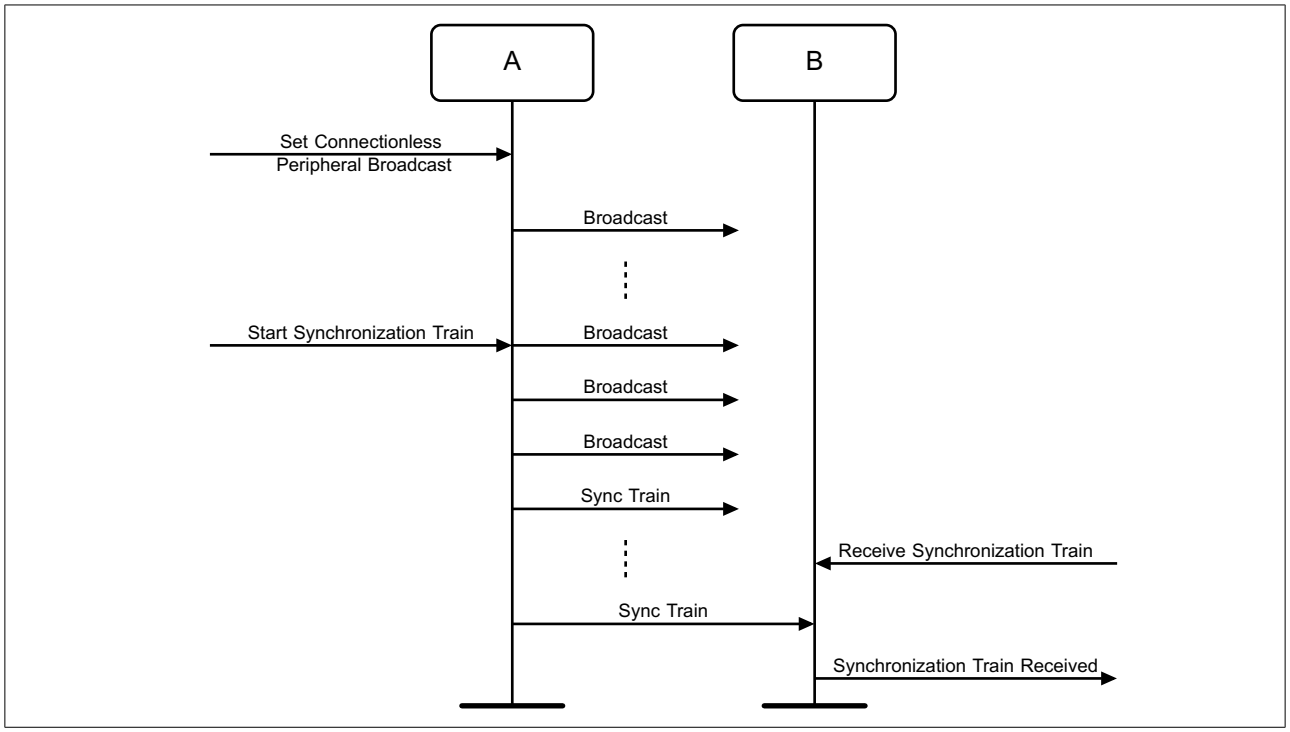



*Figure 7.7: Synchronization Establishment procedure*

### 7.5.4 Conditions

After receiving a synchronization train packet, the receiving device can listen to and receive profile data sent via Connectionless Peripheral Broadcast by device A.

Devices A and B may go through a separate Link Establishment procedure any time they desire to establish an ACL logical transport between each other. They may also use Connectionless Peripheral Broadcast procedures with an ACL logical transport already established.

The receiving device shall enter the Synchronization Scan substate using a scan interval of $T_{GAP}$(Sync_Scan_Interval) and a scan window of $T_{GAP}$(Sync_Scan_Window).

# 8 EXTENDED INQUIRY RESPONSE DATA FORMAT

The extended inquiry response data format is shown in Figure 8.1. The data is 240 octets and consists of a significant part and a non-significant part. The significant part contains a sequence of data structures. Each data structure shall have a length field of one octet, which contains the Length value, and a data field of Length octets. The first n octets of the data field contain the extended inquiry response (EIR) data type. The content of the remaining Length - n octets in the data field depends on the value of the EIR data type and is called the EIR data. The non-significant part extends the extended inquiry response to 240 octets and shall contain all-zero octets.

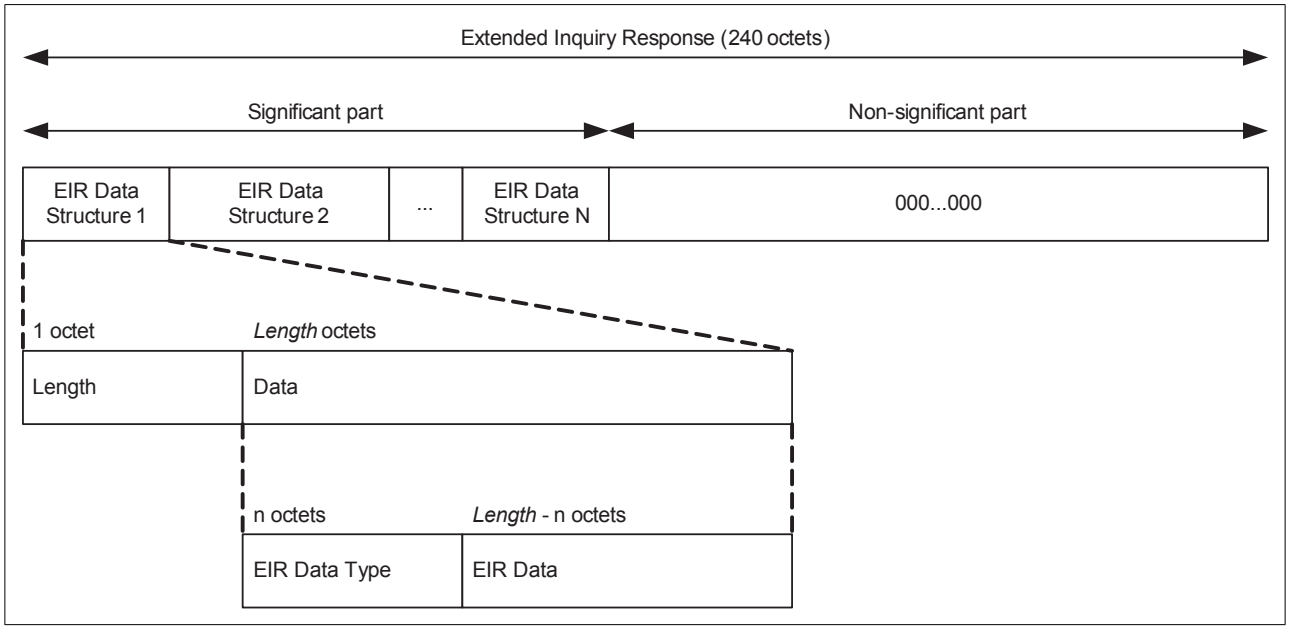



*Figure 8.1: Extended inquiry response data format*

The extended inquiry response data formats and meanings are defined in Section 1 of [4].The extended inquiry response data type values are defined in Assigned Numbers.

If the length field is set to zero, then the data field has zero octets. This shall only occur to allow an early termination of the tagged data.

To reduce interference, the Host should try to minimize the amount of EIR data such that the Baseband can use a 1-slot or 3-slot EIR packet. This is advantageous because it reduces interference and maximizes the probability that the EIR packet will be received. If applications on a Host provide more than 240 bytes of extended inquiry response data, it is up to the Host to limit it to 240 octets.

The EIR data shall be sent during the inquiry response state. In selecting the packet type to be used, FEC (DM1 or DM3) should be considered to maximize the range.

The Host shall include the device name in the EIR data according to the following rules:

1. If the device does not have a device name (i.e., no octets) and
    a. If there is no other data to be sent in the EIR packet, the Host shall send a name tag with zero length and the type field set to indicate that this is the complete name (i.e., total of 2 octets with length = 1).
    b. If there is other important data to be sent in the EIR packet and a zero octet name tag will not fit, the Host may avoid sending the name tag.
2. If the device has a device name (greater than zero octets) and
    a. If it is too long be included in the EIR packet (given the choice of packet type and any other data that is being sent), the Host may send a shortened version of the name (even no octets) and shall mark the name as 'shortened' to inform the receiver that a remote name request is required obtain the full name if the name is needed.
    b. If there are no other data to be sent in the EIR packet (given the choice of packet type selected), the Host shall maximize the length of the device name to be sent, this may be complete or shortened name (e.g., if DM1 packet is chosen and device name characters equates to greater than 15 octets, then Host sends first few characters that equates to 15 octets or less with shortened flag).

If the Host does not understand a given EIR data type value, then it should just skip over Length octets and look for the next EIR data structure.

# 9 OPERATIONAL MODES AND PROCEDURES – LE PHYSICAL TRANSPORT

Several different modes and procedures may be performed simultaneously over an LE physical transport. The following modes and procedures are defined for use over an LE physical transport:

- Broadcast mode and Observation procedure
- Discovery modes and procedures
- Connection modes and procedures
- Bonding modes and procedures
- Periodic advertising modes and procedure
- Isochronous broadcast modes and procedures
- Channel Sounding procedures

Each of the above modes and procedures are independent from each other but are closely related since a combination of the modes and procedures are necessary for most devices to communicate with each other. Both the modes and procedures may be entered or executed respectively as a result of direct user action or autonomously by a device.

The Host shall configure the Controller with its local Link Layer feature information as defined in [Vol 6] Part B, Section 4.6 before performing any of the above modes and procedures.

The types of advertising used in these modes and procedures for each of the associated GAP roles are defined in Section 2.2.2 Table 2.1.

## 9.1 Broadcast mode and Observation procedure

The Broadcast mode and Observation procedure allow two devices to communicate in a unidirectional connectionless manner using the advertising events. The requirements for a device operating in a specific GAP role to support the Broadcast mode and Observation procedure are defined in Table 9.1.

| Broadcast Mode and Observation procedure | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| Broadcast mode | 9.1.1 | E | E | M | E |
| Observation procedure | 9.1.2 | E | E | E | M |

*Table 9.1: Broadcast mode and observation procedure requirements*

### 9.1.1 Broadcast mode

#### 9.1.1.1 Definition

The Broadcast mode provides a method for a device to send connectionless data in advertising events.

#### 9.1.1.2 Conditions

A device in the Broadcast mode shall send data using non-connectable advertising events.

A device in the Broadcast mode may send non-connectable and non-scannable undirected or non-connectable and non-scannable directed advertising events anonymously by omitting the Broadcaster's address.

The advertising data shall be formatted using the Advertising Data (AD) type format as defined in Section 1.3 of [4]. A device in the Broadcast mode shall not set the ‘LE General Discoverable Mode’ flag or the ‘LE Limited Discoverable Mode’ flag in the Flags AD Type as defined in Section 1.3 of [4].

Note: All data sent by a device in the Broadcast mode is considered unreliable since there is no acknowledgment from any device that might have received the data.

The device may configure and enable multiple independent advertising sets. Each advertising set may have an independent advertising filter policy.

### 9.1.2 Observation procedure

#### 9.1.2.1 Definition

The Observation procedure provides a method for a device to receive connectionless data from a device that is sending advertising events.

#### 9.1.2.2 Conditions

A device performing the Observation procedure may use passive scanning or active scanning to receive advertising events.

A device performing the Observation procedure may use active scanning to also receive scan response data sent by any device in the Broadcast mode that advertises using scannable advertising events.

When a device performing the Observation procedure receives a resolvable private address in the advertising event, the device may resolve the private address by using the resolvable private address resolution procedure as defined in Section 10.8.2.3.

In use cases where a device in the Broadcast mode sends dynamic data, the receiving device should disable duplicate filtering capability in the Controller so that the Host receives all advertising packets received by the Controller.

## 9.2 Discovery modes and procedures

All devices shall be in either non-discoverable mode or one of the discoverable modes. A device in the discoverable mode shall be in either the general discoverable mode or the limited discoverable mode. A device in the non-discoverable mode is not discoverable. Devices operating in either the general discoverable mode or the limited discoverable mode can be found by the discovering device. A device that is discovering other devices performs either the limited discovery procedure as defined in Section 9.2.5 or the general discovery procedure as defined in Section 9.2.6.

Some devices may only scan for advertising events using legacy advertising PDUs. It is therefore recommended that devices using advertising events with the extended advertising PDUs also use an advertising set with advertising events that use legacy advertising PDUs.

If the device is in one of the discoverable modes, and if multiple advertising sets are used with the same Identity Address or the same IRK, then those advertising sets shall also share the same advertising filter policy.

### 9.2.1 Requirements

| Discovery modes and procedures | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| Non-Discoverable mode | 9.2.2 | M | E | E | E |
| Limited Discoverable mode | 9.2.3 | O | E | E | E |
| General Discoverable mode | 9.2.4 | C.1 | E | E | E |
| Limited Discovery procedure | 9.2.5 | E | O | E | E |
| General Discovery procedure | 9.2.6 | E | M | E | E |
| Name Discovery procedure | 9.2.7 | O | O | E | E |
| C.1: Optional if limited discoverable mode is supported, otherwise mandatory. | | | | | |

*Table 9.2: Device Discovery requirements*

### 9.2.2 Non-discoverable mode

#### 9.2.2.1 Description

A device configured in non-discoverable mode will not be discovered by any device that is performing either the general discovery procedure or the limited discovery procedure.

#### 9.2.2.2 Conditions

A device in the non-discoverable mode may send advertising events. If the device sends advertising events, it shall not set the ‘LE General Discoverable Mode’ flag or ‘LE Limited Discoverable Mode’ flag in the Flags AD type (see Section 1.3 of [4]).

If the device sends advertising events, then it is recommended that the Host configures the Controller as follows:

- The Host should set the advertising filter policy for all advertising sets to either ‘process scan and connection requests only from devices in the Filter Accept List’ or ‘process scan and connection requests from all devices’.
- The Host should set the advertising intervals as defined in Section 9.3.11.

### 9.2.3 Limited Discoverable mode

#### 9.2.3.1 Description

Devices configured in the limited discoverable mode are discoverable for a limited period of time by other devices performing the limited or general device discovery procedure. Devices typically enter the limited discoverable mode when a user performs a specific action.

There are two common reasons to use limited discoverable mode:

- Limited discoverable mode can be used to allow remote devices using the general discovery procedure to prioritize or otherwise identify devices in limited discoverable mode when presenting discovered devices to the end user because, typically, the user is interacting with them.
- Limited discoverable mode can also be used to allow remote devices using the limited discovery procedure to filter out devices using the general discoverable mode.

#### 9.2.3.2 Conditions

A device in the limited discoverable mode shall send advertising event types with the advertising data including the Flags AD type as defined in Section 1.3 of [4] with all the following flags set as described:

- The LE Limited Discoverable Mode flag set to one.
- The LE General Discoverable Mode flag set to zero.
- For an LE-only implementation with all the following flags set as described:
  a. The ‘BR/EDR Not Supported’ flag to set one.
  b. The ‘Simultaneous LE and BR/EDR to Same Device Capable (Controller)’ flag set to zero.

The advertising data should also include the following AD types to enable a faster connectivity experience:

- TX Power Level AD type defined in Section 1.5 of [4].
- Local Name AD type defined in Section 1.2 of [4].
- Service or Service Class UUIDs AD type defined in Section 1.1 of [4].
- Peripheral Connection Interval Range AD type as defined in Section 1.9 of [4].

Devices shall remain in the limited discoverable mode no longer than $T_{GAP}$(lim_adv_timeout).

While a device is in limited discoverable mode the Host configures the Controller as follows:

- The Host shall set the advertising filter policy for all advertising sets that share the same Identity Address or the same IRK to ‘process scan and connection requests from all devices’.
- The Host should set the advertising intervals as defined in Section 9.3.11.

The device shall remain in limited discoverable mode until a connection is established or the Host terminates the mode.

Note: The choice of advertising interval is a trade-off between power consumption and device discovery time.

The device may configure and enable multiple independent advertising sets.

### 9.2.4 General Discoverable mode

#### 9.2.4.1 Description

Devices configured in the general discoverable mode are discoverable for an indefinite period of time by devices performing the general discovery procedure. Devices typically enter general discoverable mode autonomously.

Devices in the general discoverable mode will not be discovered by devices performing the limited discovery procedure. General discoverable mode should not be used if it is known that the device performing discovery will be using the limited discovery procedure (see Section 9.2.5).

#### 9.2.4.2 Conditions

A device in general discoverable mode shall send advertising events with the advertising data including the Flags AD data type as defined in Section 1.3 of [4] with all the following flags set as described:

- The LE Limited Discoverable Mode flag set to zero.
- The LE General Discoverable Mode flag set to one.
- For an LE-only implementation with all the following flags set as described:
  a. The ‘BR/EDR Not Supported’ flag set to one.
  b. The ‘Simultaneous LE and BR/EDR to Same Device Capable (Controller)’ flag set to zero.

The advertising data should also include the following AD types to enable a faster connectivity experience:

- TX Power Level AD type as defined in Section 1.5 of [4].
- Local Name AD type as defined in Section 1.2 of [4].
- Service or Service Class UUIDs AD type as defined in Section 1.1 of [4].
- Peripheral Connection Interval Range AD type as defined in Section 1.9 of [4].

While a device is in general discoverable mode the Host configures the Controller as follows:

- The Host shall set the advertising filter policy for all advertising sets that share the same Identity Address or the same IRK to ‘process scan and connection requests from all devices’.
- The Host should set the advertising intervals as defined in Section 9.3.11.

The device shall remain in general discoverable mode until a connection is established or the Host terminates the mode.

Host data used in legacy advertising events that change frequently should be placed in the advertising data and static data should be placed in the scan response data.

Note: The choice of advertising interval is a trade-off between power consumption and device discovery time.

### 9.2.5 Limited Discovery procedure

#### 9.2.5.1 Description

A device performing the limited discovery procedure receives the device address, advertising data and scan response data from devices in the limited discoverable mode only.

The limited discovery procedure should only be used when it is known that the devices to be discovered are using limited discoverable mode. The general discovery procedure (see Section 9.2.6) should be used for general purpose discovery when it is desired to discover all devices regardless of whether they are using limited discoverable mode or general discoverable mode.

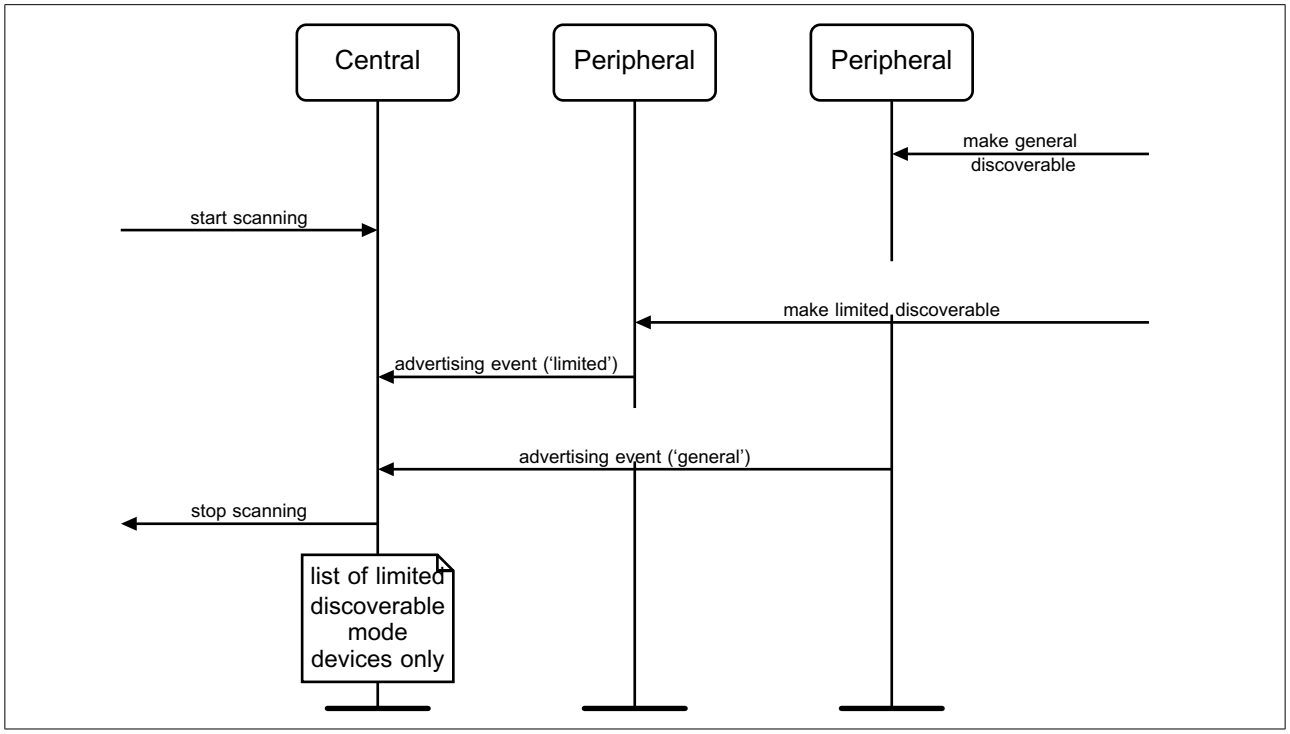



*Figure 9.1: A Central performing limited discovery procedure discovering Peripherals in the limited discoverable mode*

#### 9.2.5.2 Conditions

It is recommended that the device scan on all the PHYs it supports.

When a Host performs the limited discovery procedure, the Host configures the Controller as follows:

1. The Host shall set the scanning filter policy to an unfiltered scanning policy (see [Vol 6] Part B, Section 4.3.3).
2. The Host should set the scan interval and scan window as defined in Section 9.3.11.
3. The Host should configure the Controller to use active scanning.

The Host shall begin scanning for advertising packets and should continue for a minimum of $T_{GAP}$(lim_disc_scan_min) when scanning on the LE 1M PHY and $T_{GAP}$(lim_disc_scan_min_coded) when scanning on the LE Coded PHY, unless the Host ends the limited discovery procedure.

The Host shall check for the Flags AD type in the advertising data. If the Flags AD type is present and the LE Limited Discoverable Flag is set to one then the Host shall consider the device as a discovered device, otherwise the advertising data shall be ignored. The Flag AD type is defined in Section 1.3 of [4]. The advertising data of the discovered device may contain data with other AD types, e.g. Service or Service Class UUIDs AD type, TX Power Level AD type, Local Name AD type, Peripheral Connection Interval Range AD type. The Host may use the data in performing any of the connection establishment procedures.

The Host shall ignore the 'Simultaneous LE and BR/EDR to Same Device Capable (Controller)' bit in the Flags AD type.

### 9.2.6 General Discovery procedure

#### 9.2.6.1 Description

A device performing the general discovery procedure receives the device address, advertising data and scan response data from devices in the limited discoverable mode or the general discoverable mode.

The general discovery procedure should be used for general purpose discovery, i.e. to discover all discoverable devices regardless of whether they are in general discoverable mode or limited discoverable mode. A device which discovers devices using the general discovery procedure and presents them to users in some fashion should distinguish devices in the limited discoverable mode from those in the general discoverable mode, e.g., by sorting them to the top of a list of discovered devices or highlighting them in some way.

Note: The rationale for distinguishing the devices in limited discoverable mode to the end user is that devices typically enter limited discoverable mode only after explicit action by the end user, indicating that the user's immediate goal is to discover and interact with that specific device.

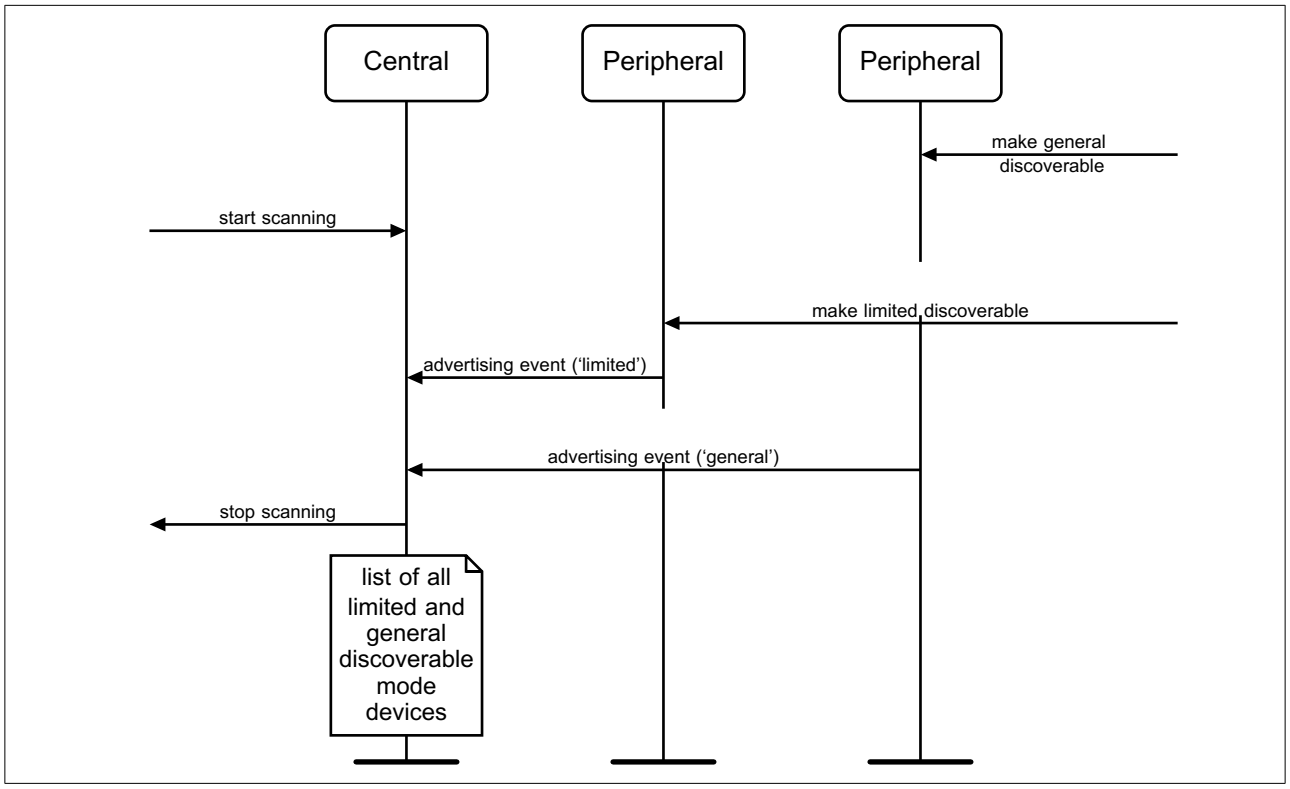



*Figure 9.2: A Central performing General Discovery procedure discovering Peripherals in the Limited Discoverable mode and General Discoverable mode*

### 9.2.6.2 Conditions

It is recommended that the device scan on all the PHYs it supports.

When a Host performs the general discovery procedure, the Host configures the Controller as follows:

1. The Host shall set the scanning filter policy to an unfiltered scanning policy (see [Vol 6] Part B, Section 4.3.3).
2. The Host should set the scan interval and scan window as defined in Section 9.3.11.
3. The Host should configure the Controller to use active scanning.

The Host shall begin scanning for advertising packets and should continue for a minimum of $T_{GAP}$(gen_disc_scan_min) when scanning on the LE 1M PHY and

$T_{GAP}$(gen_disc_scan_min_coded) when scanning on the LE Coded PHY. The procedure may be terminated early by the Host.

The Host shall check for the Flags AD type in the advertising data. If the Flags AD type (see Section 1.3 of [4]) is present and either the LE General Discoverable Mode flag is set to one or the LE Limited Discoverable Mode flag is set to one then the Host shall consider the device as a discovered device, otherwise the advertising data shall be ignored. The advertising data of the discovered device may contain data with other AD types, e.g., Service or Service Class UUIDs AD type, TX Power Level AD type, Local Name AD type, Peripheral Connection Interval Range AD type. The Host may use the data in performing any of the connection establishment procedures as defined in Section 9.3.

The Host shall ignore the 'Simultaneous LE and BR/EDR to Same Device Capable (Controller)' bit in the Flags AD type.

### 9.2.7 Name Discovery procedure

#### 9.2.7.1 Description

The name discovery procedure is used to obtain the Bluetooth Device Name of a remote connectable device.

#### 9.2.7.2 Conditions

If the complete device name is not acquired while performing either the limited discovery procedure or the general discovery procedure, then the name discovery procedure may be performed.

The name discovery procedure shall be performed as follows:

1. The Host shall establish a connection using one of the connection establishment procedures as defined in Section 9.3.
2. The Host shall read the device name characteristic using the GATT procedure Read Using Characteristic UUID [Vol 3] Part G, Section 4.8.2
3. The connection may be terminated after the GATT procedure has completed.

## 9.3 Connection modes and procedures

The connection modes and procedures allow a device to establish a connection to another device.

When devices are connected, the parameters of the connection can be updated with the Connection Parameter Update procedure. The connected device may terminate the connection using the Terminate Connection procedure. The requirements for a

device to support the connection modes and procedures are defined in Table 9.3. These requirements refer to the specific role a device is operating in. Devices supporting multiple roles shall support the specified modes and procedures for a given role while operating in that role.

### 9.3.1 Requirements

| Connection Modes and Procedures | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| Non-connectable mode | 9.3.2 | M | E | M | M |
| Directed connectable mode | 9.3.3 | O | E | E | E |
| Undirected connectable mode | 9.3.4 | M | E | E | E |
| Auto connection establishment procedure | 9.3.5 | E | O | E | E |
| General connection establishment procedure | 9.3.6 | E | O | E | E |
| Selective connection establishment procedure | 9.3.7 | E | O | E | E |
| Direct connection establishment procedure | 9.3.8 | E | M | E | E |
| Periodic Advertising Connection procedure | 9.3.17 | C.3 | C.4 | E | E |
| Connection parameter update procedure | 9.3.9 | O | M | E | E |
| Terminate connection procedure | 9.3.10 | M | M | E | E |
| Connected Isochronous Stream Central Establishment procedure | 9.3.13 | E | C.1 | E | E |
| Connected Isochronous Stream Peripheral Establishment procedure | 9.3.14 | C.5 | E | E | E |
| Connected Isochronous Stream Terminate procedure | 9.3.15 | C.5 | C.1 | E | E |
| Connection Subrate procedure | 9.3.16 | C.2 | C.2 | E | E |

C.1: Mandatory if the Connected Isochronous Stream - Central feature is supported, otherwise excluded.

C.2: Mandatory if the Connection Subrating feature is supported, otherwise excluded.

C.3: Mandatory if the Periodic Advertising with Responses - Scanner feature is supported, otherwise Excluded

C.4: Mandatory if the Periodic Advertising with Responses - Advertiser feature is supported, otherwise Excluded

C.5: Mandatory if the Connected Isochronous Stream - Peripheral feature is supported, otherwise excluded.

*Table 9.3: Connection modes and procedures requirements*

### 9.3.2 Non-connectable mode

#### 9.3.2.1 Description

A device in the non-connectable mode shall not allow a connection to be established.

#### 9.3.2.2 Conditions

A Peripheral in the non-connectable mode may send non-connectable advertising events. In this case it is recommended that the Host configures the Controller as follows:

- The Host should set the advertising filter policy to either ‘process scan and connection requests only from devices in the Filter Accept List’ or ‘process scan and connection requests from all devices’.
- The Host should set the advertising intervals as defined in Section 9.3.11.

The device may configure and enable multiple independent advertising sets. Each advertising set may have an independent advertising filter policy.

### 9.3.3 Directed Connectable mode

#### 9.3.3.1 Description

A device in the directed connectable mode shall accept a connection request from a known peer device performing the auto connection establishment procedure or the general connection establishment procedure.

#### 9.3.3.2 Conditions

A Peripheral shall send connectable directed advertising events.

The device may configure and enable multiple independent advertising sets. Each advertising set may have an independent advertising filter policy.

### 9.3.4 Undirected Connectable mode

#### 9.3.4.1 Description

A device in the undirected connectable mode shall accept a connection request from a device performing the auto connection establishment procedure or the general connection establishment procedure.

#### 9.3.4.2 Conditions

A Peripheral should follow the guidelines defined in Section 9.3.11. A Peripheral shall send either connectable and scannable undirected advertising events or connectable undirected advertising events.

The device may configure and enable multiple independent advertising sets. Each advertising set may have an independent advertising filter policy.

### 9.3.5 Auto Connection Establishment procedure

#### 9.3.5.1 Description

The auto connection establishment procedure allows the Host to configure the Controller to autonomously establish a connection with one or more devices in the directed connectable mode or the undirected connectable mode. This procedure uses the Filter Accept List in the initiator to store the addresses of the devices that can be connected to. The Controller autonomously establishes a connection with a device with the device address that matches the address stored in the Filter Accept List.

#### 9.3.5.2 Conditions

Figure 9.3 shows the flow chart for a device performing the auto connection establishment procedure.

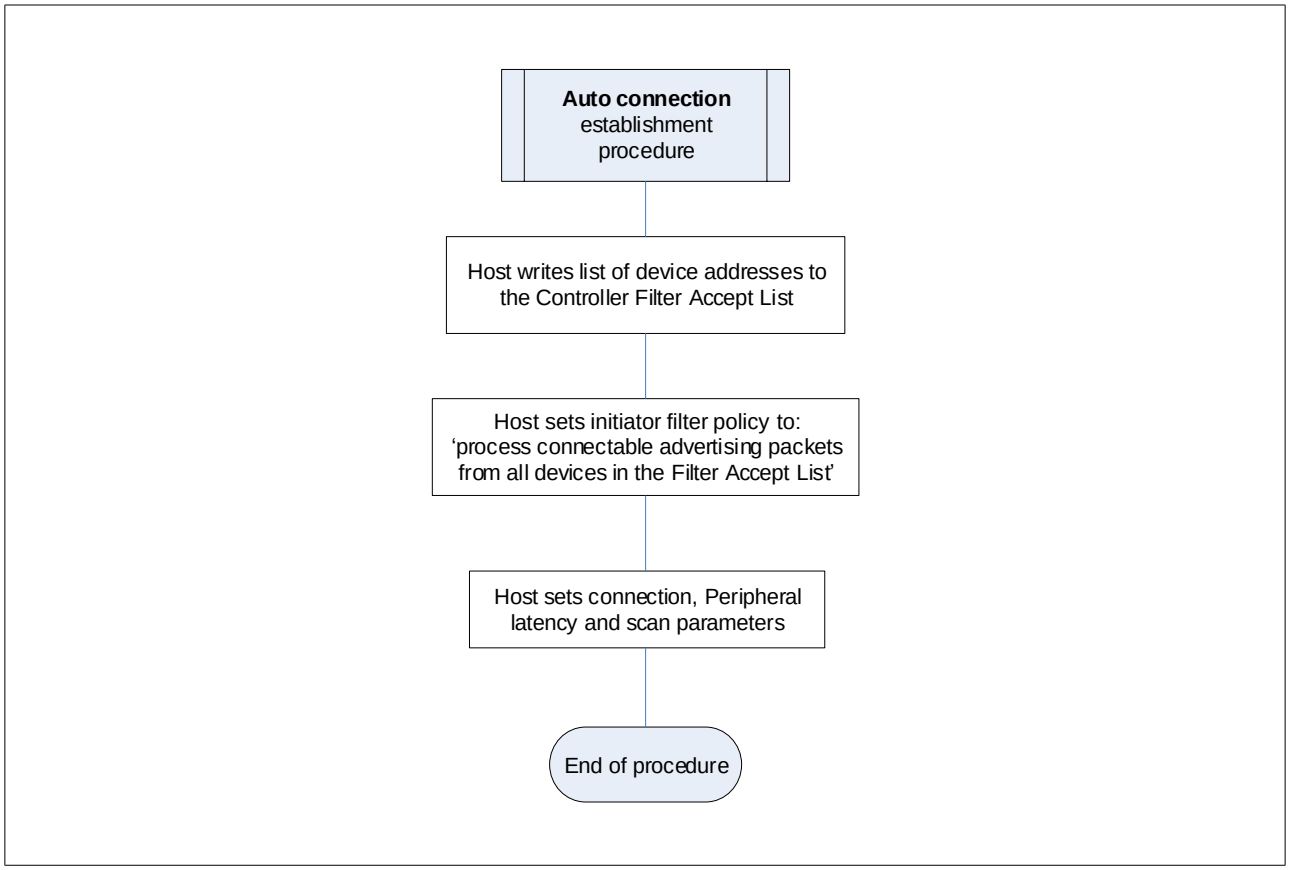



*Figure 9.3: Flow chart for a device performing the Auto Connection Establishment procedure*

When a Host performs the auto connection establishment procedure, the Host configures the Controller as follows:

1. The Host shall write the list of device addresses that are to be auto connected to into the Filter Accept List.
2. The Host shall set the initiator filter policy to 'process connectable advertising packets from all devices in the Filter Accept List'.
3. The Host should set the scan interval and scan window as defined in Section 9.3.11.
4. The Host should set connection parameters as defined in Section 9.3.12.

This procedure is terminated when a connection is established or when the Host terminates the procedure.

### 9.3.6 General Connection Establishment procedure

#### 9.3.6.1 Description

The general connection establishment procedure allows the Host to establish a connection with a set of known peer devices in the directed connectable mode or the undirected connectable mode.

### 9.3.6.2 Conditions

Figure 9.4 shows the flow chart for a device performing the general connection establishment procedure.

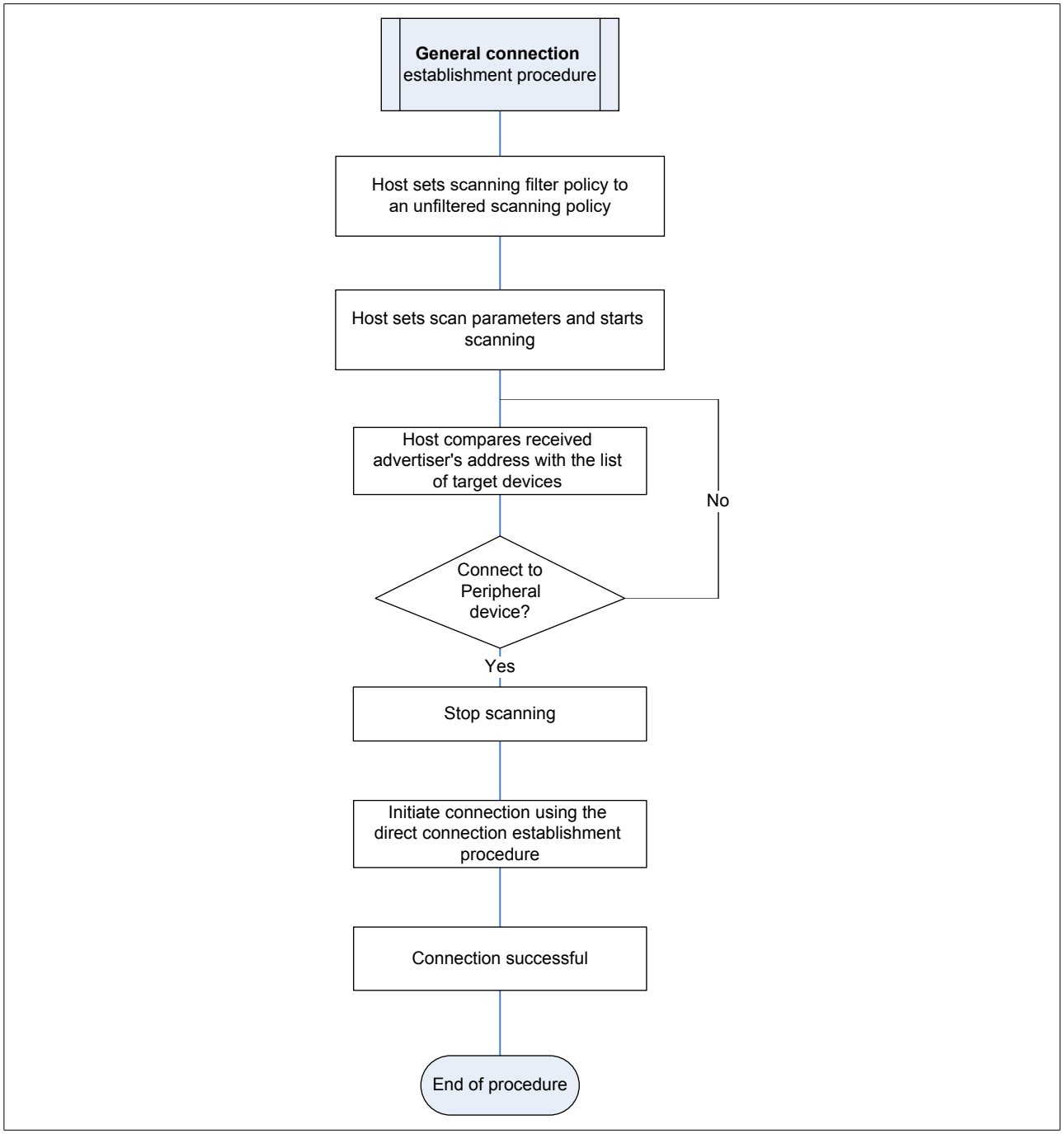



*Figure 9.4: Flow chart for a device performing the General Connection Establishment procedure*

When a Host performs the general connection establishment procedure, the Host configures the Controller as follows:

1. The Host shall set the scanning filter policy to an unfiltered scanning policy (see [Vol 6] Part B, Section 4.3.3).
2. The Host should set the scan interval as defined in Section 9.3.11.
3. The Host should set the scan window as defined in Section 9.3.11.
4. The Host shall start active scanning or passive scanning.
5. The Host should set connection parameters as defined in Section 9.3.12.

When the Host discovers a device to which the Host may attempt to connect, the Host shall stop the scanning, and initiate a connection using the direct connection establishment procedure.

This procedure is terminated when a connection is established or when the Host terminates the procedure.

### 9.3.7 Selective Connection Establishment procedure

#### 9.3.7.1 Description

The selective connection establishment procedure allows the Host to establish a connection, using the Host selected connection configuration parameters, with any device whose address is stored in the Filter Accept List.

#### 9.3.7.2 Conditions

Figure 9.5 shows the flow chart for a device performing the selective connection establishment procedure.

When a Host performs the selective connection establishment procedure, the Host configures the Controller as follows:

1. The Host shall write the list of device addresses that are to be selectively connected into the Filter Accept List.
2. The Host shall set the scanning filter policy to a filtered scanning policy (see [Vol 6] Part B, Section 4.3.3).
3. The Host should set the scan interval as defined in Section 9.3.11.
4. The Host should set the scan window as defined in Section 9.3.11.
5. The Host shall start active scanning or passive scanning.

When the Host discovers one of the peer devices it is connecting to, the Host shall stop scanning, and initiate a connection using the direct connection establishment procedure with the connection configuration parameters for that peer device.

This procedure is terminated when a connection is established or when the Host terminates the procedure.

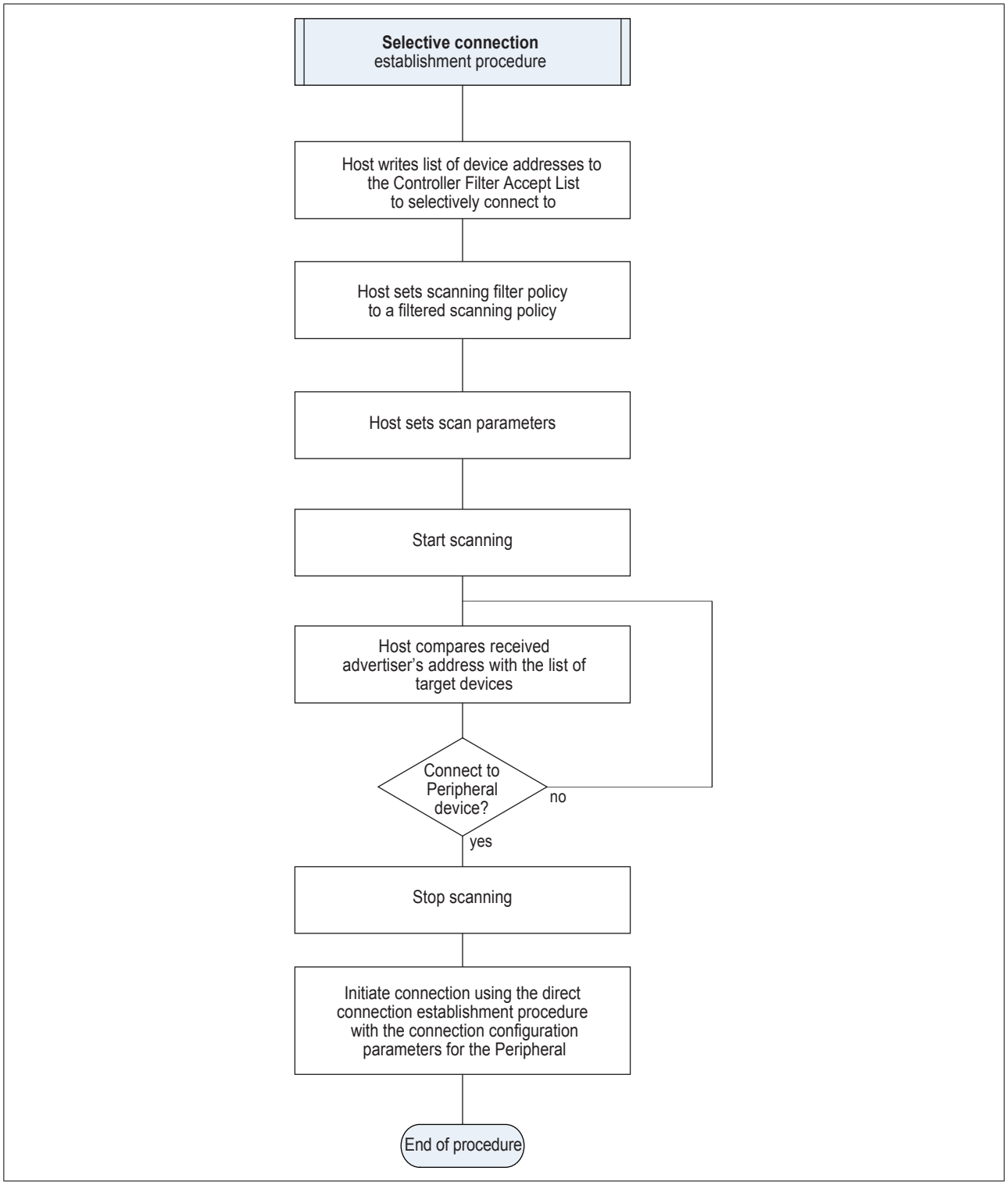



*Figure 9.5: Flow chart for a device performing the Selective Connection Establishment procedure*

### 9.3.8 Direct Connection Establishment procedure

#### 9.3.8.1 Description

The direct connection establishment procedure allows the Host to establish a connection with the Host selected connection configuration parameters with a single peer device.

#### 9.3.8.2 Conditions

Figure 9.6 shows the flow chart for a device performing the direct connection establishment procedure.

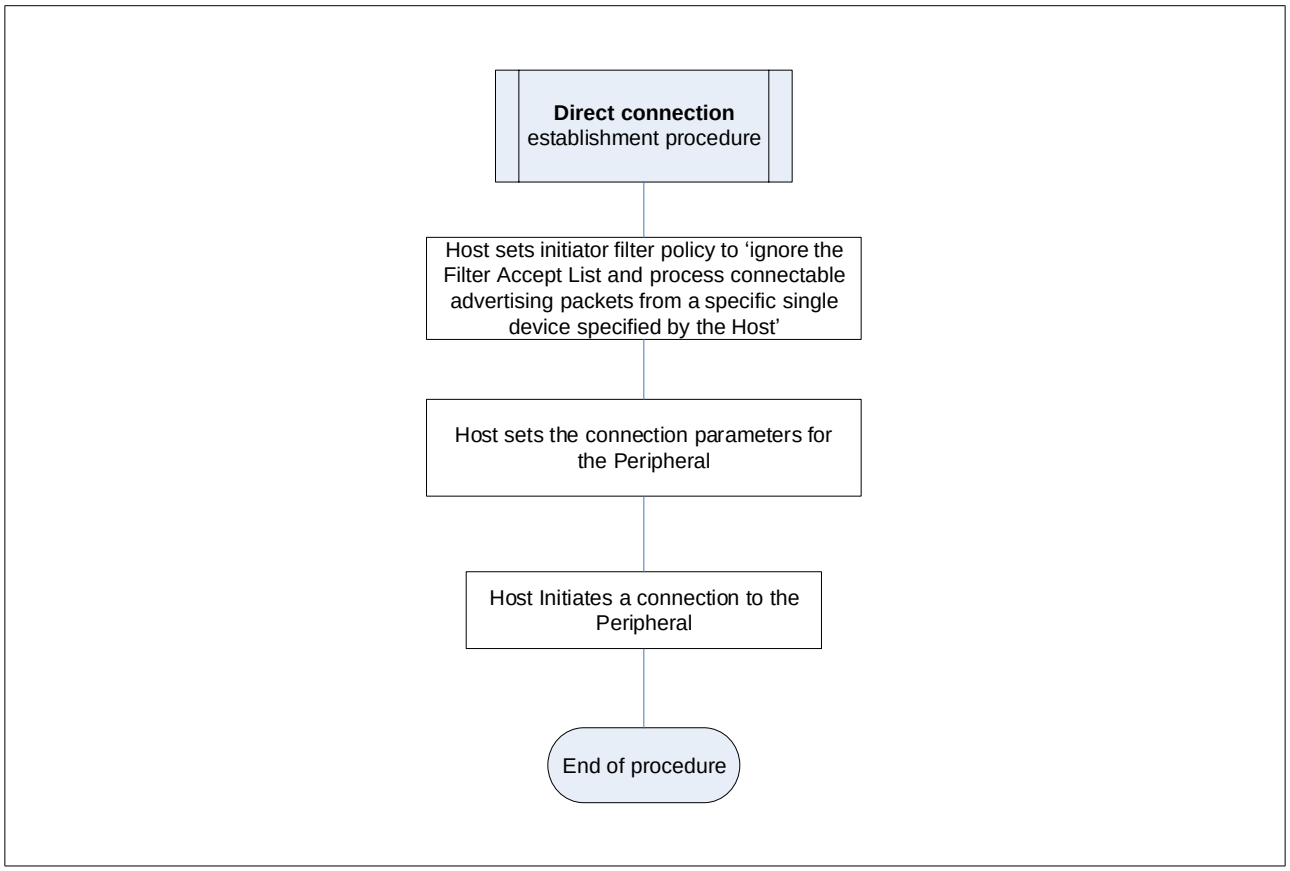



*Figure 9.6: Flow chart for a device performing the Direct Connection Establishment procedure*

When a Host performs the direct connection establishment procedure, the Host configures the Controller as follows:

1. The Host shall set the initiator filter policy to 'ignore the Filter Accept List and process connectable advertising packets from a specific single device specified by the Host'.
2. The Host shall set the peer address to the device address of the specific single device.

3. The Host should set connection parameters as defined in Section 9.3.12.
4. The Host shall initiate a connection.

This procedure is terminated when a connection is established or when the Host terminates the procedure.

### 9.3.9 Connection Parameter Update procedure

#### 9.3.9.1 Description

The connection parameter update procedure allows a Peripheral or Central to update the parameters of an established ACL connection.

#### 9.3.9.2 Conditions

A Central initiating the connection parameter update procedure shall use the Link Layer Connection Update procedure defined in [Vol 6] Part B, Section 5.1.1 with the required connection parameters if either the Central or the Peripheral does not support the Connection Parameters Request Link Layer Control procedure, the Connection Rate Request Link Layer Control procedure, or the Connection Rate Update Link Layer Control procedure.

If both the Central and Peripheral support the Connection Parameters Request Link Layer control procedure, the Connection Rate Request Link Layer Control procedure, or the Connection Rate Update Link Layer Control procedure, then the Central or Peripheral initiating the connection parameter update procedure should use the Connection Parameters Request Link Layer Control procedure defined in [Vol 6] Part B, Section 5.1.7, the Connection Rate Request Link Layer Control procedure defined in [Vol 6] Part B, Section 5.1.33, or the Connection Rate Update Link Layer Control procedure defined in [Vol 6] Part B, Section 5.1.32 with the required connection parameters.

If either the Central or the Peripheral does not support the Connection Parameters Request Link Layer Control procedure or the Connection Rate Request Link Layer Control procedure, then the Peripheral initiating the connection parameter update procedure shall use the L2CAP_CONNECTION_PARAMETER_UPDATE_REQ command defined in [Vol 3] Part A, Section 4.20 with the required connection parameters. The Peripheral shall not send an L2CAP_CONNECTION_PARAMETER_UPDATE_REQ command within $T_{GAP}$(conn_param_timeout) of an L2CAP_CONNECTION_PARAMETER_UPDATE_RSP being received. When the Central accepts the Peripheral initiated Connection Parameter Update, the Central should initiate the Link Layer Connection Update procedure defined in [Vol 6] Part B, Section 5.1.1 with the required connection parameters within TGAP(conn_param_timeout).

If the requested or updated connection parameters are unacceptable to the Central or Peripheral then it may disconnect the connection with the error code 0x3B (Unacceptable Connection Parameters). Devices should be tolerant of connection parameters given to them by the remote device.

### 9.3.10 Terminate Connection procedure

#### 9.3.10.1 Description

The Terminate Connection procedure allows a Host to terminate the connection with a peer device.

#### 9.3.10.2 Conditions

The Host should first terminate any associated CIS(es) prior to terminating the ACL.

The Host initiating the terminate connection procedure shall use the Link Layer ACL Termination procedure defined in [Vol 6] Part B, Section 5.1.6.

### 9.3.11 Connection Establishment Timing parameters

#### 9.3.11.1 Description

The connection establishment timing parameters are used during initial connection establishment between a Central and a Peripheral.

A Central should use one of the GAP connection establishment procedures to initiate a connection to a Peripheral in a connectable mode. The procedures and modes that may use these timing parameters are defined in Section 9.3.4 to Section 9.3.8.

#### 9.3.11.2 Conditions

A Central starting a user-initiated GAP connection establishment procedure should use the recommended scan interval $T_{GAP}$(scan_fast_interval) and scan window $T_{GAP}$(scan_fast_window) for $T_{GAP}$(scan_fast_period) when scanning on the LE 1M PHY and should use scan interval $T_{GAP}$(scan_fast_interval_coded) and scan window $T_{GAP}$(scan_fast_window_coded) for $T_{GAP}$(scan_fast_period) when scanning on the LE Coded PHY.

A Central that is background scanning (i.e. as part of a GAP connection establishment procedure that is not user-initiated) should use the recommended scan interval $T_{GAP}$(scan_slow_interval1) and scan window $T_{GAP}$(scan_slow_window1) when scanning on the LE 1M PHY and should use scan interval $T_{GAP}$(scan_slow_interval1_coded) and scan window $T_{GAP}$(scan_slow_window1_coded) when scanning on the LE Coded PHY. Alternatively the Central may use $T_{GAP}$(scan_slow_interval2) and scan window $T_{GAP}$(scan_slow_window2) when

scanning on the LE 1M PHY and should use $T_{GAP}$(scan_slow_interval2_coded) and scan window $T_{GAP}$(scan_slow_window2_coded) when scanning on the LE Coded PHY.

A Peripheral entering any of the following GAP Modes should use the recommended advertising interval $T_{GAP}$(adv_fast_interval1) for $T_{GAP}$(adv_fast_period) when advertising on the LE 1M PHY and should use $T_{GAP}$(adv_fast_interval1_coded) for $T_{GAP}$(adv_fast_period) when advertising on the LE Coded PHY:

- Undirected Connectable Mode
- Limited Discoverable Mode and sending connectable undirected advertising events
- General Discoverable Mode and sending connectable undirected advertising events
- Directed Connectable Mode and sending low duty cycle connectable directed advertising events

A Peripheral when entering any of the following GAP Modes and sending non-connectable advertising events should use the recommended advertising interval $T_{GAP}$(adv_fast_interval2) for $T_{GAP}$(adv_fast_period) when advertising on the LE 1M PHY and should use $T_{GAP}$(adv_fast_interval2_coded) for $T_{GAP}$(adv_fast_period) when advertising on the LE Coded PHY:

- Non-Discoverable Mode
- Non-Connectable Mode
- Limited Discoverable Mode
- General Discoverable Mode

A Peripheral that is background advertising in any GAP Mode other than GAP Directed Connectable Mode with high duty cycle connectable directed advertising events should use the recommended advertising interval $T_{GAP}$ (adv_slow_interval) when advertising on the LE 1M PHY and should use $T_{GAP}$ (adv_slow_interval_coded) when advertising on the LE Coded PHY.

When advertising interval values of less than 100 ms are used for non-connectable or scannable undirected advertising in environments where the advertiser can interfere with other devices, steps should be taken to minimize the interference. For example, the advertising might be alternately enabled for only a few seconds and disabled for several minutes.

### 9.3.12 Connection interval timing parameters

#### 9.3.12.1 Description

The connection interval timing parameters are used within a connection. Initial connection interval is used to ensure procedures such as bonding, encryption setup and

service discovery are completed quickly. The connection interval should be changed to the value in the Peripheral Preferred Connection Parameters characteristic after initiating procedures are complete.

#### 9.3.12.2 Conditions

The Central should either read the Peripheral Preferred Connection Parameters characteristic (see Section 12.3) or retrieve the parameters from advertising data (see Section 12.3).

The connection interval should be set to $T_{GAP}$(initial_conn_interval) when establishing a connection on the LE 1M PHY and to $T_{GAP}$(initial_conn_interval_coded) when establishing a connection on the LE Coded PHY and *connPeripheralLatency* should be set to zero. These parameters should be used until the Central has no further pending actions to perform or until the Peripheral performs a Connection Parameter Update procedure (see Section 9.3.9).

After the Central has no further pending actions to perform and the Peripheral has not initiated any other actions within $T_{GAP}$(conn_pause_central), then the Central should invoke the Connection Parameter Update procedure (see Section 9.3.9) and change the connection interval to that specified in the Peripheral Preferred Connection Parameters characteristic.

If the Central has not read the Peripheral Preferred Connection Parameters characteristic, then the Central may choose the connection parameters to be used.

After the Peripheral has no further pending actions to perform and the Central has not initiated any other actions within $T_{GAP}$(conn_pause_central), then the Peripheral may perform a Connection Parameter Update procedure (see Section 9.3.9). The Peripheral should not perform a Connection Parameter Update procedure within $T_{GAP}$(conn_pause_peripheral) after establishing a connection.

At any time a key refresh or encryption setup procedure is required and the current connection interval is greater than $T_{GAP}$(initial_conn_interval) when connected on the LE 1M PHY or LE 2M PHY or greater than $T_{GAP}$(initial_conn_interval_coded) when connected on the LE Coded PHY, the key refresh or encryption setup procedure should be preceded with a Connection Parameter Update procedure (see Section 9.3.9). The connection interval should be set to $T_{GAP}$(initial_conn_interval) when connected on the LE 1M PHY or the LE 2M PHY and $T_{GAP}$(initial_conn_interval_coded) when connected on the LE Coded PHY, *connSubrateFactor* should be set to 1, and *connPeripheralLatency* should be set to zero. This fast connection interval should be maintained until the key refresh or encryption setup procedure is complete. It should then switch to the value in the Peripheral Preferred Connection Parameters characteristic.

### 9.3.13 Connected Isochronous Stream Central Establishment procedure

#### 9.3.13.1 Description

The Connected Isochronous Stream Central Establishment procedure allows the Host of a Central to establish a CIS with a Peripheral using the Host selected parameters.

#### 9.3.13.2 Conditions

When two devices are connected, a Central may establish one or more CISes with a Peripheral. A CIS is established by the Central using the Connected Isochronous Stream Creation procedure (see [Vol 6] Part B, Section 5.1.15). The Central and or Peripheral may send isochronous data over the established CIS.

### 9.3.14 Connected Isochronous Stream Peripheral Establishment procedure

#### 9.3.14.1 Description

The Connected Isochronous Stream Peripheral Establishment procedure allows the Host of the Peripheral to accept or reject the request from a Central to establish a CIS.

#### 9.3.14.2 Conditions

When two devices are connected, the Peripheral may receive a request from the Central to establish a CIS. Once the request is received, the Host in the Peripheral shall either accept or reject the request. If it accepts the request, the CIS can be established using the Connected Isochronous Stream Creation procedure defined in [Vol 6] Part B, Section 5.1.15.

### 9.3.15 Connected Isochronous Stream Terminate procedure

#### 9.3.15.1 Description

The Connected Isochronous Stream Terminate procedure allows a Host to terminate a CIS with a peer device. The CIS shall also be terminated when the ACL between the Central and Peripheral is terminated, using the Terminate Connection procedure (Section 9.3.10).

#### 9.3.15.2 Conditions

The Host initiating the Connected Isochronous Stream Terminate procedure shall use the Connected Isochronous Stream Termination control procedure defined in [Vol 6] Part B, Section 5.1.16.

### 9.3.16 Connection Subrate procedure

#### 9.3.16.1 Description

The Connection Subrate procedure allows a Peripheral or Central to modify the connection subrating and other connection parameters of an established ACL connection.

#### 9.3.16.2 Conditions

The Central initiating this procedure shall use the Link Layer Connection Subrate Update procedure defined in [Vol 6] Part B, Section 5.1.19 or the Connection Rate Update Link Layer Control procedure defined in [Vol 6] Part B, Section 5.1.32, and the Peripheral initiating this procedure shall use the Connection Subrate Request procedure defined in [Vol 6] Part B, Section 5.1.20 or the Connection Rate Request Link Layer Control procedure defined in [Vol 6] Part B, Section 5.1.33.

If the requested subrate connection parameters are unacceptable to the Central, then it may reject them. If the updated subrate connection parameters are unacceptable to the Peripheral, then it cannot reject them but may follow up by using the Connection Parameters Request procedure (see [Vol 6] Part B, Section 5.1.7), the Connection Subrate Request procedure (see [Vol 6] Part B, Section 5.1.20), or the Connection Rate Request Link Layer Control procedure (see [Vol 6] Part B, Section 5.1.33) to change them. Devices should be tolerant of the connection parameters requested by the peer device.

### 9.3.17 Periodic Advertising Connection procedure

#### 9.3.17.1 Definition

The periodic advertising connection procedure provides a method for a periodic advertiser to initiate a Link Layer connection with a synchronized device.

#### 9.3.17.2 Conditions

A device performing the periodic advertising connection procedure shall initiate the Link Layer connection procedure using periodic advertising trains with responses (see [Vol 6] Part B, Section 4.4.2.12.2).

## 9.4 Bonding modes and procedures

Bonding allows two connected devices to exchange and store security and identity information to create a trusted relationship. The security and identity information as defined in [Vol 3] Part H, Section 2.4.1 is also known as the bonding information. When the devices store the bonding information, it is known as the phrases ‘devices have bonded’ or ‘a bond is created’.

There are two modes for bonding, non-bondable mode and bondable mode. Bonding may only occur between two devices in bondable mode. The requirements for a device to support the bonding modes and procedure are defined in Table 9.4.

### 9.4.1 Requirements

| Bonding | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| Non-Bondable mode | 9.4.2 | M | M | E | E |
| Bondable mode | 9.4.3 | O | O | E | E |
| Bonding procedure | 9.4.4 | C.1 | C.1 | E | E |
| C.1: Mandatory if Bondable mode is supported, otherwise Excluded. | | | | | |

*Table 9.4: Bonding requirements*

### 9.4.2 Non-bondable mode

#### 9.4.2.1 Description

A device in the non-bondable mode does not allow a bond to be created with a peer device.

#### 9.4.2.2 Conditions

If a device does not support pairing as defined in the Security Manager section then it is considered to be in non-bondable mode.

If Security Manager pairing is supported, the Host shall set the Bonding_Flags to ‘No Bonding’ as defined in [Vol 3] Part H, Section 3.5.1 and bonding information shall not be exchanged or stored.

### 9.4.3 Bondable mode

#### 9.4.3.1 Description

A device in the bondable mode allows a bond to be created with a peer device in the bondable mode.

#### 9.4.3.2 Conditions

The Host shall set the Bonding_Flags to ‘Bonding’ as defined in [Vol 3] Part H, Section 3.5.1 during the pairing procedure.

### 9.4.4 Bonding procedure

#### 9.4.4.1 Description

The bonding procedure may be performed when a non-bonded device tries to access a service that requires bonding. The bonding procedure may be performed for the purpose of creating a bond between two devices.

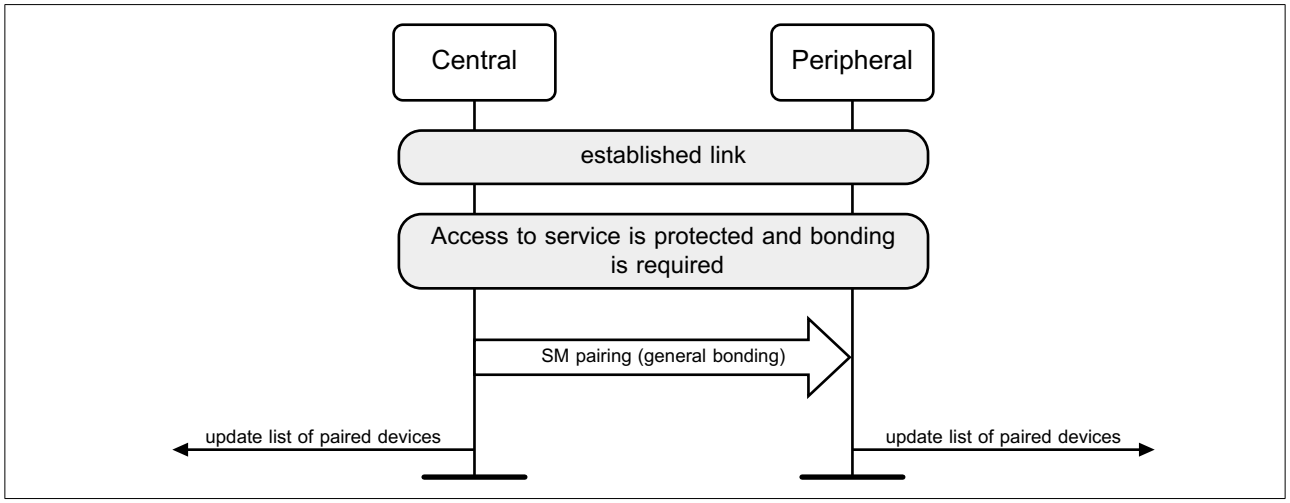



*Figure 9.7: Bonding requirements*

#### 9.4.4.2 Conditions

The Central shall be in the bondable mode and shall initiate the pairing process as defined in Section 2.1. If the Peripheral is in the bondable mode, the devices shall exchange and store the bonding information in the security database.

If a device supports the generation of resolvable private addresses as defined in Section 10.8.2.2 and generates a resolvable private address for its local address, it shall send Identity Information with SMP, including a valid IRK. If a device does not generate a resolvable private address for its own address and the Host sends Identity Information with SMP, the Host shall send an all-zero IRK. If a device supports resolving resolvable private addresses as defined in Section 10.8.2.3, it shall request the peer device to send its Identity Information with SMP. The Host can abort the pairing procedure if the authentication requirements are not sufficient to distribute the IRK.

## 9.5 Periodic advertising modes and procedure

The periodic advertising modes and procedure allow two or more devices to communicate in a connectionless manner using extended advertising events and periodic advertising events. These modes and procedures can also make use of existing connections. The requirements for a device operating in a specific GAP role to support these modes and procedures are defined in Table 9.5.

| Mode and Procedures | Ref. | Broadcaster | Observer | Peripheral | Central |
|---|---|---|---|---|---|
| Periodic Advertising Synchronizability mode | 9.5.1 | O | E | E | E |
| Periodic Advertising mode | 9.5.2 | C.1 | E | E | E |
| Periodic Advertising Synchronization Establishment procedure | 9.5.3 | E | O | O | O |
| Periodic Advertising Synchronization Transfer procedure | 9.5.4 | E | E | C.2 | C.2 |

C.1: Mandatory if Periodic Advertising Synchronizability mode is supported, otherwise excluded.

C.2: Optional if Periodic Advertising Mode is supported or the Periodic Advertising Synchronization Establishment procedure is supported, otherwise excluded.

*Table 9.5: Periodic advertising modes and periodic advertising procedure requirements*

## 9.5.1 Periodic Advertising Synchronizability mode

### 9.5.1.1 Definition

The periodic advertising synchronizability mode provides a method for a device to send synchronization information about a periodic advertising train (with or without responses) using advertisements.

### 9.5.1.2 Conditions

A device in the periodic advertising synchronizability mode shall send synchronization information for a periodic advertising train (with or without responses) in non-connectable and non-scannable extended advertising events. The advertising interval used is unrelated to the interval between the periodic advertising events.

A device shall not be in periodic advertising synchronizability mode unless it is also in periodic advertising mode. It may leave, and possibly re-enter, periodic advertising synchronizability mode while remaining in periodic advertising mode.

## 9.5.2 Periodic Advertising mode

### 9.5.2.1 Definition

On periodic advertising trains without responses, the periodic advertising mode provides a method for a device to send advertising data at periodic and deterministic intervals. On periodic advertising trains with responses, a device may send advertising data in one or more subevents to synchronized devices who may also send data back to the device.

#### 9.5.2.2 Conditions

A device in the periodic advertising mode shall send periodic advertising events at the interval and using the frequency hopping sequence specified in the periodic advertising synchronization information. On periodic advertising trains with responses, the interval may be divided into subevents. During a subevent, a synchronized device may send data back to the device in response slots.

A device entering periodic advertising mode shall also enter periodic advertising synchronizability mode for at least long enough to complete one extended advertising event (see [Vol 6] Part B, Section 4.4.2.12).

### 9.5.3 Periodic Advertising Synchronization Establishment procedure

#### 9.5.3.1 Definition

The periodic advertising synchronization establishment procedure provides a method for a device to receive periodic advertising synchronization information and to synchronize to a periodic advertising train.

#### 9.5.3.2 Conditions

A device performing the periodic advertising synchronization establishment procedure shall scan for non-connectable and non-scannable advertising events containing synchronization information about a periodic advertising train or shall accept periodic advertising synchronization information over an existing connection by taking part in the Link Layer Periodic Advertising Sync Transfer procedure defined in [Vol 6] Part B, Section 5.1.13. When a device receives synchronization information for a periodic advertising train, it may listen for periodic advertising events at the intervals and using the frequency hopping sequence specified in the periodic advertising synchronization information. If a device receives synchronization information about periodic advertising with responses, it may listen to one or more subevents in the interval and may send data to the periodic advertiser.

### 9.5.4 Periodic Advertising Synchronization Transfer procedure

#### 9.5.4.1 Definition

The periodic advertising synchronization transfer procedure provides a method for a device to send synchronization information about a periodic advertising train over an existing connection.

#### 9.5.4.2 Conditions

A device performing the periodic advertising synchronization transfer procedure shall initiate the Link Layer Periodic Advertising Sync Transfer procedure defined in [Vol 6] Part B, Section 5.1.13.

### 9.5.5 [This section is no longer used]

The Periodic Advertising Connection procedure is described in Section 9.3.17.

## 9.6 Isochronous Broadcast modes and procedures

The Isochronous Broadcast modes and procedures allow two or more devices to communicate in a unidirectional, connectionless manner by using extended advertising events, periodic advertising events, and BIG and BIS events. The requirements for a device that operates in a specific GAP role to support these modes and procedures are defined in Table 9.6.

| Modes and Procedures | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| Broadcast Isochronous Synchronizability mode | 9.6.1 | E | E | C.1 | E |
| Broadcast Isochronous Broadcasting mode | 9.6.2 | E | E | O | E |
| Broadcast Isochronous Synchronization Establishment procedure | 9.6.3 | E | E | E | O |
| Broadcast Isochronous Channel Map Update procedure | 9.6.4 | E | E | C.1 | C.2 |
| Broadcast Isochronous Terminate procedure | 9.6.5 | E | E | C.1 | C.2 |
| C.1: Mandatory if Broadcast Isochronous Broadcasting mode is supported, otherwise excluded. | | | | | |
| C.2: Mandatory if Broadcast Isochronous Synchronization Establishment procedure is supported, otherwise excluded. | | | | | |

*Table 9.6: Isochronous Broadcast modes and procedure requirements*

### 9.6.1 Broadcast Isochronous Synchronizability mode

#### 9.6.1.1 Definition

The Broadcast Isochronous Synchronizability mode provides a method for a device to transmit the synchronization information of a BIG.

#### 9.6.1.2 Conditions

A device shall also be in the Broadcast Isochronous Broadcasting mode while it is in the Broadcast Isochronous Synchronizability mode. A device in the Broadcast Isochronous Synchronizability mode shall send the BIGInfo in the ACAD field which is located in the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU of a periodic advertisement ([Vol 6] Part B, Section 2.3.4.8).

### 9.6.2 Broadcast Isochronous Broadcasting mode

#### 9.6.2.1 Definition

The Broadcast Isochronous Broadcasting mode provides a method for a device in the Broadcaster role to send encrypted or unencrypted Broadcast Isochronous PDUs in subevents of BISes of a BIG ([Vol 6] Part B, Section 4.4.6).

#### 9.6.2.2 Conditions

A device in the Broadcast Isochronous Broadcasting mode shall send isochronous PDUs in subevents of BISes of a BIG ([Vol 6] Part B, Section 4.4.6).

### 9.6.3 Broadcast Isochronous Synchronization Establishment procedure

#### 9.6.3.1 Definition

The Broadcast Isochronous Synchronization Establishment procedure provides a way for a device to synchronize to a BIS.

#### 9.6.3.2 Conditions

A device that performs the Broadcast Isochronous Synchronization Establishment procedure shall first perform the Periodic Advertising Synchronization Establishment procedure (Section 9.5.3) and receive the synchronization information. The synchronization information is used to synchronize to the required BIS in the BIG ([Vol 6] Part B, Section 4.4.6).

### 9.6.4 Broadcast Isochronous Channel Map Update procedure

#### 9.6.4.1 Definition

The Broadcast Isochronous Channel Map Update procedure allows a Broadcaster to use and transmit a new channel map of a BIG, or an Observer to receive and use a new channel map for a BIG.

#### 9.6.4.2 Conditions

In this procedure, the Broadcaster sends the channel map command in the BIG events ([Vol 6] Part B, Section 4.4.6.4). When an Observer receives a channel map message, it uses the new channel when receiving data from the BIG.

### 9.6.5 Broadcast Isochronous Terminate procedure

#### 9.6.5.1 Definition

The Broadcast Isochronous Terminate procedure allows a Host of a Broadcaster to terminate a BIG, or the Host of an Observer to terminate synchronization with a BIG.

#### 9.6.5.2 Conditions

The Host initiating the Broadcast Isochronous Stream Terminate procedure shall use the Broadcast Isochronous Stream Termination control procedure defined in [Vol 6] Part B, Section 5.6.2.

## 9.7 Channel Sounding procedures

The Channel Sounding procedures allow two devices to exchange information that may be used for distance approximation between the two. During this exchange, each device must select the alternate procedure type.

| Channel Sounding Procedures | Ref. | Peripheral | Central | Broadcaster | Observer |
|---|---|---|---|---|---|
| CS initiator procedure | 9.7.1 | O | O | E | E |
| CS reflector procedure | 9.7.2 | O | O | E | E |

*Table 9.7: Channel Sounding procedure requirements*

### 9.7.1 Channel Sounding initiator procedure

#### 9.7.1.1 Description

The CS initiator procedure allows a Peripheral or Central to initiate a CS procedure.

#### 9.7.1.2 Conditions

If the CS initiator role is supported by the Controller, then the Host may enable the initiator role by commanding the Controller to set the initiator role flag in the Configuration Exchange procedure as described in [Vol 6] Part B, Section 5.1.25.

If both the Central and Peripheral support the CS procedure, then the Central or Peripheral initiating the CS procedure shall do so in the manner described in [Vol 6] Part B, Section 5.1.26.

### 9.7.2 Channel Sounding reflector procedure

#### 9.7.2.1 Description

The CS reflector procedure allows a Peripheral or Central to respond to a CS initiator procedure.

#### 9.7.2.2 Conditions

If the CS reflector role is supported by the Controller, then the Host may enable the reflector role by commanding the Controller to set the Responder Role flag in the Configuration Exchange procedure as described in [Vol 6] Part B, Section 5.1.25.

The Central or Peripheral responding to the CS procedure shall do so in the manner described in [Vol 6] Part B, Section 5.1.26.

# 10 SECURITY ASPECTS – LE PHYSICAL TRANSPORT

This section defines the modes and procedures that relate to the security of either an ACL connection or broadcast. The modes and procedures that relate to the security of a CIS shall be the same as that used in its associated ACL. The following modes and procedures are defined:

- LE security mode 1
- LE security mode 3
- Authentication procedure
- Authorization procedure
- Encrypted Advertising Data procedure

Requirements for a device to support the LE security modes and procedures is shown in Table 10.1.

## 10.1 Requirements

| Security Modes and Procedures | Ref. | Broadcaster | Observer | Peripheral | Central |
|---|---|---|---|---|---|
| LE Security mode 1 | 10.2.1 | E | E | O | O |
| LE Security mode 3 | 10.2.5 | O | O | E | E |
| Authentication procedure | 10.3 | E | E | O | O |
| Authorization procedure | 10.5 | E | E | O | O |
| Encrypted Advertising Data procedure | 10.10 | O | O | O | O |

*Table 10.1: Requirements related to security modes and procedures*

## 10.2 LE security modes

The security requirements of a device, a service or a service request are expressed in terms of a security mode and security level. Each service or service request may have its own security requirement. The device may also have a security requirement.

There are two LE security modes, LE security mode 1 and LE security mode 3.

Note: LE security mode 2 was part of the data signing feature, which has been deprecated and removed.

### 10.2.1 LE security mode 1

LE security mode 1 has the following security levels:

1. No security (No authentication and no encryption)
2. Unauthenticated pairing with encryption
3. Authenticated pairing with encryption
4. Authenticated LE Secure Connections pairing with encryption using a 128-bit strength encryption key.

For certain services that require LE security mode 1, levels 2 or 3, a device may enforce the use of LE Secure Connections pairing before those services can be used.

A connection operating in LE security mode 1 level 2 shall also satisfy the security requirements for LE security mode 1 level 1.

A connection operating in LE security mode 1 level 3 shall also satisfy the security requirements for LE security mode 1 level 2 or LE security mode 1 level 1.

A connection operating in LE security mode 1 level 4 shall also satisfy the security requirements for LE security mode 1 level 3 or LE security mode 1 level 2 or LE security mode 1 level 1.

### 10.2.2 [This section is no longer used]

### 10.2.3 [This section is no longer used]

### 10.2.4 Secure Connections Only mode

A device may be in a Secure Connections Only mode. When in Secure Connections Only mode only security mode 1 level 4 shall be used except for services that only require security mode 1 level 1.

The device shall only accept new outgoing and incoming service level connections for services that require Security Mode 1, Level 4 when the remote device supports LE Secure Connections and authenticated pairing is used.

### 10.2.5 LE security mode 3

LE security mode 3 has three security levels:

1. No security (no authentication and no encryption)
2. Use of unauthenticated Broadcast_Code
3. Use of authenticated Broadcast_Code

LE security mode 3 shall be used to broadcast a Broadcast Isochronous Group (BIG) in an Isochronous Broadcaster or receive a BIS in a Synchronized Receiver.

A device operating in security mode 3 level 1 shall require that the isochronous data is unencrypted.

A device that operates in LE security mode 3 level 2 shall require a Broadcast_Code to encrypt the data that is transmitted in a BIS.

A device that operates in LE security mode 3 level 3 shall require a Broadcast_Code to encrypt the data that is transmitted in a BIS. If the device has not received a Broadcast_Code using an authenticated method when the service requires Level 3 security and has a user interface capable of doing so, then the device shall indicate an appropriate error to the user (e.g., Insufficient Security for Broadcast_Code).

## 10.3 Authentication procedure

The authentication procedure describes how the required security is established when a device initiates a service request to a remote device and when a device receives a service request from a remote device. The authentication procedure covers LE security mode 1. The authentication procedure shall only be initiated after a connection has been established.

Authentication in LE security mode 1 is achieved by enabling encryption as defined in Section 10.6. The security of the encryption is impacted by the type of pairing performed. There are two types of pairing: authenticated pairing or unauthenticated pairing. Authenticated pairing involves performing the pairing procedure defined in [Vol 3] Part H, Section 2.1 with the authentication set to ‘MITM protection’. Unauthenticated pairing involves performing the pairing procedure with authentication set to ‘No MITM protection’.

Note: In this section, the terms “authenticated pairing” and “unauthenticated pairing” refer to the security method used to perform pairing and are not related to the authentication of previously bonded devices during a reconnection.

Section 10.3.1 specifies the authentication procedure when a device responds to a service request. Section 10.3.2 specifies the authentication procedure when a device initiates a service request.

### 10.3.1 Responding to a service request

In this section the local device is the device responding to a service request made by a remote device. In the L2CAP protocol the local device responds with a connection response to a remote device making a connection request. In GATT, the local device is the GATT Server and the remote device is the GATT Client.

When a local device receives a service request from a remote device, it shall respond with an error code if the service request is denied. The error code is dependent on whether the current connection is encrypted or not and on the type of pairing that was performed to create the LTK or STK to be used.

When a local device receives a service request from a remote device it shall behave according to the following rules:

- The local device's security database specifies the security settings required to accept a service request. If no encryption is required, the service request shall be accepted. If encryption is required the local device shall send an error code as defined in Table 10.2.
- If neither an LTK nor an STK is available, the service request shall be rejected with the error code "Insufficient Authentication".

  Note: When the link is not encrypted, the error code "Insufficient Authentication" does not indicate that MITM protection is required.
- If an LTK or an STK is available and encryption is required (LE security mode 1) but encryption is not enabled, the service request shall be rejected with the error code "Insufficient Encryption". If the encryption is enabled with a key size that is too short then the service request shall be rejected with the error code "Encryption Key Size Too Short."
- If an authenticated pairing is required but only an unauthenticated pairing has occurred and the link is currently encrypted, the service request shall be rejected with the error code "Insufficient Authentication."

  Note: When unauthenticated pairing has occurred and the link is currently encrypted, the error code "Insufficient Authentication" indicates that MITM protection is required.
- If LE Secure Connections pairing is required but LE legacy pairing has occurred and the link is currently encrypted, the service request shall be rejected with the error code "Insufficient Authentication".

The local device will respond with the minimum security level required for access to its services. If the local device has no security requirement it should default to the minimum security level that the local device is capable of as defined in pairing phase 1, (see [Vol 3] Part H, Section 2.1).

A local device shall not require an authenticated pairing (MITM) if the local device does not support the required IO capabilities or OOB data[1].

[1]If an OOB mechanism is used, the level of MITM protection is dependent upon the properties of the OOB communications channel. See [Vol 3] Part H, Section 2.3.5.1 for more information

The local device responds to a service request from a remote device are summarized in Table 10.2.

| Link Encryption State | Local Device's Access Requirement for Service | Local Device Pairing Status | | | |
|---|---|---|---|---|---|
| | | **No LTK**<br>**No STK** | **Unauthenticated LTK (with or without LE Secure Connections) or Unauthenticated STK** | **Authenticated LTK without LE Secure Connections or Authenticated STK** | **Authenticated LTK with Secure Connections** |
| **Unencrypted** | **None** | Request succeeds | Request succeeds | Request succeeds | Request succeeds |
| | **Encryption, No MITM Protection** | Error Resp.: Insufficient Authentication | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption |
| | **Encryption, MITM Protection** | Error Resp.: Insufficient Authentication | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption |
| | **Encryption, MITM Protection, Secure Connections** | Error Resp.: Insufficient Authentication | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption | Error Resp.: Insufficient Encryption |
| **Encrypted** | **None** | N/A<br>(Not possible to be encrypted without LTK) | Request succeeds | Request succeeds | Request succeeds |
| | **Encryption, No MITM Protection** | | Request succeeds | Request succeeds | Request succeeds |
| | **Encryption, MITM Protection** | | Error Resp.: Insufficient Authentication | Request succeeds | Request succeeds |
| | **Encryption, MITM Protection, Secure Connections** | | Error Resp.: Insufficient Authentication | Error Resp.: Insufficient Authentication | Request succeeds |

*Table 10.2: Local device responds to a service request*

Figure 10.1 shows how a server handles a service request.

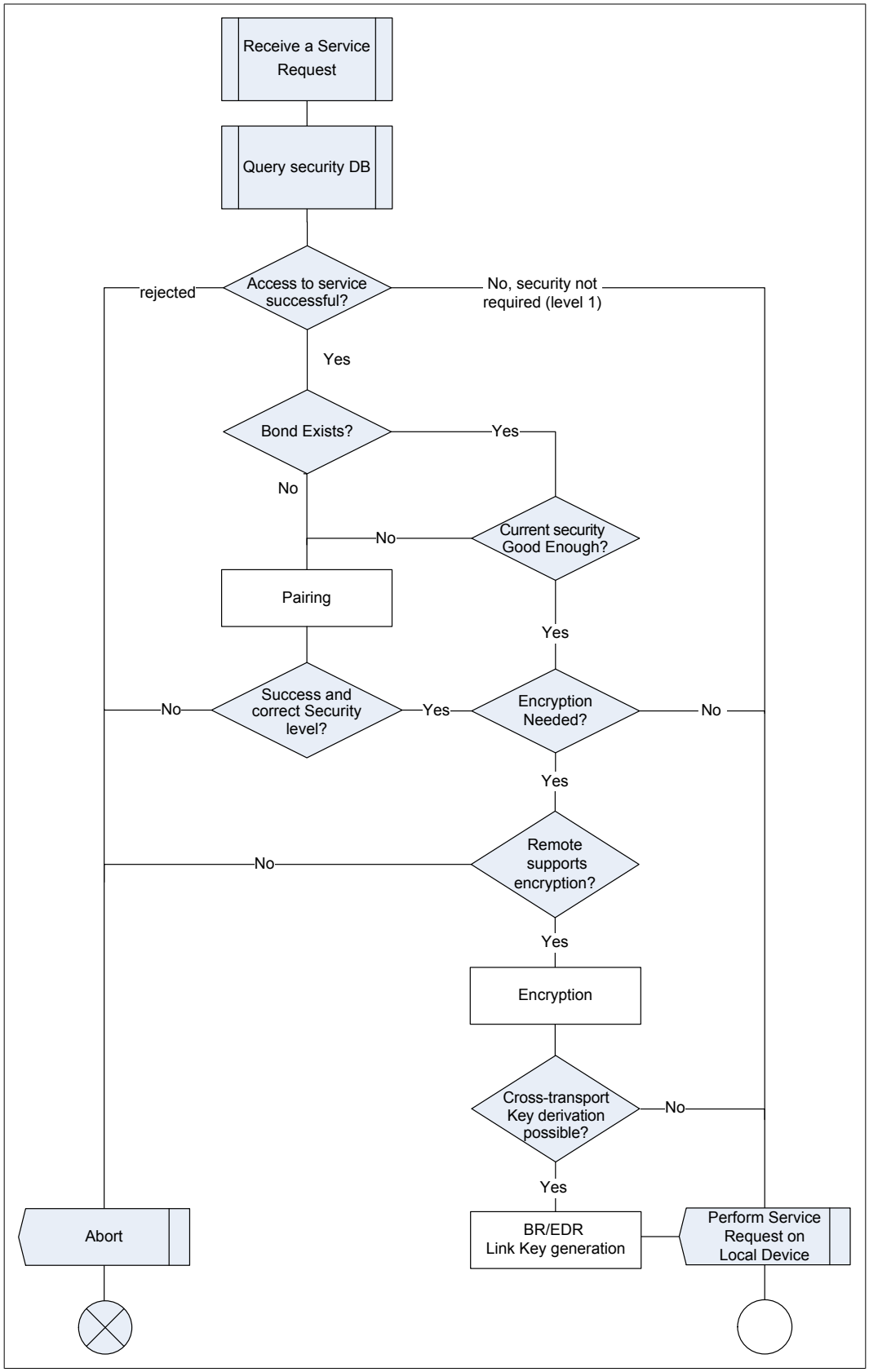



*Figure 10.1: Flow chart for a local device handling a service request from a remote device*

#### 10.3.1.1 Handling of GATT indications and notifications

A client “requests” a server to send indications and notifications by appropriately configuring the server via a Client Characteristic Configuration Descriptor. Since the configuration is persistent across a disconnection and reconnection, the security requirements for the connection shall be checked against the configuration upon a reconnection before sending indications or notifications. When a server reconnects to a client to send an indication or notification for which a specific level of security for the connection is required, the server shall initiate or request encryption with the client prior to sending an indication or notification. If the client does not have an LTK, indicating that the client has lost the bond, enabling encryption will fail.

#### 10.3.1.2 Cross-transport key generation

After encryption is enabled, the correct security level has been achieved, and if both devices support cross-transport key generation, then both devices may perform BR/EDR link key derivation.

Note: If the LTK has an encryption key size that is shorter than 16 octets (128 bits), then the BR/EDR link key is derived before the LTK gets masked.

### 10.3.2 Initiating a service request

In this section the local device is the device initiating a service request to a remote device. In the L2CAP protocol the local device sends the connection request and the remote device sends the connection response. In GATT, the local device is the GATT Client and the remote device is the GATT Server.

When a local device initiates a service request to a remote device it shall behave according to the following rules:

- The local device’s security database specifies the security required to initiate a service request. If no encryption is required by the local device then the service request may proceed without encryption or pairing.
- If an LTK is not available but encryption is required, the pairing procedure shall be initiated with the local device’s required authentication settings. If the pairing procedure fails then the service request shall be aborted.

  Note: When encryption is not enabled, the error code “Insufficient Authentication” does not indicate to the local device that MITM protection is required.

  If the local device is a Peripheral then it may send a Peripheral Initiated Security Request as defined in [Vol 3] Part H, Section 2.4.6.

- If pairing has occurred but the encryption key size is insufficient the pairing procedure shall be executed with the required encryption key size. If the pairing procedure fails then the service request shall be aborted.
- If an LTK is available and encryption is required (LE security mode 1) then encryption shall be enabled before the service request proceeds as defined in Section 10.6. Once encryption is enabled the service request shall proceed. If encryption fails, then either the bond no longer exists on the remote device or the wrong device has been connected. If the local device does not abandon the service request, it shall trigger a user interaction to confirm the remote device and then re-bond, perform service discovery, and reconfigure the remote device (reconfiguring the remote device can involve actions such as re-enabling indications and notifications on the relevant characteristics). If the local device had previously determined that the remote device did not have the «Service Changed» characteristic, or if the local device determines by reading the «Database Hash» characteristic that the database has not changed, then service discovery may be skipped.
- If an authenticated pairing is required but only an unauthenticated pairing has occurred and the link is currently encrypted, the pairing procedure shall be executed with the required authentication settings. If the pairing procedure fails or an authenticated pairing cannot be performed with the IO capabilities of the local device and remote device, then the service request shall be aborted.

  When a bond has been created between two devices, any reconnection should result in the local device enabling or requesting encryption with the remote device before initiating any service request.

  If a local device does not enable encryption before initiating a service request and relies on the error codes to determine the security requirements, the local device shall not request pairing with MITM protection in response to receiving an “Insufficient Authentication” error code from the remote device while the link is unencrypted. The local device shall only set the MITM protection required flag if the local device itself requires MITM protection.
- If encryption is not enabled at the time of the service request, the error code “Insufficient Authentication” is received, and the local device currently has an LTK, then the encryption procedure should be started (see Section 10.6). If this fails (likely indicating that the remote device has lost the bond and no longer has the LTK) or the local device does not have the correct LTK, then it should re-pair. If it re-pairs, it shall trigger a user interaction to confirm the remote device before starting the pairing procedure. IO capabilities are exchanged in pairing phase 1, (see [Vol 3] Part H, Section 2.1) and the security level shall be determined by the devices’ IO capabilities and MITM requirements.
- If encryption is not enabled at the time of the service request, the error code “Insufficient Encryption” is received, and the local device currently has an LTK, then the encryption procedure shall be started (see Section 10.6). If starting encryption

fails (likely indicating that the remote device has lost the bond and no longer has the LTK) or the local device does not have the correct LTK, then it should re-pair. If it re-pairs, it shall trigger a user interaction to confirm the remote device before starting the pairing procedure.

- If LE Secure Connections authenticated pairing is required but the remote device does not support LE Secure Connections, then the service request shall be aborted.

Figure 10.2 shows how a client issues a service request.

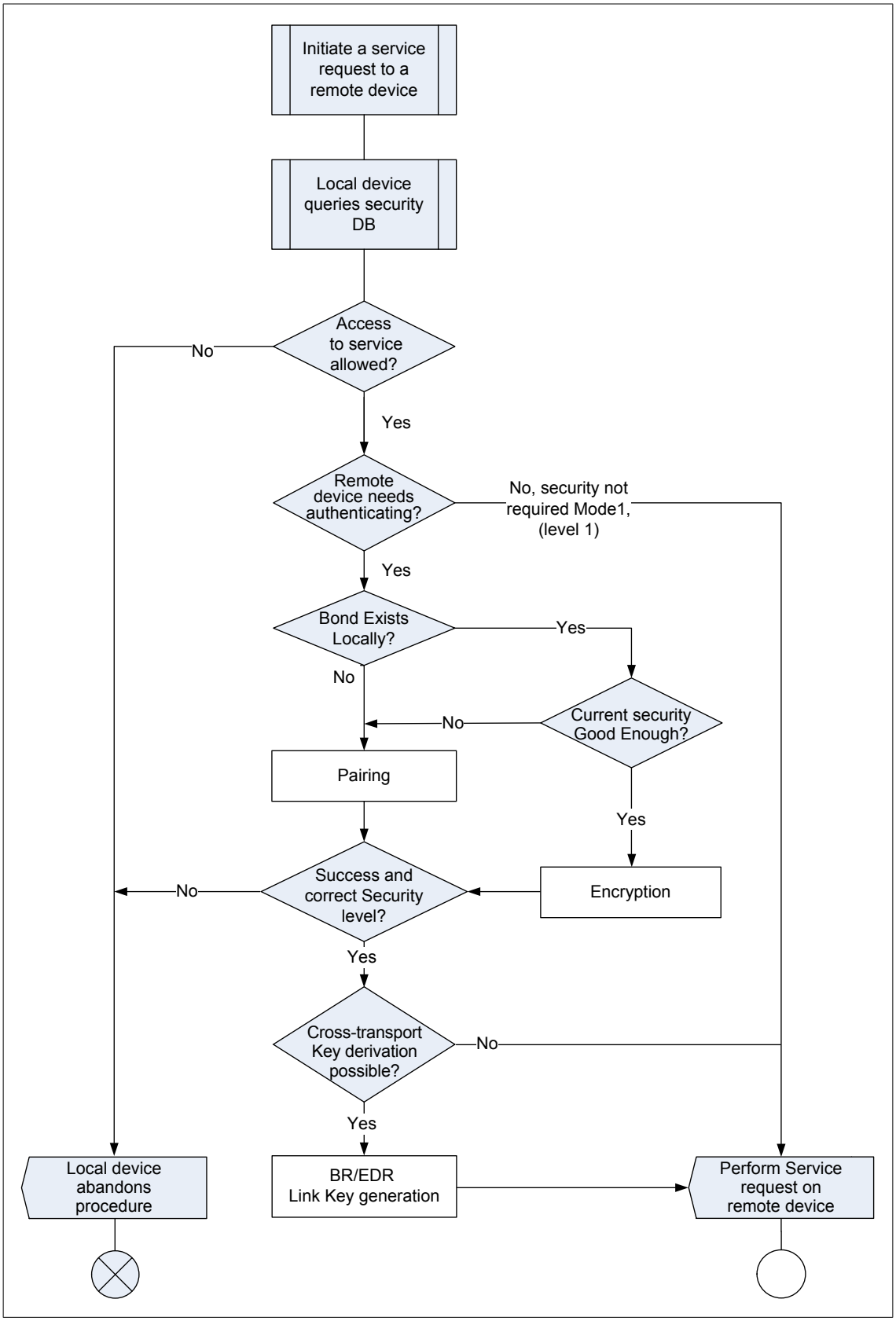



*Figure 10.2: Flow chart for a local device issuing a service request to a remote device*

#### 10.3.2.1 Cross-transport key generation

After encryption is enabled, the correct security level has been achieved, and both devices support cross-transport key generation, the both devices may perform BR/EDR link key derivation.

Note: If the LTK has an encryption key size that is shorter than 16 octets (128 bits), the BR/EDR link key is derived before the LTK gets masked.

#### 10.3.2.2 Handling of GATT indications and notifications

A client requests a server to send indications and notifications by appropriately configuring the server via a Client Characteristic Configuration Descriptor. Since the configuration is persistent across a disconnection and reconnection, the client shall check the security requirements for the connection against the configuration upon a reconnection before processing any indications or notifications from the server. Any notifications received before the security requirements are met shall be ignored. Any indications received before the security requirements are met shall be confirmed and then discarded. When a client reconnects to a server and expects to receive indications or notifications for which a specific level of security for the connection is required, the client shall enable encryption with the server. If the server does not have an LTK, indicating that the server has lost the bond, enabling encryption will fail.

## 10.4 [This section is no longer used]

## 10.5 Authorization procedure

A service may require authorization before allowing access. Authorization is a confirmation by the user to continue with the procedure. Authentication does not necessarily provide authorization. Authorization may be granted by user confirmation after successful authentication.

## 10.6 Encryption procedure

A Central may encrypt a connection using the Encryption Session Setup as defined in [Vol 3] Part H, Section 2.4.4 to provide integrity and confidentiality.

A Peripheral may encrypt a connection using the Peripheral Initiated Security Request as defined in [Vol 3] Part H, Section 2.4.6 to provide integrity and confidentiality.

If the encryption procedure fails and either the Central or Peripheral used a Resolvable Private Address for the connection establishment, then the current Resolvable Private Address(es) shall be immediately discarded and new Resolvable Private Address(es) shall be generated.

## 10.7 Privacy feature

The privacy feature provides a level of privacy which makes it more difficult for an attacker to track a device over a period of time. The requirements for a device to support the privacy feature are defined in Table 10.3.

| Privacy Requirements | Ref. | Broadcaster | Observer | Peripheral | Central |
|---|---|---|---|---|---|
| Privacy feature | 10.7 | O | O | O | O |
| Non-resolvable private address generation procedure | 10.8.2.1 | C.2 | C.4 | O | O |
| Resolvable private address generation procedure | 10.8.2.2 | C.3 | C.5 | C.1 | C.1 |
| Resolvable private address resolution procedure | 10.8.2.3 | E | O | C.1 | C.1 |
| Bondable Mode | 9.4.3 | E | E | C.1 | C.1 |
| Bonding procedure | 9.4.4 | E | E | C.1 | C.1 |

C.1: Mandatory if privacy feature is supported, otherwise optional

C.2: Mandatory if privacy feature is supported and resolvable private address generation procedure is not supported, otherwise optional

C.3: Mandatory if privacy feature is supported and non-resolvable private address generation procedure is not supported, otherwise optional

C.4: Mandatory if privacy feature and active scanning are supported and resolvable private address generation procedure is not supported, otherwise optional

C.5: Mandatory if privacy feature and active scanning are supported and non-resolvable private address generation procedure is not supported, otherwise optional

*Table 10.3: Requirements related to privacy feature*

Two modes of privacy exist:

- Device Privacy Mode: When a device is in device privacy mode, it is only concerned about its own privacy. It should accept advertising packets from peer devices that contain their Identity Addresses as well as their private address, even if the peer device has distributed its IRK.
- Network Privacy Mode. When a device is in network privacy mode, it shall not accept advertising packets containing the Identity Address of peer devices that have distributed their IRK.

If the Resolvable Private Address Only characteristic is not present in the GAP Service of the remote device, then the remove device may use its Identity Address over the air.

A device may use different modes for different peers.

If a device, i.e. Host and Controller, claims support for the privacy feature, the requirements in this section shall be met.

A device may support either just Host-based privacy or both Host-based and Controller-based privacy. When a device supports Controller-based privacy, the Host configures the Controller to perform some of the privacy functionality.

If a device supports Controller-based privacy, the requirements in the following paragraphs shall be met.

- The Host may maintain a resolving list by adding and removing device identities. A device identity consists of the peer's Identity Address and a local and peer's IRK pair. The local or peer's IRK shall be an all-zero key if not applicable for the particular device identity.
- If a peer device provides an all-zero Identity Address during pairing, the Host shall choose a unique identifier to substitute the peer's device Identity Address. The Host shall ensure that all identities provided to the Controller are unique.
- When address resolution is enabled in the Controller, all references to peer devices that are included in the resolving list from Host to the Controller shall be done using the peer's device Identity Address. Likewise, all incoming events from the Controller to the Host will use the peer's device identity, if the peer's device address has been resolved.
- To select device privacy mode, the Host shall so instruct the Controller for each peer in the resolving list.
- If the Host requires network privacy mode, then it shall only populate entries in the Controller's resolving list that have non-zero IRKs and shall not instruct the Controller to use device privacy mode.

### 10.7.1 Privacy feature in a Peripheral

The privacy-enabled Peripheral shall use a resolvable private address as the advertiser's device address when in connectable mode.

A Peripheral shall use non-resolvable or resolvable private addresses when in non-connectable mode as defined in Section 9.3.2.

If a privacy-enabled Peripheral, that has a stored bond, receives a resolvable private address, the Host may resolve the resolvable private address by performing the 'resolvable private address resolution procedure' as defined in Section 10.8.2.3. If the resolution is successful, the Host may accept the connection. If the resolution procedure fails, then the Host shall either accept the connection from the new, unresolved device, disconnect with the error code "Authentication failure", or perform the pairing procedure, or perform the authentication procedure as defined in Section 10.3. Accepting the

connection from the new, unresolved device, can result in exposing the device name or unique data to the Central.

The device should not send the device name or unique data in the advertising data that can be used to recognize the device.

#### 10.7.1.1 Privacy feature in a Peripheral with Controller-based privacy

A privacy-enabled Peripheral shall use either the undirected connectable mode as defined in Section 9.3.4 or directed connectable mode as defined in Section 9.3.3. The directed connectable mode shall only be used if the peer device supports Address Resolution in the Controller.

The Host shall enable resolvable private address generation by enabling it in the Controller and populating the resolving list.

By default, network privacy mode is used when private addresses are resolved and generated by the Controller.

If the advertising data or the scan response data change regularly then those changes should be synchronized with any changes in private addresses (both local and remote). For this purpose, the Host should either instruct the Controller to synchronize address changes with those data changes or, if the Controller does not support such synchronization, generate private addresses as described in Section 10.7.1.2 instead of offloading private address generation to the Controller.

#### 10.7.1.2 Privacy feature in a Peripheral with Host-based privacy

A privacy-enabled Peripheral should use the undirected connectable mode as defined in Section 9.3.2, to create a connection.

The Host shall generate a resolvable private address using the ‘resolvable private address generation procedure’ as defined in Section 10.8.2.2 or non-resolvable private address procedure as defined in Section 10.8.2.1. The Host shall set a timer equal to $T_{GAP}$(private_addr_int). The Host shall generate a new resolvable private address or non-resolvable private address when the timer $T_{GAP}$(private_addr_int) expires.

Note: $T_{GAP}$(private_addr_int) timer need not be run if a Peripheral is not advertising.

If the advertising data or the scan response data change regularly, then those changes should be synchronized with any changes in private addresses (both local and remote).

### 10.7.2 Privacy feature in a Central

The privacy-enabled Central shall use a resolvable private address as the initiator's device address.

During active scanning, a privacy enabled Central shall use a non-resolvable or resolvable private address.

If, a privacy-enabled Central, that has a stored bond, receives a resolvable private address, the Host may resolve the resolvable private address by performing the "resolvable private address resolution procedure" as defined in Section 10.8.2.3.

#### 10.7.2.1 Privacy feature in a Central with Controller-based privacy

A privacy-enabled Central with Address Resolution enabled in the Controller can use any of the connection establishment procedures defined in Section 9.3.

By default, network privacy mode is used when private addresses are resolved and generated by the Controller.

#### 10.7.2.2 Privacy feature in a Central with Host-based privacy

A privacy-enabled Central should use the general connection establishment procedure defined in Section 9.3.6 to create a connection.

The Host shall generate a resolvable private address using the ‘resolvable private address generation procedure’ as defined in Section 10.8.2.2 or non-resolvable private address procedure as defined in Section 10.8.2.1. The Host shall set a timer equal to $T_{GAP}$(private_addr_int). The Host shall generate a new resolvable private address or non-resolvable private address when the timer $T_{GAP}$(private_addr_int) expires.

Note: $T_{GAP}$(private_addr_int) timer need not be run if a Central is not scanning or connected.

### 10.7.3 Privacy feature in a Broadcaster

A privacy-enabled Broadcaster shall use the Broadcast mode defined in Section 9.1.1. The Broadcaster shall use either a resolvable private address or non-resolvable private address.

If Address Resolution is not supported or disabled in the Controller, the following applies to the Host: The Host shall generate a resolvable private address using the ‘resolvable private address generation procedure’ as defined in Section 10.8.2.2 or non-resolvable private address procedure as defined in Section 10.8.2.1. The Host shall set a timer to $T_{GAP}$(private_addr_int). The Host shall generate a new resolvable private address or non-resolvable private address when the timer $T_{GAP}$(private_addr_int) expires.

Note: $T_{GAP}$(private_addr_int) timer need not be run if a Broadcaster is not advertising.

The device should not send the device name or unique data in the advertising data which can be used to recognize the device.

### 10.7.4 Privacy feature in an Observer

A privacy-enabled Observer shall use the Observation procedure defined in Section 9.1.2. During active scanning, a privacy enabled Observer shall use either a resolvable private address or non-resolvable private address.

If Address Resolution is not supported or disabled in the Controller, the following applies to the Host: The Host shall generate a resolvable private address using the 'resolvable private address generation procedure' as defined in Section 10.8.2.2 or non-resolvable private address procedure as defined in Section 10.8.2.1. The Host shall set a timer equal to $T_{GAP}$(private_addr_int). The Host shall generate a new resolvable private address or non-resolvable private address when the timer $T_{GAP}$(private_addr_int) expires. The value of $T_{GAP}$(private_addr_int) shall not be greater than 1 hour.

Note: $T_{GAP}$(private_addr_int) timer need not be run if an Observer is not scanning.

## 10.8 Random Device address

For the purposes of this profile, the random device address may be of either of the following two sub-types:

- Static address
- Private address

The term random device address refers to both static and private address types.

The transmission of a random device address is optional. A device shall accept the reception of a random device address from a remote device.

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

A bonded device shall process a resolvable private address as defined in Section 10.8.2.3 or by establishing a connection and then performing the authentication procedure as defined in Section 10.3. A device that generates a resolvable private address for its local address shall always request to distribute its IRK value as defined in [Vol 3] Part H, Section 3.6.4 if both sides are bondable, unless keys have been pre-distributed.

After a device has distributed its IRK, it should use resolvable private addresses when establishing a connection with a peer device to which the IRK has been distributed.

Note: Section 10.7.1 and Section 10.7.2 require a privacy-enabled device to use resolvable private addresses in this case.

### 10.8.1 Static address

The Host can generate a static address using the procedure described in [Vol 6] Part B, Section 1.3.2.1.

### 10.8.2 Private address

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

#### 10.8.2.1 Non-Resolvable Private Address Generation procedure

The Host can generate a non resolvable private address using the procedure described in [Vol 6] Part B, Section 1.3.2.2.

#### 10.8.2.2 Resolvable Private Address Generation procedure

The Host can generate a resolvable private address where the Host has its IRK using the procedure described in [Vol 6] Part B, Section 1.3.2.2.

#### 10.8.2.3 Resolvable Private Address Resolution procedure

The Host can resolve a resolvable private address where the Host has the peer device's IRK or the local device's IRK, using the procedure described in [Vol 6] Part B, Section 1.3.2.3.

## 10.9 Encrypted Broadcast Isochronous Group

A device with a service that requires using an unauthenticated encrypted BIG shall set its security to LE security mode 3 level 2. A device with a service that requires using an authenticated encrypted BIG shall set its security to LE security mode 3 level 3.

When a user initiates a service that includes broadcasting an encrypted BIG, the Host shall provide the Broadcast_Code associated with the encrypted BIG to the Controller.

When a user initiates a service that requires the reception of an encrypted BIS, the device needs to know the Broadcast_Code for that encrypted BIS. If the device does not have the Broadcast_Code or cannot obtain the Broadcast_Code and has a user interface, then it shall indicate an appropriate error (e.g., Code Unavailable) to the user.

## 10.10 Encrypted Advertising Data procedure

A Broadcaster may encrypt advertising data using an Encrypted Data data type. The data is encrypted using key material that is known by both devices as defined in Section 1.23.3 of [7].

The key material may be exposed using the Encrypted Data Key Material characteristic on the Broadcaster or may be supplied to scanning devices using an out-of-band mechanism.

The scanning device shall read the Encrypted Data Key Material characteristic on an advertising device or use an out-of-band mechanism to obtain the key material for a device needed to interpret the encrypted advertising data being sent.

## 10.11 LE Channel Sounding

A device with a service that requires using the CS procedure described in Section 9.7 must first encrypt the underlying LE connection as described in [Vol 6] Part B, Section 4.5.18.2.

### 10.11.1 Channel Sounding security

Channel Sounding provides the following levels of channel measurement security for the application layer:

1. Either CS tone or CS RTT
2. 150 ns CS RTT accuracy and CS tones
3. 10 ns CS RTT accuracy and CS tones
4. Level 3 with the addition of CS RTT sounding sequence or random sequence payloads, and support of the Normalized Attack Detector Metric requirements as described in [Vol 6] Part H, Section 3.5.1.

A device that operates in security level 1 shall use CS tone or CS RTT within a CS procedure.

A device that operates in security level 2 shall use 150 ns or better CS RTT accuracy and CS tones within a CS procedure.

A device that operates in security level 3 shall use 10 ns or better CS RTT accuracy and CS tones within a CS procedure.

A device that operates in security level 4 shall meet the requirements of security level 3 and shall also require that the CS procedure uses either CS RTT with sounding sequence or CS RTT with random sequence, and that the device shall also support

the Normalized Attack Detector Metric requirements as described in [Vol 6] Part H, Section 3.5.1.

# 11 ADVERTISING AND SCAN RESPONSE DATA FORMAT

The format of Advertising, Periodic Advertising, and Scan Response data is shown in Figure 11.1. The data consists of a significant part and a non-significant part. The significant part contains a sequence of AD structures. Each AD structure shall have a Length field of one octet, which contains the Length value and shall not be zero, and a Data field of Length octets. The first octet of the Data field shall contain the AD type field. The content of the remaining Length - 1 octets in the Data field depends on the value of the AD type field and is called the AD data. The non-significant part shall only be present when necessary to fill a fixed-length field and shall contain all-zero octets.

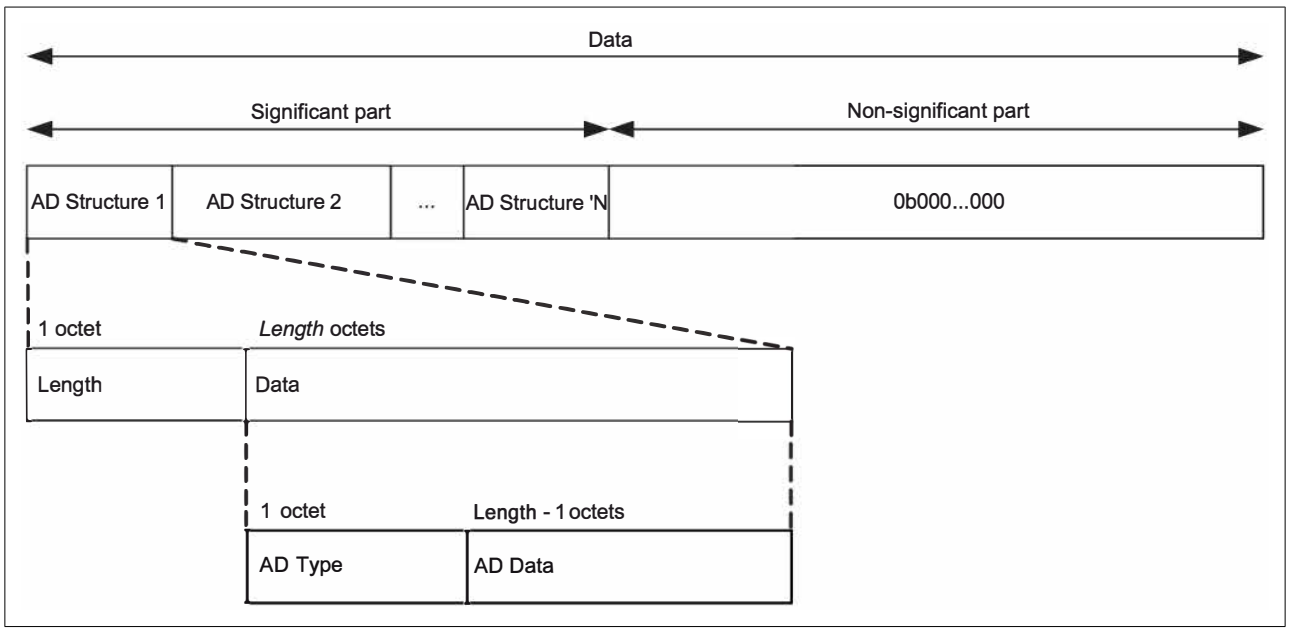



*Figure 11.1: Advertising and Scan Response data format*

Only the significant part of the data should be sent over the air.

The data is sent in advertising or periodic advertising events. Host Advertising data is placed in the AdvData field of ADV_IND, ADV_NONCONN_IND, ADV_SCAN_IND, AUX_ADV_IND, and AUX_CHAIN_IND PDUs. Additional Controller Advertising Data is placed in the ACAD field of AUX_ADV_IND, AUX_SYNC_IND, and AUX_SCAN_RSP PDUs. Periodic Advertising data is placed in the AdvData field of AUX_SYNC_IND, AUX_SYNC_SUBEVENT_IND, AUX_SYNC_SUBEVENT_RSP, and AUX_CHAIN_IND PDUs. Scan Response data is sent in the ScanRspData field of SCAN_RSP PDUs or the AdvData field of AUX_SCAN_RSP PDUs. If the complete data cannot fit in the AdvData field of an AUX_ADV_IND, AUX_SYNC_IND, or AUX_SCAN_RSP PDU,

AUX_CHAIN_IND PDUs are used to send the remaining fragments of the data. In this case, an AD Structure may be fragmented over two or more PDUs.

The AD type data formats and meanings are defined in Section 1 of [4]. The AD type identifier values are defined in Assigned Numbers.

# 12 GAP SERVICE AND CHARACTERISTICS FOR GATT SERVER

The GATT Server shall contain the GAP Service as defined in the GAP Service requirements in Table 12.1. A device shall have only one instance of the GAP Service in the GATT Server. The GAP Service shall be a GATT based primary service with the service UUID as «GAP Service» defined in Assigned Numbers.

| | BR/EDR GAP Role | LE Broadcaster | LE Observer | LE Peripheral | LE Central |
|---|---|---|---|---|---|
| **GAP Service** | C.1 | E | E | M | M |
| C.1: Mandatory if the GATT Profile is supported on the BR/EDR physical transport, otherwise excluded | | | | | |

*Table 12.1: GAP Service requirements*

The GAP Service shall contain the characteristics specified in Table 12.2.

| Characteristics | Ref. | BR/EDR GAP Role | LE Broadcaster | LE Observer | LE Peripheral | LE Central |
|---|---|---|---|---|---|---|
| Device Name | 12.1 | C.1 | E | E | M | M |
| Appearance | 12.2 | C.1 | E | E | M | M |
| Peripheral Preferred Connection Parameters | 12.3 | O | E | E | O | E |
| Central Address Resolution | 12.4 | O | E | E | C.3 | C.2 |
| Resolvable Private Address Only | 12.5 | O | E | E | C.3 | C.3 |
| Encrypted Data Key Material | 12.6 | O | E | E | O | E |
| LE GATT Security Levels | 12.7 | E | E | E | O | O |
| C.1: Mandatory if the GATT Profile is supported on the BR/EDR physical transport, otherwise excluded<br>C.2: Mandatory if Link Layer privacy is supported, otherwise excluded<br>C.3: Optional if Link Layer privacy is supported, otherwise excluded | | | | | | |

*Table 12.2: Requirements related to GAP Service characteristics*

A device that supports multiple GAP roles shall contain all the characteristics required by all the supported roles. The device shall continue to make all the contained characteristics accessible irrespective of the role the device is operating in.

## 12.1 Device Name characteristic

The Device Name characteristic shall contain the name of the device as an UTF-8 string as defined in Section 3.2.2. When the device is discoverable, the Device Name characteristic value shall be readable without authentication or authorization. When the device is not discoverable, the Device Name Characteristic should not be readable without authentication or authorization. The Device Name characteristic value may be writable. If writable, authentication and authorization may be defined by a higher layer specification or be implementation specific.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2A00 – UUID for «Device Name» | Device Name | Readable without authentication or authorization when discoverable.<br>Optionally writable, authentication and authorization may be defined by a higher layer specification or be implementation specific. |

*Table 12.3: Device Name characteristic*

The Device Name characteristic value shall be 0 to 248 octets in length.

A device shall have only one instance of the Device Name characteristic.

## 12.2 Appearance characteristic

The Appearance characteristic defines the representation of the external appearance of the device. This enables the discovering device to represent the device to the user using an icon, or a string, or similar. The Appearance characteristic value shall be readable without authentication or authorization. The Appearance characteristic value may be writable. If writable, authentication and authorization may be defined by a higher layer specification or be implementation specific.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2A01 – UUID for «Appearance» | Appearance | Readable without authentication or authorization.<br>Optionally writable, authentication and authorization may be defined by a higher layer specification or be implementation specific. |

*Table 12.4: Appearance characteristic*

The Appearance characteristic value shall be the enumerated value as defined in Assigned Numbers. The Appearance characteristic value shall be 2 octets in length. A device shall have only one instance of the Appearance characteristic.

## 12.3 Peripheral Preferred Connection Parameters characteristic

The Peripheral Preferred Connection Parameters (PPCP) characteristic contains the preferred connection parameters of the Peripheral.

The Peripheral Preferred Connection Parameters characteristic value shall be readable. Authentication and authorization may be defined by a higher layer specification or be implementation specific.

The Peripheral Preferred Connection Parameters characteristic value shall not be writable.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2A04 – UUID for «Peripheral Preferred Connection Parameters» | Peripheral Preferred Connection Parameters | Readable, authentication and authorization may be defined by a higher layer specification or be implementation specific<br>Not writable |

*Table 12.5: Peripheral Preferred Connection Parameters characteristic*

The Peripheral Preferred Connection Parameters characteristic value shall be 8 octets in length. A device shall have only one instance of the Peripheral Preferred Connection Parameters characteristic. The format of the value is specified in Table 12.6.

| Name | Size (Octet) |
|---|---|
| Interval_Min | 2 |
| Interval_Max | 2 |
| Latency | 2 |
| Timeout | 2 |

*Table 12.6: Format of the Peripheral Preferred Connection Parameters data*

Each field shall have the same meaning and requirements as the field of the LL_CONNECTION_PARAM_REQ PDU (see [Vol 6] Part B, Section 2.4.2.16) with the same name, or shall contain the value 0xFFFF which indicates that no specific value is requested.

## 12.4 Central Address Resolution characteristic

The Peripheral should check if the peer device supports address resolution by reading the Central Address Resolution characteristic before using directed advertisement where the target address is set to a Resolvable Private Address (RPA).

The Central Address Resolution characteristic defines whether the device supports privacy with address resolution. See Table 12.7.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2AA6 - UUID of «Central Address Resolution» | Central Address Resolution Support | Readable without authentication or authorization.<br>Not writable. |

*Table 12.7: Central Address Resolution characteristic*

The Central Address Resolution characteristic value shall be 1 octet in length:

0 = address resolution is not supported in this device

1 = address resolution is supported in this device

All other values are reserved for future use.

A device shall have only one instance of the Central Address Resolution characteristic. If the Central Address Resolution characteristic is not present, then it shall be assumed that Central Address Resolution is not supported.

## 12.5 Resolvable Private Address Only characteristic

The device should check if the peer will only use Resolvable Private Addresses (RPAs) after bonding by reading the Resolvable Private Address Only characteristic, in order to determine if it will satisfy its privacy mode as defined in Section 10.7.

The Resolvable Private Address Only characteristic defines whether the device will only use Resolvable Private Addresses (RPAs) as local addresses. See Table 12.8.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2AC9 - UUID of «Resolvable Private Address Only» | Resolvable Private Address Only | Readable without authentication or authorization.<br>Not writable. |

*Table 12.8: Resolvable Private Address Only characteristic*

The Resolvable Private Address Only characteristic value shall be 1 octet in length:

0 = only Resolvable Private Addresses will be used as local addresses after bonding

All other values are reserved for future use.

A device shall have only one instance of the Resolvable Private Address Only characteristic. If the Resolvable Private Address Only characteristic is not present, then it cannot be assumed that only Resolvable Private Addresses will be used over the air.

## 12.6 Encrypted Data Key Material

The Encrypted Data Key Material characteristic allows advertising data associated with the GAP Service to be decrypted and authenticated using the key material. The Encrypted Data Key characteristic value shall not be writable. The Encrypted Data Key Material characteristic shall only be readable when authenticated and authorized. When read, the key material is stored on the local device. The key material may be discarded at any time on a local device.

The Encrypted Data Key Material characteristic may support indications. If the characteristic supports indications, the client has configured the characteristic for indications, and the characteristic value changes after being authenticated and authorized, then the characteristic shall be indicated by the server to the client.

The Encrypted Data Key Material characteristic can be used by other services to allow those services to expose separate key material for encrypted advertising data used by those services.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2B88 - UUID for «Encrypted Data Key Material» | Key Material | Readable when authenticated and authorized.<br>Indicatable when authenticated and authorized.<br>Not writable. |

*Table 12.9: Encrypted Data Key Material characteristic*

The Key Material is composed of a 128-bit value that is used as the session key and a 64-bit value that is used as the IV for encrypting and authenticating the Encrypted Data as defined in Section 1.23.3 of [4], as shown in Table 12.10. The server should update the Key Material periodically.

| Field | Size (octets) | Description |
|---|---|---|
| Session Key | 16 | The shared session key. |
| IV | 8 | The initialization vector. |

*Table 12.10: Key Material format*

## 12.7 LE GATT Security Levels Characteristic

This section specifies the LE GATT Security Levels characteristic.

The LE GATT Security Levels characteristic shall contain the highest security requirements of the GATT server when operating on a LE connection. The value of the LE GATT Security Levels characteristic shall be static during a connection.

The format of the LE GATT Security Levels characteristic is given in Table 12.11.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xMMMM | 0x2BF5 - UUID for «LE GATT Security Levels» | Sequence of one or more Security Level Requirements | Read Only,<br>No Encryption required,<br>No Authentication,<br>No Authorization |

*Table 12.11: LE GATT Security Levels characteristic*

The Attribute Value is a sequence of Security Level Requirements, each with the type uint8[2]. Each Security Level Requirement consists of a Security Mode field followed by a Security Level field. The Security Mode and Security Level shall be expressed as the same number as used in their definitions; e.g., mode 1 is represented as 0x01 and level 4 is represented as 0x04.

Meeting any one of the security requirements provided for a given mode shall be sufficient for the GATT server to allow the GATT client to use all the GATT procedures permitted by the characteristic properties for all characteristics on the server operating in that mode. If any one of the security requirements specified in the LE GATT Security Levels characteristic is met, no GATT procedure will fail for a security-related reason (such as insufficient authentication). For example, the attribute value 0x01 0x04 for the LE GATT Security Levels characteristic means that the GATT server requires level 4 when operating in security mode 1 on a LE connection.

Note: The Security Level value is not a minimum; specifying (say) level 2 for a mode does not mean that level 3, level 4, or even level 87 would suffice instead. However, in many cases a given security level satisfies the security requirements of other levels as well. For example, LE security mode 1 levels 3 and 4, by definition, both satisfy the requirements for mode 1 level 2 and so could be used with a GATT server that requires mode 1 level 2.

The security modes and levels for LE are defined in Section 10.

A device shall have at most one instance of a LE GATT Security Levels characteristic.

# 13 BR/EDR/LE OPERATION

This section describes the requirements for BR/EDR/LE implementations. The implementation may support any LE GAP roles allowed by the Controller over the LE physical channel. The requirements for each role are defined in Section 9.

| Feature | Ref. | Broadcaster | Observer | Peripheral | Central |
|---|---|---|---|---|---|
| BR/EDR/LE modes and procedures | 13.1 and 13.2 | O | O | O | O |

*Table 13.1: Requirements for the modes of BR/EDR/LE implementations*

## 13.1 Modes, procedures, and security aspects

All modes, procedures and security aspects shall follow the requirements as specified for the physical transport over which they operate. Section 4, Section 5, Section 6 and Section 7 specify the requirements for the modes, procedures and security aspects for operations performed over the BR/EDR physical transport. Section 9 and Section 10 specify the requirements for the modes, procedures and security aspects performed over the LE physical transport. It is optional to support both physical transports simultaneously to the same remote device.

### 13.1.1 Discoverable mode requirements

A device shall expose the capabilities of both physical transports for both Limited and General Discoverable Mode using the advertisement Flag AD Type as follows:

a. The ‘BR/EDR Not Supported’ bit in the Flags AD type shall be set to 0 as defined in Section 1.3 of [4].

b. The 'Simultaneous LE and BR/EDR to Same Device Capable (Controller)' bit in the Flags AD type shall be set to 0.

The ‘LE Supported (Controller)’ and ‘LE Supported (Host)’ bits in the LMP features shall be set as defined in [Vol 2] Part C, Section 3.2.

## 13.2 Bonding for BR/EDR/LE implementations

The requirements for BR/EDR/LE implementations are shown in Table 13.2.

| Bonding Requirement | Ref. | Peripheral | Central |
|---|---|---|---|
| Bonding | 6.5 / 9.4.4 | O | O |

*Table 13.2: Requirements for the bonding of BR/EDR/LE implementations*

If the remote device supports the BR/EDR physical transport, the bonding procedures for the BR/EDR physical transport as defined in Section 6.5 shall be used.

If the remote device supports the LE physical transport, the bonding procedures for the LE physical transport as defined in Section 9.4.4 shall be used.

## 13.3 Relationship between physical transports

To determine if both the BR/EDR physical transport and the LE physical transport are established to the same peer device, the device shall use either the public address used on the LE advertising physical channel or the public address from the BD_ADDR field contained in the SMP Identity Address Information packet ([Vol 3] Part H, Section 3.6.5) if it has been received.

# 14 BR/EDR/LE SECURITY ASPECTS

The requirements for BR/EDR/LE implementations are shown in Table 14.1.

| Security aspects | Ref. | Peripheral | Central |
|---|---|---|---|
| Security aspects | 14 | M | M |

*Table 14.1: Requirements for the security aspects of BR/EDR/LE implementations*

If the remote device supports the BR/EDR physical transport the security procedures for the BR/EDR physical transport as defined in Section 5 shall be used.

If the remote device supports the LE physical transport the security procedures for the LE physical transport as defined in Section 10 shall be used.

If the LE transport in a BR/EDR/LE device supports Secure Connections, the security procedures in Section 14.1 may also be used.

## 14.1 Cross-transport key derivation

If both the local and remote devices support Secure Connections over the BR/EDR and LE transports, devices may generate keys of identical strength and the same MITM protection for both transports as part of a single pairing procedure (see [Vol 3] Part H, Section 2.3.5.7).

If both the local and remote devices support Secure Connections over the LE transport but not over the BR/EDR transport, then the devices may generate the BR/EDR keys of identical strength and the same MITM protection as the LE keys as part of the LE pairing procedure (see [Vol 3] Part H, Section 2.3.5.7).

If an LE LTK already exists and the BR/EDR link key is weaker in either strength or MITM protection, then neither device shall generate an LE LTK using cross-transport key derivation from a BR/EDR link key. If either device receives a request to generate an LE LTK using cross-transport key derivation, it shall respond with a Pairing Failed message with reason "Cross-transport Key Derivation/Generation not allowed".

If the BR/EDR link key has been generated by a Controller that does not perform remote public key validation (see [Vol 2] Part H, Section 7.6), then the local device shall not generate an LE LTK using cross-transport key derivation from a BR/EDR link key. If the local device receives a request to generate an LE LTK using cross-transport key derivation, it shall respond with a Pairing Failed message with reason "Cross-transport Key Derivation/Generation not allowed".

If a BR/EDR link key already exists and the LE LTK is weaker in either strength or MITM protection, then the local device shall not generate a BR/EDR link key using

cross-transport key derivation from the LE LTK. If both devices request cross-transport key derivation during pairing, then the local device shall either silently skip deriving the BR/EDR link key or stop the pairing procedure by sending a Pairing Failed message with reason "Cross-transport Key Derivation/Generation not allowed".

Note: The Host can use the HCI_Read_Local_Simple_Pairing_Options command (see [Vol 4] Part E, Section 7.4.9) or vendor-specific methods to determine whether the Controller performs remote public key validation.

If the LE LTK has been generated (e.g., using the HCI_LE_Generate_DHKey command; see [Vol 4] Part E, Section 7.8.37) by a Controller that does not perform remote public key validation (see [Vol 3] Part H, Section 2.3.5.6.1), then the BR/EDR link key shall not be generated from such an LE LTK using cross-transport key derivation.

Note: The Host can use the Remote Public Key Validation feature bit (see [Vol 6] Part B, Section 4.6) or vendor-specific methods to determine whether the HCI_LE_Generate_DHKey command performs the remote public key validation.

## 14.2 Collision handling

If pairing has been initiated by the local device on the BR/EDR transport, and a pairing request is received from the same remote device on the LE transport, the LE pairing shall be rejected with SMP error code *BR/EDR Pairing in Progress* (0x0D) if both sides support LE Secure Connections.

If a BR/EDR/LE device supports LE Secure Connections, then it shall initiate pairing on only one transport at a time to the same remote device.

## 14.3 Secure Connections Only Mode

If a BR/EDR/LE device is in Secure Connections Only Mode, then this mode applies to both transports and the requirements in both Section 5.2.2 and Section 10.2.4 shall apply.

# 15 BLUETOOTH DEVICE REQUIREMENTS

## 15.1 Bluetooth Device address

All Bluetooth devices shall have a Bluetooth Device Address (BD_ADDR) that uniquely identifies the device to another Bluetooth device. The specific Bluetooth Device Address requirements depend on the type of Bluetooth device.

### 15.1.1 Bluetooth Device Address types

#### 15.1.1.1 Public Bluetooth address

A Bluetooth public address used as the BD_ADDR for the BR/EDR physical channel is defined in [Vol 2] Part B, Section 1.2. A Bluetooth public address used as the BD_ADDR for the LE physical channel is defined in [Vol 6] Part B, Section 1.3.

#### 15.1.1.2 Random Bluetooth address

A random device address used as the BD_ADDR on the LE physical channel is defined in Section 10.8.

## 15.2 GATT Profile requirements

The requirements for supporting a GATT Client or GATT Server are specified in Table 15.1.

| | BR/EDR GAP Role | LE Broadcaster | LE Observer | LE Peripheral | LE Central |
|---|---|---|---|---|---|
| **GATT Client** | C.1 | E | E | O | O |
| **GATT Server** | C.2 | E | E | M | M |

C.1 Optional if the GATT Profile is supported on the BR/EDR physical transport; otherwise excluded

C.2: Mandatory if the GATT Profile is supported on the BR/EDR physical transport; otherwise excluded

*Table 15.1: Requirements based on GAP roles supported*

## 15.3 SDP requirements

The requirements for supporting an SDP Client or SDP Server are specified in Table 15.2. There shall be no more than one active SDP Server per device.

| | BR/EDR-Only and BR/EDR/LE Implementations | LE-Only Implementations |
|---|---|---|
| **SDP Client** | C.1 | E |
| **SDP Server** | C.2 | E |

C.1: Optional if SDP Server is supported, otherwise mandatory.

C.2: Mandatory to support if GATT Server is supported, otherwise optional.

*Table 15.2: Requirements based on GAP roles supported*

## 15.4 SDP service record requirement

A BR/EDR or BR/EDR/LE device that supports a GATT Server accessible over the BR/EDR physical transport and that supports only one of ATT or EATT shall publish the SDP record shown below in Table 15.3; if both ATT and EATT are supported, the device shall publish the SDP record shown below in Table 15.4. The GAP Service start handle shall be set to the attribute handle of the GAP Service service declaration. The GAP Service end handle shall be set to the attribute handle of the last attribute within the GAP Service service definition group.

| Item | Type | Value | Meaning |
|---|---|---|---|
| Attribute ID | uint16 | 0x0001 | ServiceClassIDList |
| Attribute Value | Data element sequence (1 item) | | |
| Service Class | UUID | «GAP Service» | |
| Attribute ID | uint16 | 0x0004 | ProtocolDescriptorList |
| Attribute Value | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x001F *or* 0x0027 | PSM = ATT *or* PSM = EATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GAP Service start handle |
| Parameter 1 | uint16 | 0xHHHH | GAP Service end handle |
| Attribute ID | uint16 | 0x0005 | BrowseGroupList |
| Attribute Value | Data element sequence (1 item) | | |
| Group | UUID | «PublicBrowseRoot» | |

*Table 15.3: SDP record for the Generic Access Profile, if only one of ATT or EATT is supported*

| Item | Type | Value | Meaning |
|---|---|---|---|
| Attribute ID | uint16 | 0x0001 | ServiceClassIDList |
| Attribute Value | Data element sequence (1 item) | | |
| Service class | UUID | «GAP Service» | |
| Attribute ID | uint16 | 0x0004 | ProtocolDescriptorList |
| Attribute Value | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x001F | PSM = ATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GAP Service start handle |
| Parameter 1 | uint16 | 0xHHHH | GAP Service end handle |
| Attribute ID | uint16 | 0x000D | AdditionalProtocolDescriptorLists |
| Attribute Value | Data element sequence (1 item) | | |
| Protocol Descriptor List | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x0027 | PSM = EATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GAP Service start handle |
| Parameter 1 | uint16 | 0xHHHH | GAP Service end handle |
| Attribute ID | uint16 | 0x0005 | BrowseGroupList |
| Attribute Value | Data element sequence (1 item) | | |
| Group | UUID | «PublicBrowseRoot» | |

*Table 15.4: SDP record for the Generic Access Profile, if both ATT and EATT are supported*

If a BR/EDR or BR/EDR/LE device supports a GATT-based service on the BR/EDR transport, the service shall exist in the SDP Server and the GATT Server.

# 16 DEFINITIONS

In the following, terms written with capital letters refer to states.

Most definitions in this section are BR/EDR specific.

## 16.1 General definitions

**Mode:** A set of directives that defines how a device will respond to certain events.

**Idle:** As seen from a remote device, a Bluetooth device is idle, or is in idle mode, when there is no link established between them.

**Bond:** A relation between two Bluetooth devices defined by creating, exchanging and storing a common link key. The bond is created through the bonding or LMP-pairing procedures.

## 16.2 Connection-related definitions

**Physical channel:** See [Vol 2] Part B, Section 2.1.

**Piconet:** A set of Bluetooth devices sharing the same physical channel defined by the Central's parameters (clock and BD_ADDR).

**Physical link:** A Baseband-level connection[1] between two devices established using paging. A physical link comprises a sequence of transmission slots on a physical channel alternating between Central and Peripheral transmission slots.

**ACL link:** An asynchronous (packet-switched) connection[1] between two devices created on LMP level. Traffic on an ACL link uses ACL packets to be transmitted.

**SCO link:** A synchronous (circuit-switched) connection[1] for reserved bandwidth communications; e.g. voice between two devices, created on the LMP level by reserving slots periodically on a physical channel. Traffic on a SCO link uses SCO packets to be transmitted. SCO links can be established only after an ACL link has first been established.

**Link:** Shorthand for an ACL link.

**PAGE:** A Baseband state where a device transmits page trains, and processes any eventual responses to the page trains.

[1]The term 'connection' used here is not identical to the definition below. It is used in the absence of a more concise term.

**PAGE_SCAN:** A Baseband state where a device listens for page trains.

**Page:** The transmission by a device of page trains containing the Device Access Code of the device to which the physical link is requested.

**Page scan:** The listening by a device for page trains containing its own Device Access Code.

**Channel:** A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.

**Connection:** A connection between two peer applications or higher layer protocols mapped onto a channel.

**Connecting:** A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)

**Connect (to service):** The establishment of a connection to a service. If not already done, this includes establishment of a physical link, link and channel as well.

## 16.3 Device-related definitions

**Discoverable device:** A Bluetooth device in range that will respond to an inquiry (normally in addition to responding to page).

**Silent device:** A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device. A device may be silent due to being non-discoverable or due to baseband congestion while being discoverable.

**Connectable device:** Bluetooth device in range that will respond to a page.

**Trusted device:** A paired device that is explicitly marked as trusted.

**Paired device:** A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

**Pre-paired device:** A Bluetooth device with which a link key was exchanged, and the link key is stored, before link establishment.

**Un-paired device:** A Bluetooth device for which there was no exchanged link key available before connection establishment was requested.

**Known device:** A Bluetooth device for which at least the BD_ADDR is stored.

**Un-known device:** Bluetooth device for which no information (BD_ADDR, link key or other) is stored.

**Authenticated device:** A Bluetooth device whose identity has been verified during the lifetime of the current link, based on the authentication procedure.

## 16.4 Procedure-related definitions

**Paging:** A procedure for establishing a physical link of ACL type on Baseband level, consisting of a page action of the initiator and a page scan action of the responding device.

**Link establishment:** A procedure for establishing a link on LMP level. A link is established when both devices have agreed that LMP setup is completed.

**Channel establishment:** A procedure for establishing a channel on L2CAP level.

**Connection establishment:** A procedure for creating a connection mapped onto a channel.

**Creation of a trusted relationship:** A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).

**Creation of a secure connection:** A procedure of establishing a connection, including authentication and encryption.

**Device discovery:** A procedure for retrieving the Bluetooth Device Address, clock, and Class of Device from discoverable devices.

**Name discovery:** A procedure for retrieving the user-friendly name (the Bluetooth Device Name) of a connectable device.

**Service discovery:** Procedures for querying and browsing for services offered by or through another Bluetooth device.

## 16.5 Security-related definitions

**Authentication:** A generic procedure based on LMP-authentication if a link key exists or on LMP-pairing if no link key exists.

**LMP-authentication:** An LMP level procedure for verifying the identity of a remote device. The procedure is based on a challenge-response mechanism using a random number, a secret key and the BD_ADDR of the non-initiating device. The secret key used can be a previously exchanged link key.

**Authorization:** A procedure where a user of a Bluetooth device grants a specific (remote) Bluetooth device access to a specific service. Authorization implies that the identity of the remote device can be verified through authentication.

**Authorize:** The act of granting a specific Bluetooth device access to a specific service. It may be based upon user confirmation, or given the existence of a trusted relationship.

**LMP-pairing:** A procedure that authenticates two devices, based on a PIN, and subsequently creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. The procedure consists of the steps: creation of an initialization key (based on a random number and a PIN), creation and exchange of a common link key and LMP-authentication based on the common link key.

**Bonding:** A dedicated procedure for performing the first authentication, where a common link key is created and stored for future use.

**Trusting:** The marking of a paired device as trusted. Trust marking can be done by the user, or automatically by the device (e.g. when in bondable mode) after a successful pairing.

# 17 REFERENCES

[1] RFCOMM

[2] Telephony Control Specification

[3] Assigned Numbers Specification: https://www.bluetooth.com/specifications/assigned-numbers

[4] Core Specification Supplement, Part A, Data Types Specification

[5] Core Specification Supplement, Part C, Services Permitted to use Security Mode 4 Level 0

# Appendix A Timers and Constants

The following timers are required by this profile.

| Timer name | Value | Description | Requirement or Recommendation |
|---|---|---|---|
| $T_{GAP}(100)$ | 10.24 s | Time span that a Bluetooth device performs device discovery | Recommended value |
| $T_{GAP}(101)$ | 10.625 ms | A discoverable Bluetooth device enters INQUIRY_SCAN for at least TGAP(101) every TGAP(102) | Required value |
| $T_{GAP}(102)$ | 2.56 s | Maximum time between repeated INQUIRY_SCAN enterings when power consumption or bandwidth is more important than discovery speed | Recommended value |
| $T_{GAP}(103)$ | 30.72 s | Minimum time span that a device is in discoverable mode | Required value |
| $T_{GAP}(104)$ | 1 min | Maximum time span that a device is in limited discoverable mode | Recommended value |
| $T_{GAP}(105)$ | 100 ms | Maximum time between INQUIRY_SCAN enterings when discovery speed is more important than power consumption or bandwidth | Recommended value |
| $T_{GAP}(106)$ | 100 ms | Maximum time between PAGE_SCAN enterings when connection speed is more important than power consumption or bandwidth | Recommended value |
| $T_{GAP}(107)$ | 1.28 s | Maximum time between PAGE_SCAN enterings (R1 page scan) when power consumption or bandwidth is more important than connection speed | Recommended value |
| $T_{GAP}(108)$ | 2.56 s | Maximum time between PAGE_SCAN enterings (R2 page scan) when power consumption or bandwidth is more important than connection speed | Recommended value |

| Timer name | Value | Description | Requirement or Recommendation |
|---|---|---|---|
| $T_{GAP}$(adv_fast_-interval1_coded) | 90 ms to 180 ms | Minimum to maximum advertising interval in the following GAP Modes on the LE Coded PHY when user initiated:<br>1. Undirected Connectable Mode<br>2. Limited Discoverable Mode and sending connectable undirected advertising events<br>3. General Discoverable Mode and sending connectable undirected advertising events<br>4. Directed Connectable Mode and sending low duty cycle connectable directed advertising events | Recommended value |
| $T_{GAP}$(adv_fast_-interval1) | 30 ms to 60 ms | Minimum to maximum advertising interval in the following GAP Modes on the LE 1M PHY when user initiated:<br>1. Undirected Connectable Mode<br>2. Limited Discoverable Mode and sending connectable undirected advertising events<br>3. General Discoverable Mode and sending connectable undirected advertising events<br>4. Directed Connectable Mode and sending low duty cycle directed advertising events | Recommended value |
| $T_{GAP}$(adv_fast_-interval2_coded) | 300 ms to 450 ms | Minimum to maximum advertising interval in the following GAP Modes on the LE Coded PHY when user initiated and sending non-connectable advertising events:<br>1. Non-Discoverable Mode<br>2. Non-Connectable Mode<br>3. Limited Discoverable Mode<br>4. General Discoverable Mode | Recommended value |

| Timer name | Value | Description | Requirement or Recommendation |
|---|---|---|---|
| $T_{GAP}$(adv_fast_-interval2) | 100 ms to 150 ms | Minimum to maximum advertising interval in the following GAP Modes on the LE 1M PHY when user initiated and sending non-connectable advertising events:<br>1. Non-Discoverable Mode<br>2. Non-Connectable Mode<br>3. Limited Discoverable Mode<br>4. General Discoverable Mode | Recommended value |
| $T_{GAP}$(adv_fast_period) | 30 s | Minimum time to perform advertising when user initiated | Recommended value |
| $T_{GAP}$(adv_slow_-interval_coded) | 3 s to 3.6 s | Minimum to maximum advertisement interval in any discoverable or connectable mode when background advertising on the LE Coded PHY | Recommended value |
| $T_{GAP}$(adv_slow_-interval) | 1 s to 1.2 s | Minimum to maximum advertisement interval in any discoverable or connectable mode when background advertising on the LE 1M PHY | Recommended value |
| $T_{GAP}$(conn_param_-timeout) | 30 s | Connection parameter update notification timer when performing the connection parameter update procedure | Recommended value |
| $T_{GAP}$(conn_pause_-central) | 1 s | Central idle timer | Recommended value |
| $T_{GAP}$(conn_pause_-peripheral) | 5 s | Minimum time upon connection establishment before the Peripheral starts a connection update procedure | Recommended value |
| $T_{GAP}$(gen_disc_scan_-min_coded) | 30.72 s | Minimum time to perform scanning when performing the general discovery procedure on the LE Coded PHY | Recommended value |
| $T_{GAP}$(gen_disc_-scan_min) | 10.24 s | Minimum time to perform scanning when performing the general discovery procedure on the LE 1M PHY | Recommended value |
| $T_{GAP}$(initial_conn_-interval_coded) | 90 ms to 150 ms | Minimum to maximum connection interval upon any connection establishment on the LE Coded PHY | Recommended value |
| $T_{GAP}$(initial_conn_-interval) | 30 ms to 50 ms | Minimum to maximum connection interval upon any connection establishment on the LE 1M PHY | Recommended value |
| $T_{GAP}$(lim_adv_timeout) | 180 s | Maximum time to remain advertising when in the limited discoverable mode | Required value |

| Timer name | Value | Description | Requirement or Recommendation |
|---|---|---|---|
| $T_{GAP}$(lim_disc_scan_int_coded) | 33.75 ms | Scan interval used in the limited discovery procedure on the LE Coded PHY | Recommended value |
| $T_{GAP}$(lim_disc_scan_int) | 11.25 ms | Scan interval used in the limited discovery procedure on the LE 1M PHY | Recommended value |
| $T_{GAP}$(lim_disc_scan_min_coded) | 30.72 s | Minimum time to perform scanning when performing the limited discovery procedure on the LE Coded PHY | Recommended value |
| $T_{GAP}$(lim_disc_scan_min) | 10.24 s | Minimum time to perform scanning when performing the limited discovery procedure on the LE 1M PHY | Recommended value |
| $T_{GAP}$(private_addr_int) | 15 min | Maximum time interval between private address change | Recommended value |
| $T_{GAP}$ (scan_fast_interval_coded | 90 ms to 180 ms | Scan interval in any discovery or connection establishment procedure when user initiated on the LE Coded PHY | Recommended value |
| $T_{GAP}$(scan_fast_interval) | 30 ms to 60 ms | Scan interval in any discovery or connection establishment procedure when user initiated on the LE 1M PHY | Recommended value |
| $T_{GAP}$(scan_fast_period) | 30.72 s | Minimum time to perform scanning when user initiated | Recommended value |
| $T_{GAP}$(scan_fast_window_coded) | 90 ms | Scan window in any discovery or connection establishment procedure when user initiated on the LE Coded PHY | Recommended value |
| $T_{GAP}$(scan_fast_window) | 30 ms | Scan window in any discovery or connection establishment procedure when user initiated on the LE 1M PHY | Recommended value |
| $T_{GAP}$(scan_slow_interval1_coded) | 3.84 s | Scan interval in any discovery or connection establishment procedure when background scanning on the LE Coded PHY | Recommended value |
| $T_{GAP}$(scan_slow_interval1) | 1.28 s | Scan interval in any discovery or connection establishment procedure when background scanning on the LE 1M PHY | Recommended value |
| $T_{GAP}$(scan_slow_interval2_coded) | 7.68 s | Scan interval in any discovery or connection establishment procedure when background scanning on the LE Coded PHY | Recommended value |

| Timer name | Value | Description | Requirement or Recommendation |
|---|---|---|---|
| $T_{GAP}$(scan_slow_-interval2) | 2.56 s | Scan interval in any discovery or connection establishment procedure when background scanning on the LE 1M PHY | Recommended value |
| $T_{GAP}$(scan_slow_-window1_coded) | 33.75 ms | Scan window in any discovery or connection establishment procedure when background scanning on the LE Coded PHY | Recommended value |
| $T_{GAP}$(scan_slow_-window1) | 11.25 ms | Scan window in any discovery or connection establishment procedure when background scanning on the LE 1M PHY | Recommended value |
| $T_{GAP}$(scan_slow_-window2_coded) | 67.5 ms | Scan window in any discovery or connection establishment procedure when background scanning on the LE Coded PHY | Recommended value |
| $T_{GAP}$(scan_slow_-window2) | 22.5 ms | Scan window in any discovery or connection establishment procedure when background scanning on the LE 1M PHY | Recommended value |
| $T_{GAP}$(Sync_Scan_-Interval) | 320 ms | Interval between the start of adjacent synchronization scan windows | Recommended value |
| $T_{GAP}$(Sync_Scan_-Window) | 91.25 ms | Duration of Synchronization scan window | Recommended value |
| $T_{GAP}$(Sync_Train_-Duration) | 30.72 s | Duration of synchronizability mode | Required value |
| $T_{GAP}$(Sync_Train_-Interval) | 80 ms | Interval between Synchronization Train events | Recommended value |

*Table A.1: Defined GAP timers*

# Appendix B Information Flows of Related Procedures

## B.1 LMP – authentication

Authentication at the link level is specified in [Vol 2] Part C, Section 4.2.1.

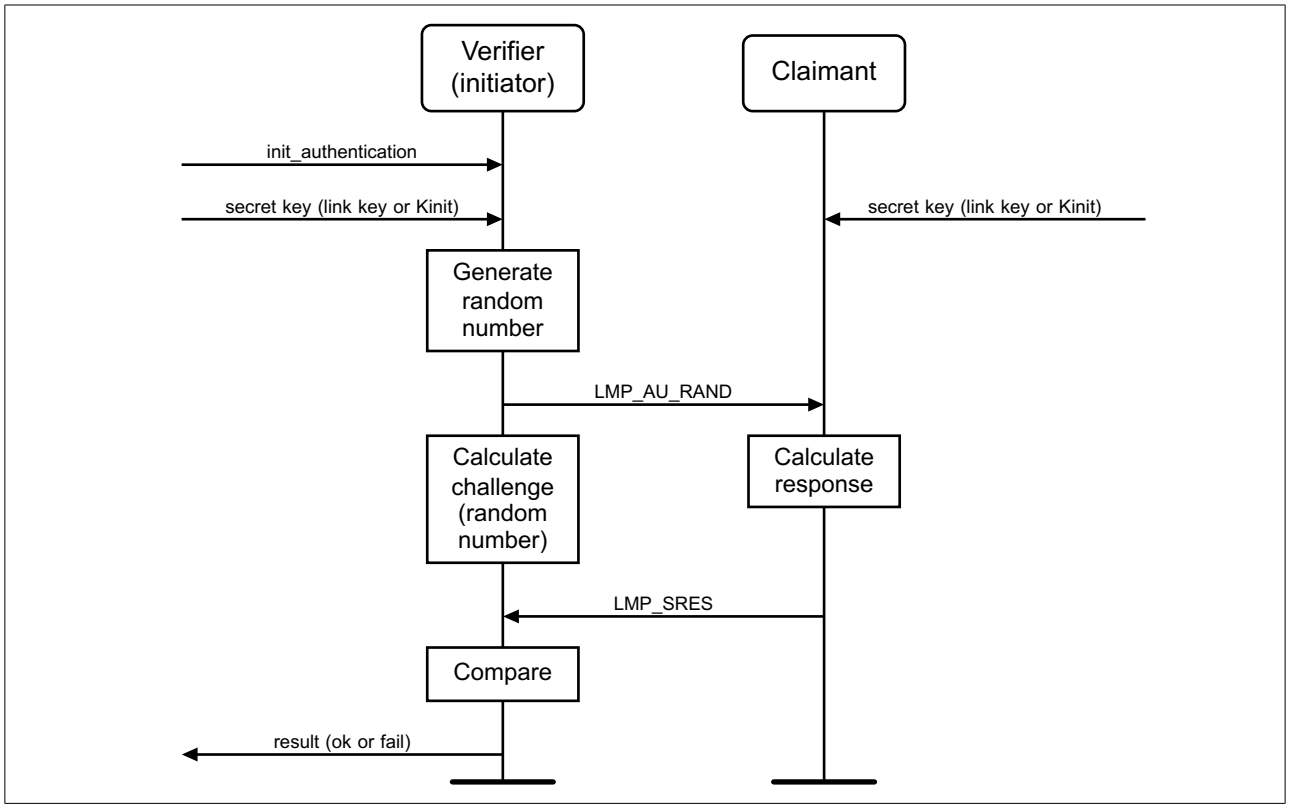



*Figure B.1: LMP authentication*

The secret key used here is an already exchanged link key.

## B.2 LMP – pairing

Pairing at the link level is specified in [Vol 2] Part C, Section 4.2.2.

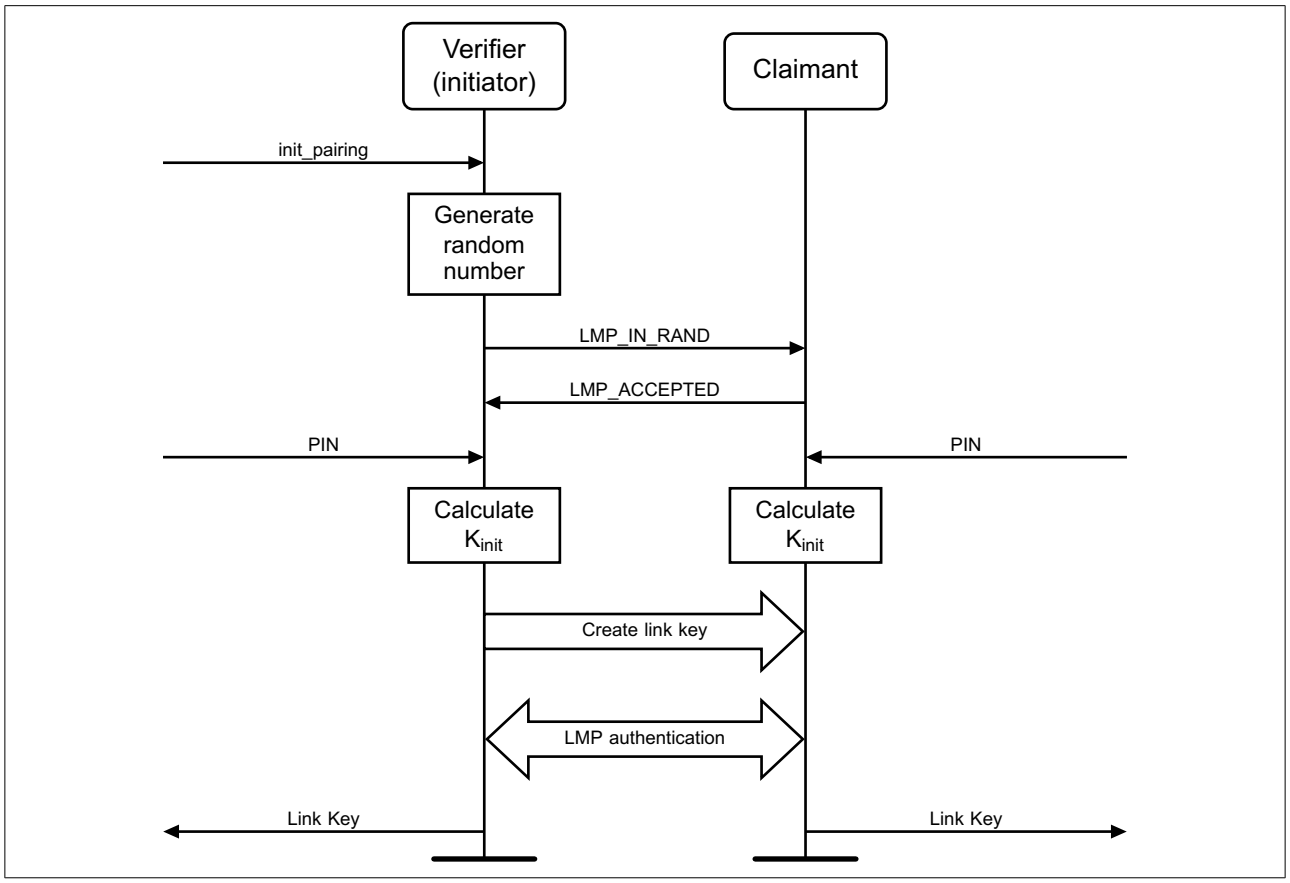



*Figure B.2: LMP pairing*

The PIN used here is PINBB.

The create link key procedure is described in [Vol 2] Part C, Section 4.2.2.4 and [Vol 2] Part H, Section 3.2. A mutual authentication takes place irrespective of the current security mode.

## B.3 Service Discovery

The Service Discovery Protocol specifies what PDUs are used over-the-air to inquire about services and service attributes. The procedures for discovery of supported

services and capabilities using the Service Discovery Protocol are described in higher layer specifications. This is just an example.

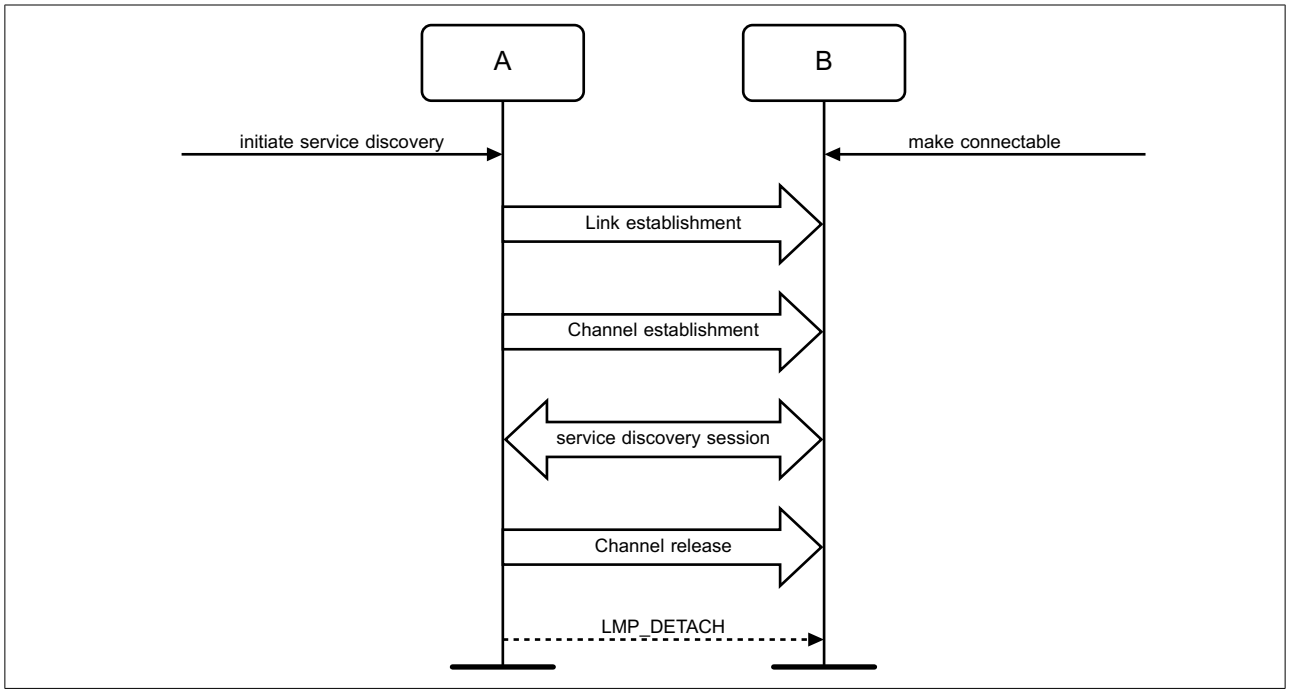



*Figure B.3: Service Discovery procedure*

## B.4 Generating a resolvable private address

Generating a resolvable private address is described in Section 10.8.2.2.

## B.5 Resolving a resolvable private address

Resolving a resolvable private address is described in Section 10.8.2.3.

Host
Part D

# TEST SUPPORT

# CONTENTS

**1 Test methodology ..... 1480**
1.1 BR/EDR test scenarios ..... 1480
1.1.1 Test setup ..... 1480
1.1.2 Transmitter test ..... 1481
1.1.2.1 Packet format ..... 1481
1.1.2.2 Pseudorandom sequence ..... 1483
1.1.2.3 Control of transmit parameters ..... 1484
1.1.2.4 Power control ..... 1484
1.1.2.5 Switch between different frequency settings ..... 1484
1.1.2.6 Adaptive Frequency Hopping ..... 1485
1.1.3 LoopBack test ..... 1485
1.1.4 Pause test ..... 1489
1.2 [This section is no longer used] ..... 1489
1.3 References ..... 1489

**2 [This section is no longer used] ..... 1490**

# 1 TEST METHODOLOGY

This section describes the test modes for hardware and low-level functionality tests of Bluetooth devices.

The BR/EDR Test mode supports testing of the Bluetooth transmitter and receiver including transmitter tests (packets with constant bit patterns) and loopback tests. It is intended mainly for testing of the radio and Baseband and may also be used for in-production and after-sales testing.

## 1.1 BR/EDR test scenarios

A device in BR/EDR Test mode shall not support normal operation. For security reasons the BR/EDR Test mode is designed such that it offers no benefit to the user. Therefore, no data output or acceptance on a HW or SW interface shall be allowed.

### 1.1.1 Test setup

The setup consists of an implementation under test (IUT) and a Test System. Additional measurement equipment may be used.

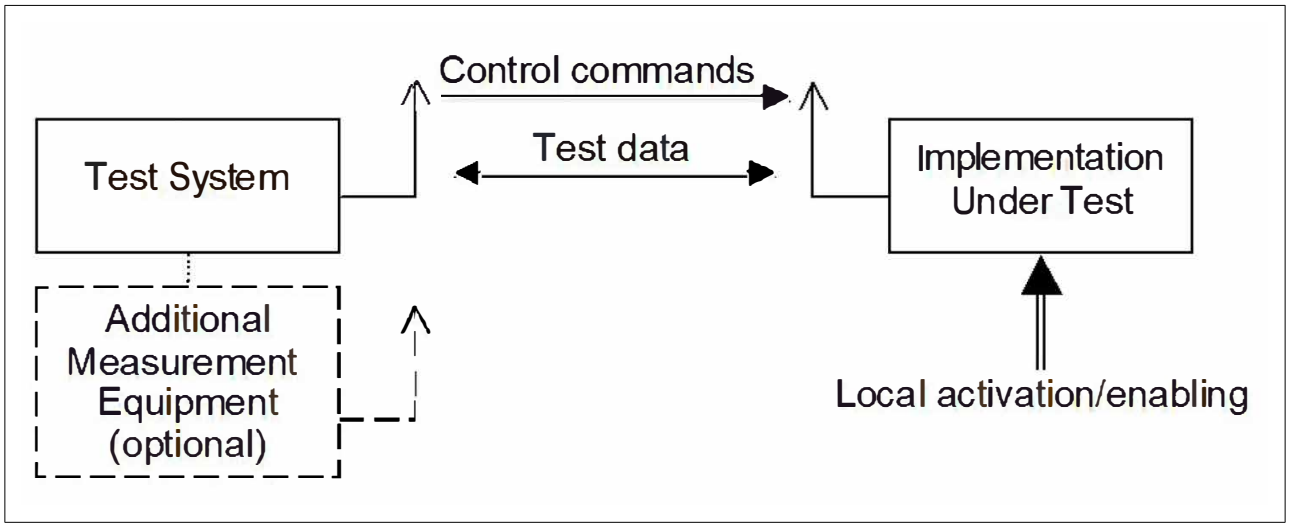



*Figure 1.1: Setup for Test mode*

The Lower Tester and IUT form a piconet where the Lower Tester acts as the Central and has full control over the test procedure. The IUT acts as the Peripheral.

The Lower Tester control is done via the air interface using LMP commands (see [Vol 2] Part C, Section 4.7.3). Hardware interfaces to the IUT may exist, but are not subject to standardization.

The test mode is a special state of the Bluetooth model. For security and type approval reasons, a Bluetooth device in test mode shall not support normal operation. When the IUT leaves the test mode it enters the standby state. After power-off the Bluetooth device shall return to the standby state.

### 1.1.2 Transmitter test

The Bluetooth device transmits a constant bit pattern. This pattern is transmitted periodically with packets aligned to the Peripheral TX timing of the piconet formed by the Lower Tester and IUT. The same test packet is repeated for each transmission.

The transmitter test is started when the Central sends the first POLL packet. In non-hopping mode the agreed frequency is used for this POLL packet.

The Lower Tester (Central) transmits control or POLL packets in the Central-to-Peripheral transmission slots. The IUT (Peripheral) shall transmit packets in the following Peripheral-to-Central transmission slot. The Lower Tester's polling interval is fixed and defined by the LMP_TEST_CONTROL PDU. The implementation under test may transmit its burst according to the normal timing even if no packet from the Lower Tester was received. In this case, the ARQN bit is shall be set to NAK.

The burst length may exceed the length of a one slot packet. In this case the Lower Tester may take the next free Central TX slot for polling. The timing is illustrated in Figure 1.2.

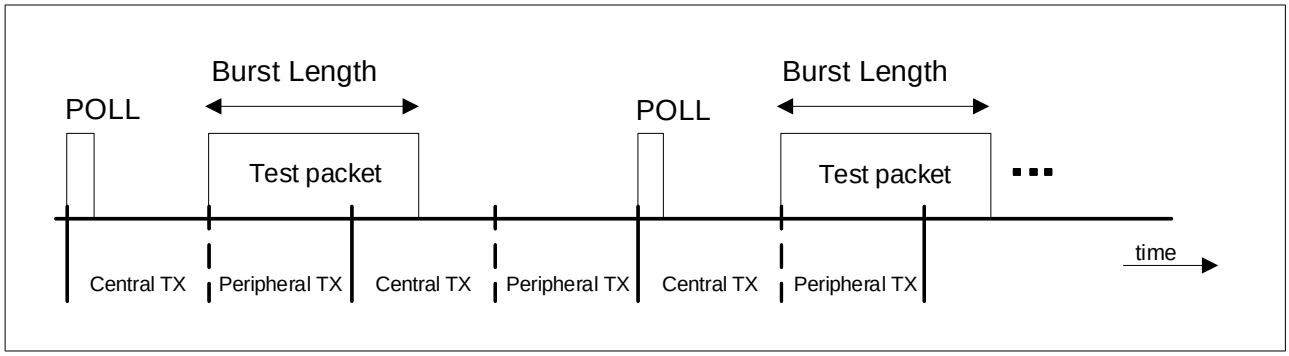



*Figure 1.2: Timing for transmitter test*

#### 1.1.2.1 Packet format

The test packet is a normal Bluetooth packet, see Figure 1.3. For the payload itself see below.

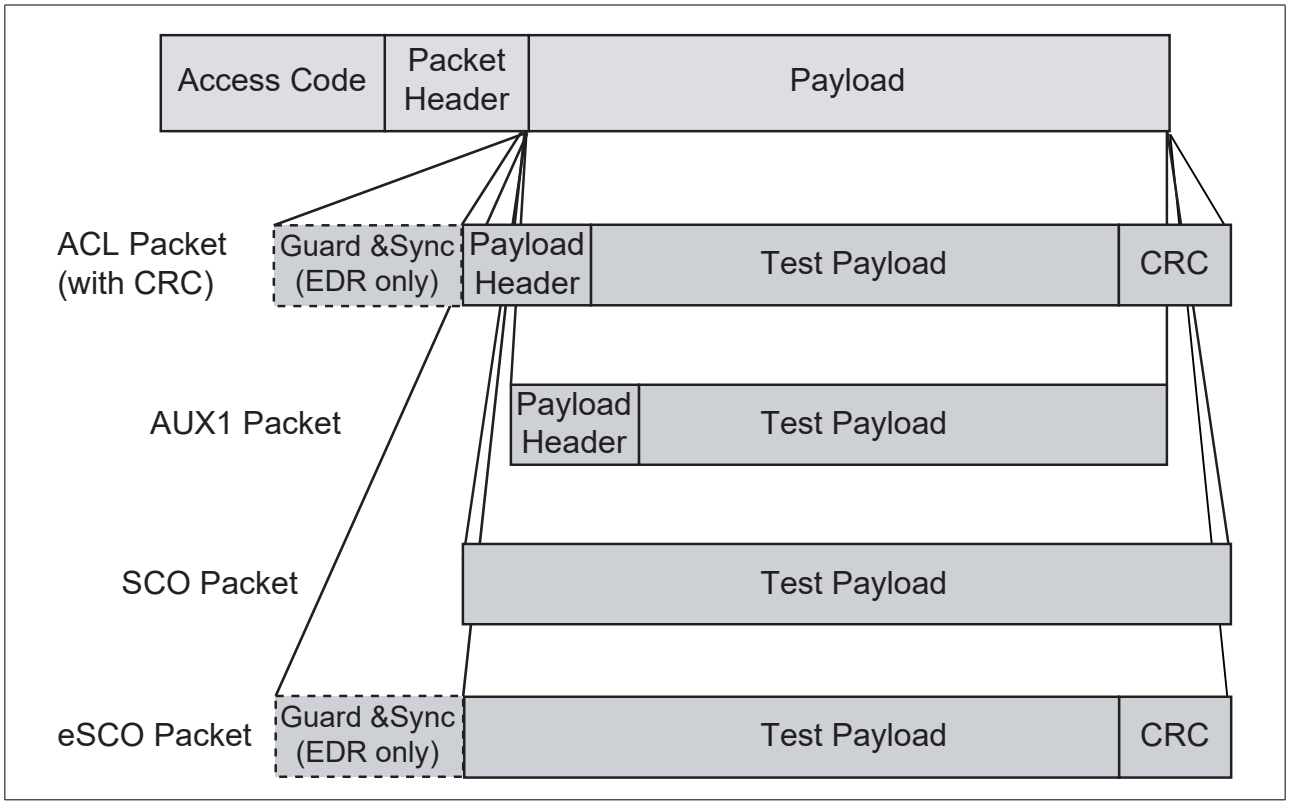



*Figure 1.3: General format of TX packet*

During configuration the Lower Tester defines:

- the packet type to be used
- payload length

For the payload length, the restrictions from the Baseband specification shall apply (see [Vol 2] Part B, Section 6.5). In case of ACL, SCO and eSCO packets the payload structure defined in the Baseband specification is preserved as well, see Figure 1.3.

For the transmitter test mode, only packets without FEC should be used; i.e. HV3, EV3, EV5, DH1, DH3, DH5, 2-EV3, 2-EV5, 3-EV3, 3-EV5, 2-DH1, 2-DH3, 2-DH5, 3-DH1, 3-DH3, 3-DH5 and AUX1 packets.

In transmitter test mode, the packets exchanged between the Lower Tester and the IUT shall not be scrambled with the whitening sequence. Whitening shall be turned off when the IUT has accepted to enter the transmitter test mode, and shall be turned on when the IUT has accepted to exit the transmitter test mode, see Figure 1.4. Implementations shall ensure that retransmissions of the LMP_ACCEPTED messages use the same whitening status as used in the original LMP_ACCEPTED.

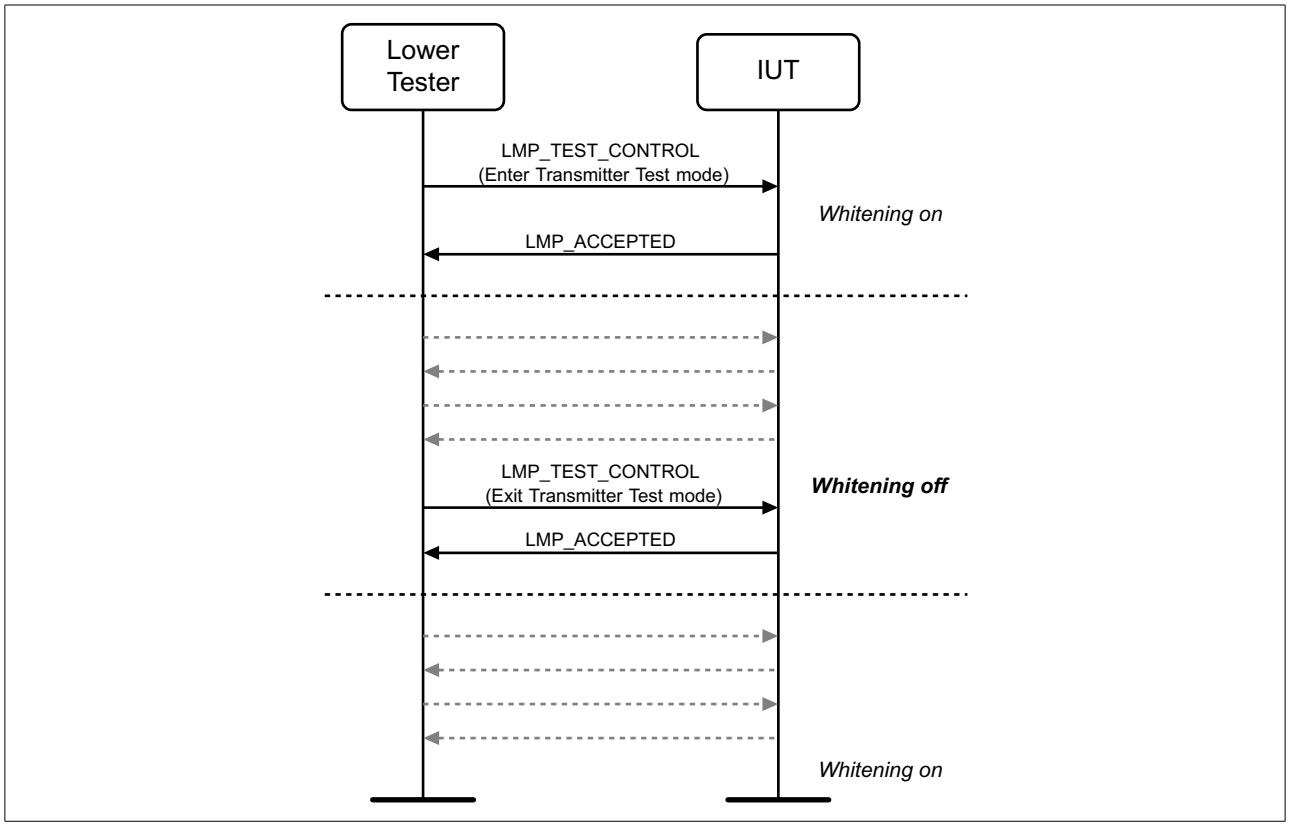



*Figure 1.4: Use of whitening in Transmitter mode*

### 1.1.2.2 Pseudorandom sequence

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated). A PRBS9 sequence is used, see [1] and [2].

The properties of this sequence are as follows (see [2]). The sequence may be generated in a nine-stage shift register whose 5th and 9th stage outputs are XORed (see Figure 1.5), and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONEs; i.e. the shift register is initialized with nine ones.

- Number of shift register stages: 9
- Length of pseudo-random sequence: $2^9$-1 = 511 bits
- Longest sequence of zeros: 8 (non-inverted signal)

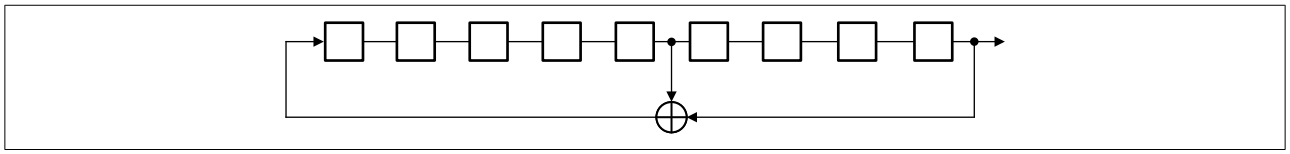


*Figure 1.5: Linear feedback shift register for generation of the PRBS sequence*

#### 1.1.2.3 Control of transmit parameters

The following parameters can be set to configure the transmitter test:

1. Bit pattern:
   - Constant zero
   - Constant one
   - Alternating 1010...[1]
   - Alternating 1111 0000 1111 0000...[1]
   - Pseudorandom bit pattern
   - Transmission off
2. Frequency selection:
   - Single frequency
   - Normal hopping
3. TX frequency
   - (2402 + k) MHz for channel k
4. Default poll period in TDD frames (n × 1.25 ms)
5. Packet Type
6. Length of test payload

#### 1.1.2.4 Power control

When the legacy power control mechanism is tested the IUT shall start transmitting at the maximum power and shall reduce/increase its power by one step on every LMP_INCR_POWER_REQ or LMP_DECR_POWER_REQ PDU received. When the enhanced power control mechanism is tested the IUT shall start transmitting at the maximum power and shall reduce/increase its power by one step or go to the maximum power level when a LMP_POWER_CONTROL_REQ PDU is received.

#### 1.1.2.5 Switch between different frequency settings

A change in the frequency selection becomes effective when the LMP procedure is completed:

When the Lower Tester receives the LMP_ACCEPTED PDU it shall then transmit POLL packets containing the ACK for at least 8 slots (4 transmissions). When

[1]It is recommended that the sequence starts with a one; but, as this is irrelevant for measurements, it is also allowed to start with a zero.

these transmissions have been completed the Lower Tester shall change to the new frequency hop and whitening settings.

After sending LMP_ACCEPTED the IUT shall wait for the LC level ACK for the LMP_ACCEPTED. When this is received it shall change to the new frequency hop and whitening settings.

There will be an implementation defined delay after sending the LMP_ACCEPTED before the TX or loopback test starts. The Lower Tester shall be able to cope with this.

Note: Loss of the LMP_ACCEPTED PDU will eventually lead to a loss of frequency synchronization that cannot be recovered. Similar problems occur in normal operation, when the hopping pattern changes.

#### 1.1.2.6 Adaptive Frequency Hopping

Adaptive Frequency Hopping (AFH) shall only be used when the Hopping Mode is set to 79 channels (e.g., Hopping Mode = 1) in the LMP_TEST_CONTROL PDU. If AFH is used, the normal LMP commands and procedures shall be used. When AFH is enabled prior to entering test mode it shall continue to be used with the same parameters if Hopping Mode = 1 until the AFH parameters are changed by the LMP_SET_AFH PDU.

The channel classification reporting state shall be retained upon entering or exiting Test Mode. The IUT shall change the channel classification reporting state in Test Mode based on control messages from the Lower Tester (e.g., LMP_CHANNEL_CLASSIFICATION_REQ) and from the Host (HCI_Write_AFH_Channel_Assessment_Mode).

### 1.1.3 LoopBack test

In loopback, the implementation under test receives normal Baseband packets containing payload *Accepted* from the Lower Tester. The received packets shall be decoded in the IUT, and the payload shall be sent back using the same packet type. The return packet shall be sent back in either the Peripheral-to-Central transmission slot directly following the transmission of the Lower Tester, or it is delayed and sent back in the Peripheral-to-Central transmission slot after the next transmission of the Lower Tester (see Figure 1.7 to Figure 1.9).

There is no signaling to determine or control the mode. The device behavior shall be fixed or adjusted by other means, and shall not change randomly.

The Lower Tester can select whether whitening is on or off. This setting holds for both uplink and downlink. For switching the whitening status, the same rules as in Section 1.1.2 (Figure 1.4) shall apply.

The following rules apply (for illustration see Figure 1.6):

- If the synch word was not detected, the IUT shall not reply.
- If the header error check (HEC) fails, the IUT shall either reply with a NULL packet with the ARQN bit set to NAK or send nothing.
- If the packet contains an LMP message relating to the control of the test mode this command shall be executed and the packet shall not be returned, though ACK or NAK shall be returned as per the usual procedure. Other LMP commands shall be ignored and no packet returned.
- The payload FEC is decoded and the payload shall be encoded again for transmission. This allows testing of the FEC handling. If the pure bit error rate is to be determined, then the Lower Tester shall choose a packet type without FEC.
- The CRC shall be evaluated. In the case of a failure, ARQN=NAK shall be returned. The payload shall be returned as received.

  A new CRC for the return packet shall be calculated for the returned payload regardless of whether the CRC was valid or not.
- If the CRC fails for a packet with a CRC and a payload header, the number of bytes as indicated in the (possibly erroneous) payload header shall be looped back.

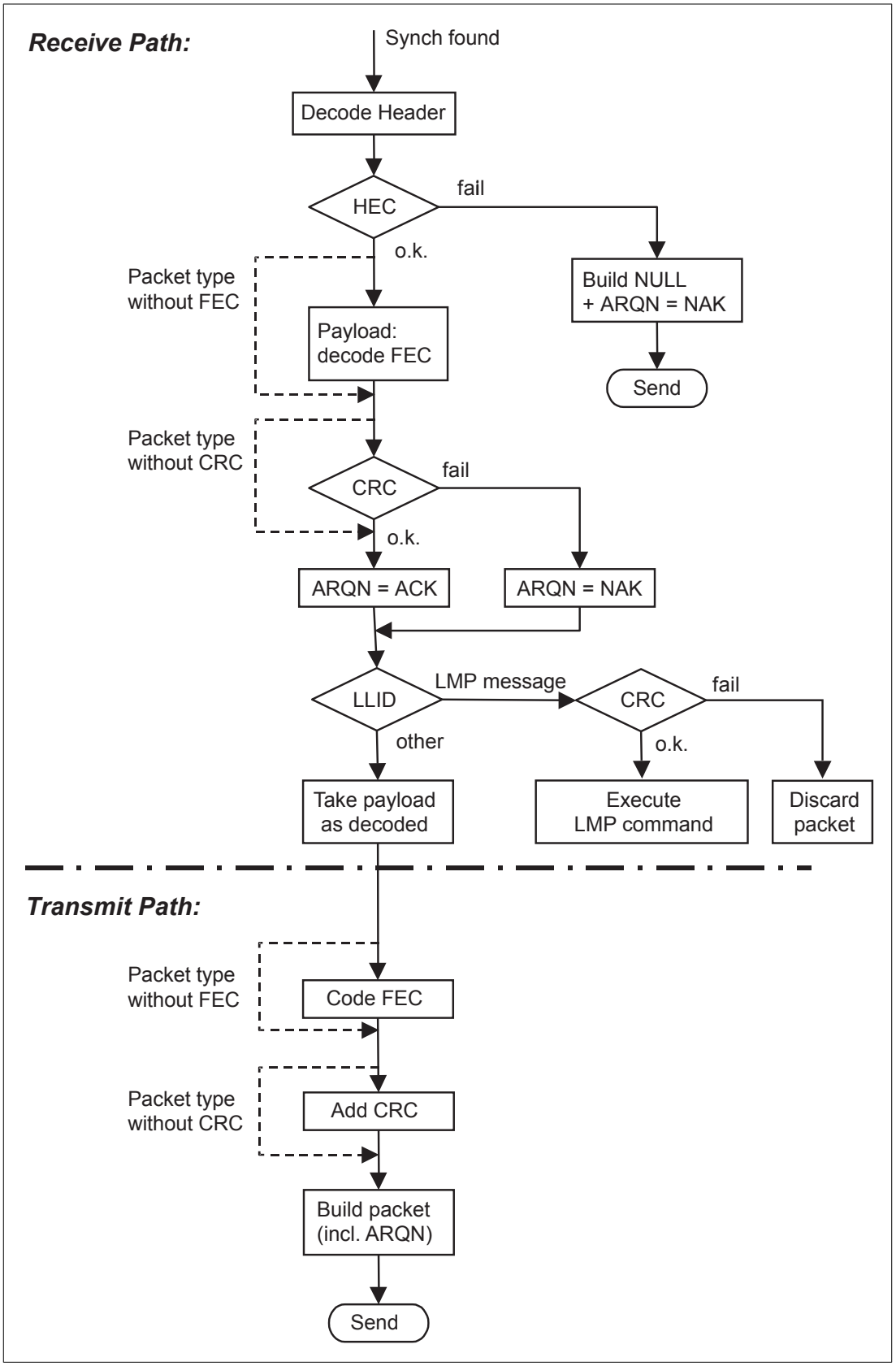



*Figure 1.6: IUT packet handling in loopback test*

The timing for normal and delayed loopback is illustrated in Figure 1.7 to Figure 1.9:

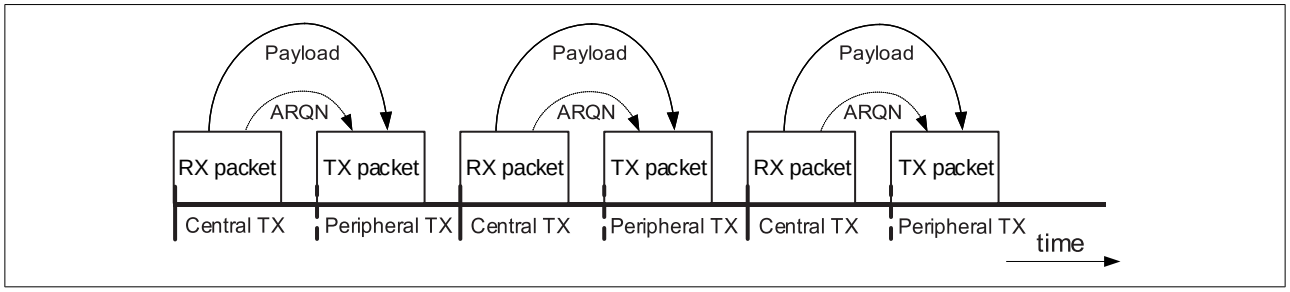



*Figure 1.7: Payload and ARQN handling in normal loopback*

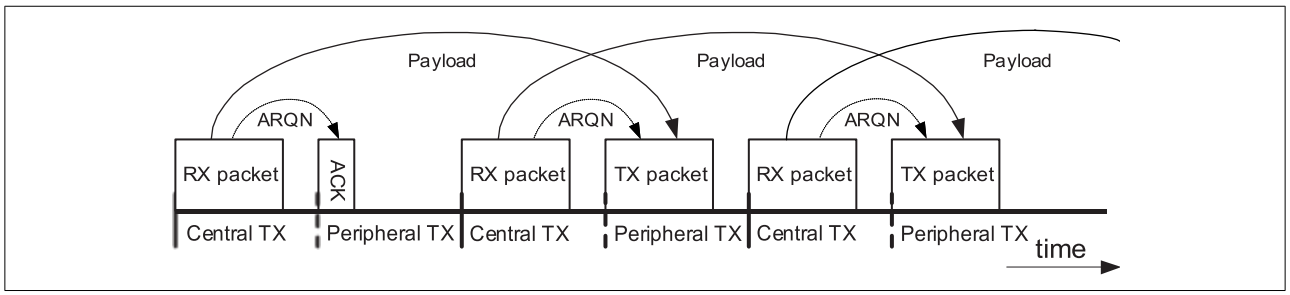



*Figure 1.8: Payload and ARQN handling in delayed loopback – start*

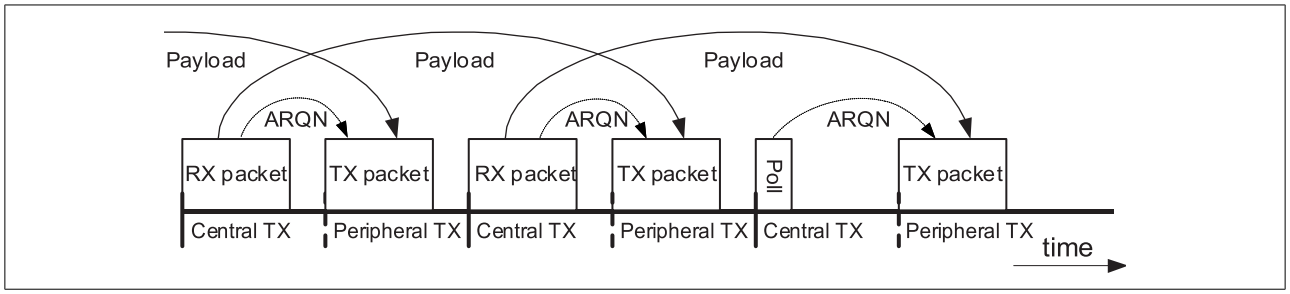



*Figure 1.9: Payload and ARQN handling in delayed loopback – end*

The whitening is performed in the same way as it is used in normal Active mode.

The following parameters can be set to configure the loop back test:

1. Packet Class[1] ACL packets
   SCO packets
   eSCO packets
   ACL packets without whitening
   SCO packets without whitening
   eSCO packets without whitening

[1]This is included because the packet type numbering is ambiguous.

2. Frequency Selection

   Single frequency (independent for RX and TX)

   Normal hopping

3. Power level: (To be used according radio specification requirements)

   power control or fixed TX power

The switch of the frequency setting is done exactly as for the transmitter test (see Section 1.1.2.5).

### 1.1.4 Pause test

Pause test is used by the Lower Tester to put the implementation under test into Pause Test mode from either the loopback or transmitter test modes.

When an LMP_TEST_CONTROL PDU that specifies Pause Test is received the IUT shall stop the current test and enter Pause Test mode. In the case of a transmitter test this means that no more packets shall be transmitted. While in Pause Test mode the IUT shall respond normally to POLL packets (i.e. responds with a NULL packet). The IUT shall also respond normally to all the LMP packets that are allowed in test mode.

When the test scenario is set to Pause Test all the other fields in the LMP_TEST_CONTROL PDU shall be ignored. There shall be no change in hopping scheme or whitening as a result of a request to pause test.

## 1.2 [This section is no longer used]

## 1.3 References

[1] CCITT Recommendation O.153 (1992), Basic parameters for the measurement of error performance at bit rates below the primary rate.

[2] ITU-T Recommendation O.150 (1996), General requirements for instrumentation for performance measurements on digital transmission equipment.

# 2 [THIS SECTION IS NO LONGER USED]

# [THIS PART IS NO LONGER USED]

Host
Part F

# ATTRIBUTE PROTOCOL (ATT)

*This Part defines the Attribute Protocol; a protocol for discovering, reading, and writing attributes on a peer device*

# CONTENTS

**1 Introduction ..... 1495**
1.1 Scope ..... 1495
1.2 [This section is no longer used] ..... 1495
1.3 Conventions ..... 1495

**2 Protocol overview ..... 1496**

**3 Protocol requirements ..... 1498**
3.1 Introduction ..... 1498
3.2 Basic concepts ..... 1498
3.2.1 Attribute type ..... 1498
3.2.2 Attribute handle ..... 1498
3.2.3 Attribute handle grouping ..... 1499
3.2.4 Attribute value ..... 1499
3.2.5 Attribute permissions ..... 1499
3.2.6 Control-point attributes ..... 1501
3.2.7 Protocol methods ..... 1501
3.2.8 Exchanging MTU size ..... 1502
3.2.9 Long attribute values ..... 1502
3.2.10 Atomic operations ..... 1502
3.2.11 ATT bearers ..... 1503
3.3 Attribute PDU ..... 1504
3.3.1 Attribute PDU format ..... 1505
3.3.2 Sequential protocol ..... 1505
3.3.3 Transaction ..... 1506
3.4 Attribute Protocol PDUs ..... 1507
3.4.1 Error handling ..... 1507
3.4.1.1 ATT_ERROR_RSP ..... 1507
3.4.2 MTU exchange ..... 1509
3.4.2.1 ATT_EXCHANGE_MTU_REQ ..... 1509
3.4.2.2 ATT_EXCHANGE_MTU_RSP ..... 1510
3.4.3 Find information ..... 1511
3.4.3.1 ATT_FIND_INFORMATION_REQ ..... 1511
3.4.3.2 ATT_FIND_INFORMATION_RSP ..... 1512
3.4.3.3 ATT_FIND_BY_TYPE_VALUE_REQ ..... 1513
3.4.3.4 ATT_FIND_BY_TYPE_VALUE_RSP ..... 1514
3.4.4 Reading attributes ..... 1515
3.4.4.1 ATT_READ_BY_TYPE_REQ ..... 1515
3.4.4.2 ATT_READ_BY_TYPE_RSP ..... 1517
3.4.4.3 ATT_READ_REQ ..... 1518

3.4.4.4 ATT_READ_RSP ..... 1519
3.4.4.5 ATT_READ_BLOB_REQ ..... 1519
3.4.4.6 ATT_READ_BLOB_RSP ..... 1521
3.4.4.7 ATT_READ_MULTIPLE_REQ ..... 1521
3.4.4.8 ATT_READ_MULTIPLE_RSP ..... 1522
3.4.4.9 ATT_READ_BY_GROUP_TYPE_REQ ..... 1523
3.4.4.10 ATT_READ_BY_GROUP_TYPE_RSP ..... 1525
3.4.4.11 ATT_READ_MULTIPLE_VARIABLE_REQ 1526
3.4.4.12 ATT_READ_MULTIPLE_VARIABLE_RSP . 1527
3.4.5 Writing attributes ..... 1527
3.4.5.1 ATT_WRITE_REQ ..... 1527
3.4.5.2 ATT_WRITE_RSP ..... 1529
3.4.5.3 ATT_WRITE_CMD ..... 1529
3.4.5.4 [This section is no longer used] ..... 1530
3.4.6 Queued writes ..... 1530
3.4.6.1 ATT_PREPARE_WRITE_REQ ..... 1530
3.4.6.2 ATT_PREPARE_WRITE_RSP ..... 1532
3.4.6.3 ATT_EXECUTE_WRITE_REQ ..... 1533
3.4.6.4 ATT_EXECUTE_WRITE_RSP ..... 1534
3.4.7 Server initiated ..... 1534
3.4.7.1 ATT_HANDLE_VALUE_NTF ..... 1534
3.4.7.2 ATT_HANDLE_VALUE_IND ..... 1535
3.4.7.3 ATT_HANDLE_VALUE_CFM ..... 1536
3.4.7.4 ATT_MULTIPLE_HANDLE_VALUE_NTF .. 1536
3.4.8 Attribute Opcode summary ..... 1536
3.4.9 Attribute PDU response summary ..... 1538

**4 Security considerations ..... 1544**

**5 References ..... 1546**

**Appendix A Changes to PDU names ..... 1547**

# 1 INTRODUCTION

## 1.1 Scope

The Attribute Protocol allows a device referred to as the server to expose a set of attributes and their associated values to a peer device referred to as the client. These attributes exposed by the server can be discovered, read, and written by a client, and can be indicated and notified by the server.

## 1.2 [This section is no longer used]

## 1.3 Conventions

In this Part PDU names appear in italics. Error codes defined in Table 3.4 (see Section 3.4.1.1) appear in italics followed by the numeric code (e.g. *Attribute Not Found* (0x0A)). Other names such as parameters appear in Roman text.

# 2 PROTOCOL OVERVIEW

The Attribute Protocol defines two roles; a server role and a client role. It allows a server to expose a set of attributes to a client that are accessible using the Attribute Protocol.

An attribute is a discrete value that has the following three properties associated with it:

a. attribute type, defined by a UUID

b. attribute handle

c. a set of permissions that are defined by each higher layer specification that uses the attribute; these permissions cannot be accessed using the Attribute Protocol.

The attribute type specifies what the attribute represents. Bluetooth SIG defined attribute types are defined in Assigned Numbers and used by an associated higher layer specification. Non-Bluetooth SIG attribute types may also be defined.

The attribute handle uniquely identifies an attribute on a server, allowing a client to reference the attribute in read or write requests; see Section 3.4.4, Section 3.4.5, and Section 3.4.6. It allows a client to uniquely identify the attribute being notified or indicated, see Section 3.4.7. Clients are able to discover the handles of the server's attributes; see Section 3.4.3. Permissions may be applied to an attribute to prevent applications from obtaining or altering an attribute's value. An attribute may be defined by a higher layer specification to be readable or writable or both, and may have additional security requirements. For more information, see Section 3.2.5.

A client may send Attribute Protocol requests to a server, and the server shall respond to all requests that it receives. A device can implement both client and server roles, and both roles can function concurrently in the same device and between the same devices. There shall be only one instance of a server on each Bluetooth device; this implies that the attribute handles shall be identical for all supported bearers. For a given client, the server shall have one set of attributes, which shall have the same value and properties irrespective of which bearer is used. The server can support multiple clients. The attribute values can be the same or different for each client as defined by GATT or a higher layer specification.

Multiple services may be exposed on a single server by allocating separate ranges of handles for each service. The discovery of these handle ranges is defined by a higher layer specification.

The Attribute Protocol has notification and indication capabilities that provide an efficient way of sending attribute values to a client without the need for them to be read; see Section 3.3.

All Attribute Protocol requests are sent over an ATT bearer. There can be multiple ATT bearers established between two devices. Each ATT bearer uses a separate L2CAP channel and can have a different configuration.

In LE, there is a single ATT bearer that uses a fixed channel that is available as soon as the ACL connection is established. Additional ATT bearers can be established using L2CAP (see Section 3.2.11).

In BR/EDR, one or more ATT bearers can be established using L2CAP (see Section 3.2.11).

# 3 PROTOCOL REQUIREMENTS

## 3.1 Introduction

Each attribute has an attribute type that identifies, by means of a UUID (Universally Unique IDentifier), what the attribute represents so that a client can understand the attributes exposed by a server. Each attribute has an attribute handle that is used for accessing the attribute on a server, as well as an attribute value.

An attribute value is accessed using its attribute handle. The attribute handles are discovered by a client using Attribute Protocol PDUs (Protocol Data Unit). Attributes that have the same attribute type may exist more than once in a server. Attributes also have a set of permissions that controls whether they can be read or written, or whether the attribute value shall be sent over an encrypted link. Security aspects of the Attribute Protocol are defined in Section 4.

## 3.2 Basic concepts

### 3.2.1 Attribute type

A universally unique identifier (UUID) is used to identify every attribute type. A UUID is considered unique over all space and time. A UUID can be independently created by anybody and distributed or published as required. There is no central registry for UUIDs, as they are based off a unique identifier that is not duplicated. The Attribute Protocol allows devices to identify attribute types using UUIDs regardless of the local handle used to identify them in a read or write request.

Universal unique identifiers are defined in SDP [Vol 3] Part B, Section 2.5.1.

All 32-bit Attribute UUIDs shall be converted to 128-bit UUIDs when the Attribute UUID is contained in an ATT PDU.

### 3.2.2 Attribute handle

An attribute handle is a 16-bit value that is assigned by each server to its own attributes to allow a client to reference those attributes. An attribute handle shall not be reused while an ATT bearer exists between a client and its server.

Attribute handles on any given server shall have unique, non-zero values. Attributes are ordered by attribute handle.

An attribute handle of value 0x0000 is reserved for future use. An attribute handle of value 0xFFFF is known as the maximum attribute handle.

Note: Attributes can be added or removed while an ATT bearer is established; however, an attribute that has been removed cannot be replaced by another attribute with the same handle while any ATT bearer is established.

### 3.2.3 Attribute handle grouping

Grouping is defined by a specific attribute placed at the beginning of a range of other attributes that are grouped with that attribute, as defined by a higher layer specification. Clients can request the first and last handles associated with a group of attributes.

### 3.2.4 Attribute value

An attribute value is an octet array that may be either fixed or variable length. For example, it can be a one octet value, or a four octet integer, or a variable length string. An attribute may contain a value that is too large to transmit in a single PDU and can be sent using multiple PDUs. The values that are transmitted are opaque to the Attribute Protocol. The encoding of these octet arrays is defined by the attribute type.

When transmitting attribute values in a request, a response, a notification or an indication, the attribute value length is not sent in any field of the majority of PDUs. The length of a variable length field in those PDUs is implicitly given by the length of the packet that carries this PDU. This implies that:

- Only one attribute value can be placed in a single request, response, notification or indication unless the attribute values have lengths known by both the server and client, as defined by the attribute type.
- This attribute value will always be the only variable length field of a request, response, notification or indication.
- The bearer protocol (e.g. L2CAP) preserves datagram boundaries.

Note: Some responses include multiple attribute values, for example when client requests multiple attribute reads. For the client to determine the attribute value boundaries, the attribute values must have a fixed size defined by the attribute type.

There are some PDUs where the length of attribute values is included as a field within the PDU and therefore the above implications do not apply to these PDUs.

### 3.2.5 Attribute permissions

An attribute has a set of permission values associated with it. The permissions associated with an attribute specifies that it may be read and/or written. The permissions associated with the attribute specifies the security level required for read and/or write access, as well as notification and/or indication. The permissions of a given attribute are defined by a higher layer specification, and are not discoverable using the Attribute Protocol.

If access to a secure attribute requires an authenticated link, and the client is not already authenticated with the server with sufficient security, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). When a client receives this error code it may try to authenticate the link, and if the authentication is successful, it can then access the secure attribute.

If access to a secure attribute requires an encrypted link, and the link is not encrypted, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). When a client receives this error code it may try to encrypt the link and if the encryption is successful, it can then access the secure attribute.

If access to a secure attribute requires an encrypted link, and the link is encrypted but with an encryption key size that is too short for the level of security required, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). When a client receives this error code it may try to encrypt the link with a longer key size, and if the encryption is successful, it can then access the secure attribute.

Attribute permissions are a combination of access permissions, encryption permissions, authentication permissions and authorization permissions.

The following access permissions are possible:

- Readable
- Writable
- Readable and writable

The following encryption permissions are possible:

- Encryption required
- No encryption required

The following authentication permissions are possible:

- Authentication required
- No authentication required

The following authorization permissions are possible:

- Authorization required
- No authorization required

Encryption, authentication, and authorization permissions can have different possibilities; for example, a specific attribute could require a particular kind of

authentication or a certain minimum encryption key length. An attribute can have several combinations of permissions that apply; for example, a specific attribute could allow any of the following:

- Read if encrypted (authentication not required)
- Write if authenticated and encrypted
- Read or write if authenticated and authorized (irrespective of encryption)

Access permissions are used by a server to determine if a client can read and/or write an attribute value.

Encryption permissions are used by a server to determine if an encrypted physical link is required when a client attempts to access an attribute or if the server is going to send a notification or indication to a client.

Authentication permissions are used by a server to determine if an authenticated physical link is required when a client attempts to access an attribute. Authentication permissions are also used by a server to determine if an authenticated physical link is required before sending a notification or indication to a client.

Authorization permissions determine if a client needs to be authorized before accessing an attribute value.

Different bearers for the same client may be on links with different security properties. Therefore, the server must not assume that when a client has been authenticated on the link carrying one bearer, it has been authenticated on all bearers. The different security properties might have other implications that an implementation needs to take into account.

### 3.2.6 Control-point attributes

Attributes that cannot be read, but can only be written, notified or indicated are called control-point attributes. These control-point attributes can be used by higher layers to enable device specific procedures, for example the writing of a command or the indication when a given procedure on a device has completed.

### 3.2.7 Protocol methods

The Attribute Protocol uses methods defined in Section 3.4 to find, read, write, notify, and indicate attributes. A method is categorized as either a command, a request, a response, a notification, an indication, or a confirmation; see Section 3.3. The number identifying the method is known as the opcode.

### 3.2.8 Exchanging MTU size

ATT_MTU is defined as the maximum size of any packet sent between a client and a server. A higher layer specification defines the default ATT_MTU value.

When using an L2CAP channel with a fixed CID, the client and server may exchange the maximum size of a packet that can be received using the ATT_EXCHANGE_MTU_REQ and ATT_EXCHANGE_MTU_RSP PDUs. Both devices then use the minimum of these exchanged values for all further communication (see Section 3.4.2). A device that is acting as a server and client at the same time shall use the same value for Client Rx MTU and Server Rx MTU.

When using an L2CAP channel with a dynamically allocated CID, the ATT_MTU shall be set to the L2CAP MTU size.

The ATT_MTU value is a per ATT bearer value. A device with multiple ATT bearers may have a different ATT_MTU value for each ATT bearer.

### 3.2.9 Long attribute values

The longest attribute that can be sent in a single packet is (ATT_MTU-1) octets in size. At a minimum, the Attribute Opcode is included in an Attribute PDU.

An attribute value may be defined to be larger than (ATT_MTU-1) octets in size. These attributes are called long attributes.

To read the entire value of an attributes larger than (ATT_MTU-1) octets, the ATT_READ_BLOB_REQ PDU is used. It is possible to read the first (ATT_MTU-1) octets of a long attribute value using the ATT_READ_REQ PDU.

To write the entire value of an attribute larger than (ATT_MTU-3) octets, the ATT_PREPARE_WRITE_REQ and ATT_EXECUTE_WRITE_REQ PDUs are used. It is possible to write the first (ATT_MTU-3) octets of a long attribute value using the ATT_WRITE_CMD PDU.

It is not possible to determine if an attribute value is longer than (ATT_MTU-3) octets using this protocol. A higher layer specification will state that a given attribute can have a maximum length larger than (ATT_MTU-3) octets.

The maximum length of an attribute value shall be 512 octets.

Note: The protection of an attribute value changing when reading the value using multiple Attribute Protocol PDUs is the responsibility of the higher layer.

### 3.2.10 Atomic operations

The server shall treat each request or command as an atomic operation that cannot be affected by another ATT bearer sending a request or command at the same time. If an

ATT bearer is terminated for any reason (user action or loss of the radio link), the value of any modified attribute is the responsibility of the higher layer specification.

Long attributes cannot be read or written in a single atomic operation.

### 3.2.11 ATT bearers

An ATT bearer is a channel used to send Attribute Protocol PDUs. Each ATT bearer uses an L2CAP channel which shall be either a dynamically allocated channel or the LE Attribute Protocol fixed channel (see [Vol 3] Part A, Section 2.1). A device may have any number of dynamically allocated channels and at most one fixed channel as ATT bearers to a peer device.

An ATT bearer connects an ATT Client on one device to an ATT Server on the peer device and may also connect an ATT Server on the first device to an ATT Client on the peer device. Whether a received Attribute PDU is intended for the ATT Client or for the ATT Server is determined by the PDU type (see Section 3.3).

The L2CAP channel mode determines the behavior of Attribute Protocol on that ATT bearer. If the L2CAP channel is using Enhanced Credit Based Flow Control mode or (on BR/EDR) Enhanced Retransmission mode, then the ATT bearer is known as an Enhanced ATT bearer. Any ATT bearer that is not an Enhanced ATT bearer, using any other L2CAP channel mode, is known as an Unenhanced ATT bearer. Except where explicitly stated, the behavior of an Enhanced ATT bearer shall be identical to the behavior of an Unenhanced ATT bearer.

An ATT bearer is terminated when either the L2CAP channel (if dynamically allocated) or the underlying physical link is disconnected.

An LE fixed channel can only be terminated by disconnecting the physical link.

A higher-layer specification may require an Enhanced ATT bearer.

A device that supports L2CAP over multiple logical transports may, subject to any other requirements in this specification, support ATT bearers on some or all of those transports.

## 3.3 Attribute PDU

Attribute PDUs have one of six types, which are indicated by the suffix to the PDU name as shown in Table 3.1:

| Type | Purpose | Suffix |
|---|---|---|
| Commands | PDUs sent to a server by a client that do not invoke a response. | CMD |
| Requests | PDUs sent to a server by a client that invoke a response. | REQ |
| Responses | PDUs sent to a client by a server in response to a request. | RSP |
| Notifications | Unsolicited PDUs sent to a client by a server that do not invoke a confirmation. | NTF |
| Indications | Unsolicited PDUs sent to a client by a server that invoke a confirmation. | IND |
| Confirmations | PDUs sent to a server by a client to confirm receipt of an indication. | CFM |

*Table 3.1: Attribute PDUs*

A server shall be able to receive and properly respond to the following request PDUs:

- ATT_FIND_INFORMATION_REQ
- ATT_READ_REQ

Support for all other PDU types in a server can be specified in a higher layer specification, see Section 3.4.8.

If a client sends a request, then the client shall support all possible response PDUs for that request.

If a server receives a request that it does not support, then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Request Not Supported* (0x06), with the Attribute Handle In Error set to 0x0000.

If a server receives a command that it does not support, indicated by the Command Flag of the PDU set to one, then the server shall ignore the command.

If the server receives an invalid request – for example, the PDU is the wrong length – then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid PDU* (0x04), with the Attribute Handle In Error set to 0x0000.

If a server does not have sufficient resources to process a request, then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Resources* (0x11), with the Attribute Handle In Error set to 0x0000.

If a server cannot process a request because an error was encountered during the processing of this request, then the server shall respond with the ATT_ERROR_RSP

PDU with the Error Code parameter set to *Unlikely Error* (0x0E), with the Attribute Handle In Error set to 0x0000.

### 3.3.1 Attribute PDU format

Attribute PDUs have the following format:

| Name | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | The attribute PDU operation code<br>bit 7: *previously used*<br>bit 6: Command Flag<br>bits 5-0: Method |
| Attribute Parameters | 0 to (ATT_MTU - 1) | The attribute PDU parameters |

*Table 3.2: Format of attribute PDU*

Multi-octet fields within the Attribute Protocol shall be sent least significant octet first (little-endian) with the exception of the Attribute Value field. The endian-ness of the Attribute Value field is defined by a higher layer specification.

The Attribute Opcode is composed of two fields, the Command Flag and the Method. The Method is a 6-bit value that determines the format and meaning of the Attribute Parameters.

If the Command Flag of the Attribute Opcode is set to one, the PDU shall be considered to be a command.

### 3.3.2 Sequential protocol

Many Attribute Protocol PDUs use a sequential request-response protocol.

Once a client sends a request to a server, that client shall send no other request to the same server on the same ATT bearer until a response PDU has been received.

Indications sent from a server also use a sequential indication-confirmation protocol. No other indications shall be sent to the same client from this server on the same ATT bearer until a confirmation PDU has been received. The client, however, is free to send commands and requests prior to sending a confirmation.

For notifications, which do not have a response PDU, there is no flow control and a notification can be sent at any time.

For commands, which do not have a response PDU, there is no flow control and a command can be sent at any time.

Note: A server can be flooded with commands, and a higher layer specification can define how to prevent this from occurring.

Commands that are received but cannot be processed, due to buffer overflows or a change-unaware client (see [Vol 3] Part G, Section 2.5.2.1), shall be discarded. Therefore, those PDUs must be considered to be unreliable.

On an Unenhanced ATT bearer, notifications that are received but cannot be processed due to buffer overflows shall be discarded. Therefore, those PDUs must be considered to be unreliable.

On an Enhanced ATT bearer, notifications shall always be processed when received.

Note: Flow control for each client and a server is independent.

Note: It is possible for a server to receive a request, send one or more notifications, and then the response to the original request. The flow control of requests is not affected by the transmission of the notifications.

Note: It is possible for a server to receive a request and then a command before responding to the original request. The flow control of requests is not affected by the transmission of commands.

Note: It is possible for a notification from a server to be sent after an indication has been sent but the confirmation has not been received. The flow control of indications is not affected by the transmission of notifications.

Note: It is possible for a client to receive an indication from a server and then send a request or command to that server before sending the confirmation of the original indication.

### 3.3.3 Transaction

An Attribute Protocol request and response or indication-confirmation pair is considered a single transaction. A transaction shall always be performed on one ATT bearer, and shall not be split over multiple ATT bearers.

On the client, a transaction shall start when the request is sent by the client. A transaction shall complete when the response is received by the client.

On a server, a transaction shall start when a request is received by the server. A transaction shall complete when the response is sent by the server.

On a server, a transaction shall start when an indication is sent by the server. A transaction shall complete when the confirmation is received by the server.

On a client, a transaction shall start when an indication is received by the client. A transaction shall complete when the confirmation is sent by the client.

A transaction not completed within 30 seconds shall time out. Such a transaction shall be considered to have failed and the local higher layers shall be informed of this failure. No more Attribute Protocol requests, commands, indications or notifications shall be sent to the target device on this ATT bearer.

To send another Attribute Protocol PDU, a new ATT bearer must be established between these devices. The existing ATT bearer may need to be terminated before the new ATT bearer is established.

If the ATT bearer is terminated during a transaction, then the transaction shall be considered to be closed, and any values that were being modified on the server will be in an undetermined state.

Note: Each ATT_PREPARE_WRITE_REQ and each ATT_READ_BLOB_REQ PDU starts a separate request and therefore a separate transaction.

## 3.4 Attribute Protocol PDUs

### 3.4.1 Error handling

#### 3.4.1.1 ATT_ERROR_RSP

The ATT_ERROR_RSP PDU is used to state that a given request cannot be performed, and to provide the reason.

Note: Commands do not generate this response.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x01 = ATT_ERROR_RSP PDU |
| Request Opcode In Error | 1 | The request that generated this ATT_ERROR_RSP PDU |
| Attribute Handle In Error | 2 | The attribute handle that generated this ATT_ERROR_RSP PDU |
| Error Code | 1 | The reason why the request has generated an ATT_ERROR_RSP PDU |

*Table 3.3: Format of the ATT_ERROR_RSP PDU*

The Request Opcode In Error parameter shall be set to the Attribute Opcode of the request that generated this error.

The Attribute Handle In Error parameter shall be set to the attribute handle in the original request that generated this error. If there was no attribute handle in the original

request or if the request is not supported, then the value 0x0000 shall be used for this field.

The Error Code parameter shall be set to one of the following values:

| Name | Error Code | Description |
|---|---|---|
| Invalid Handle | 0x01 | The attribute handle given was not valid on this server. |
| Read Not Permitted | 0x02 | The attribute cannot be read. |
| Write Not Permitted | 0x03 | The attribute cannot be written. |
| Invalid PDU | 0x04 | The attribute PDU was invalid. |
| Insufficient Authentication | 0x05 | The attribute requires authentication before it can be read or written. |
| Request Not Supported | 0x06 | ATT Server does not support the request received from the client. |
| Invalid Offset | 0x07 | Offset specified was past the end of the attribute. |
| Insufficient Authorization | 0x08 | The attribute requires authorization before it can be read or written. |
| Prepare Queue Full | 0x09 | Too many prepare writes have been queued. |
| Attribute Not Found | 0x0A | No attribute found within the given attribute handle range. |
| Attribute Not Long | 0x0B | The attribute cannot be read using the ATT_READ_BLOB_REQ PDU. |
| Encryption Key Size Too Short[1] | 0x0C | The Encryption Key Size used for encrypting this link is too short. |
| Invalid Attribute Value Length | 0x0D | The attribute value length is invalid for the operation. |
| Unlikely Error | 0x0E | The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested. |
| Insufficient Encryption | 0x0F | The attribute requires encryption before it can be read or written. |
| Unsupported Group Type | 0x10 | The attribute type is not a supported grouping attribute as defined by a higher layer specification. |
| Insufficient Resources | 0x11 | Insufficient Resources to complete the request. |
| Database Out Of Sync | 0x12 | The server requests the client to rediscover the database. |
| Value Not Allowed | 0x13 | The attribute parameter value was not allowed. |
| Application Error | 0x80 – 0x9F | Application error code defined by a higher layer specification. |

| Name | Error Code | Description |
|---|---|---|
| Common Profile and Service Error Codes | 0xE0 – 0xFF | Common profile and service error codes defined in [1] |
| Reserved for future use | All other values | Reserved for future use. |

*Table 3.4: Error codes*

[1]This was previously "Insufficient Encryption Key Size".

The Error Code values listed in Section 3.4.2 to Section 3.4.8 are not necessarily the only ones permitted in response to those PDUs. See Section 3.4.9 for the definitive list of which Error Code values are permitted.

If more than one error code applies, then it is vendor-specific which error code is transmitted in the ATT_ERROR_RSP PDU.

If an error code is received in the ATT_ERROR_RSP PDU that is not understood by the client, for example an error code that was reserved for future use that is now being used in a future version of the specification, then the ATT_ERROR_RSP PDU shall still be considered to state that the given request cannot be performed for an unknown reason.

Sending an ATT_ERROR_RSP PDU should not cause the ATT Server to disconnect from the client. The client may upgrade the security and retry the request, so the server should give the client sufficient time to perform such an upgrade.

## 3.4.2 MTU exchange

### 3.4.2.1 ATT_EXCHANGE_MTU_REQ

The ATT_EXCHANGE_MTU_REQ PDU is used by the client to inform the server of the client's maximum receive MTU size and request the server to respond with its maximum receive MTU size.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x02 = ATT_EXCHANGE_MTU_REQ |
| Client Rx MTU | 2 | Client receive MTU size |

*Table 3.5: Format of ATT_EXCHANGE_MTU_REQ PDU*

The Client Rx MTU shall be greater than or equal to the default ATT_MTU.

This request shall only be sent once during a connection by the client. The Client Rx MTU parameter shall be set to the maximum size of the Attribute Protocol PDU that the client can receive.

#### 3.4.2.2 ATT_EXCHANGE_MTU_RSP

The ATT_EXCHANGE_MTU_RSP PDU is sent in reply to a received ATT_EXCHANGE_MTU_REQ PDU.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x03 = ATT_EXCHANGE_MTU_RSP |
| Server Rx MTU | 2 | ATT Server receive MTU size |

*Table 3.6: Format of ATT_EXCHANGE_MTU_RSP PDU*

The Server Rx MTU shall be greater than or equal to the default ATT_MTU.

The Server Rx MTU parameter shall be set to the maximum size of the Attribute Protocol PDU that the server can receive.

The server and client shall set ATT_MTU to the minimum of the Client Rx MTU and the Server Rx MTU. The size is the same to ensure that a client can correctly detect the final packet of a long attribute read.

This ATT_MTU value shall be applied in the server after this response has been sent and before any other Attribute Protocol PDU is sent.

This ATT_MTU value shall be applied in the client after this response has been received and before any other Attribute Protocol PDU is sent.

If either Client Rx MTU or Service Rx MTU are incorrectly less than the default ATT_MTU, then the ATT_MTU shall not be changed and the ATT_MTU shall be the default ATT_MTU.

If a device is both a client and a server, the following rules shall apply:

1. A device's ATT_EXCHANGE_MTU_REQ PDU shall contain the same MTU as the device's ATT_EXCHANGE_MTU_RSP PDU (i.e. the MTU shall be symmetric).
2. If MTU is exchanged in one direction, that is sufficient for both directions.
3. It is permitted, (but not necessary - see 2.) to exchange MTU in both directions, but the MTUs shall be the same in each direction (see 1.)
4. If an Attribute Protocol Request is received after the MTU Exchange Request is sent and before the MTU Exchange Response is received, the associated Attribute Protocol Response shall use the default MTU. Figure 3.1 shows an example that is covered by this rule. In this case device A and device B both use the default MTU for the Attribute Protocol Response.

5. Once the MTU Exchange Request has been sent, the initiating device shall not send an Attribute Protocol Indication or Notification until after the MTU Exchange Response has been received.

   Note: This stops the risk of a cross-over condition where the MTU size is unknown for the Indication or Notification.

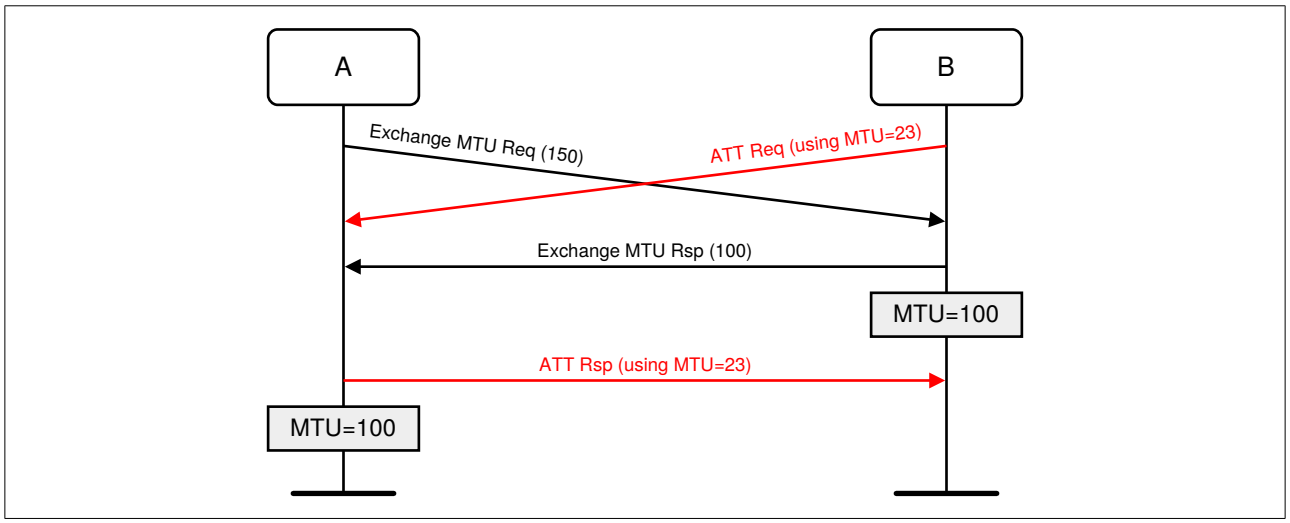



*Figure 3.1: MTU Request and Response exchange*

## 3.4.3 Find information

### 3.4.3.1 ATT_FIND_INFORMATION_REQ

The ATT_FIND_INFORMATION_REQ PDU is used to obtain the mapping of attribute handles with their associated types. This allows a client to discover the list of attributes and their types on a server.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x04 = ATT_FIND_INFORMATION_REQ |
| Starting Handle | 2 | First requested handle number |
| Ending Handle | 2 | Last requested handle number |

*Table 3.7: Format of ATT_FIND_INFORMATION_REQ PDU*

Only attributes with attribute handles between the Starting Handle parameter and the Ending Handle parameter will be returned. To read all attributes, the Starting Handle parameter shall be set to 0x0001, and the Ending Handle parameter shall be set to 0xFFFF. The Starting Handle parameter shall be less than or equal to the Ending Handle parameter.

If one or more attributes will be returned, an ATT_FIND_INFORMATION_RSP PDU shall be sent.

If a server receives an ATT_FIND_INFORMATION_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01); the Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attributes will be returned (e.g., because there are no attributes in the range), an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A); the Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

The server shall not respond to the ATT_FIND_INFORMATION_REQ PDU with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05), *Insufficient Authorization* (0x08), *Encryption Key Size Too Short* (0x0C), *Database Out of Sync* (0x12), *Application Error* (0x80 to 0x9F), or *Common Profile and Service Error Codes* (0xE0 to 0xFF).

#### 3.4.3.2 ATT_FIND_INFORMATION_RSP

The ATT_FIND_INFORMATION_RSP PDU is sent in reply to a received ATT_FIND_INFORMATION_REQ PDU and contains information about attributes on this server.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x05 = ATT_FIND_INFORMATION_RSP |
| Format | 1 | The format of the information data. |
| Information Data | 4 to (ATT_MTU-2) | The information data whose format is determined by the Format field |

*Table 3.8: Format of ATT_FIND_INFORMATION_RSP PDU*

The *Find Information Response* shall have complete handle-UUID pairs. Such pairs shall not be split across response packets; this also implies that a handle-UUID pair shall fit into a single response packet. The handle-UUID pairs shall be returned in ascending order of attribute handles without omissions.

The Format parameter can contain one of two possible values.

| Name | Format | Description |
|---|---|---|
| Handle(s) and 16-bit Bluetooth UUID(s) | 0x01 | A list of 1 or more handles with their 16-bit Bluetooth UUIDs |
| Handle(s) and 128-bit UUID(s) | 0x02 | A list of 1 or more handles with their 128-bit UUIDs |

*Table 3.9: Format field values*

The information data field is comprised of a list of data defined in Table 3.10 and Table 3.11 depending on the value chosen for the format.

| Handle | 16-bit Bluetooth UUID |
|---|---|
| 2 octets | 2 octets |

*Table 3.10: Format 0x01 – handle and 16-bit Bluetooth UUIDs*

| Handle | 128-bit UUID |
|---|---|
| 2 octets | 16 octets |

*Table 3.11: Format 0x02 – handle and 128-bit UUIDs*

If sequential attributes have differing UUID sizes, the ATT_FIND_INFORMATION_RSP PDU shall end with the first attribute of the pair even though this may mean that it is not filled with the maximum possible amount of (handle, UUID) pairs. This is because it is not possible to include attributes with differing UUID sizes into a single response packet. In this situation, the client must use another ATT_FIND_INFORMATION_REQ PDU with its starting handle updated to fetch the second attribute of the pair and any further ones in its original request. However, the server may convert a 16-bit UUID to the corresponding 128-bit UUID to allow it to be included in the response packet.

#### 3.4.3.3 ATT_FIND_BY_TYPE_VALUE_REQ

The ATT_FIND_BY_TYPE_VALUE_REQ PDU is used to obtain the handles of attributes that have a 16-bit UUID attribute type and attribute value.This allows the range of handles associated with a given attribute to be discovered when the attribute type determines the grouping of a set of attributes.

Note: GATT defines grouping of attributes by attribute type.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x06 = ATT_FIND_BY_TYPE_VALUE_REQ PDU |
| Starting Handle | 2 | First requested handle number |
| Ending Handle | 2 | Last requested handle number |
| Attribute Type | 2 | 2 octet UUID to find |
| Attribute Value | 0 to (ATT_MTU-7) | Attribute value to find |

*Table 3.12: Format of ATT_FIND_BY_TYPE_VALUE_REQ PDU*

Only attributes with attribute handles between the Starting Handle parameter and the Ending Handle parameter that match the requested attribute type and the attribute value that have sufficient permissions to allow reading will be returned. To read all attributes, the Starting Handle parameter shall be set to 0x0001, and the Ending Handle parameter shall be set to 0xFFFF.

If one or more handles will be returned, an ATT_FIND_BY_TYPE_VALUE_RSP PDU shall be sent.

Note: Attribute values will be compared in terms of length and binary representation.

Note: It is not possible to use this request on an attribute that has a value longer than (ATT_MTU-7).

If a server receives an ATT_FIND_BY_TYPE_VALUE_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attributes will be returned, an ATT_ERROR_RSP PDU shall be sent by the server with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The server shall not respond to the ATT_FIND_BY_TYPE_VALUE_REQ PDU with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05), *Insufficient Authorization* (0x08), *Encryption Key Size Too Short* (0x0C), *Insufficient Encryption* (0x0F), *Database Out of Sync* (0x12), *Application Error* (0x80 to 0x9F), or *Common Profile and Service Error Codes* (0xE0 to 0xFF).

#### 3.4.3.4 ATT_FIND_BY_TYPE_VALUE_RSP

The ATT_FIND_BY_TYPE_VALUE_RSP PDU is sent in reply to a received ATT_FIND_BY_TYPE_VALUE_REQ PDU and contains information about this server.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x07 = ATT_FIND_BY_TYPE_VALUE_RSP PDU |
| Handles Information List | 4 to (ATT_MTU-1) | A list of 1 or more Handle Informations |

*Table 3.13: Format of ATT_FIND_BY_TYPE_VALUE_RSP PDU*

The Handles Information List field is a list of one or more Handle Informations. The Handles Information field is an attribute handle range as defined in Table 3.14.

| Found Attribute Handle | Group End Handle |
|---|---|
| 2 octets | 2 octets |

*Table 3.14: Format of the Handles Information*

The ATT_FIND_BY_TYPE_VALUE_RSP PDU shall contain one or more complete Handles Information. Such Handles Information shall not be split across response

packets. The Handles Information List is ordered sequentially based on the found attribute handles.

For each handle that matches the attribute type and attribute value in the ATT_FIND_BY_TYPE_VALUE_REQ PDU a Handles Information shall be returned. The Found Attribute Handle shall be set to the handle of the attribute that has the exact attribute type and attribute value from the ATT_FIND_BY_TYPE_VALUE_REQ PDU. If the attribute type in the ATT_FIND_BY_TYPE_VALUE_REQ PDU is a grouping attribute as defined by a higher layer specification, the Group End Handle shall be defined by that higher layer specification. If the attribute type in the ATT_FIND_BY_TYPE_VALUE_REQ PDU is not a grouping attribute as defined by a higher layer specification, the Group End Handle shall be equal to the Found Attribute Handle. The Group End Handle may be greater than the Ending Handle in the ATT_FIND_BY_TYPE_VALUE_REQ PDU.

If a server receives an ATT_FIND_BY_TYPE_VALUE_REQ PDU, the server shall respond with the ATT_FIND_BY_TYPE_VALUE_RSP PDU containing as many handles for attributes that match the requested attribute type and attribute value that exist in the server that will fit into the maximum PDU size of (ATT_MTU-1).

### 3.4.4 Reading attributes

#### 3.4.4.1 ATT_READ_BY_TYPE_REQ

The ATT_READ_BY_TYPE_REQ PDU is used to obtain the values of attributes where the attribute type is known but the handle is not known.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x08 = ATT_READ_BY_TYPE_REQ PDU |
| Starting Handle | 2 | First requested handle number |
| Ending Handle | 2 | Last requested handle number |
| Attribute Type | 2 or 16 | 2 or 16 octet UUID |

*Table 3.15: Format of ATT_READ_BY_TYPE_REQ PDU*

Only the attributes with attribute handles between the Starting Handle and the Ending Handle with the attribute type that is the same as the Attribute Type given will be returned. To search through all attributes, the starting handle shall be set to 0x0001 and the ending handle shall be set to 0xFFFF.

Note: All attribute types are effectively compared as 128-bit UUIDs, even if a 16-bit UUID is provided in this request or defined for an attribute. See [Vol 3] Part B, Section 2.5.1.

The starting handle shall be less than or equal to the ending handle. If a server receives an ATT_READ_BY_TYPE_REQ PDU with the Starting Handle parameter

greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attribute with the given type exists within the handle range, then no attribute handle and value will be returned, and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The attributes returned shall be the attributes with the lowest handles within the handle range. These are known as the requested attributes.

If the attributes with the requested type within the handle range have attribute values that have the same length, then these attributes can all be read in a single request. However, if those attributes have different lengths, then multiple ATT_READ_BY_TYPE_REQ PDUs must be issued.

The ATT Server shall include as many attributes as possible in the response in order to minimize the number of PDUs required to read attributes of the same type.

When multiple attributes match, then the rules below shall be applied to each in turn.

- Only attributes that can be read shall be returned in an ATT_READ_BY_TYPE_RSP PDU.
- If an attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then this attribute cannot be included in an ATT_READ_BY_TYPE_RSP PDU and the attributes before this attribute shall be returned.
- If the first attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then no other attributes in the requested attributes can be considered.

The server shall respond with an ATT_READ_BY_TYPE_RSP PDU if the requested attributes have sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the requested attribute's value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

Note: If there are multiple attributes with the requested type within the handle range, and the client would like to get the next attribute with the requested type, it would have to issue another ATT_READ_BY_TYPE_REQ PDU with its starting handle updated. The client can be sure there are no more such attributes remaining once it gets an ATT_ERROR_RSP PDU with the Error Code parameter set to *Attribute Not Found* (0x0A).

#### 3.4.4.2 ATT_READ_BY_TYPE_RSP

The ATT_READ_BY_TYPE_RSP PDU is sent in reply to a received ATT_READ_BY_TYPE_REQ PDU and contains the handles and values of the attributes that have been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x09 = ATT_READ_BY_TYPE_RSP PDU |
| Length | 1 | The size of each attribute handle-value pair |
| Attribute Data List | 2 to (ATT_MTU-2) | A list of Attribute Data |

*Table 3.16: Format of ATT_READ_BY_TYPE_RSP PDU*

The ATT_READ_BY_TYPE_RSP PDU shall contain complete handle-value pairs. Such pairs shall not be split across response packets. The handle-value pairs shall be returned sequentially based on the attribute handle.

The Length parameter shall be set to the size of one attribute handle-value pair.

The maximum length of an attribute handle-value pair is 255 octets, bounded by the Length parameter that is one octet. Therefore, the maximum length of an attribute value returned in this response is (Length – 2) = 253 octets.

The attribute handle-value pairs shall be set to the value of the attributes identified by the attribute type within the handle range within the request. If the attribute value is longer than (ATT_MTU - 4) or 253 octets, whichever is smaller, then the first (ATT_MTU - 4) or 253 octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) can be used to read the remaining octets of a long attribute value.

The Attribute Data field is comprised of a list of attribute handle and value pairs as defined in Table 3.17.

| Attribute Handle | Attribute Value |
|---|---|
| 2 octets | (Length – 2) octets |

*Table 3.17: Format of the Attribute Data*

#### 3.4.4.3 ATT_READ_REQ

The ATT_READ_REQ PDU is used to request the server to read the value of an attribute and return its value in an ATT_READ_RSP PDU.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0A = ATT_READ_REQ PDU |
| Attribute Handle | 2 | The handle of the attribute to be read |

*Table 3.18: Format of ATT_READ_REQ PDU*

The attribute handle parameter shall be set to a valid handle.

The server shall respond with an ATT_READ_RSP PDU if the handle is valid and the attribute has sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

#### 3.4.4.4 ATT_READ_RSP

The ATT_READ_RSP PDU is sent in reply to a received *Read Request* and contains the value of the attribute that has been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0B = ATT_READ_RSP PDU |
| Attribute Value | 0 to (ATT_MTU-1) | The value of the attribute with the handle given |

*Table 3.19: Format of ATT_READ_RSP PDU*

The attribute value shall be set to the value of the attribute identified by the attribute handle in the request. If the attribute value is longer than (ATT_MTU-1), then the first (ATT_MTU-1) octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) can be used to read the remaining octets of a long attribute value.

#### 3.4.4.5 ATT_READ_BLOB_REQ

The ATT_READ_BLOB_REQ PDU is used to request the server to read part of the value of an attribute at a given offset and return a specific part of the value in an ATT_READ_BLOB_RSP PDU.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0C = ATT_READ_BLOB_REQ PDU |
| Attribute Handle | 2 | The handle of the attribute to be read |
| Value Offset | 2 | The offset of the first octet to be read |

*Table 3.20: Format of ATT_READ_BLOB_REQ PDU*

The attribute handle parameter shall be set to a valid handle.

The value offset parameter is based from zero; the first value octet has an offset of zero, the second octet has a value offset of one, etc.

The server shall respond with an ATT_READ_BLOB_RSP PDU if the handle is valid and the attribute and value offset is not greater than the length of the attribute value and has sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

If the value offset of the *Read Blob Request* is greater than the length of the attribute value, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Offset* (0x07).

If the attribute value has a fixed length that is less than or equal to (ATT_MTU - 1) octets in length, then an ATT_ERROR_RSP PDU may be sent with the Error Code parameter set to *Attribute Not Long* (0x0B).

If the value offset of the ATT_READ_BLOB_REQ PDU is equal to the length of the attribute value, then the length of the part attribute value in the response shall be zero.

Note: If the attribute is longer than (ATT_MTU-1) octets, the ATT_READ_BLOB_REQ PDU is the only way to read the additional octets of a long attribute. The first (ATT_MTU-1) octets can be read using an ATT_READ_RSP PDU; the first (ATT_MTU-3) octets can be received in an ATT_HANDLE_VALUE_NTF or an ATT_HANDLE_VALUE_IND PDU.

Note: Some, but not all, long attributes have their length specified by a higher layer specification. If the long attribute has a variable length, the only way to get to the end of it is to read it part by part until the value in the ATT_READ_BLOB_RSP PDU has a length shorter than (ATT_MTU-1) or an ATT_ERROR_RSP PDU is sent with the Error Code parameter set to *Invalid Offset* (0x07).

Note: The value of a Long Attribute can change between the server receiving one ATT_READ_BLOB_REQ PDU and the next ATT_READ_BLOB_REQ PDU. Higher level specifications are expected to be aware of this and define appropriate behavior.

#### 3.4.4.6 ATT_READ_BLOB_RSP

The ATT_READ_BLOB_RSP PDU is sent in reply to a received ATT_READ_BLOB_REQ PDU and contains part of the value of the attribute that has been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0D = ATT_READ_BLOB_RSP |
| Part Attribute Value | 0 to (ATT_MTU-1) | Part of the value of the attribute with the handle given |

*Table 3.21: Format of ATT_READ_BLOB_RSP PDU*

The part attribute value shall be set to part of the value of the attribute identified by the attribute handle and the value offset in the request. If the value offset is equal to the length of the attribute value, then the length of the part attribute value shall be zero. If the attribute value is longer than (Value Offset + ATT_MTU-1) then (ATT_MTU-1) octets from Value Offset shall be included in this response.

#### 3.4.4.7 ATT_READ_MULTIPLE_REQ

The ATT_READ_MULTIPLE_REQ PDU is used to request the server to read two or more values of a set of attributes and return their values in an ATT_READ_MULTIPLE_RSP PDU. Only values that have a known fixed size can be read, with the exception of the last value that can have a variable length. The knowledge of whether attributes have a known fixed size is defined in a higher layer specification.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0E = ATT_READ_MULTIPLE_REQ PDU |
| Set Of Handles | 4 to (ATT_MTU-1) | A set of two or more attribute handles. |

*Table 3.22: Format of ATT_READ_MULTIPLE_REQ PDU*

The attribute handles in the Set Of Handles parameter shall be valid handles.

The server shall respond with an ATT_READ_MULTIPLE_RSP PDU if all the handles are valid and all attributes have sufficient permissions to allow reading.

Note: The attribute values for the attributes in the Set Of Handles parameters do not have to all be the same size.

Note: The attribute handles in the Set Of Handles parameter do not have to be in attribute handle order; they are in the order that the values are required in the response.

If the client has insufficient authorization to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If any of the handles are invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If any of the attribute values cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

If an ATT_ERROR_RSP PDU is sent, the Attribute Handle In Error parameter shall be set to the handle of the first attribute causing the error.

#### 3.4.4.8 ATT_READ_MULTIPLE_RSP

The ATT_READ_MULTIPLE_RSP PDU is sent in reply to a received ATT_READ_MULTIPLE_REQ PDU and contains the values of the attributes that have been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x0F = ATT_READ_MULTIPLE_RSP PDU |
| Set Of Values | 0 to (ATT_MTU-1) | A set of two or more values |

*Table 3.23: Format of ATT_READ_MULTIPLE_RSP PDU*

The Set Of Values parameter shall be a concatenation of attribute values for each of the attribute handles in the request in the order that they were requested. If the Set Of Values parameter is longer than (ATT_MTU-1) then only the first (ATT_MTU-1) octets shall be included in this response.

Note: If a client uses this request for attributes when the Set Of Values parameter could be (ATT_MTU-1) octets long, then it will not be possible to determine if the last attribute value is complete or if it overflowed.

#### 3.4.4.9 ATT_READ_BY_GROUP_TYPE_REQ

The ATT_READ_BY_GROUP_TYPE_REQ PDU is used to obtain the values of attributes where the attribute type is known, the type of a grouping attribute as defined by a higher layer specification, but the handle is not known.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x10 = ATT_READ_BY_GROUP_TYPE_REQ PDU |
| Starting Handle | 2 | First requested handle number |
| Ending Handle | 2 | Last requested handle number |
| Attribute Group Type | 2 or 16 | 2 or 16 octet UUID |

*Table 3.24: Format of ATT_READ_BY_GROUP_TYPE_REQ PDU*

Only the attributes with attribute handles between the Starting Handle and the Ending Handle with the attribute type that is the same as the Attribute Group Type given will be returned. To search through all attributes, the starting handle shall be set to 0x0001 and the ending handle shall be set to 0xFFFF.

Note: All attribute types are effectively compared as 128-bit UUIDs, even if a 16-bit UUID is provided in this request or defined for an attribute. See [Vol 3] Part B, Section 2.5.1.

The starting handle shall be less than or equal to the ending handle. If a server receives an ATT_READ_BY_GROUP_TYPE_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If the Attribute Group Type is not a supported grouping attribute as defined by a higher layer specification then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Unsupported Group Type* (0x10). The Attribute Handle In Error parameter shall be set to the Starting Handle.

If no attribute with the given type exists within the handle range, then no attribute handle and value will be returned, and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The attributes returned shall be the attributes with the lowest handles within the handle range. These are known as the requested attributes.

If the attributes with the requested type within the handle range have attribute values that have the same length, then these attributes can all be read in a single request. However, if those attributes have different lengths, then multiple ATT_READ_BY_GROUP_TYPE_REQ PDUs must be issued.

The ATT Server shall include as many attributes as possible in the response in order to minimize the number of PDUs required to read attributes of the same type.

When multiple attributes match, then the rules below shall be applied to each in turn.

- Only attributes that can be read shall be returned in an ATT_READ_BY_GROUP_TYPE_RSP PDU.
- If an attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then this attribute cannot be included in an ATT_READ_BY_GROUP_TYPE_RSP PDU and the attributes before this attribute shall be returned.
- If the first attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then no other attributes in the requested attributes can be considered.

The server shall respond with an ATT_READ_BY_GROUP_TYPE_RSP PDU if the requested attributes have sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the requested attribute's value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

Note: If there are multiple attributes with the requested type within the handle range, and the client would like to get the next attribute with the requested type, it would have to issue another ATT_READ_BY_GROUP_TYPE_REQ PDU with its starting handle updated. The client can be sure there are no more such attributes remaining once it gets an ATT_ERROR_RSP PDU with the Error Code parameter set to *Attribute Not Found* (0x0A).

The server shall not respond to the ATT_READ_BY_GROUP_TYPE_REQ PDU with an ATT_ERROR_RSP PDU with the error code *Database Out of Sync* (0x12).

#### 3.4.4.10 ATT_READ_BY_GROUP_TYPE_RSP

The ATT_READ_BY_GROUP_TYPE_RSP PDU is sent in reply to a received ATT_READ_BY_GROUP_TYPE_REQ PDU and contains the handles and values of the attributes that have been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x11 = ATT_READ_BY_GROUP_TYPE_RSP PDU |
| Length | 1 | The size of each Attribute Data |
| Attribute Data List | 4 to (ATT_MTU-2) | A list of Attribute Data |

*Table 3.25: Format of ATT_READ_BY_GROUP_TYPE_RSP PDU*

The ATT_READ_BY_GROUP_TYPE_RSP PDU shall contain complete Attribute Data. An Attribute Data shall not be split across response packets. The Attribute Data List is ordered sequentially based on the attribute handles.

The Length parameter shall be set to the size of the one Attribute Data.

The maximum length of an Attribute Data is 255 octets, bounded by the Length parameter that is one octet. Therefore, the maximum length of an attribute value returned in this response is (Length – 4) = 251 octets.

The Attribute Data List shall be set to the value of the attributes identified by the attribute type within the handle range within the request. If the attribute value is longer than (ATT_MTU - 6) or 251 octets, whichever is smaller, then the first (ATT_MTU - 6) or 251 octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) can be used to read the remaining octets of a long attribute value.

The Attribute Data List is comprised of a list of Attribute Data as defined in Table 3.26.

| Attribute Handle | End Group Handle | Attribute Value |
|---|---|---|
| 2 octets | 2 octets | (Length - 4) octets |

*Table 3.26: Format of the Attribute Data*

#### 3.4.4.11 ATT_READ_MULTIPLE_VARIABLE_REQ

The ATT_READ_MULTIPLE_VARIABLE_REQ PDU is used to request that the server read two or more values of a set of attributes that have a variable or unknown value length and return their values in an ATT_READ_MULTIPLE_VARIABLE_RSP PDU.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x20 = ATT_READ_MULTIPLE_VARIABLE_REQ PDU |
| Set Of Handles | 4 to (ATT_MTU-1) | A set of two or more attribute handles |

*Table 3.27: Format of ATT_READ_MULTIPLE_VARIABLE_REQ PDU*

The attribute handles in the Set Of Handles parameter shall all be valid handles.

The server shall respond with an ATT_READ_MULTIPLE_VARIABLE_RSP PDU if all attributes have sufficient permissions to allow reading.

Note: The attribute values for the attributes in the Set Of Handles parameters do not have to all be the same size.

Note: The attribute handles in the Set Of Handles parameter do not have to be in attribute handle order; they are in the order that the values are required in the response.

If the client has insufficient authorization to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Authorization*.

If the client has insufficient security to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Authentication*.

If the client has an encryption key size that is too short to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Encryption Key Size Too Short*.

If the client has not enabled encryption, and encryption is required to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Encryption*.

If any of the handles are invalid, then an ATT_ERROR_RSP PDU shall be sent with the error code *Invalid Handle*.

If any of the attribute values cannot be read due to permissions, then an ATT_ERROR_RSP PDU shall be sent with the error code *Read Not Permitted*.

If an ATT_ERROR_RSP PDU is sent, the Attribute Handle In Error parameter in the ATT_ERROR_RSP PDU (see Section 3.4.1.1) shall be set to the handle of the first attribute causing the error.

#### 3.4.4.12 ATT_READ_MULTIPLE_VARIABLE_RSP

The ATT_READ_MULTIPLE_VARIABLE_RSP PDU is sent in reply to a received ATT_READ_MULTIPLE_VARIABLE_REQ PDU and contains the lengths and values of the attributes that have been read.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x21 = ATT_READ_MULTIPLE_VARIABLE_RSP |
| Length Value Tuple List | 4 to (ATT_MTU-1) | A list of Length Value Tuples |

*Table 3.28: Format of ATT_READ_MULTIPLE_VARIABLE_RSP PDU*

The Length Value Tuple List shall be a concatenation of Length Value Tuples for each of the attribute handles in the request in the order that they were requested. If the Length Value Tuple List is longer than (ATT_MTU-1) octets, then it shall be truncated after (ATT_MTU-1) or, if that would be within the Value Length field of a Length Value Tuple, at the start of the Length Value Tuple.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) can be used to read the remaining octets of a long attribute value.

The Value Length field in a Length Value Tuple shall be set to the length of the Attribute Value field. The Attribute Value field in a Length Value Tuple shall be set to the value of the attribute being read.

| Value Length | Attribute Value |
|---|---|
| 2 octets | (Value Length) octets |

*Table 3.29: Format of the Length Value Tuple*

Note: If a Length Value Tuple is truncated, then the amount of Attribute Value will be less than the value of the Value Length field. The client must therefore not use the Value Length to determine the amount of the Attribute Value actually included in the PDU.

### 3.4.5 Writing attributes

#### 3.4.5.1 ATT_WRITE_REQ

The ATT_WRITE_REQ PDU is used to request the server to write the value of an attribute and acknowledge that this has been achieved in an ATT_WRITE_RSP PDU.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x12 = ATT_WRITE_REQ PDU |
| Attribute Handle | 2 | The handle of the attribute to be written |
| Attribute Value | 0 to (ATT_MTU-3) | The value to be written to the attribute |

*Table 3.30: Format of ATT_WRITE_REQ PDU*

The Attribute Handle shall be set to a valid handle.

The Attribute Value shall be set to the new value of the attribute.

If the attribute value has a variable length, then the attribute value shall be truncated or lengthened to match the length of the Attribute Value parameter.

Note: If an attribute value has a variable length and if the Attribute Value parameter is of zero length, the attribute value will be fully truncated.

If the attribute value has a fixed length and the Attribute Value parameter length is less than or equal to the length of the attribute value, the octets of the attribute value parameter length shall be written; all other octets in this attribute value shall be unchanged.

The server shall respond with an ATT_WRITE_RSP PDU if the handle is valid, the attribute has sufficient permissions to allow writing, and the attribute value has a valid size and format, and it is successful in writing the attribute.

If the attribute value has a variable length and the Attribute Value parameter length exceeds the maximum valid length of the attribute value then the server shall respond with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the attribute value has a fixed length and the requested attribute value parameter length is greater than the length of the attribute value then the server shall respond with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the client has insufficient authorization to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to write the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be written due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Write Not Permitted* (0x03).

If the attribute value cannot be written due to an application error then an ATT_ERROR_RSP PDU shall be sent with an error code defined by a higher layer specification.

#### 3.4.5.2 ATT_WRITE_RSP

The ATT_WRITE_RSP PDU is sent in reply to a valid ATT_WRITE_REQ PDU and acknowledges that the attribute has been successfully written.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x13 = ATT_WRITE_RSP PDU |

*Table 3.31: Format of ATT_WRITE_RSP*

The ATT_WRITE_RSP PDU shall be sent after the attribute value is written.

#### 3.4.5.3 ATT_WRITE_CMD

The ATT_WRITE_CMD PDU is used to request the server to write the value of an attribute, typically into a control-point attribute.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x52 = ATT_WRITE_CMD PDU |
| Attribute Handle | 2 | The handle of the attribute to be set |
| Attribute Value | 0 to (ATT_MTU-3) | The value of be written to the attribute |

*Table 3.32: Format of ATT_WRITE_CMD PDU*

The attribute handle parameter shall be set to a valid handle.

The attribute value parameter shall be set to the new value of the attribute.

If the attribute value has a variable length, then the attribute value shall be truncated or lengthened to match the length of the attribute value parameter.

Note: If an attribute value has a variable length and if the attribute value parameter is of zero length, the attribute value will be fully truncated.

If the attribute value has a fixed length and the attribute value parameter length is less than or equal to the length of the attribute value, the octets up to the attribute value parameter length shall be written; all other octets in this attribute value shall be unchanged.

If the attribute value has a variable length and the attribute value parameter length exceeds the maximum valid length of the attribute value then the server shall ignore the command.

If the attribute value has a fixed length and the requested attribute value parameter length is greater than the length of the attribute value then the server shall ignore the command.

No ATT_ERROR_RSP or ATT_WRITE_RSP PDUs shall be sent in response to this command. If the server cannot write this attribute for any reason the command shall be ignored.

#### 3.4.5.4 [This section is no longer used]

### 3.4.6 Queued writes

The purpose of queued writes is to queue up writes of values of multiple attributes in a first-in first-out queue and then execute the write on all of them in a single atomic operation.

Each client's queued values are separate; the execution of one queue shall not affect the preparation or execution of any other client's queued values. Each client has a single queue regardless of how many ATT bearers are currently established.

#### 3.4.6.1 ATT_PREPARE_WRITE_REQ

The ATT_PREPARE_WRITE_REQ PDU is used to request the server to prepare to write the value of an attribute. The server will respond to this request with an ATT_PREPARE_WRITE_RSP PDU, so that the client can verify that the value was received correctly.

A client may send more than one ATT_PREPARE_WRITE_REQ PDU to a server, which will queue and send a response for each handle value pair.

A server may limit the number of prepared writes that it can queue. A higher layer specification should define this limit.

After an ATT_PREPARE_WRITE_REQ PDU has been issued, and the response received, any other attribute command or request can be issued from the same client to the same server.

Any actions on attributes that exist in the prepare queue shall proceed as if the prepare queue did not exist, and the prepare queue shall be unaffected by these actions. A subsequent execute write will write the values in the prepare queue even if the value of the attribute has changed since the prepared writes were started.

The Attribute Protocol makes no determination on the validity of the Part Attribute Value or the Value Offset. A higher layer specification determines the meaning of the data.

Each ATT_PREPARE_WRITE_REQ PDU will be queued even if the attribute handle is the same as a previous ATT_PREPARE_WRITE_REQ PDU. These will then be executed in the order received, causing multiple writes for this attribute to occur.

If all ATT bearers belonging to the same client are lost while a number of pending prepared write values have been queued, the queue will be cleared and no writes will be executed.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x16 = ATT_PREPARE_WRITE_REQ PDU |
| Attribute Handle | 2 | The handle of the attribute to be written |
| Value Offset | 2 | The offset of the first octet to be written |
| Part Attribute Value | 0 to (ATT_MTU-5) | The value of the attribute to be written |

*Table 3.33: Format of ATT_PREPARE_WRITE_REQ PDU*

The Attribute Handle parameter shall be set to a valid handle.

The Value Offset parameter shall be set to the offset of the first octet where the Part Attribute Value parameter is to be written within the attribute value. The Value Offset parameter is based from zero; the first octet has an offset of zero, the second octet has an offset of one, etc.

The server shall respond with an ATT_PREPARE_WRITE_RSP PDU if the handle is valid, the attribute has sufficient permissions to allow writing at this time, and the prepare queue has sufficient space.

Note: The Attribute Value validation is done when an ATT_EXECUTE_WRITE_REQ PDU is received. Hence, any *Invalid Offset* (0x07) or *Invalid Attribute Value Length* (0x0D) errors are generated when an ATT_EXECUTE_WRITE_REQ PDU is received.

If the client has insufficient authorization to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to write the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the server does not have sufficient space to queue this request then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Prepare Queue Full* (0x09).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be written then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Write Not Permitted* (0x03).

The server shall not change the value of the attribute until an ATT_EXECUTE_WRITE_REQ PDU is received.

If an ATT_PREPARE_WRITE_REQ PDU was invalid, and therefore an ATT_ERROR_RSP PDU has been issued, then this prepared write will be considered to have not been received. All existing prepared writes in the prepare queue shall not be affected by this invalid request.

#### 3.4.6.2 ATT_PREPARE_WRITE_RSP

The ATT_PREPARE_WRITE_RSP PDU is sent in response to a received ATT_PREPARE_WRITE_REQ PDU and acknowledges that the value has been successfully received and placed in the prepare write queue.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x17 = ATT_PREPARE_WRITE_RSP PDU |
| Attribute Handle | 2 | The handle of the attribute to be written |
| Value Offset | 2 | The offset of the first octet to be written |

| Parameter | Size (octets) | Description |
|---|---|---|
| Part Attribute Value | 0 to (ATT_MTU-5) | The value of the attribute to be written |

*Table 3.34: Format of ATT_PREPARE_WRITE_RSP PDU*

The attribute handle shall be set to the same value as in the corresponding ATT_PREPARE_WRITE_REQ PDU.

The value offset and part attribute value shall be set to the same values as in the corresponding ATT_PREPARE_WRITE_REQ PDU.

#### 3.4.6.3 ATT_EXECUTE_WRITE_REQ

The ATT_EXECUTE_WRITE_REQ PDU is used to request the server to write or cancel the write of all the prepared values currently held in the prepare queue from this client. This request shall be handled by the server as an atomic operation.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x18 = ATT_EXECUTE_WRITE_REQ PDU |
| Flags | 1 | 0x00 – Cancel all prepared writes<br>0x01 – Immediately write all pending prepared values |

*Table 3.35: Format of ATT_EXECUTE_WRITE_REQ PDU*

When the flags parameter is set to 0x01, all pending prepared write values that are currently queued shall be written in the order they were received in the corresponding ATT_PREPARE_WRITE_REQ PDUs. The queue shall then be cleared and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

When the flags parameter is set to 0x00 all pending prepared write values shall be discarded for this client. The queue shall then be cleared, and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

Note: If multiple ATT bearers were used to send the prepared write requests, then the order that writes sent on different ATT bearers are executed is not specified and is not reported to the client. In addition, if the ATT_EXECUTE_WRITE_REQ PDU is sent before the client has received responses to all the prepared write requests, the requests for which the client has not yet received a response might form part of a subsequent write rather than this one; this is also not reported to the client.

If there are no pending prepared write values, then no values are written and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

If the prepared Attribute Value exceeds the maximum valid length of the attribute value, then all pending prepared write values shall be discarded for this client, the queue shall

then be cleared, and then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the prepared Value Offset is greater than the current length of the attribute value, then all pending prepared write values shall be discarded for this client, the queue shall be cleared, and then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Offset* (0x07).

If the pending prepared write values cannot be written due to an application error, then the queue shall be cleared and then an ATT_ERROR_RSP PDU shall be sent with a higher layer specification defined error code. The Attribute Handle In Error parameter shall be set to the attribute handle of the attribute from the prepare queue that caused this application error. The state of the attributes that were to be written from the prepare queue is not defined in this case.

If the pending prepared write values cannot be written due to a database change, then the queue shall be cleared and then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Database Out Of Sync* (0x12). The Attribute Handle In Error parameter shall be set to the attribute handle of the attribute from the prepare queue that caused this error. The state of the attributes that were to be written from the prepare queue is not defined in this case.

#### 3.4.6.4 ATT_EXECUTE_WRITE_RSP

The ATT_EXECUTE_WRITE_RSP PDU is sent in response to a received ATT_EXECUTE_WRITE_REQ PDU.

| Parameter | Size | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x19 - ATT_EXECUTE_WRITE_RSP PDU |

*Table 3.36: Format of ATT_EXECUTE_WRITE_RSP PDU*

The ATT_EXECUTE_WRITE_RSP PDU shall be sent after the attributes are written. In case an action is taken in response to the write, an indication may be used once the action is complete.

### 3.4.7 Server initiated

#### 3.4.7.1 ATT_HANDLE_VALUE_NTF

A server can send a notification of an attribute's value at any time.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x1B = ATT_HANDLE_VALUE_NTF PDU |
| Attribute Handle | 2 | The handle of the attribute |

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Value | 0 to (ATT_MTU-3) | The current value of the attribute |

*Table 3.37: Format of ATT_HANDLE_VALUE_NTF PDU*

The attribute handle shall be set to a valid handle.

The attribute value shall be set to the current value of the attribute identified by the attribute handle.

If the attribute value is longer than (ATT_MTU-3) octets, then only the first (ATT_MTU-3) octets of this attributes value can be sent in a notification.

Note: For a client to get a long attribute, it must use the ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) following the receipt of this notification.

If the attribute handle or the attribute value is invalid, then this notification shall be ignored upon reception.

#### 3.4.7.2 ATT_HANDLE_VALUE_IND

A server can send an indication of an attribute's value.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x1D = ATT_HANDLE_VALUE_IND PDU |
| Attribute Handle | 2 | The handle of the attribute |
| Attribute Value | 0 to (ATT_MTU-3) | The current value of the attribute |

*Table 3.38: Format of ATT_HANDLE_VALUE_IND PDU*

The attribute handle shall be set to a valid handle.

The attribute value shall be set to the current value of the attribute identified by the attribute handle.

If the attribute value is longer than (ATT_MTU-3) octets, then only the first (ATT_MTU - 3) octets of this attributes value can be sent in an indication.

Note: For a client to get a long attribute, it must use the ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5) following the receipt of this indication.

The client shall send an ATT_HANDLE_VALUE_CFM PDU in response to an ATT_HANDLE_VALUE_IND PDU. No further indications to this client shall occur until the confirmation has been received by the server.

If the attribute handle or the attribute value is invalid, the client shall send an ATT_HANDLE_VALUE_CFM PDU in response and shall discard the handle and value from the received indication.

#### 3.4.7.3 ATT_HANDLE_VALUE_CFM

The ATT_HANDLE_VALUE_CFM PDU is sent in response to a received ATT_HANDLE_VALUE_IND PDU and confirms that the client has received an indication of the given attribute.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x1E = ATT_HANDLE_VALUE_CFM PDU |

*Table 3.39: Format of ATT_HANDLE_VALUE_CFM PDU*

#### 3.4.7.4 ATT_MULTIPLE_HANDLE_VALUE_NTF

A server can send a notification of two or more attributes' values at any time.

| Parameter | Size (octets) | Description |
|---|---|---|
| Attribute Opcode | 1 | 0x23 = ATT_MULTIPLE_-HANDLE_VALUE_NTF |
| Handle Length Value Tuple List | 8 to (ATT_MTU-1) | A list of Handle Length Value Tuples |

*Table 3.40: Format of ATT_MULTIPLE_HANDLE_VALUE_NTF PDU*

The Handle Length Value Tuple List shall be a concatenation of Handle Length Value Tuples for each of the attributes being notified.

The server shall not truncate a Handle Length Value Tuple.

The Attribute Handle field in a Handle Length Value Tuple shall be set to the handle of the attribute being notified. The Value Length field in a Handle Length Value Tuple shall be set to the length of the Attribute Value field. The Attribute Value field in a Handle Length Value Tuple shall be set to the value of the attribute being notified.

| Attribute Handle | Value Length | Attribute Value |
|---|---|---|
| 2 octets | 2 octets | (Value Length) octets |

*Table 3.41: Format of the Handle Length Value Tuple*

If an attribute handle or an attribute value is invalid, then the client shall ignore that attribute when receiving this notification.

### 3.4.8 Attribute Opcode summary

Table 3.42 gives a summary of the Attribute Protocol PDUs.

| Attribute PDU Name | Attribute Opcode | Parameters |
|---|---|---|
| **ATT_ERROR_RSP** | 0x01 | Request Opcode in Error,<br>Attribute Handle In Error,<br>Error Code |
| **ATT_EXCHANGE_MTU_REQ** | 0x02 | Client Rx MTU |
| **ATT_EXCHANGE_MTU_RSP** | 0x03 | Server Rx MTU |
| **ATT_FIND_INFORMATION_REQ** | 0x04 | Starting Handle,<br>Ending Handle |
| **ATT_FIND_INFORMATION_RSP** | 0x05 | Format,<br>Information Data |
| **ATT_FIND_BY_TYPE_VALUE_REQ** | 0x06 | Starting Handle,<br>Ending Handle,<br>Attribute Type,<br>Attribute Value |
| **ATT_FIND_BY_TYPE_VALUE_RSP** | 0x07 | Handles Information List |
| **ATT_READ_BY_TYPE_REQ** | 0x08 | Starting Handle,<br>Ending Handle,<br>UUID |
| **ATT_READ_BY_TYPE_RSP** | 0x09 | Length,<br>Attribute Data List |
| **ATT_READ_REQ** | 0x0A | Attribute Handle |
| **ATT_READ_RSP** | 0x0B | Attribute Value |
| **ATT_READ_BLOB_REQ** | 0x0C | Attribute Handle,<br>Value Offset |
| **ATT_READ_BLOB_RSP** | 0x0D | Part Attribute Value |
| **ATT_READ_MULTIPLE_REQ** | 0x0E | Handle Set |
| **ATT_READ_MULTIPLE_RSP** | 0x0F | Value Set |
| **ATT_READ_BY_GROUP_TYPE_REQ** | 0x10 | Start Handle,<br>Ending Handle,<br>UUID |
| **ATT_READ_BY_GROUP_TYPE_RSP** | 0x11 | Length,<br>Attribute Data List |
| **ATT_WRITE_REQ** | 0x12 | Attribute Handle,<br>Attribute Value |
| **ATT_WRITE_RSP** | 0x13 | *none* |

| Attribute PDU Name | Attribute Opcode | Parameters |
|---|---|---|
| **ATT_WRITE_CMD** | 0x52 | Attribute Handle,<br>Attribute Value |
| **ATT_PREPARE_WRITE_REQ** | 0x16 | Attribute Handle,<br>Value Offset,<br>Part Attribute Value |
| **ATT_PREPARE_WRITE_RSP** | 0x17 | Attribute Handle,<br>Value Offset,<br>Part Attribute Value |
| **ATT_EXECUTE_WRITE_REQ** | 0x18 | Flags |
| **ATT_EXECUTE_WRITE_RSP** | 0x19 | *none* |
| **ATT_READ_MULTIPLE_VARIABLE_REQ** | 0x20 | Set Of Handles |
| **ATT_READ_MULTIPLE_VARIABLE_RSP** | 0x21 | Length Value Tuple List |
| **ATT_MULTIPLE_HANDLE_VALUE_NTF** | 0x23 | Handle Length Value Tuple List |
| **ATT_HANDLE_VALUE_NTF** | 0x1B | Attribute Handle,<br>Attribute Value |
| **ATT_HANDLE_VALUE_IND** | 0x1D | Attribute Handle,<br>Attribute Value |
| **ATT_HANDLE_VALUE_CFM** | 0x1E | *none* |

*Table 3.42: Attribute Protocol summary*

Note: Attribute Opcode 0xD2 is previously used (see [Vol 1] Part E, Section 2.4.2).

### 3.4.9 Attribute PDU response summary

Table 3.43 gives a summary of the Attribute PDU Method responses that are allowed. Each method indicates the method that should be sent as a successful response, and whether an ATT_ERROR_RSP PDU can be sent in response instead. If an ATT_ERROR_RSP PDU can be sent, then Table 3.43 also indicates the error codes that are valid within this ATT_ERROR_RSP PDU for the given method.

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_EXCHANGE_-MTU_REQ** | ATT_EXCHANGE_-MTU_RSP | Yes | *Request Not Supported* (0x06) |
| **ATT_FIND_-INFORMATION_REQ** | ATT_FIND_-INFORMATION_RSP | Yes | *Invalid Handle* (0x01),<br>*Attribute Not Found* (0x0A) |

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_FIND_BY_-TYPE_VALUE_REQ** | ATT_FIND_BY_-TYPE_VALUE_RSP | Yes | *Invalid Handle* (0x01),<br>*Request Not Supported* (0x06),<br>*Attribute Not Found* (0x0A) |
| **ATT_READ_BY_-TYPE_REQ** | ATT_READ_BY_-TYPE_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Attribute Not Found* (0x0A),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |
| **ATT_READ_REQ** | ATT_READ_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_READ_BLOB_-REQ** | ATT_READ_BLOB_-RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Invalid Offset* (0x07),<br>*Attribute Not Long* (0x0B),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |
| **ATT_READ_-MULTIPLE_REQ** | ATT_READ_-MULTIPLE_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_READ_BY_-GROUP_TYPE_REQ** | ATT_READ_BY_-GROUP_TYPE_RSP | Yes | *Invalid Handle* (0x01),<br>*Request Not Supported* (0x06),<br>*Attribute Not Found* (0x0A),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Unsupported Group Type* (0x10),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |
| **ATT_READ_-MULTIPLE_-VARIABLE_REQ** | ATT_READ_-MULTIPLE_-VARIABLE_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Read Not Permitted* (0x02),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_WRITE_REQ** | ATT_WRITE_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Write Not Permitted* (0x03),<br>*Invalid Attribute Value Length* (0x0D),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |
| **ATT_WRITE_CMD** | *none* | No | *none* |
| **ATT_PREPARE_-WRITE_REQ** | ATT_PREPARE_-WRITE_RSP | Yes | *Invalid Handle* (0x01),<br>*Database Out Of Sync* (0x12),<br>*Request Not Supported* (0x06),<br>*Insufficient Authorization* (0x08),<br>*Insufficient Authentication* (0x05),<br>*Write Not Permitted* (0x03),<br>*Prepare Queue Full* (0x09),<br>*Insufficient Encryption* (0x0F),<br>*Encryption Key Size Too Short* (0x0C),<br>*Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF) |

| Attribute PDU Method | Successful Response PDU | ATT_ERROR_-RSP Allowed | Valid Error Codes |
|---|---|---|---|
| **ATT_EXECUTE_-WRITE_REQ** | ATT_EXECUTE_-WRITE_RSP | Yes | *Application Error* (0x80 to 0x9F),<br>*Common Profile and Service Error Codes* (0xE0 to 0xFF),<br>*Invalid Offset* (0x07),<br>*Invalid Attribute Value Length* (0x0D),<br>*Database Out Of Sync* (0x12) |
| **ATT_HANDLE_-VALUE_NTF** | *none* | No | *none* |
| **ATT_HANDLE_-VALUE_IND** | ATT_HANDLE_-VALUE_CFM | No | *none* |
| **ATT_MULTIPLE_-HANDLE_VALUE_-NTF** | *none* | No | *none* |

*Table 3.43: Attribute request and response summary*

# 4 SECURITY CONSIDERATIONS

The Attribute Protocol can be used to access information that may require both authorization and an authenticated and encrypted physical link before an attribute can be read or written.

If such a request is issued when the client has not been authorized to access this information, the server shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authorization* (0x08). The authorization requirements for access to a given attribute are not defined in this Part. Each device implementation will determine how authorization occurs. Authorization procedures are defined in GAP, and may be further refined in a higher layer specification.

If such a request is issued when the physical link is unauthenticated, the server shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05). To read or write this attribute, a client can then request that the physical link be authenticated, and once this has been completed, send the request again.

The Attribute Protocol can be used to notify or indicate the value of an attribute that may require an authenticated and encrypted physical link before an attribute notification or indication is performed. To notify or indicate this attribute, a server can then request that the physical link be authenticated, and once this has been completed, send the notification or indication.

The list of attributes that a device supports is not considered private or confidential information, and therefore the ATT_FIND_INFORMATION_REQ PDU shall always be permitted. This implies that the error code *Insufficient Authorization* (0x08) or *Insufficient Authentication* (0x05) shall not be used in an ATT_ERROR_RSP PDU for an ATT_FIND_INFORMATION_REQ PDU.

For example, an attribute value may be allowed to be read by any device, but only written by an authenticated device. An implementation should take this into account, and not assume that just because it can read an attribute’s value, it will also be able to write the value. Similarly, just because an attribute value can be written, does not mean that an attribute value can also be read. Each individual attribute could have different security requirements.

When a client accesses an attribute, the order of checks that are performed on the server will have security implications. A server shall check authentication and authorization requirements before any other check is performed.

Note: For example, if the authentication and authorization requirement checks are not performed first then the size of an attribute could be determined by sending repeated ATT_READ_BLOB_REQ PDUs for an attribute that a client does not have access to, because either the error code *Invalid Offset* (0x07) or *Insufficient Authentication* (0x05) would be returned.

# 5 REFERENCES

[1] Core Specification Supplement, Part B, Common Profile and Service Error Codes

# Appendix A Changes to PDU names

Previous versions of this specification used different names for the PDUs defined in Section 3.4. Table A.1 shows the previous and current names of these PDUs.

| Previous name | Current name |
|---|---|
| Error Response | ATT_ERROR_RSP |
| Exchange MTU Request | ATT_EXCHANGE_MTU_REQ |
| Exchange MTU Response | ATT_EXCHANGE_MTU_RSP |
| Execute Write Request | ATT_EXECUTE_WRITE_REQ |
| Execute Write Response | ATT_EXECUTE_WRITE_RSP |
| Find By Type Value Request | ATT_FIND_BY_TYPE_VALUE_REQ |
| Find By Type Value Response | ATT_FIND_BY_TYPE_VALUE_RSP |
| Find Information Request | ATT_FIND_INFORMATION_REQ |
| Find Information Response | ATT_FIND_INFORMATION_RSP |
| Handle Value Confirmation | ATT_HANDLE_VALUE_CFM |
| Handle Value Indication | ATT_HANDLE_VALUE_IND |
| Handle Value Notification | ATT_HANDLE_VALUE_NTF |
| Prepare Write Request | ATT_PREPARE_WRITE_REQ |
| Prepare Write Response | ATT_PREPARE_WRITE_RSP |
| Read Blob Request | ATT_READ_BLOB_REQ |
| Read Blob Response | ATT_READ_BLOB_RSP |
| Read by Group Type Request | ATT_READ_BY_GROUP_TYPE_REQ |
| Read by Group Type Response | ATT_READ_BY_GROUP_TYPE_RSP |
| Read By Type Request | ATT_READ_BY_TYPE_REQ |
| Read By Type Response | ATT_READ_BY_TYPE_RSP |
| Read Multiple Request | ATT_READ_MULTIPLE_REQ |
| Read Multiple Response | ATT_READ_MULTIPLE_RSP |
| Read Request | ATT_READ_REQ |
| Read Response | ATT_READ_RSP |
| Write Command | ATT_WRITE_CMD |
| Write Request | ATT_WRITE_REQ |
| Write Response | ATT_WRITE_RSP |

*Table A.1: Changes to PDU names*

Host
Part G

# GENERIC ATTRIBUTE PROFILE (GATT)

*This Part defines the Generic Attribute Profile that describes a service framework using the Attribute Protocol for discovering services, and for reading and writing characteristic values on a peer device.*

# CONTENTS

**1 Introduction ..... 1552**
1.1 Scope ..... 1552
1.2 Profile dependency ..... 1552
1.3 [This section is no longer used] ..... 1552
1.4 [This section is no longer used] ..... 1552
1.5 Conventions ..... 1552

**2 Profile overview ..... 1553**
2.1 Protocol stack ..... 1553
2.2 Configurations and roles ..... 1553
2.3 User requirements and scenarios ..... 1554
2.4 Profile fundamentals ..... 1555
2.5 Attribute Protocol ..... 1555
2.5.1 Overview ..... 1555
2.5.2 Attribute caching ..... 1556
2.5.2.1 Robust Caching ..... 1558
2.5.3 Attribute grouping ..... 1562
2.5.4 UUIDs ..... 1562
2.6 GATT Profile hierarchy ..... 1562
2.6.1 Overview ..... 1562
2.6.2 Service ..... 1563
2.6.3 Included services ..... 1564
2.6.4 Characteristic ..... 1564
2.7 Configured Broadcast ..... 1564

**3 Service interoperability requirements ..... 1566**
3.1 Service definition ..... 1566
3.2 Include definition ..... 1567
3.3 Characteristic definition ..... 1567
3.3.1 Characteristic declaration ..... 1568
3.3.1.1 Characteristic Properties ..... 1569
3.3.1.2 Characteristic Value Attribute Handle ..... 1569
3.3.1.3 Characteristic UUID ..... 1570
3.3.2 Characteristic Value declaration ..... 1570
3.3.3 Characteristic descriptor declarations ..... 1570
3.3.3.1 Characteristic Extended Properties ..... 1571
3.3.3.2 Characteristic User Description ..... 1571
3.3.3.3 Client Characteristic Configuration ..... 1572
3.3.3.4 Server Characteristic Configuration ..... 1573
3.3.3.5 Characteristic Presentation Format ..... 1574

3.3.3.6 Characteristic Aggregate Format ..... 1576
3.4 Summary of GATT Profile attribute types ..... 1577

**4 GATT feature requirements ..... 1578**
4.1 Overview ..... 1578
4.2 Feature support and procedure mapping ..... 1578
4.3 Server configuration ..... 1580
4.3.1 Exchange MTU ..... 1580
4.4 Primary Service Discovery ..... 1581
4.4.1 Discover All Primary Services ..... 1581
4.4.2 Discover Primary Service by Service UUID ..... 1582
4.5 Relationship Discovery ..... 1583
4.5.1 Find Included Services ..... 1584
4.6 Characteristic discovery ..... 1585
4.6.1 Discover All Characteristics of a Service ..... 1585
4.6.2 Discover Characteristics by UUID ..... 1586
4.7 Characteristic Descriptor Discovery ..... 1588
4.7.1 Discover All Characteristic Descriptors ..... 1588
4.8 Characteristic Value Read ..... 1589
4.8.1 Read Characteristic Value ..... 1589
4.8.2 Read Using Characteristic UUID ..... 1590
4.8.3 Read Long Characteristic Value ..... 1591
4.8.4 Read Multiple Characteristic Values ..... 1592
4.8.5 Read Multiple Variable Length Characteristic Values 1593
4.9 Characteristic Value Write ..... 1594
4.9.1 Write Without Response ..... 1594
4.9.2 [This section is no longer used] ..... 1594
4.9.3 Write Characteristic Value ..... 1594
4.9.4 Write Long Characteristic Value ..... 1595
4.9.5 Characteristic Value Reliable Writes ..... 1596
4.10 Characteristic Value Notification ..... 1598
4.10.1 Single Notification ..... 1599
4.10.2 Multiple Variable Length Notifications ..... 1599
4.11 Characteristic Value Indication ..... 1599
4.11.1 Indication ..... 1600
4.12 Characteristic Descriptors ..... 1600
4.12.1 Read Characteristic Descriptor ..... 1600
4.12.2 Read Long Characteristic Descriptor ..... 1601
4.12.3 Write Characteristic Descriptor ..... 1602
4.12.4 Write Long Characteristic Descriptor ..... 1603
4.13 GATT procedure mapping to ATT protocol opcodes ..... 1604
4.14 Procedure timeouts ..... 1607

**5 L2CAP interoperability requirements ..... 1608**
5.1 BR/EDR L2CAP interoperability requirements ..... 1608
5.1.1 ATT_MTU ..... 1608
5.1.2 BR/EDR channel requirements ..... 1608
5.1.3 [This section is no longer used] ..... 1609
5.2 LE L2CAP interoperability requirements ..... 1609
5.2.1 ATT_MTU ..... 1609
5.2.2 LE channel requirements ..... 1609
5.3 Enhanced ATT bearer L2CAP interoperability requirements .... 1610
5.3.1 ATT_MTU ..... 1610
5.3.2 Channel Requirements ..... 1610
5.4 L2CAP collision mitigation ..... 1610
5.5 Bearer support ..... 1611

**6 GAP interoperability requirements ..... 1612**
6.1 BR/EDR GAP interoperability requirements ..... 1612
6.1.1 Connection Establishment ..... 1612
6.2 LE GAP interoperability requirements ..... 1612
6.2.1 Connection Establishment ..... 1612
6.2.2 Profile roles ..... 1613
6.3 Disconnected events ..... 1613
6.3.1 Notifications and indications while disconnected ..... 1613

**7 Defined GATT service ..... 1614**
7.1 Service Changed ..... 1614
7.2 Client Supported Features ..... 1615
7.3 Database Hash ..... 1617
7.3.1 Database Hash calculation ..... 1618
7.4 Server Supported Features ..... 1619

**8 Security considerations ..... 1620**
8.1 Authentication requirements ..... 1620
8.2 Authorization requirements ..... 1621

**9 SDP interoperability requirements ..... 1622**

**10 References ..... 1624**

**Appendix A Example ATT Server contents ..... 1625**

**Appendix B Example Database Hash ..... 1628**

# 1 INTRODUCTION

## 1.1 Scope

The Generic Attribute profile (GATT) defines a service framework using the Attribute Protocol. This framework defines procedures and formats of services and their characteristics. The procedures defined include discovering, reading, writing, notifying and indicating characteristics, as well as configuring the broadcast of characteristics.

## 1.2 Profile dependency

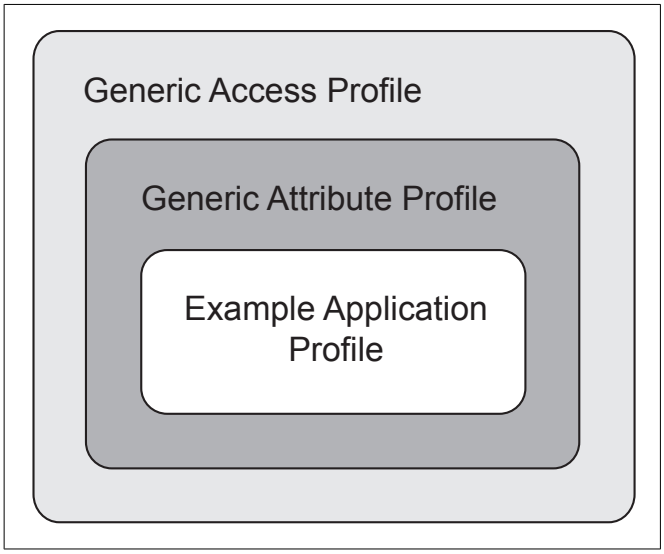



*Figure 1.1: Profile dependencies*

Figure 1.1 depicts the structure and the dependencies of the profiles. A profile is dependent upon another profile if it re-uses parts of that profile by implicitly or explicitly referencing it.

## 1.3 [This section is no longer used]

## 1.4 [This section is no longer used]

## 1.5 Conventions

In this Part the use of literal terms such as procedure, PDUs, opcodes, or function names appear in italics. Specific names of fields in structures, packets, etc. also appear in italics. The use of « » (e.g. «Primary Service») indicates a UUID. Attribute Protocol error codes (see [Vol 3] Part F, Table 3.4) appear in italics followed by the numeric error code.

# 2 PROFILE OVERVIEW

The GATT profile is designed to be used by an application or another profile, so that a client can communicate with a server. The server contains a number of attributes, and the GATT Profile defines how to use the Attribute Protocol to discover, read, write and obtain indications of these attributes, as well as configuring broadcast of attributes.

## 2.1 Protocol stack

Figure 2.1 shows the peer protocols used by this profile.

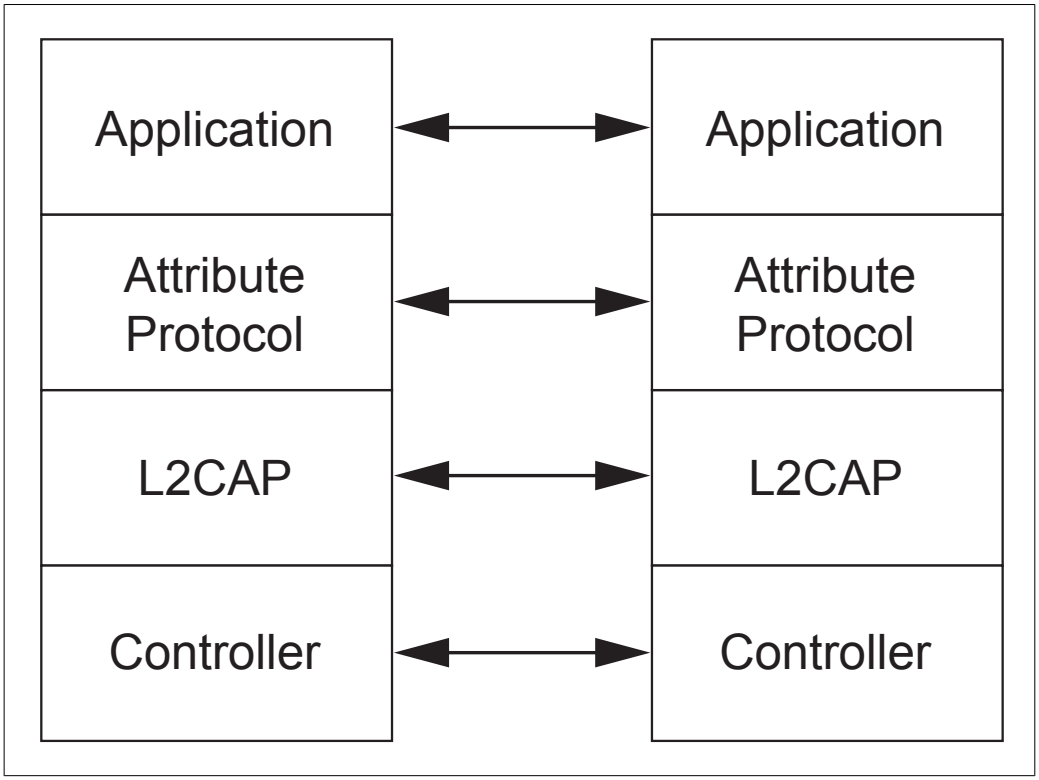



*Figure 2.1: Protocol model*

## 2.2 Configurations and roles

The following roles are defined for devices that implement this profile:

Client—This is the device that initiates commands and requests towards the server and can receive responses, indications and notifications sent by the server.

Server—This is the device that accepts incoming commands and requests from the client and sends responses, indications and notifications to a client.

Note: The roles are not fixed to the device. The roles are determined when a device initiates a defined procedure, and they are released when the procedure ends.

A device can act in both roles at the same time.

An example of configurations illustrating the roles for this profile is depicted in Figure 2.2.

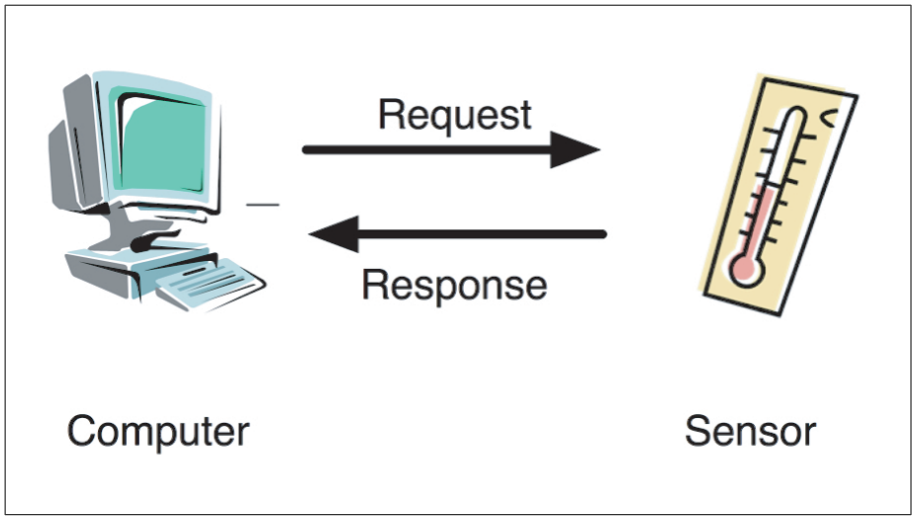



*Figure 2.2: Examples of configuration*

In Figure 2.2, the computer is the temperature service client and the sensor is the temperature service server. The computer initiates procedures to configure the sensor or to read the sensor values. In this example the sensor provides information about the characteristics the sensor device exposes as part of the temperature service and may permit some characteristics to be written. Also, the sensor responds to read requests with the appropriate values.

## 2.3 User requirements and scenarios

The following scenarios are covered by this profile:

- Exchanging configuration
- Discovery of services and characteristics on a device
- Reading a characteristic value
- Writing a characteristic value
- Notification of a characteristic value
- Indication of a characteristic value

## 2.4 Profile fundamentals

This profile can be used over any physical link, using the Attribute Protocol L2CAP channel, known as the ATT Bearer. Here is a brief summary of lower layer requirements communication between the client and the server.

- An ATT bearer is established using “Channel Establishment” as defined in Section 6.
- The profile roles are not tied to the Controller roles (i.e. Central or Peripheral).
- On an LE Physical link, use of security features such as authorization, authentication and encryption are optional. On a BR/EDR physical link encryption is mandatory.
- Multi-octet fields within the GATT profile shall be sent least significant octet first (little-endian) with the exception of the Characteristic Value field. The Characteristic Value and any fields within it shall be little-endian unless otherwise defined in the specification which defines the characteristic.
- Multiple ATT bearers may be established between a client and a server. The server can determine if ATT bearers are from the same client, and vice versa, by using connection information such as the Bluetooth Device Address of the peer device.

## 2.5 Attribute Protocol

The GATT profile requires the implementation of the Attribute Protocol (See [Vol 3] Part F) and those Attribute Protocol PDUs required by Section 4.2 and Section 4.13.

### 2.5.1 Overview

The GATT Profile uses the Attribute Protocol to transport data in the form of commands, requests, responses, indications, notifications and confirmations between devices. This data is contained in Attribute Protocol PDUs as specified in Figure 2.3.

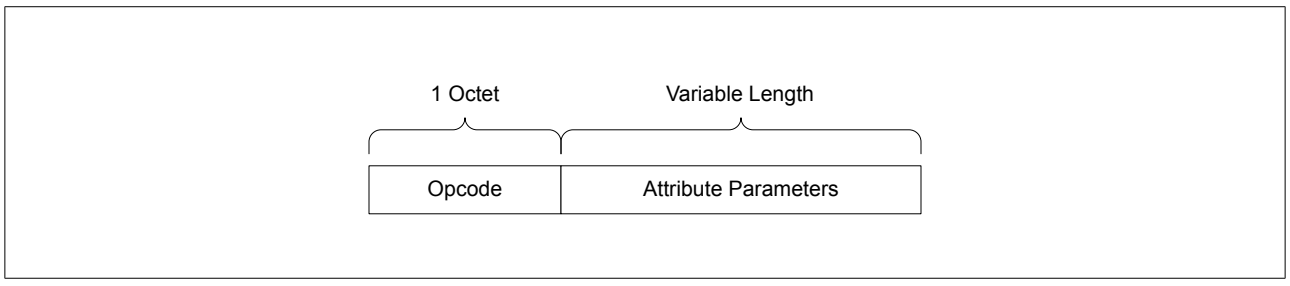



*Figure 2.3: Attribute Protocol PDU*

The *Opcode* contains the specific command, request, response, indication, notification or confirmation opcode. The *Attribute Parameters* contain data for the specific command or request or the data returned in a response, indication, or notification.

Attribute Protocol commands and requests act on values stored in Attributes on the server device. An Attribute is composed of four parts: *Attribute Handle*, *Attribute Type*,

*Attribute Value*, and *Attribute Permissions*. Figure 2.4 shows a logical representation of an Attribute. The actual representation for a given implementation is specific to that implementation.

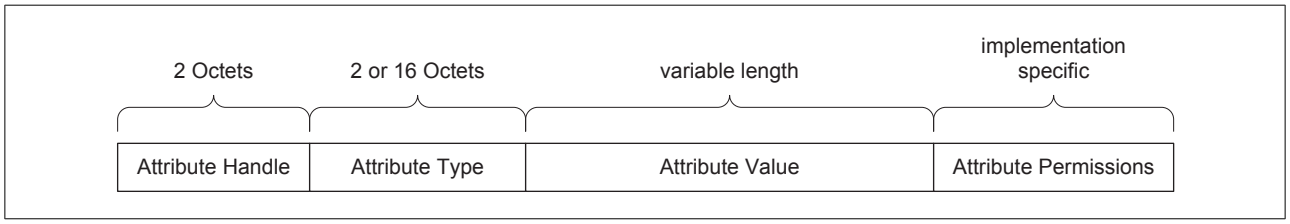



*Figure 2.4: Logical attribute representation*

The *Attribute Handle* is an index corresponding to a specific Attribute. The *Attribute Type* is a UUID that describes the *Attribute Value*. The *Attribute Value* is the data described by the *Attribute Type* and indexed by the *Attribute Handle*. The Attributes are ordered by increasing *Attribute Handle* values. *Attribute Handle* values may begin at any value between 0x0001 and 0xFFFF. Although the *Attribute Handle* values are in increasing order, following *Attribute Handle* values may differ by more than one. That is to say there may be gaps between successive *Attribute Handles*. When the specification requires two attribute handles to be adjacent or for one to immediately follow one the other, such gaps are still permitted and shall be ignored.

*Attribute Permissions* is part of the Attribute that cannot be read from or written to using the Attribute Protocol. It is used by the server to determine whether read or write access is permitted for a given attribute. *Attribute Permissions* are established by the GATT profile, a higher layer profile or are implementation specific if not specified.

### 2.5.2 Attribute caching

Attribute caching is an optimization that allows the client to discover the Attribute information such as *Attribute Handles* used by the server once and use the same Attribute information across reconnections without rediscovery. If the client does not cache the Attribute information, then it must rediscover the *Attribute* information at each reconnection. With caching, time is saved and a significant number of packets need not be exchanged between the client and server. The Attribute information that shall be cached by a client is the *Attribute Handles* of all server attributes and the GATT service characteristics values.

*Attribute Handles* used by the server should not change over time. This means that once an *Attribute Handle* is discovered by a client the *Attribute Handle* for that Attribute should not be changed.

Some circumstances may cause servers to change the *Attribute Handles* used for services, perhaps due to a factory reset or a firmware upgrade procedure being performed. The following is only required on the server if the services on the server

can be added, modified or removed. If GATT based services on the server cannot be changed during the usable lifetime of the device, the *Service Changed* characteristic shall not exist on the server and the client does not need to ever perform service discovery after the initial service discovery for that server.

To support caching when a server supports changes in GATT based services, an indication is sent by the server to clients when a service is added, removed, or modified on the server. A client may also detect a service change by reading the *Database Hash* characteristic if that characteristic exists on the server. A GATT based service is considered modified if the binding of the *Attribute Handles* to the associated Attributes grouped within a service definition are changed. Any change to the GATT service definition characteristic values other than the *Service Changed* characteristic value and the *Client Supported Features* characteristic value themselves shall also be considered a modification.

For clients that have a trusted relationship (i.e. bond) with the server, the attribute cache is valid across connections. For clients with a trusted relationship and not in a connection when a service change occurs, the server shall send an indication when the client reconnects to the server (see Section 7.1). For clients that do not have a trusted relationship with the server and that do not support reading the *Database Hash* characteristic, the attribute cache is valid only during the connection. Clients without a trusted relationship that do support reading the *Database Hash* characteristic may validate the attribute cache on connection setup. Clients without a trusted relationship shall receive an indication when the service change occurs only during the current connection.

Note: Clients without a trusted relationship that support caching must either perform service discovery or detect service changes by reading the *Database Hash* characteristic on each connection if the server supports the *Service Changed* characteristic.

The server shall send an ATT_HANDLE_VALUE_IND PDU containing the range of affected *Attribute Handles* that shall be considered invalid in the client's attribute cache. The start *Attribute Handle* shall be the start *Attribute Handle* of the service definition containing the change and the end *Attribute Handle* shall be the last *Attribute Handle* of the service definition containing the change. The value in the indication is composed of two 16-bit *Attribute Handles* concatenated to indicate the affected *Attribute Handle* range.

Note: A server can set the affected *Attribute Handle* range to 0x0001 to 0xFFFF to indicate to the client to rediscover the entire set of *Attribute Handles* on the server.

If the *Database Hash* characteristic exists on the server then, each time a service change occurs, the server shall update the *Database Hash* characteristic value with the new Database Hash (see Section 7.3).

If the *Database Hash* characteristic value has changed since the last time it was read, the client shall consider its attribute cache invalid and shall not make use of the cached information until it has performed service discovery or obtained the changed database definitions using an out-of-band mechanism.

The client, upon receiving an ATT_HANDLE_VALUE_IND PDU containing the range of affected Attribute Handles, shall consider the attribute cache invalid over the affected Attribute Handle range. Any outstanding request transaction shall be considered invalid if the Attribute Handle is contained within the affected Attribute Handle range. The client must perform service discovery before the client uses any service that has an attribute within the affected Attribute Handle range. Alternatively, the client may read the *Database Hash* characteristic and obtain the changed database definitions using an out-of-band mechanism. If the client receives an ATT_HANDLE_VALUE_IND PDU during service discovery and the client has read the *Database Hash* characteristic prior to the service discovery, the client may read the *Database Hash* characteristic again to determine if the current service discovery can be continued or if a new service discovery is required.

Once the server has received the ATT_HANDLE_VALUE_CFM PDU, the server can consider the client to be aware of the updated Attribute Handles.

The client shall consider the affected *Attribute Handle* range to be invalid in its attribute cache and perform the discovery procedures to restore the attribute cache. The server shall store service changed information for all bonded devices.

#### 2.5.2.1 Robust Caching

Robust Caching is a feature where the server sends an ATT_ERROR_RSP PDU to the client if the server does not consider the client to be aware of a service change.

If the *Database Hash* and *Service Changed* characteristics are both present on the server, then the server shall support the Robust Caching feature.

From the perspective of a server, each connected client is either "change-aware" or "change-unaware" regarding changes in the database definitions. After connecting to a server, the initial state of a client without a trusted relationship is change-aware. The initial state of a client with a trusted relationship is unchanged from the previous connection unless the database has been updated since the last connection, in which case the initial state is change-unaware.

Whenever the server updates the database definitions, all connected clients become change-unaware. A change-unaware connected client becomes change-aware when it reads the *Database Hash* characteristic and then the server receives another ATT request from the client. A change-unaware client using multiple ATT bearers shall wait

until the server has responded to all pending requests before reading the *Database Hash* characteristic (see Figure 2.5).

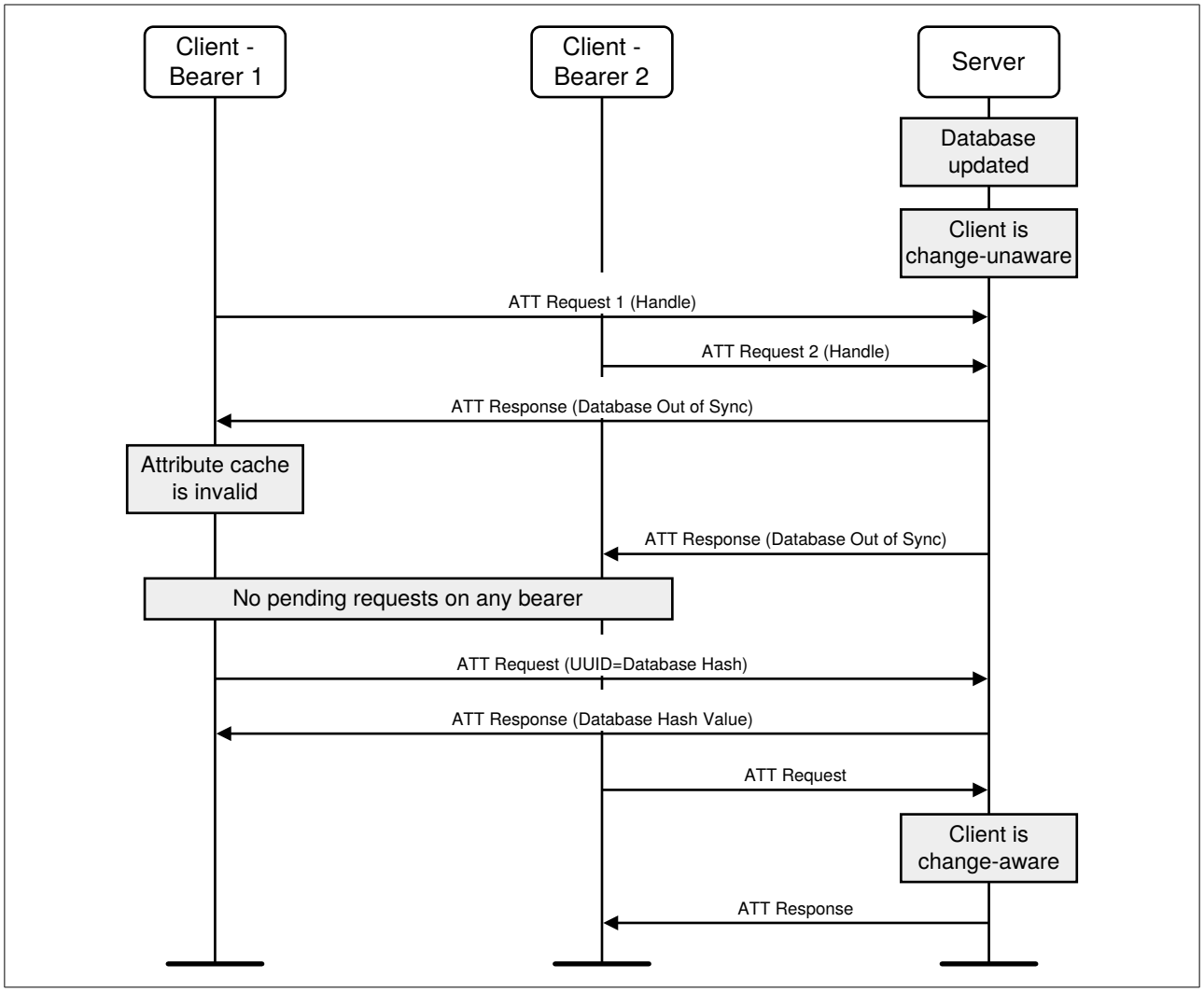



*Figure 2.5: Transition to change-aware state for a client using multiple ATT bearers*

In addition, a change-unaware connected client using exactly one ATT bearer becomes change-aware when either of the following happen:

- The client receives and confirms a *Handle Value Indication* for the *Service Changed* characteristic (see Figure 2.6).
- The server sends the client a response with the Error Code parameter set to *Database Out Of Sync* (0x12) and then the server receives another ATT request from the client (see Figure 2.7).

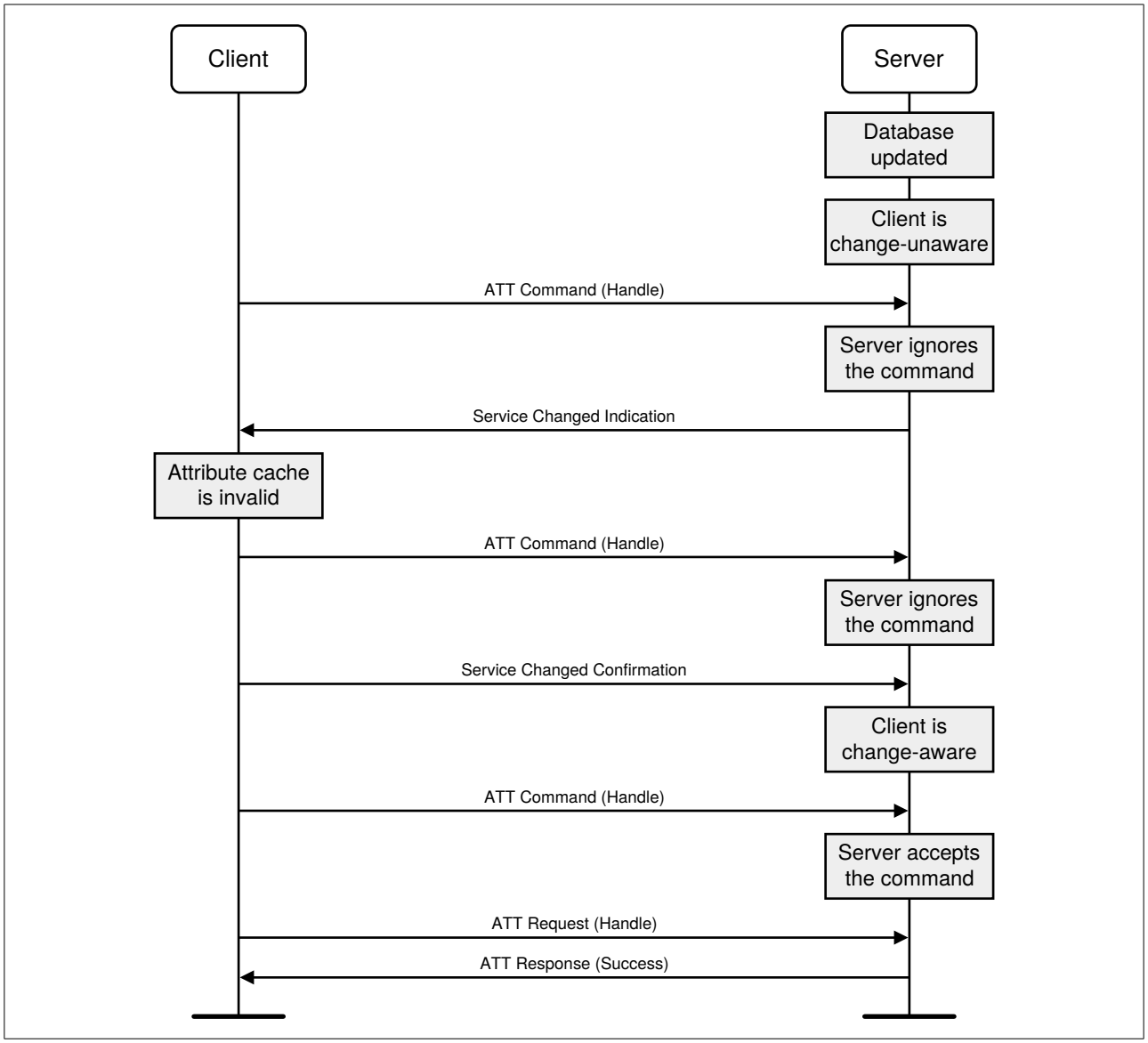



*Figure 2.6: Transition to change-aware state for a client using exactly one ATT bearer, triggered by a Service Changed confirmation*

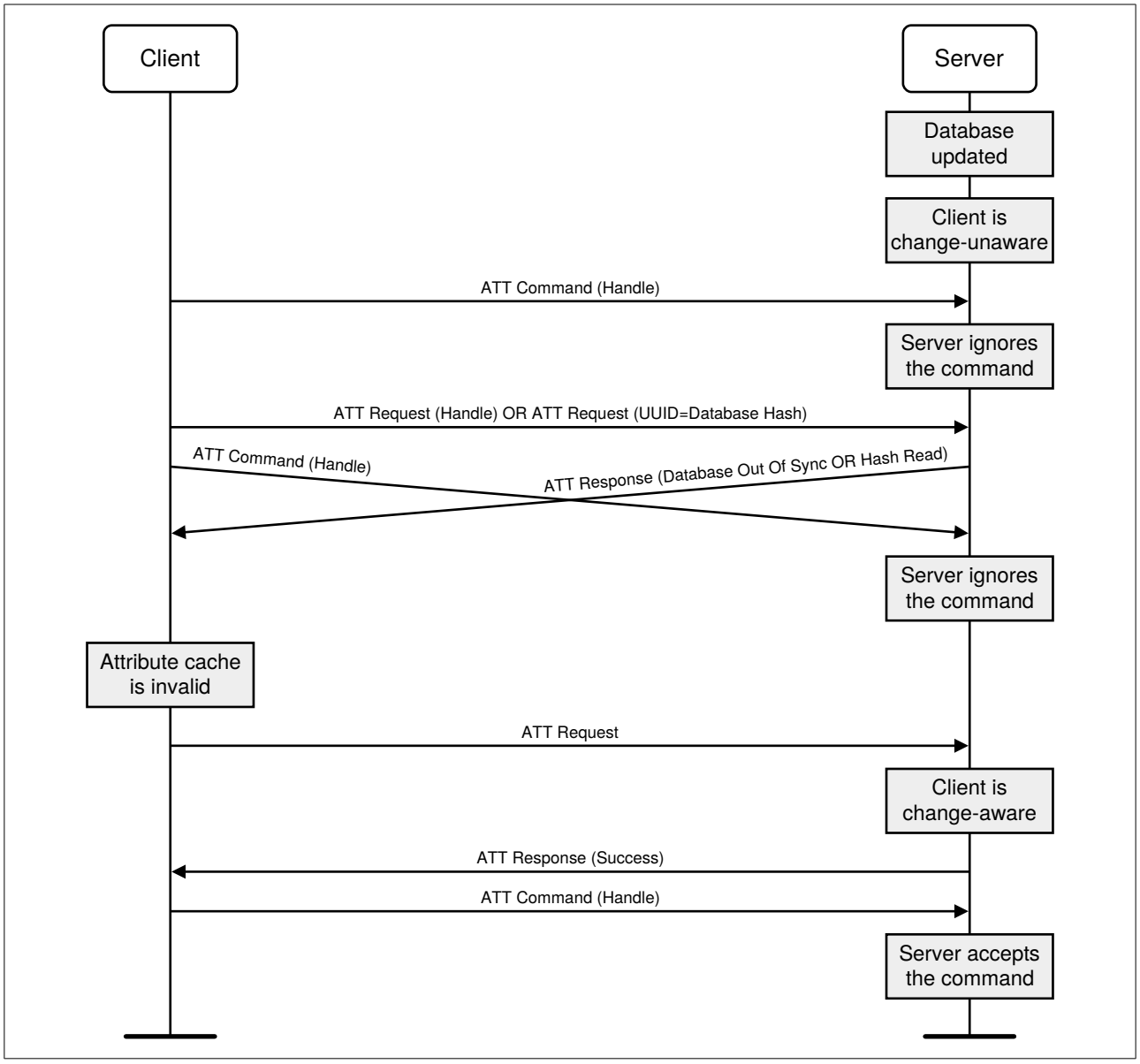



*Figure 2.7: Transition to change-aware state for a client using exactly one ATT bearer, triggered by the ATT response "Database Out Of Sync" or "Hash Read"*

If a client that has indicated support for robust caching (by setting the *Robust Caching* bit in the *Client Supported Features* characteristic) is change-unaware then the server shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Database Out Of Sync* (0x12) when either of the following happen:

- That client requests an operation at any *Attribute Handle* or list of *Attribute Handles* by sending an ATT request.
- That client sends an ATT_READ_BY_TYPE_REQ PDU with Attribute Type other than «Include» or «Characteristic» and an *Attribute Handle* range other than 0x0001 to 0xFFFF.

The ATT_ERROR_RSP PDU is sent only once per bearer after the client becomes change-unaware, unless the client disconnects or the database changes again before the client becomes change-aware in which case the ATT_ERROR_RSP PDU shall be sent again. If a change-unaware client sends an ATT command, the server shall ignore it. Except for a *Handle Value Indication* for the *Service Changed* characteristic, which the server should send as soon as possible after a service change, the server shall not send notifications and indications to such a client until it becomes change-aware. If a client is change-aware, then the server shall perform the operation normally.

If a client receives an ATT_ERROR_RSP PDU with the Error Code parameter set to *Database Out Of Sync* (0x12), it shall consider its attribute cache invalid and shall not make use of the cached information until it has performed service discovery or obtained the changed database definitions using an out-of-band mechanism.

### 2.5.3 Attribute grouping

GATT defines the grouping of attributes for three attribute types: «Primary Service», «Secondary Service» and «Characteristic». A group begins with a declaration, and ends as defined in Section 3.1 for services and Section 3.3 for characteristics. Not all of the grouping attributes can be used in the ATT_READ_BY_GROUP_TYPE_REQ PDU. The «Primary Service» and «Secondary Service» grouping types may be used in the ATT_READ_BY_GROUP_TYPE_REQ PDU. The «Characteristic» grouping type shall not be used in the ATT_READ_BY_GROUP_TYPE_REQ PDU.

### 2.5.4 UUIDs

All 16-bit UUIDs shall be contained in exactly 2 octets. All 128-bit UUIDs shall be contained in exactly 16 octets.

All 32-bit UUIDs shall be converted to 128-bit UUIDs when the UUID is contained in an ATT PDU. See [Vol 3] Part B, Section 2.5.1 for the method of conversion.

## 2.6 GATT Profile hierarchy

### 2.6.1 Overview

The GATT Profile specifies the structure in which profile data is exchanged. This structure defines basic elements such as services and characteristics, used in a profile. All of the elements are contained by Attributes. Attributes used in the Attribute Protocol are containers that carry this profile data.

The top level of the hierarchy is a profile. A profile is composed of one or more services necessary to fulfill a use case. A service is composed of characteristics or inclusions of other services. Each characteristic contains a value and may contain information about the value. The service and characteristic and the components of the characteristic (i.e.

value and descriptors) contain the profile data and are all stored in Attributes on the server.

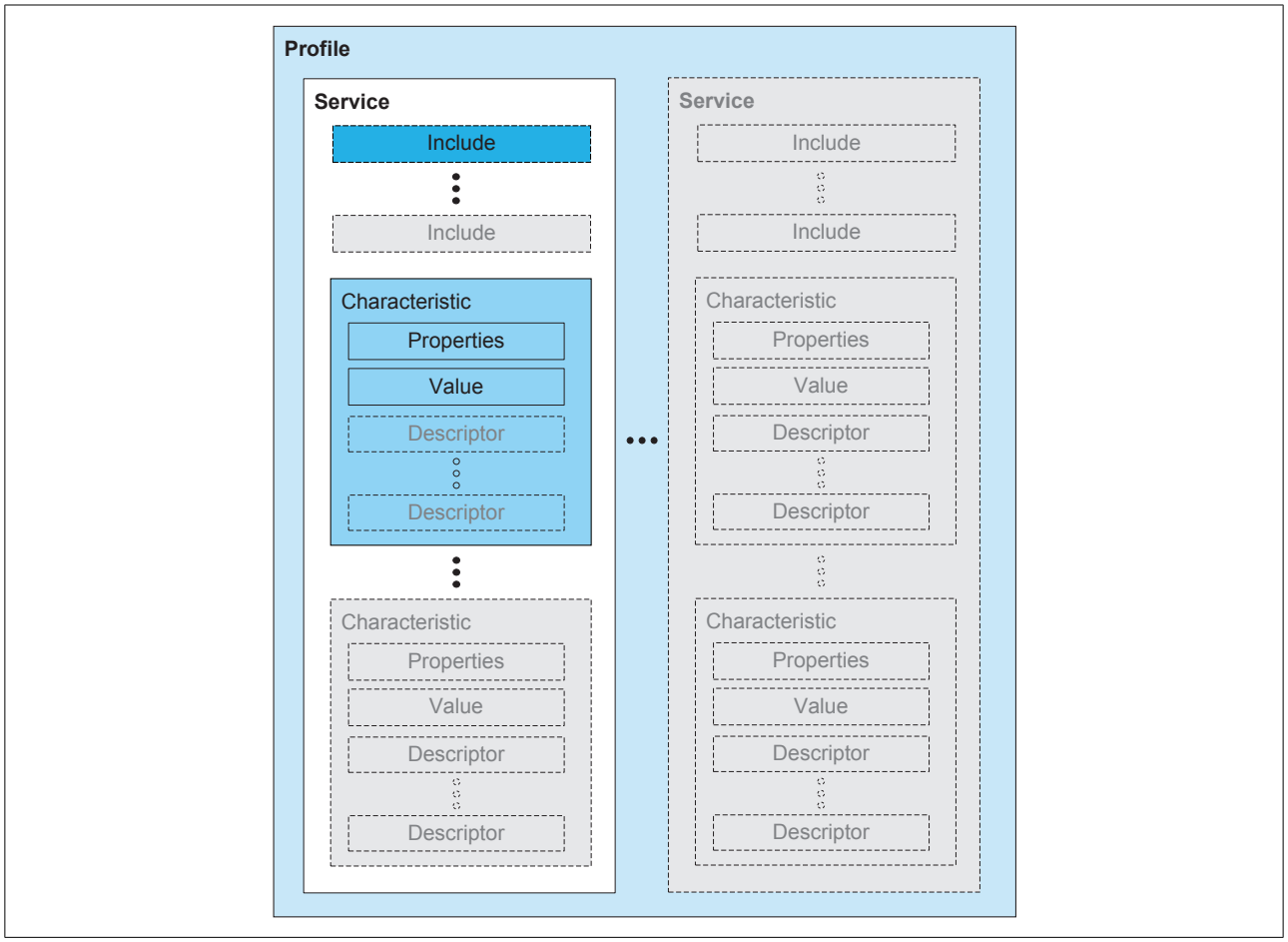



*Figure 2.8: GATT Profile hierarchy*

### 2.6.2 Service

A service is a collection of data and associated behaviors to accomplish a particular function or feature. In GATT, a service is defined by its service definition. A service definition may contain included services, mandatory characteristics, and optional characteristics.

To maintain backward compatibility with earlier clients, later versions of a service definition can only add new included services or optional characteristics. Later versions of a service definition are forbidden from changing behaviors from previous versions of the service definition.

There are two types of services: primary service and secondary service. A primary service is a service that exposes functionality of this device. A primary service can be included by another service. Primary services can be discovered using Primary Service

Discovery procedures. A secondary service is a service that should only be included from a primary service or another secondary service or other higher layer specification. A secondary service is only relevant in the context of the entity that includes it.

The determination of whether a service is either a primary or secondary service can be mandated by a higher layer specification.

Note: There is no procedure for discovering secondary services.

Services may be used in one or more higher layer specifications to fulfill a particular use case.

The service definition is described in Section 3.1.

### 2.6.3 Included services

An included service is a method to reference another service definition existing on the server into the service being defined. To include another service, an include definition is used at the beginning of the service definition. When a service definition uses an include definition to include a service, the entire included service definition becomes part of the new service definition. This includes all the included services and characteristics of the included service. The included service still exists as an independent service. A service that is included by another service shall not be changed by the act of inclusion or by the including service. There are no limits to the number of include definitions or the depth of nested includes in a service definition.

The include definition is described in Section 3.2.

### 2.6.4 Characteristic

A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. In GATT, a characteristic is defined by its characteristic definition. A characteristic definition contains a characteristic declaration, characteristic properties, and a value and may contain descriptors that describe the value or permit configuration of the server with respect to the characteristic.

The characteristic definition is described in Section 3.3.

## 2.7 Configured Broadcast

For LE physical links, Configured Broadcast is a method for a client to indicate to a server which *Characteristic Value* shall be broadcast in the advertising data when the server is executing the Broadcast mode procedure. For BR/EDR physical links, Configured Broadcast is not supported.

To configure a *Characteristic Value* to be broadcast by the server when in Broadcast mode, the client sets the broadcast configuration bit described in Section 3.3.3.4. The frequency of the broadcast is part of the service behavior definition. The data shall be broadcast as part of the Service Data Advertising Data type as defined in Section 1.11 of [3]. If multiple characteristics can simultaneously be enabled for broadcast, the service specification defines how the characteristics are to be formatted in the service data which follows the service UUID in the Service Data Advertising Data type payload.

# 3 SERVICE INTEROPERABILITY REQUIREMENTS

## 3.1 Service definition

A service definition shall contain a service declaration and may contain include definitions and characteristic definitions. The service definition ends before the next service declaration or after the maximum *Attribute Handle* is reached. Service definitions appear on the server in an order based on *Attribute Handle*.

All include definitions and characteristic definitions contained within the service definition are considered to be part of the service. All include definitions shall immediately follow the service declaration and precede any characteristic definitions. A service definition may have zero or more include definitions. All characteristic definitions shall be immediately following the last include definition or, in the event of no include definitions, immediately following the service declaration. A service definition may have zero or more characteristic definitions. There is no upper limit for include or characteristic definitions.

A service declaration is an Attribute with the *Attribute Type* set to the UUID for «Primary Service» or «Secondary Service». The *Attribute Value* shall be the 16-bit Bluetooth UUID or 128-bit UUID for the service, known as the service UUID. A client shall support the use of both 16-bit and 128-bit UUIDs. A client may ignore any service definition with an unknown service UUID. An unknown service UUID is a UUID for an unsupported service. The *Attribute Permissions* shall be read-only and shall not require authentication or authorization.

When multiple services exist, services definitions with service declarations using 16-bit Bluetooth UUID should be grouped together (i.e. listed sequentially) and services definitions with service declarations using 128-bit UUID should be grouped together.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permission |
|---|---|---|---|
| 0xNNNN | 0x2800 – UUID for «Primary Service» OR 0x2801 for «Secondary Service» | 16-bit Bluetooth UUID or 128-bit UUID for Service | Read Only,<br>No Authentication,<br>No Authorization |

*Table 3.1: Service declaration*

A device or higher level specification may have multiple service definitions and may have multiple service definitions with the same service UUID.

All Attributes on a server shall either contain a service declaration or exist within a service definition.

Service definitions contained in a server may appear in any order; a client shall not assume the order of service definitions on a server.

## 3.2 Include definition

An include definition shall contain only one include declaration.

The include declaration is an Attribute with the *Attribute Type* set to the UUID for «Include». The *Attribute Value* shall be set to the included service *Attribute Handle*, the End Group Handle, and the *service UUID*. The Service UUID shall only be present when the UUID is a 16-bit Bluetooth UUID.The *Attribute Permissions* shall be read only and not require authentication or authorization.

| Attribute Handle | Attribute Type | Attribute Value | | | Attribute Permission |
|---|---|---|---|---|---|
| 0xNNNN | 0x2802 – UUID for «Include» | Included Service Attribute Handle | End Group Handle | Service UUID | Read Only,<br>No Authentication,<br>No Authorization |

*Table 3.2: Include declaration*

A server shall not contain a service definition with an include definition to another service that includes the original service. This applies to each of the services the included definition references. This is referred to as a circular reference.

If the client detects a circular reference or detects nested include declarations to a greater level than it expects, it should terminate or stop using the ATT bearer.

## 3.3 Characteristic definition

A characteristic definition shall contain a characteristic declaration, a Characteristic Value declaration and may contain characteristic descriptor declarations. A characteristic definition ends at the start of the next characteristic declaration or service declaration or after the maximum *Attribute Handle*. Characteristic definitions appear on the server within a service definition in an order based on *Attribute Handle*.

Each declaration above is contained in a separate Attribute. The two required declarations are the characteristic declaration and the Characteristic Value declaration. The Characteristic Value declaration shall exist immediately following the characteristic declaration. Any optional characteristic descriptor declarations are placed after the Characteristic Value declaration. The order of the optional characteristic descriptor declarations is not significant.

A characteristic definition may be defined to concatenate several Characteristic Values into a single aggregated Characteristic Value. This may be used to optimize read

and writes of multiple Characteristic Values through the reading and writing of a single aggregated Characteristic Value. This type of characteristic definition is the same as a normal characteristic definition. The characteristic declaration shall use a characteristic UUID that is unique to the aggregated characteristic definition. The aggregated characteristic definition may also contain a characteristic aggregate format descriptor that describes the display format of the aggregated Characteristic Value.

### 3.3.1 Characteristic declaration

A characteristic declaration is an Attribute with the Attribute Type set to the UUID for «Characteristic» and *Attribute Value* set to the Characteristic Properties, Characteristic Value *Attribute Handle* and Characteristic UUID. The Attribute Permissions shall be readable and not require authentication or authorization.

If the server changes any characteristic declaration *Attribute Value* while the server has a trusted relationship with any client, then it shall send each client a Service Changed Indication indicating a change in the service holding the Characteristic Declaration (see Section 7.1).

| Attribute Handle | Attribute Types | Attribute Value | | | Attribute Permissions |
|---|---|---|---|---|---|
| 0xNNNN | 0x2803–UUID for «Characteristic» | Characteristic Properties | Characteristic Value Attribute Handle | Characteristic UUID | Read Only,<br>No Authentication,<br>No Authorization |

*Table 3.3: Characteristic declaration*

The *Attribute Value* of a characteristic declaration is read only.

| Attribute Value | Size | Description |
|---|---|---|
| **Characteristic Properties** | 1 octets | Bit field of characteristic properties |
| **Characteristic Value Handle** | 2 octets | Handle of the Attribute containing the value of this characteristic |
| **Characteristic UUID** | 2 or 16 octets | 16-bit Bluetooth UUID or 128-bit UUID for Characteristic Value |

*Table 3.4: Attribute Value field in characteristic declaration*

A service may have multiple characteristic definitions with the same Characteristic UUID.

Within a service definition, some characteristics may be mandatory and those characteristics shall be located after the include declarations and before any optional characteristics within the service definition. A client shall not assume any order of those characteristics that are mandatory or any order of those characteristics

that are optional within a service definition. Whenever possible and within the requirements stated earlier, characteristics definitions with characteristic declarations using 16-bit Bluetooth UUIDs should be grouped together (i.e. listed sequentially) and characteristics definitions with characteristic declarations using 128-bit UUIDs should be grouped together.

#### 3.3.1.1 Characteristic Properties

The Characteristic Properties bit field determines how the Characteristic Value can be used, or how the characteristic descriptors (see Section 3.3.3) can be accessed. If the bits defined in Table 3.5 are set, the action described is permitted. Multiple characteristic properties can be set.

These bits shall be set according to the procedures allowed for this characteristic, as defined by higher layer specifications, without regard to security requirements.

| Properties | Value | Description |
|---|---|---|
| Broadcast | 0x01 | If set, permits broadcasts of the Characteristic Value using Server Characteristic Configuration Descriptor. If set, the Server Characteristic Configuration Descriptor shall exist. |
| Read | 0x02 | If set, permits reads of the Characteristic Value using procedures defined in Section 4.8 |
| Write Without Response | 0x04 | If set, permit writes of the Characteristic Value without response using procedures defined in Section 4.9.1. |
| Write | 0x08 | If set, permits writes of the Characteristic Value with response using procedures defined in Section 4.9.3 or Section 4.9.4. |
| Notify | 0x10 | If set, permits notifications of a Characteristic Value without acknowledgment using the procedure defined in Section 4.10. If set, the Client Characteristic Configuration Descriptor shall exist. |
| Indicate | 0x20 | If set, permits indications of a Characteristic Value with acknowledgment using the procedure defined in Section 4.11. If set, the Client Characteristic Configuration Descriptor shall exist. |
| | 0x40 | *Previously used* |
| Extended Properties | 0x80 | If set, additional characteristic properties are defined in the Characteristic Extended Properties Descriptor defined in Section 3.3.3.1. If set, the Characteristic Extended Properties Descriptor shall exist. |

*Table 3.5: Characteristic Properties bit field*

#### 3.3.1.2 Characteristic Value Attribute Handle

The Characteristic Value *Attribute Handle* field is the *Attribute Handle* of the Attribute that contains the *Characteristic Value*.

#### 3.3.1.3 Characteristic UUID

The *Characteristic UUID* field is a 16-bit Bluetooth UUID or 128-bit UUID that describes the type of *Characteristic Value*. A client shall support the use of both 16-bit and 128-bit *Characteristic UUIDs*. A client may ignore any characteristic definition with an unknown *Characteristic UUID*. An unknown characteristic UUID is a UUID for an unsupported characteristic.

### 3.3.2 Characteristic Value declaration

The *Characteristic Value* declaration contains the value of the characteristic. It is the first Attribute after the characteristic declaration. All characteristic definitions shall have a *Characteristic Value* declaration.

A Characteristic Value declaration is an Attribute with the Attribute Type set to the 16-bit Bluetooth or 128-bit UUID for the Characteristic Value used in the characteristic declaration. The *Attribute Value* is set to the *Characteristic Value*.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0xUUUU – 16-bit Bluetooth UUID or 128-bit UUID for Characteristic UUID | Characteristic Value | Higher layer profile or implementation specific |

*Table 3.6: Characteristic Value declaration*

### 3.3.3 Characteristic descriptor declarations

Characteristic descriptors are used to contain related information about the *Characteristic Value*. The GATT profile defines a standard set of characteristic descriptors that can be used by higher layer profiles. Higher layer profiles may define additional characteristic descriptors that are profile specific. Each characteristic descriptor is identified by the characteristic descriptor UUID. A client shall support the use of both 16-bit and 128-bit characteristic descriptor UUIDs. A client may ignore any characteristic descriptor declaration with an unknown characteristic descriptor UUID. An unknown characteristic descriptor UUID is a UUID for an unsupported characteristic descriptor.

Characteristic descriptors if present within a characteristic definition shall follow the *Characteristic Value* declaration. The characteristic descriptor declaration may appear in any order within the characteristic definition. The client shall not assume the order in which a characteristic descriptor declaration appears in a characteristic definition following the *Characteristic Value* declaration.

Characteristic descriptor declaration permissions may be defined by a higher layer profile or are implementation specific. A client shall not assume all characteristic descriptor declarations are readable.

#### 3.3.3.1 Characteristic Extended Properties

The *Characteristic Extended Properties* declaration is a descriptor that defines additional *Characteristic Properties*. If the *Extended Properties* bit of the *Characteristic Properties* is set then this characteristic descriptor shall exist. The characteristic descriptor may occur in any position within the characteristic definition after the Characteristic Value. Only one *Characteristic Extended Properties* declaration shall exist in a characteristic definition.

The characteristic descriptor is contained in an Attribute and the *Attribute Type* shall be set to the UUID for «Characteristic Extended Properties» and the *Attribute Value* shall be two octets in length and shall contain the *Characteristic Extended Properties Bit Field*.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0x2900 – UUID for «Characteristic Extended Properties» | Characteristic Extended Properties Bit Field | Read Only,<br>No Authentication,<br>No Authorization |

*Table 3.7: Characteristic Extended Properties declaration*

The *Characteristic Extended Properties* bit field describes additional properties on how the Characteristic Value can be used, or how the characteristic descriptors (see Section 3.3.3.3) can be accessed. If the bits defined in Table 3.8 are set, the action described is permitted. Multiple additional properties can be set.

| Bit Number | Property | Description |
|---|---|---|
| 0 | Reliable Write | If set, permits reliable writes of the Characteristic Value using the procedure defined in Section 4.9.5 |
| 1 | Writable Auxiliaries | If set, permits writes to the characteristic descriptor defined in Section 3.3.3.2 |
| All other bits | | Reserved for future use |

*Table 3.8: Characteristic Extended Properties bit field*

#### 3.3.3.2 Characteristic User Description

The *Characteristic User Description* declaration is an optional characteristic descriptor that defines a UTF-8 string of variable size that is a user textual description of the *Characteristic Value*. If the *Writable Auxiliaries* bit of the *Characteristic Extended*

*Properties* is set then this characteristic descriptor can be written. The characteristic descriptor may occur in any position within the characteristic definition after the *Characteristic Value*. Only one *Characteristic User Description* declaration shall exist in a characteristic definition.

The characteristic descriptor is contained in an Attribute and the *Attribute Type* shall be set to the UUID for «Characteristic User Description» and the *Attribute Value* shall be set to the characteristic user description UTF-8 string.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0x2901 – UUID for «Characteristic User Description» | Characteristic User Description UTF-8 String | Higher layer profile or implementation specific |

*Table 3.9: Characteristic User Description declaration*

#### 3.3.3.3 Client Characteristic Configuration

The *Client Characteristic Configuration* declaration is an optional characteristic descriptor that defines how the characteristic may be configured by a specific client. The Client Characteristic Configuration descriptor value shall be persistent across connections for bonded devices. The Client Characteristic Configuration descriptor value shall be set to the default value at each connection with non-bonded devices.The characteristic descriptor value is a bit field. When a bit is set, that action shall be enabled, otherwise it will not be used. The *Client Characteristic Configuration* descriptor may occur in any position within the characteristic definition after the Characteristic Value. Only one *Client Characteristic Configuration* declaration shall exist in a characteristic definition.

A client may write this configuration descriptor to control the configuration of this characteristic on the server for the client. Each client has its own instantiation of the *Client Characteristic Configuration*. Reads of the *Client Characteristic Configuration* only shows the configuration for that client and writes only affect the configuration of that client. Authentication and authorization may be required by the server to write the configuration descriptor. The *Client Characteristic Configuration* declaration shall be readable and writable.

The characteristic descriptor is contained in an Attribute. The *Attribute Type* shall be set to the UUID for «Client Characteristic Configuration». The *Attribute Value* shall be two octets in length and shall be set to the characteristic descriptor value.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0x2902 – UUID for «Client Characteristic Configuration» | Characteristic Configuration Bits | Readable with no authentication or authorization.<br>Writable with authentication and authorization defined by a higher layer specification or is implementation specific. |

*Table 3.10: Client Characteristic Configuration declaration*

The following Client Characteristic Configuration bits are defined:

| Bit Number | Configuration | Description |
|---|---|---|
| 0 | Notification | The Characteristic Value shall be notified. This value shall only be set if the characteristic's properties have the notify bit set. |
| 1 | Indication | The Characteristic Value shall be indicated. This value shall only be set if the characteristic's properties have the indicate bit set. |
| All other bits | | Reserved for future use. |

*Table 3.11: Client Characteristic Configuration bit field definition*

If both the Notification and Indication bits are set, then the server shall use the notification procedure (see Section 4.10) when an acknowledgment is not required by a higher-layer specification or shall use the indication procedure (see Section 4.11) when an acknowledgment is required. The server should not use both procedures to send the same characteristic value.

The server may reject a request to set both the Notification and Indication bits for the same characteristic.

The default value for the *Client Characteristic Configuration* descriptor value shall be 0x0000.

Between a client and a server there shall be a single Client Characteristic Configuration Descriptor irrespective of the number of ATT bearers between them.

#### 3.3.3.4 Server Characteristic Configuration

The *Server Characteristic Configuration* declaration is an optional characteristic descriptor that defines how the characteristic may be configured for the server. The characteristic descriptor value is a bit field. When a bit is set, that action shall be enabled, otherwise it will not be used. The *Server Characteristic Configuration* descriptor may occur in any position within the characteristic definition after the *Characteristic Value*. Only one *Server Characteristic Configuration* declaration shall exist in a characteristic definition. The *Server Characteristic Configuration* declaration shall be readable and writable.

A client may write this configuration descriptor to control the configuration of this characteristic on the server for all clients. There is a single instantiation of the *Server Characteristic Configuration* for all clients. Reads of the *Server Characteristic Configuration* shows the configuration all clients and writes affect the configuration for all clients. Authentication and authorization may be required by the server to write the configuration descriptor.

The characteristic descriptor is contained in an Attribute. The *Attribute Type* shall be set to the UUID for «Server Characteristic Configuration». The *Attribute Value* shall be two octets in length and shall be set to the characteristic descriptor value.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0x2903 – UUID for «Server Characteristic Configuration» | Characteristic Configuration Bits | Readable with no authentication or authorization.<br>Writable with authentication and authorization defined by a higher layer specification or is implementation specific. |

*Table 3.12: Server Characteristic Configuration declaration*

The following *Server Characteristic Configuration* bits are defined:

| Bit Number | Configuration | Description |
|---|---|---|
| 0 | Broadcast | The Characteristic Value shall be broadcast when the server is in the broadcast procedure if advertising data resources are available. This value can only be set if the characteristic's properties have the broadcast bit set. |
| All other bits | | Reserved for future use. |

*Table 3.13: Server Characteristic Configuration bit field definition*

#### 3.3.3.5 Characteristic Presentation Format

The *Characteristic Presentation Format* declaration is an optional characteristic descriptor that defines the format of the *Characteristic Value*. The characteristic descriptor may occur in any position within the characteristic definition after the *Characteristic Value*. If more than one *Characteristic Presentation Format* declaration exists in a characteristic definition, then a *Characteristic Aggregate Format* declaration shall exist as part of the characteristic definition.

The characteristic presentation format value is composed of five parts: format, exponent, unit, name space, and description.

The characteristic descriptor is contained in an Attribute. The *Attribute Type* shall be set to the UUID for «Characteristic Presentation Format». The *Attribute Value* shall be set to the characteristic descriptor value.

| Attribute Handle | Attribute Type | Attribute Value | | | | | Attribute Permissions |
|---|---|---|---|---|---|---|---|
| 0xNNNN | 0x2904 – UUID for «Characteristic Presentation Format» | Format | Exponent | Unit | Name Space | Description | Read only<br>No Authentication,<br>No Authorization |

*Table 3.14: Characteristic Presentation Format declaration*

The definition of the Characteristic Presentation Format descriptor Attribute Value field is the following.

| Field Name | Value Size | Description |
|---|---|---|
| Format | 1 octet | Format of the value of this characteristic as defined in [1]. |
| Exponent | 1 octet | Exponent field to determine how the value of this characteristic is further formatted. |
| Unit | 2 octets | The unit of this characteristic as defined in [1] |
| Name Space | 1 octet | The name space of the description as defined in [1] |
| Description | 2 octets | The description of this characteristic as defined in a higher layer profile. |

*Table 3.15: Characteristic Presentation Format value definition*

#### 3.3.3.5.1 Bit ordering

The bit ordering used for the Characteristic Presentation Format descriptor shall be little-endian.

#### 3.3.3.5.2 Format

The format field determines how a single value contained in the *Characteristic Value* is formatted. The values of this field are defined in Assigned Numbers [1].

#### 3.3.3.5.3 Exponent

The exponent field is used with integer data types to indicate that the actual value being represented differs from the value stored in the characteristic by a power of 10. The exponent field is only used on integer format types; this field is RFU for all other format types. The exponent field has type sint8.

$$\text{actual value} = \textit{Characteristic Value} \times 10^{\text{Exponent}}$$

As can be seen in the above equation, the actual value is a combination of the Characteristic Value and the value 10 to the power Exponent. This is sometimes known as a fixed point number.

For example, if the Exponent is 2 and the *Characteristic Value* is 23, the actual value would be 2300.

For example, if the Exponent is -3 and the *Characteristic Value* is 3892, the actual value would be 3.892.

#### 3.3.3.5.4 Unit

The Unit is a UUID as defined in Assigned Numbers [1].

#### 3.3.3.5.5 Name Space

The Name Space field is used to identify the organization, as defined in Assigned Numbers [1], that is responsible for defining the enumerations for the description field.

#### 3.3.3.5.6 Description

The Description is an enumerated value as defined in Assigned Numbers [1] from the organization identified by the Name Space field. The Description is used to distinguish between different instances of the same characteristic; for example, if a sound system has several speakers each with a "volume" characteristic, the Description would identify which speaker a given characteristic refers to.

### 3.3.3.6 Characteristic Aggregate Format

The *Characteristic Aggregate Format* declaration is an optional characteristic descriptor that defines the format of an aggregated *Characteristic Value*.

The characteristic descriptor may occur in any position within the characteristic definition after the *Characteristic Value*. Only one *Characteristic Aggregate Format* declaration shall exist in a characteristic definition.

The *Characteristic Aggregate Format* value is composed of a list of Attribute Handles of *Characteristic Presentation Format* declarations, where each *Attribute Handle* points to a *Characteristic Presentation Format* declaration.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permissions |
|---|---|---|---|
| 0xNNNN | 0x2905 – UUID for «Characteristic Aggregate Format» | List of *Attribute Handles* for the Characteristic Presentation Format Declarations | Read only<br>No authentication<br>No authorization |

*Table 3.16: Characteristic Aggregate Format declaration*

The *List of Attribute Handles* is the concatenation of multiple 16-bit *Attribute Handle* values into a single *Attribute Value*. The list shall contain at least two *Attribute Handles* for *Characteristic Presentation Format* declarations. The Characteristic Value shall be decomposed by each of the *Characteristic Presentation Format* declarations pointed to by the *Attribute Handles*. The order of the *Attribute Handles* in the list is significant.

If more than one *Characteristic Presentation Format* declaration exists in a characteristic definition, there shall also be one *Characteristic Aggregate Format* declaration. The *Characteristic Aggregate Format* declaration shall include each *Characteristic Presentation Format* declaration in the characteristic definition in the list of *Attribute Handles*. *Characteristic Presentation Format* declarations from other characteristic definitions may also be used.

A *Characteristic Aggregate Format* declaration may exist without a *Characteristic Presentation Format* declaration existing in the characteristic definition. The *Characteristic Aggregate Format* declaration may use *Characteristic Presentation Format* declarations from other characteristic definitions.

## 3.4 Summary of GATT Profile attribute types

Table 3.17 summarizes the attribute types defined by the GATT Profile.

| Attribute Type | UUID | Description |
|---|---|---|
| «Primary Service» | 0x2800 | Primary Service Declaration |
| «Secondary Service» | 0x2801 | Secondary Service Declaration |
| «Include» | 0x2802 | Include Declaration |
| «Characteristic» | 0x2803 | Characteristic Declaration |
| «Characteristic Extended Properties» | 0x2900 | Characteristic Extended Properties |
| «Characteristic User Description» | 0x2901 | Characteristic User Description Descriptor |
| «Client Characteristic Configuration» | 0x2902 | Client Characteristic Configuration Descriptor |
| «Server Characteristic Configuration» | 0x2903 | Server Characteristic Configuration Descriptor |
| «Characteristic Presentation Format» | 0x2904 | Characteristic Presentation Format Descriptor |
| «Characteristic Aggregate Format» | 0x2905 | Characteristic Aggregate Format Descriptor |

*Table 3.17: Summary of GATT Profile attribute types*

# 4 GATT FEATURE REQUIREMENTS

## 4.1 Overview

There are 11 features defined in the GATT Profile:

1. Server Configuration
2. Primary Service Discovery
3. Relationship Discovery
4. Characteristic Discovery
5. Characteristic Descriptor Discovery
6. Reading a Characteristic Value
7. Writing a Characteristic Value
8. Notification of a Characteristic Value
9. Indication of a Characteristic Value
10. Reading a Characteristic Descriptor
11. Writing a Characteristic Descriptor

Each of the features is mapped to procedures and sub-procedures. These procedures and sub-procedures describe how the Attribute Protocol is used to accomplish the corresponding feature.

## 4.2 Feature support and procedure mapping

Table 4.1 maps each feature to the procedures used for that feature, and indicates whether the procedure is optional or mandatory for that feature. The procedures are described in the referenced section.

If an ATT PDU is supported on any ATT bearer, then it shall be supported on all supported ATT bearers with the following exception:

- The Exchange MTU sub-procedure shall only be supported on the LE Fixed Channel Unenhanced ATT bearer.

| Feature | Sub-Procedure | Ref. | Support in Client | Support in Server |
|---|---|---|---|---|
| Server Configuration | Exchange MTU | 4.3.1 | O | O |
| Primary Service Discovery | Discover All Primary Services | 4.4.1 | O | M |
| | Discover Primary Service By Service UUID | 4.4.2 | O | M |
| Relationship Discovery | Find Included Services | 4.5.1 | O | M |
| Characteristic Discovery | Discover All Characteristics of a Service | 4.6.1 | O | M |
| | Discover Characteristics by UUID | 4.6.2 | O | M |
| Characteristic Descriptor Discovery | Discover All Characteristic Descriptors | 4.7.1 | O | M |
| Characteristic Value Read | Read Characteristic Value | 4.8.1 | O | M |
| | Read Using Characteristic UUID | 4.8.2 | O | M |
| | Read Long Characteristic Value | 4.8.3 | O | C.4 |
| | Read Multiple Characteristic Values | 4.8.4 | O | O |
| | Read Multiple Variable Length Characteristic Values | 4.8.5 | O | C.4 |
| Characteristic Value Write | Write Without Response | 4.9.1 | O | C.1 |
| | Write Characteristic Value | 4.9.3 | O | C.2 |
| | Write Long Characteristic Value | 4.9.4 | O | C.4 |
| | Characteristic Value Reliable Writes | 4.9.5 | O | O |
| Characteristic Value Notification | Single Notification | 4.10.1 | C.4 | C.4 |
| | Multiple Variable Length Notifications | 4.10.2 | C.4 | C.4 |
| Characteristic Value Indication | Indication | 4.11.1 | M | C.3 |
| Characteristic Descriptor Value Read | Read Characteristic Descriptor | 4.12.1 | O | C.4 |
| | Read Long Characteristic Descriptor | 4.12.2 | O | C.4 |

| Feature | Sub-Procedure | Ref. | Support in Client | Support in Server |
|---|---|---|---|---|
| Characteristic Descriptor Value Write | Write Characteristic Descriptor | 4.12.3 | O | C.4 |
| | Write Long Characteristic Descriptor | 4.12.4 | O | O |

C.1: Write Without Response is mandatory if Enhanced ATT Bearers are supported, otherwise optional.

C.2: Write Characteristic Value is mandatory if Write Long Characteristic Value or Enhanced ATT Bearers are supported, otherwise optional.

C.3: If *Service Changed Characteristic* is present, this feature is mandatory, otherwise optional.

C.4: If Enhanced ATT Bearers are supported then this feature is mandatory, otherwise optional.

*Table 4.1: GATT feature mapping to procedures*

## 4.3 Server configuration

This procedure is used by the client to configure the Attribute Protocol. This procedure has only one sub-procedure used to set the MTU sizes.

### 4.3.1 Exchange MTU

This sub-procedure is used by the client to set the ATT_MTU to the maximum possible value that can be supported by both devices when the client supports a value greater than the default ATT_MTU for the Attribute Protocol. This sub-procedure shall only be initiated once during a connection.

This sub-procedure shall not be used on a BR/EDR physical link since the MTU size is negotiated using L2CAP channel configuration procedures.

The ATT_EXCHANGE_MTU_REQ PDU is used by this sub-procedure. The Client Rx MTU parameter shall be set to the maximum MTU that this client can receive.

Two possible responses can be sent from the server for the ATT_EXCHANGE_MTU_REQ PDU: ATT_EXCHANGE_MTU_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The server shall respond to this message with an ATT_EXCHANGE_MTU_RSP PDU with the Server Rx MTU parameter set to the maximum MTU that this server can receive.

If the ATT_ERROR_RSP PDU is sent by the server with the Error Code parameter set to *Request Not Supported* (0x06), the *Attribute Opcode* is not supported and the default MTU shall be used.

Once the messages have been exchanged, the ATT_MTU shall be set to the minimum of the Client Rx MTU and Server Rx MTU values.

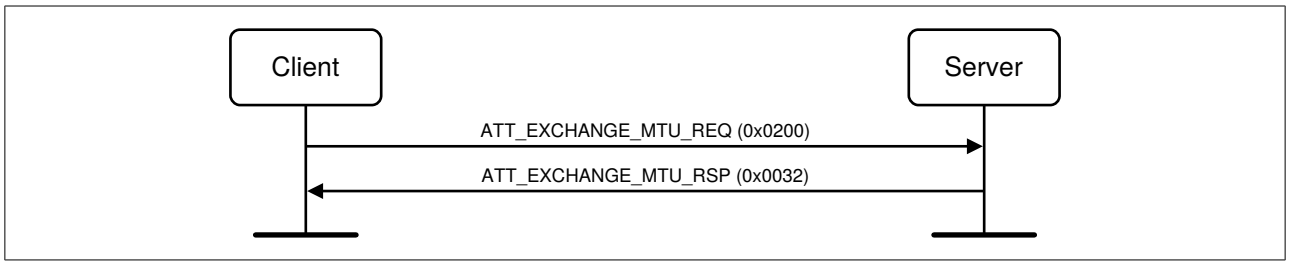



*Figure 4.1: Exchange MTU*

For example, in Figure 4.1, based on the exchanged ATT_MTU values, the ATT_MTU would be 0x0032.

## 4.4 Primary Service Discovery

This procedure is used by a client to discover primary services on a server. Once the primary services are discovered, additional information about the primary services can be accessed using other procedures, including characteristic discovery and relationship discovery to find other related primary and secondary services.

There are two sub-procedures that can be used for primary service discovery: Discover All Primary Services and Discover Primary Service by Service UUID.

### 4.4.1 Discover All Primary Services

This sub-procedure is used by a client to discover all the primary services on a server.

The ATT_READ_BY_GROUP_TYPE_REQ PDU shall be used with the Attribute Type parameter set to the UUID for «Primary Service». The *Starting Handle* shall be set to 0x0001 and the *Ending Handle* shall be set to 0xFFFF.

Two possible responses can be sent from the server for the ATT_READ_BY_GROUP_TYPE_REQ PDU: ATT_READ_BY_GROUP_TYPE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_READ_BY_GROUP_TYPE_RSP PDU returns a list of *Attribute Handle*, *End Group Handle*, and *Attribute Value* tuples corresponding to the services supported by the server. Each *Attribute Value* contained in the response is the Service UUID of a service supported by the server. The *Attribute Handle* is the handle for the service declaration. The *End Group Handle* is the handle of the last attribute within the service definition. The *End Group Handle* of the last service in a device can

be 0xFFFF. The ATT_READ_BY_GROUP_TYPE_REQ PDU shall be issued again with the *Starting Handle* set to one greater than the last *End Group Handle* in the ATT_READ_BY_GROUP_TYPE_RSP PDU.

This sub-procedure is complete when the ATT_ERROR_RSP PDU is received and the Error Code parameter is set to *Attribute Not Found* (0x0A) or when the *End Group Handle* in the *Read by Type Group Response* is 0xFFFF.

The sub-procedure may end early if a desired primary service is found prior to discovering all the primary services on the server.

The service declaration described in Section 3.1 specifies that the service declaration is readable and requires no authentication or authorization, therefore insufficient authentication or read not permitted errors shall not occur.

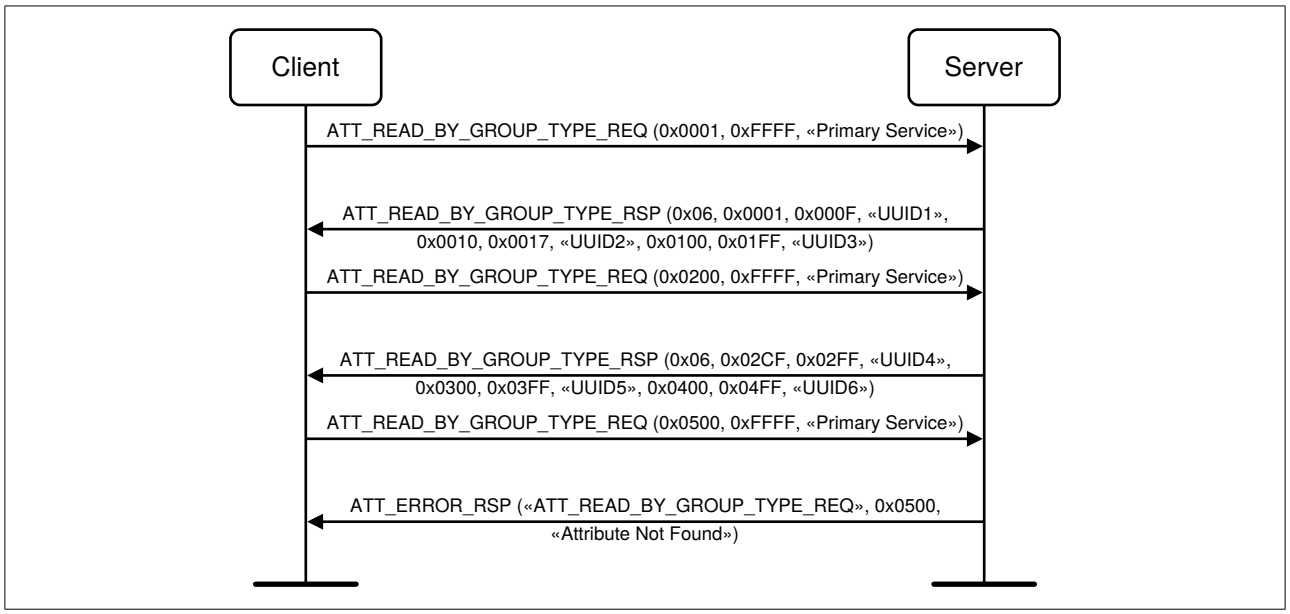



*Figure 4.2: Discover All Primary Services example*

### 4.4.2 Discover Primary Service by Service UUID

This sub-procedure is used by a client to discover a specific primary service on a server when only the Service UUID is known. The specific primary service may exist multiple times on a server. The primary service being discovered is identified by the service UUID.

The ATT_FIND_BY_TYPE_VALUE_REQ PDU shall be used with the Attribute Type parameter set to the UUID for «Primary Service» and the *Attribute Value* set to the 16-bit Bluetooth UUID or 128-bit UUID for the specific primary service. The *Starting Handle* shall be set to 0x0001 and the *Ending Handle* shall be set to 0xFFFF.

Two possible responses can be sent from the server for the ATT_FIND_BY_TYPE_VALUE_REQ PDU: ATT_FIND_BY_TYPE_VALUE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_FIND_BY_TYPE_VALUE_RSP PDU returns a list of *Attribute Handle* ranges. The *Attribute Handle* range is the starting handle and the ending handle of the service definition. The *End Group Handle* of the last service in a device can be 0xFFFF. If the *Attribute Handle* range for the Service UUID being searched is returned and the End Found Handle is not 0xFFFF, the ATT_FIND_BY_TYPE_VALUE_REQ PDU may be issued again with the *Starting Handle* set to one greater than the last *Attribute Handle* range in the ATT_FIND_BY_TYPE_VALUE_RSP PDU.

This sub-procedure is complete when the ATT_ERROR_RSP PDU is received and the Error Code parameter is set to *Attribute Not Found* (0x0A) or when the *End Group Handle* in the ATT_FIND_BY_TYPE_VALUE_RSP PDU is 0xFFFF.

The sub-procedure may end early if a desired primary service is found prior to discovering all the primary services of the specified service UUID supported on the server.

The service declaration described in Section 3.1 specifies that the service declaration is readable and requires no authentication or authorization, therefore insufficient authentication or read not permitted errors shall not occur.

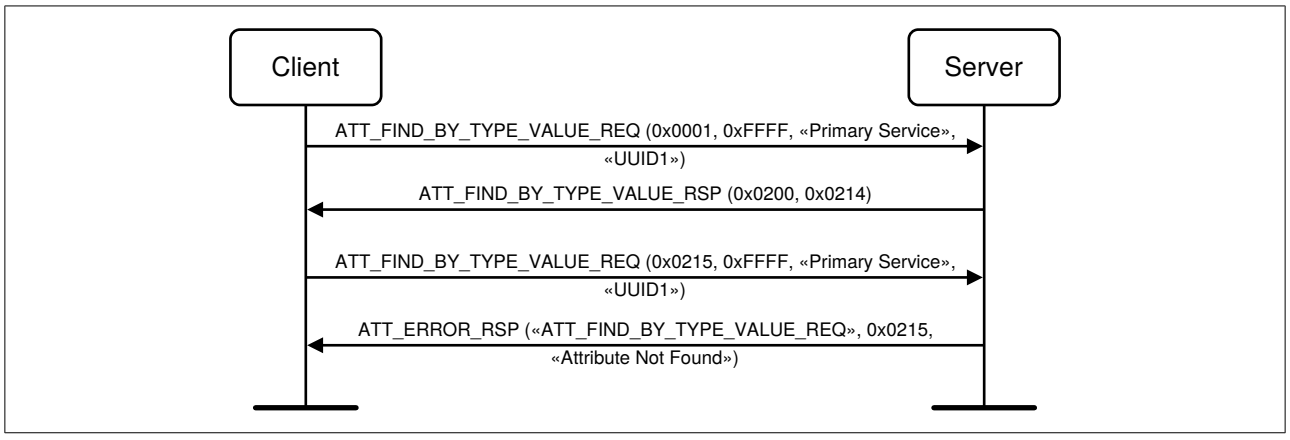



*Figure 4.3: Discover Primary Service by Service UUID example*

## 4.5 Relationship Discovery

This procedure is used by a client to discover service relationships to other services.

There is one sub-procedure that can be used for relationship discovery: Find Included Services.

### 4.5.1 Find Included Services

This sub-procedure is used by a client to find include service declarations within a service definition on a server. The service specified is identified by the service handle range.

The ATT_READ_BY_TYPE_REQ PDU shall be used with the *Attribute Type* parameter set to the UUID for «Include» The *Starting Handle* shall be set to the starting handle of the specified service and the *Ending Handle* shall be set to the ending handle of the specified service. The sub-procedure may end early if a desired included service is found prior to discovering all the included services of the specified service supported on the server.

Two possible responses can be sent from the server for the ATT_READ_BY_TYPE_REQ PDU: ATT_READ_BY_TYPE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_READ_BY_TYPE_RSP PDU returns a set of *Attribute Handle* and *Attribute Value* pairs corresponding to the included services in the service definition. Each *Attribute Value* contained in the response is composed of the *Attribute Handle* of the included service declaration and the *End Group Handle*. If the service UUID is a 16-bit Bluetooth UUID it is also returned in the response. The ATT_READ_BY_TYPE_REQ PDU shall be issued again with the *Starting Handle* set to one greater than the last *Attribute Handle* in the ATT_READ_BY_TYPE_RSP PDU.

The sub-procedure is complete when either the ATT_ERROR_RSP PDU is received with the Error Code parameter set to *Attribute Not Found* (0x0A) or the ATT_READ_BY_TYPE_RSP PDU has an *Attribute Handle* of the included service declaration that is equal to the *Ending Handle* of the request.

To get the included service UUID when the included service uses a 128-bit UUID, the ATT_READ_REQ PDU is used. The *Attribute Handle* for the ATT_READ_REQ PDU is the *Attribute Handle* of the included service.

The include declaration described in Section 3.2 specifies that the include declaration is readable and requires no authentication or authorization, therefore insufficient authentication or read not permitted errors shall not occur.

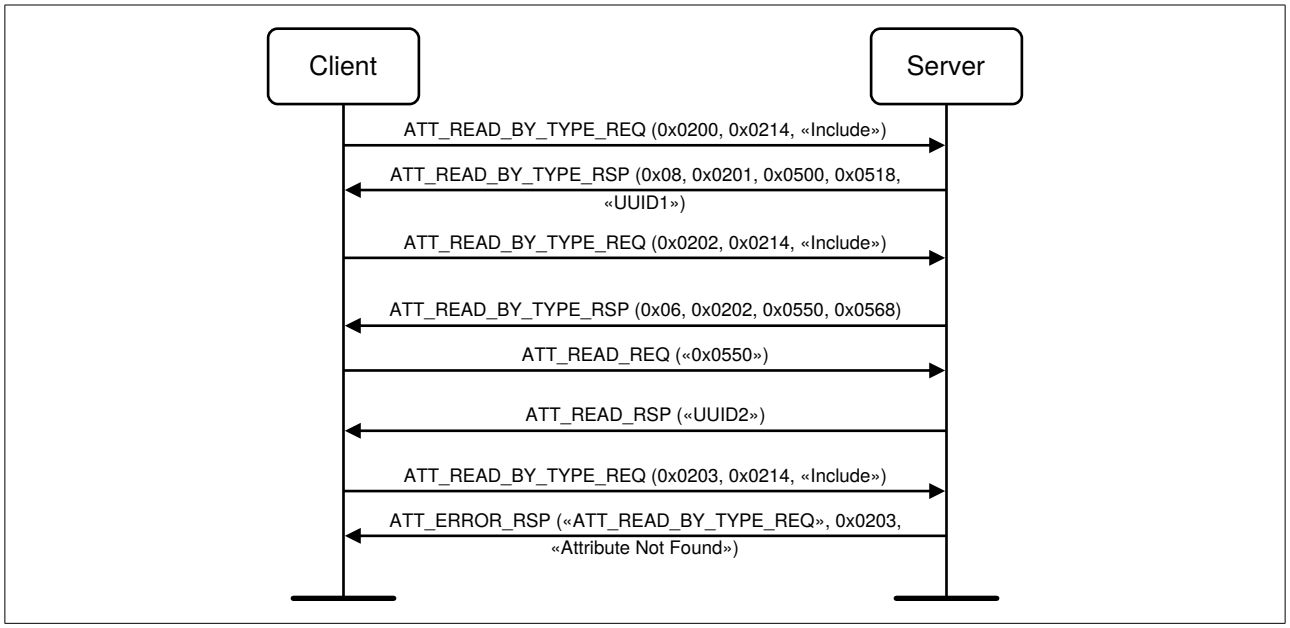



*Figure 4.4: Find Included Services example*

## 4.6 Characteristic discovery

This procedure is used by a client to discover service characteristics on a server. Once the characteristics are discovered additional information about the characteristics can be discovered or accessed using other procedures.

There are two sub-procedures that can be used for characteristic discovery: Discover All Characteristics of a Service and Discover Characteristics by UUID.

### 4.6.1 Discover All Characteristics of a Service

This sub-procedure is used by a client to find all the characteristic declarations within a service definition on a server when only the service handle range is known. The service specified is identified by the service handle range.

The ATT_READ_BY_TYPE_REQ PDU shall be used with the *Attribute Type* parameter set to the UUID for «Characteristic» The *Starting Handle* shall be set to starting handle of the specified service and the *Ending Handle* shall be set to the ending handle of the specified service.

Two possible responses can be sent from the server for the ATT_READ_BY_TYPE_REQ PDU: ATT_READ_BY_TYPE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_READ_BY_TYPE_RSP PDU returns a list of *Attribute Handle* and *Attribute Value* pairs corresponding to the characteristics in the service definition. The *Attribute*

*Handle* is the handle for the characteristic declaration. The *Attribute Value* is the Characteristic Properties, Characteristic Value Handle and Characteristic UUID. The ATT_READ_BY_TYPE_REQ PDU shall be issued again with the *Starting Handle* set to one greater than the last *Attribute Handle* in the ATT_READ_BY_TYPE_RSP PDU.

The sub-procedure is complete when the ATT_ERROR_RSP PDU is received and the Error Code parameter is set to *Attribute Not Found* (0x0A) or the ATT_READ_BY_TYPE_RSP PDU has an *Attribute Handle* that is equal to the *Ending Handle* of the request.

The sub-procedure may end early if a desired characteristic is found prior to discovering all the characteristics of the specified service supported on the server.

Note: Insufficient authentication and read not permitted errors must not occur because Section 3.3.1 specifies that characteristic declarations are readable and require no authentication or authorization.

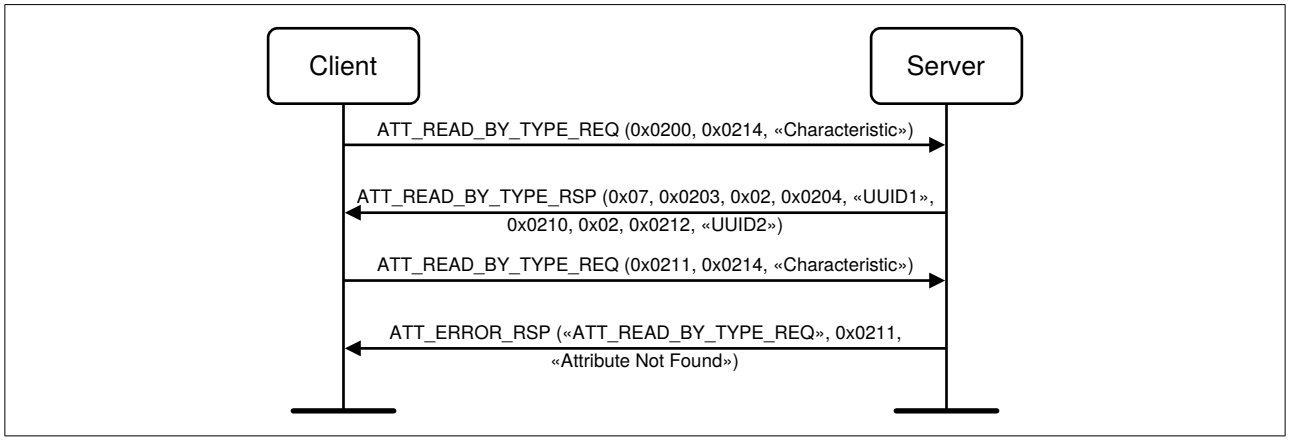



*Figure 4.5: Discover All Characteristics of a Service example*

Note: In this example «UUID1» and «UUID2» are 16 bits (2 octets).

If they were 128 bits (16 octets) then the ATT_READ_BY_TYPE_RSP PDU data would instead be:

0x15, 0x0203, 0x02, 0x0204, «UUID1», 0x0210, 0x02, 0x0212, «UUID2»

### 4.6.2 Discover Characteristics by UUID

This sub-procedure is used by a client to discover service characteristics on a server when only the service handle ranges are known and the characteristic UUID is known. The specific service may exist multiple times on a server. The characteristics being discovered are identified by the characteristic UUID.

The ATT_READ_BY_TYPE_REQ PDU is used to perform the beginning of the sub-procedure. The *Attribute Type* is set to the UUID for «Characteristic» and the *Starting Handle* and *Ending Handle* parameters shall be set to the service handle range.

Two possible responses can be sent from the server for the ATT_READ_BY_TYPE_REQ PDU: ATT_READ_BY_TYPE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_READ_BY_TYPE_RSP PDU returns a list of *Attribute Handle* and *Attribute Value* pairs corresponding to the characteristics contained in the handle range provided. Each *Attribute Value* in the list is the *Attribute Value* for the characteristic declaration. The *Attribute Value* contains the characteristic properties, Characteristic *Value Handle* and characteristic UUID. The *Attribute Value* for each *Attribute Handle* and *Attribute Value* pairs are checked for a matching characteristic UUID. Once found, the sub-procedure continues until the end of the service handle range is exhausted. The ATT_READ_BY_TYPE_REQ PDU is issued again with the *Starting Handle* set to one greater than the last *Attribute Handle* in the ATT_READ_BY_TYPE_RSP PDU.

If the ATT_ERROR_RSP PDU is sent by the server with the Error Code parameter set to *Attribute Not Found* (0x0A), the characteristic does not exist on the server within the handle range provided.

The sub-procedure may end early if a desired characteristic is found prior to discovering all the characteristics for the specified service supported on the server.

The characteristic declaration described in Section 3.3 specifies that the characteristic declaration is readable and requires no authentication or authorization, therefore insufficient authentication or read not permitted errors shall not occur.

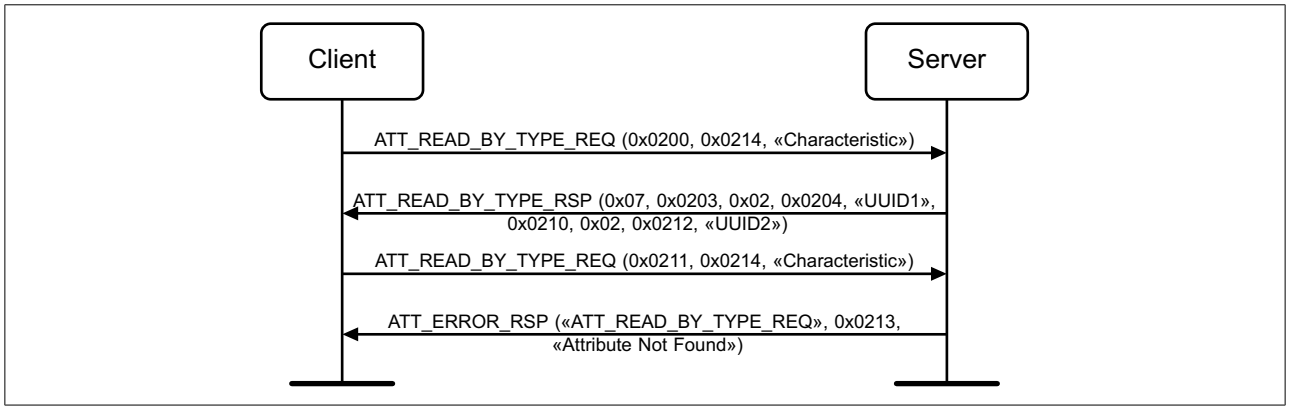



*Figure 4.6: Discover Characteristics by UUID example*

## 4.7 Characteristic Descriptor Discovery

This procedure is used by a client to discover characteristic descriptors of a characteristic. Once the characteristic descriptors are discovered additional information about the characteristic descriptors can be accessed using other procedures.

There is one sub-procedure that can be used for characteristic descriptor discovery: Discover All Characteristic Descriptors.

### 4.7.1 Discover All Characteristic Descriptors

This sub-procedure is used by a client to find all the characteristic descriptor's Attribute Handles and Attribute Types within a characteristic definition when only the characteristic handle range is known. The characteristic specified is identified by the characteristic handle range.

The ATT_FIND_INFORMATION_REQ PDU shall be used with the Starting Handle set to the handle of the specified characteristic value + 1 and the *Ending Handle* set to the ending handle of the specified characteristic.

Two possible responses can be sent from the server for the ATT_FIND_INFORMATION_REQ PDU: ATT_FIND_INFORMATION_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_FIND_INFORMATION_RSP PDU returns a list of *Attribute Handle* and *Attribute Type* pairs corresponding to the characteristic descriptors in the characteristic definition. The *Attribute Handle* is the handle for the characteristic descriptor declaration. The *Attribute Type* is the characteristic descriptor UUID. The ATT_FIND_INFORMATION_REQ PDU shall be issued again with the *Starting Handle* set to one greater than the last *Attribute Handle* in the ATT_FIND_INFORMATION_RSP PDU.

The sub-procedure is complete when the ATT_ERROR_RSP PDU is received and the Error Code parameter is set to *Attribute Not Found* (0x0A) or the ATT_FIND_INFORMATION_RSP PDU has an *Attribute Handle* that is equal to the *Ending Handle* of the request.

The sub-procedure may end early if a desired Characteristic Descriptor is found prior to discovering all the characteristic descriptors of the specified characteristic.

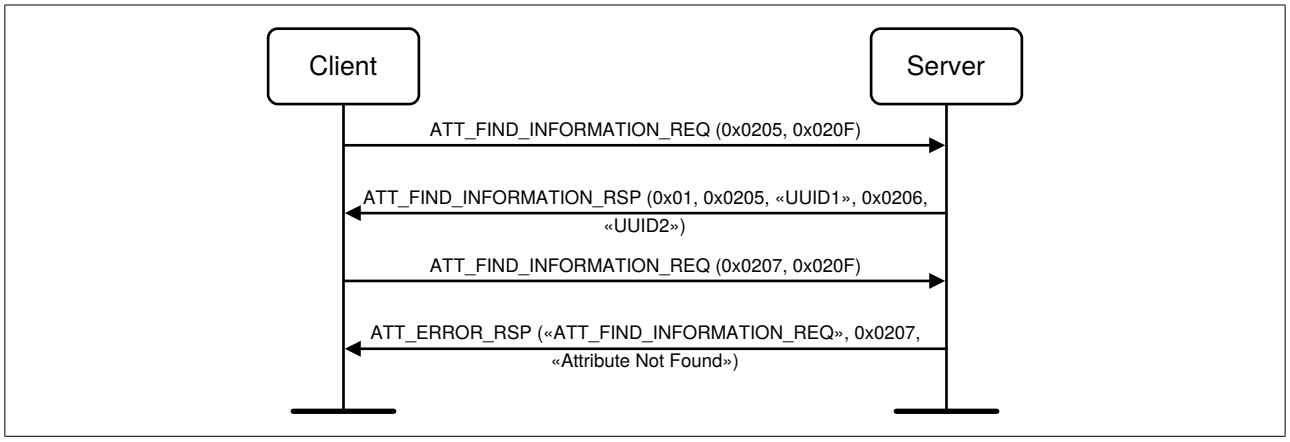



*Figure 4.7: Discover All Characteristic Descriptors example*

## 4.8 Characteristic Value Read

This procedure is used to read a *Characteristic Value* from a server. There are four sub-procedures that can be used to read a *Characteristic Value*: Read Characteristic Value, Read Using Characteristic UUID, Read Long Characteristic Value, and Read Multiple Characteristic Values.

### 4.8.1 Read Characteristic Value

This sub-procedure is used to read a *Characteristic Value* from a server when the client knows the *Characteristic Value Handle*. The ATT_READ_REQ PDU is used with the *Attribute Handle* parameter set to the *Characteristic Value Handle*. The ATT_READ_RSP PDU returns the *Characteristic Value* in the *Attribute Value* parameter.

The ATT_READ_RSP PDU only contains the complete *Characteristic Value* if that is less than or equal to (ATT_MTU – 1) octets in length. If the *Characteristic Value* is greater than (ATT_MTU – 1) octets in length, the ATT_READ_RSP PDU only contains the first portion of the *Characteristic Value* and the *Read Long Characteristic Value* procedure may be used if the rest is required.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, or if a read operation is not permitted on the *Characteristic Value*. The *Error Code* parameter is set as specified in the Attribute Protocol.

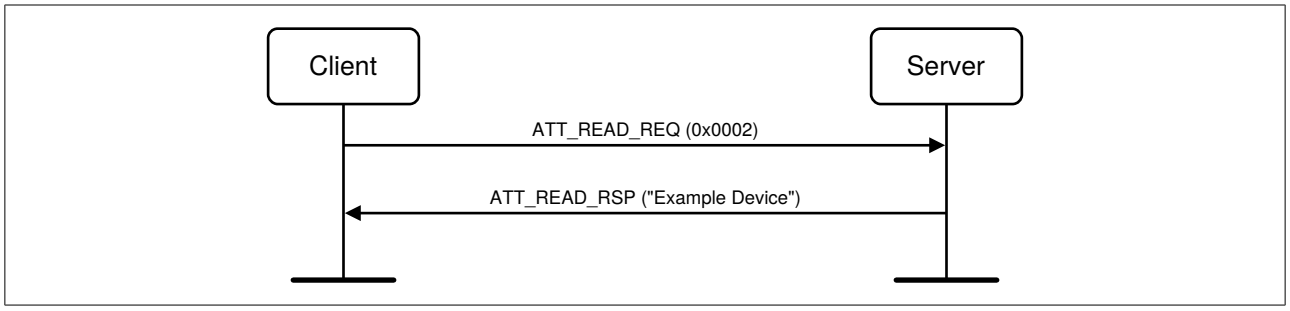



*Figure 4.8: Read Characteristic Value example*

### 4.8.2 Read Using Characteristic UUID

This sub-procedure is used to read a *Characteristic Value* from a server when the client only knows the characteristic UUID and does not know the handle of the characteristic.

The ATT_READ_BY_TYPE_REQ PDU is used to perform the sub-procedure. The Attribute Type is set to the known characteristic UUID and the Starting Handle and Ending Handle parameters shall be set to the range over which this read is to be performed. This is typically the handle range for the service in which the characteristic belongs.

Two possible responses can be sent from the server for the ATT_READ_BY_TYPE_REQ PDU: ATT_READ_BY_TYPE_RSP and ATT_ERROR_RSP PDUs.

An ATT_ERROR_RSP PDU is returned if an error occurred on the server.

The ATT_READ_BY_TYPE_RSP PDU returns a list of *Attribute Handle* and *Attribute Value* pairs corresponding to the first characteristics contained in the handle range that will fit into the ATT_READ_BY_TYPE_RSP PDU. This procedure does not return the complete list of all characteristics with the given characteristic UUID within the range of values. If such an operation is required, then the *Discover Characteristics by UUID* sub procedure shall be used.

If the ATT_ERROR_RSP PDU is sent by the server with the Error Code parameter set to *Attribute Not Found* (0x0A), the characteristic does not exist on the server within the handle range provided.

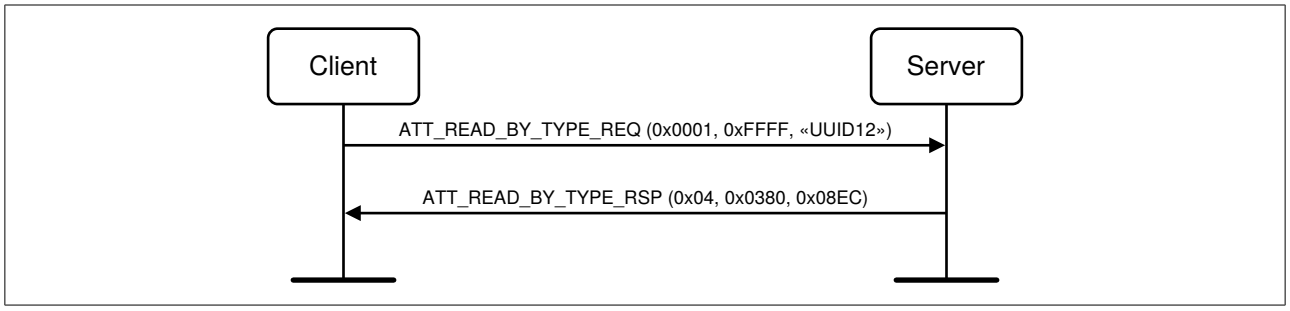



*Figure 4.9: Read Using Characteristic UUID example*

### 4.8.3 Read Long Characteristic Value

This sub-procedure is used to read a *Characteristic Value* from a server when the client knows the *Characteristic Value Handle* and the length of the *Characteristic Value* is longer than can be sent in a single ATT_READ_RSP PDU.

The ATT_READ_REQ and ATT_READ_BLOB_REQ PDUs are used to perform this sub-procedure. The *Attribute Handle* shall be set to the *Characteristic Value Handle* of the *Characteristic Value* to be read. To read the complete *Characteristic Value* an ATT_READ_REQ PDU should be used for the first part of the value and ATT_READ_BLOB_REQ PDUs shall used for the rest. The Value Offset parameter of each ATT_READ_BLOB_REQ PDU shall be set to the offset of the next octet within the *Characteristic Value* that has yet to be read. The ATT_READ_BLOB_REQ PDU is repeated until the ATT_READ_BLOB_RSP PDU's *Part Attribute Value* parameter is shorter than (ATT_MTU – 1).

For each ATT_READ_BLOB_REQ PDU a ATT_READ_BLOB_RSP PDU is received with a portion of the Characteristic Value contained in the *Part Attribute Value* parameter.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_BLOB_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, or if a read operation is not permitted on the *Characteristic Value*. The Error Code parameter is set as specified in the Attribute Protocol. If the *Characteristic Value* has a fixed length that is not longer than (ATT_MTU – 1), then the server may respond to the first ATT_READ_BLOB_REQ_PDU with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Attribute Not Long* (0x0B).

Note: The ATT_READ_BLOB_REQ PDU with zero offset can be used to read the first part of the value of the Attribute.

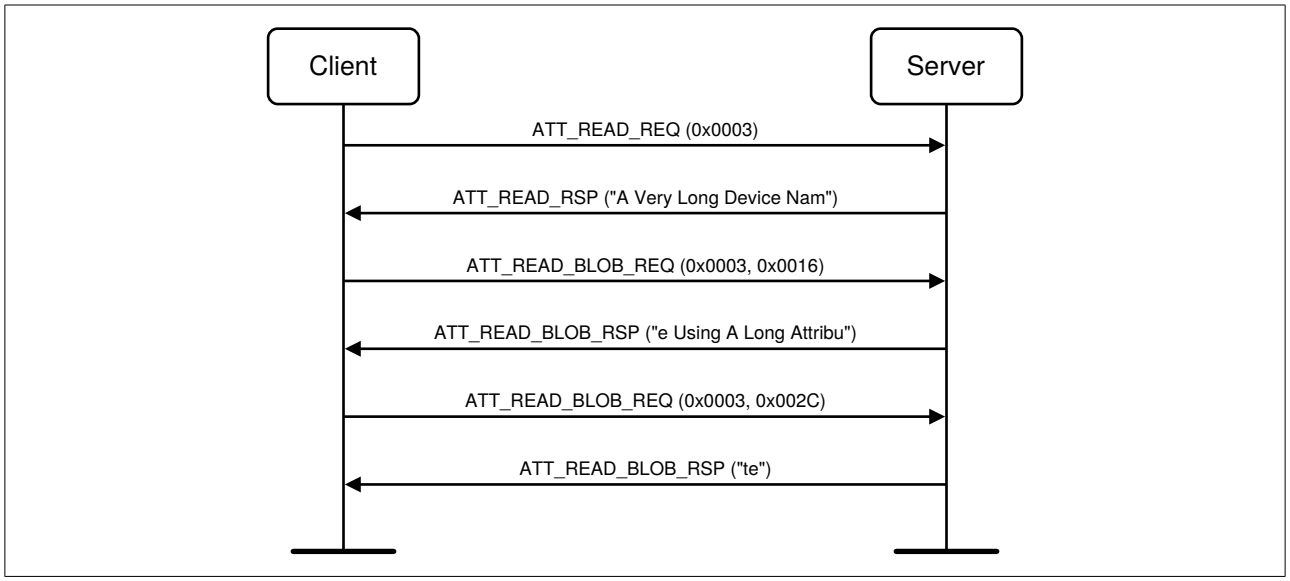



*Figure 4.10: Read Long Characteristic Value example*

### 4.8.4 Read Multiple Characteristic Values

This sub-procedure is used to read multiple *Characteristic Values* from a server when the client knows the *Characteristic Value Handles*. The ATT_READ_MULTIPLE_REQ PDU is used with the *Set Of Handles* parameter set to the *Characteristic Value Handles*. The ATT_READ_MULTIPLE_RSP PDU returns the *Characteristic Values* in the *Set Of Values* parameter.

The ATT_READ_MULTIPLE_RSP PDU only contains a set of *Characteristic Values* that is less than or equal to (ATT_MTU – 1) octets in length. If the *Set Of Values* is greater than (ATT_MTU – 1) octets in length, only the first (ATT_MTU – 1) octets are included in the response.

Note: If a client requests multiple *Characteristic Values* when the response's *Set Of Values* parameter is equal to (ATT_MTU – 1) octets in length, then it will not be possible to determine if the last *Characteristic Value* was read or additional *Characteristic Values* exist but were truncated.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_MULTIPLE_RSP PDU if insufficient authentication, insufficient authorization, a too-short encryption key size, or insufficient encryption is used by the client, or if a read operation is not permitted on any of the *Characteristic Values*. The *Error Code* parameter is set as specified in the Attribute Protocol.

Refer to the Attribute Protocol specification for the format of the *Set Of Handles* and *Set Of Values* parameter.

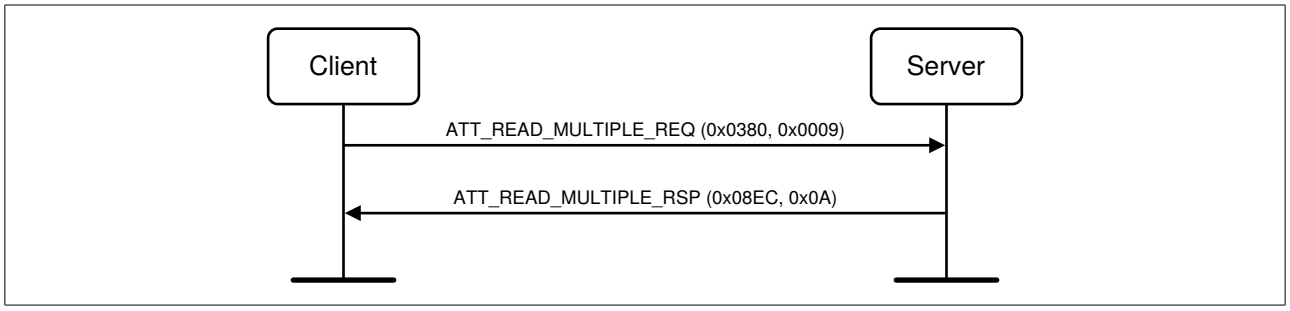



*Figure 4.11: Read Multiple Characteristic Values example*

### 4.8.5 Read Multiple Variable Length Characteristic Values

This sub-procedure is used to read multiple variable length Characteristic Values from a server when the client knows the Characteristic Value Handles. The Attribute Protocol ATT_READ_MULTIPLE_VARIABLE_REQ PDU is used with the Set Of Handles parameter set to the Characteristic Value Handles. The ATT_READ_MULTIPLE_VARIABLE_RSP PDU returns the Characteristic Values in the Length Value Tuple List parameter.

The ATT_READ_MULTIPLE_VARIABLE_RSP PDU can only contain a Length Value Tuple List that is less than or equal to (ATT_MTU – 1) octets in length. If the Length Value Tuple List is greater than (ATT_MTU – 1) octets in length, only the first (ATT_MTU – 1) octets are included in the response.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_MULTIPLE_VARIABLE_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size, or insufficient encryption is used by the client, or if a read operation is not permitted on any of the Characteristic Values. The Error Code parameter is set as specified in the Attribute Protocol.

Refer to the Attribute Protocol specification for the format of the Set Of Handles and Length Value Tuple List parameter.

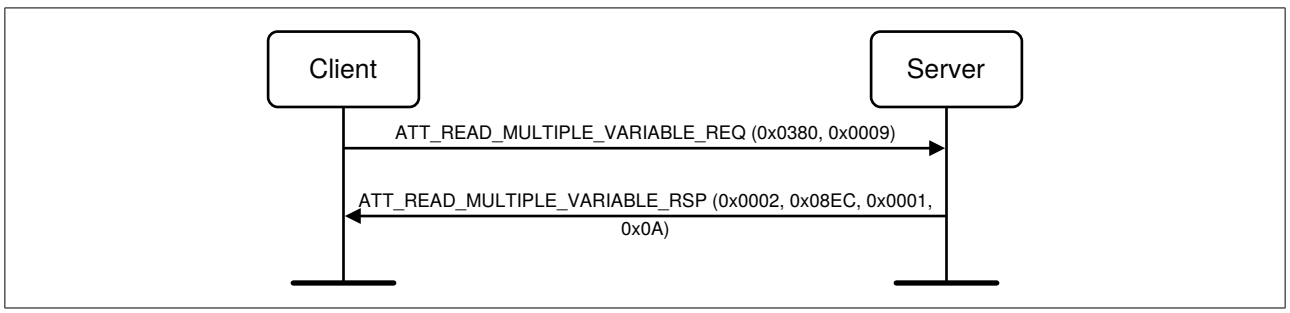



*Figure 4.12: Read Multiple Variable Length Characteristic Values example*

## 4.9 Characteristic Value Write

This procedure is used to write a *Characteristic Value* to a server.

The sub-procedures that can be used to write a *Characteristic Value* are: Write Without Response, Write Characteristic Value, Write Long Characteristic Value, and Characteristic Value Reliable Writes.

### 4.9.1 Write Without Response

This sub-procedure is used to write a *Characteristic Value* to a server when the client knows the *Characteristic Value Handle* and the client does not need an acknowledgment that the write was successfully performed. This sub-procedure only writes the first (ATT_MTU – 3) octets of a *Characteristic Value*. This sub-procedure cannot be used to write a long characteristic; instead the *Write Long Characteristic Value* sub-procedure should be used.

The ATT_WRITE_CMD PDU is used for this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle*. The *Attribute Value* parameter shall be set to the new *Characteristic Value*.

If the *Characteristic Value* write request is too long (see [Vol 3] Part F, Section 3.4.5.1) or has an invalid value as defined by the profile, then the write shall not succeed and no error shall be generated by the server.

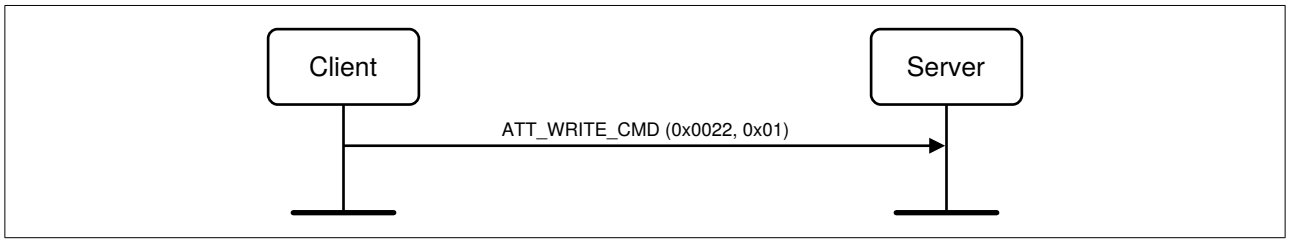



*Figure 4.13: Write Without Response example*

### 4.9.2 [This section is no longer used]

### 4.9.3 Write Characteristic Value

This sub-procedure is used to write a *Characteristic Value* to a server when the client knows the *Characteristic Value Handle*. This sub-procedure only writes the first (ATT_MTU – 3) octets of a *Characteristic Value*. This sub-procedure cannot be used to write a long Attribute; instead the *Write Long Characteristic Value* sub-procedure should be used.

The ATT_WRITE_REQ PDU is used to for this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle*. The *Attribute Value* parameter shall be set to the new characteristic.

An ATT_WRITE_RSP PDU shall be sent by the server if the write of the *Characteristic Value* succeeded.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_WRITE_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, the *Attribute Value* to be written is too long (see [Vol 3] Part F, Section 3.4.5.1), or if a write operation is not permitted on the *Characteristic Value*. The Error Code parameter is set as specified in the Attribute Protocol. If the *Characteristic Value* has an invalid value as defined by the profile, then the value shall not be written and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Application Error* (0x80 – 0x9F) by the server.

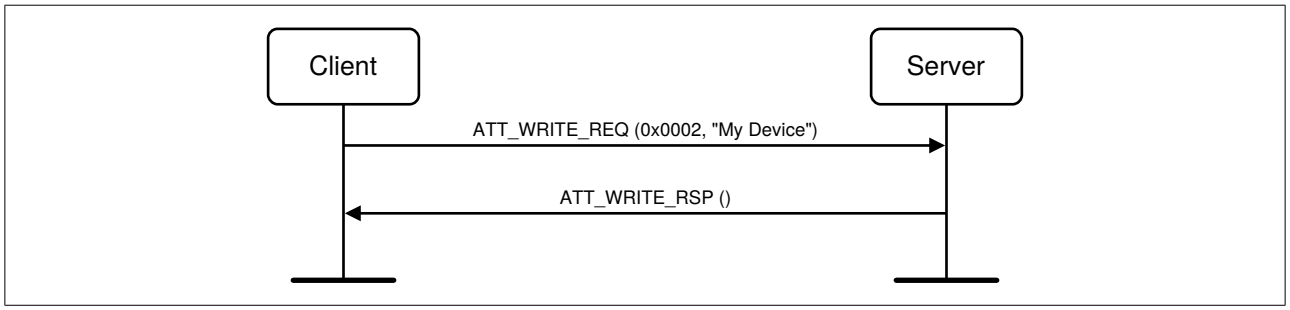



*Figure 4.14: Write Characteristic Value example*

### 4.9.4 Write Long Characteristic Value

This sub-procedure is used to write a *Characteristic Value* to a server when the client knows the *Characteristic Value Handle* but the length of the *Characteristic Value* is longer than can be sent in a single ATT_WRITE_REQ PDU.

The ATT_PREPARE_WRITE_REQ and ATT_EXECUTE_WRITE_REQ PDUs are used to perform this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle* of the *Characteristic Value* to be written. The *Part Attribute Value* parameter shall be set to the part of the *Attribute Value* that is being written. The *Value Offset* parameter shall be the offset within the *Characteristic Value* to be written. To write the complete *Characteristic Value* the offset should be set to 0x0000 for the first ATT_PREPARE_WRITE_REQ PDU. The offset for subsequent ATT_PREPARE_WRITE_REQ PDUs is the next octet that has yet to be written. The ATT_PREPARE_WRITE_REQ PDU is repeated until the complete *Characteristic Value* has been transferred, after which an ATT_EXECUTE_WRITE_REQ PDU is used to write the complete value.

Note: The values in the ATT_PREPARE_WRITE_RSP PDU do not need to be verified in this sub-procedure.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_PREPARE_WRITE_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, the prepared attribute value is too long (see [Vol 3] Part F, Section 3.4.6.3), or if a write operation is not permitted on the *Characteristic Value*. The Error Code parameter is set as specified in the Attribute Protocol. If the *Attribute Value* has an invalid value as defined by the profile, then the write shall not succeed and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Application Error* (0x80 – 0x9F) by the server.

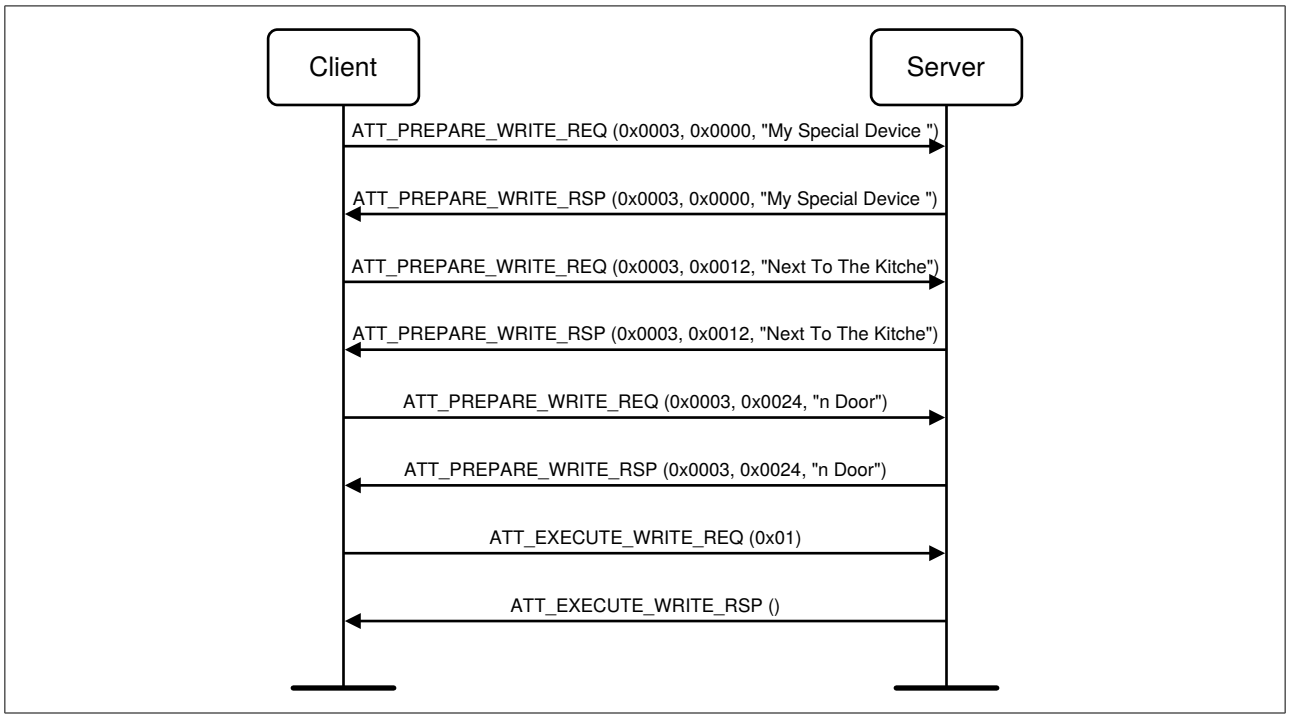



*Figure 4.15: Write Long Characteristic Value example*

### 4.9.5 Characteristic Value Reliable Writes

This sub-procedure is used to write a *Characteristic Value* to a server when the client knows the *Characteristic Value Handle*, and assurance is required that the correct *Characteristic Value* is going to be written by transferring the *Characteristic Value* to be written in both directions before the write is performed. A higher-layer protocol can also use this sub-procedure to write multiple values in order in a single operation.

The sub-procedure has two phases; the first phase prepares the *Characteristic Values* to be written. To do this, the client transfers the *Characteristic Values* to the server. The server checks the validity of the *Characteristic Values*. The client also checks each *Characteristic Value* to verify it was correctly received by the server using the server

responses. Once this is complete, the second phase performs the execution of all of the prepared Characteristic Value writes on the server from this client.

In the first phase, the ATT_PREPARE_WRITE_REQ PDU is used. The *Attribute Handle* shall be set to the *Characteristic Value Handle* that is to be prepared to write. The *Value Offset* and *Part Attribute Value* parameter shall be set to the new *Characteristic Value*.

Two possible responses can be sent from the server for the ATT_PREPARE_WRITE_-REQ PDU: ATT_PREPARE_WRITE_RSP and ATT_ERROR_RSP PDUs.

If the number of prepared write requests exceeds the number of prepared writes supported, then an ATT_ERROR_RSP PDU with the Error Code parameter set to *Prepare Queue Full* (0x09) shall be sent by the server.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_PREPARE_WRITE_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, the prepared attribute value is too long (see [Vol 3] Part F, Section 3.4.6.3), or if a write operation is not permitted on the *Characteristic Value*. The *Error Code* parameter is set as specified in the Attribute Protocol.

If a *Characteristic Value* is prepared two or more times during this sub-procedure, then all prepared values are written to the same *Characteristic Value* in the order that they were prepared.

If an ATT_PREPARE_WRITE_RSP PDU is returned, then the *Value Offset* and *Part Attribute Value* parameter in the response shall be checked with the *Value Offset* and *Part Attribute Value* parameter that was sent in the ATT_PREPARE_WRITE_REQ PDU; if they are different, then the value has been corrupted during transmission, and the sub-procedure shall be aborted by sending an ATT_EXECUTE_WRITE_REQ PDU with the Flags parameter set to 0x00 to cancel all prepared writes. The complete sub-procedure may be restarted.

Multiple ATT_PREPARE_WRITE_REQ PDUs can be sent by a client, each of which will be queued by the server.

In the second phase, the ATT_EXECUTE_WRITE_REQ PDU is used. The Attribute Flags parameter shall be set to 0x01 to immediately write all pending prepared values in the order that they were prepared. The server shall write the prepared writes once it receives this request and shall only send the ATT_EXECUTE_WRITE_RSP PDU once all the prepared values have been successfully written. If the *Characteristic Value* has an invalid value as defined by the profile, then the write shall not succeed, and an ATT_ERROR_RSP PDU with the Error Code parameter set to *Application Error* (0x80 – 0x9F) shall be sent by the server. The state of the Characteristic Values that were prepared is undefined.

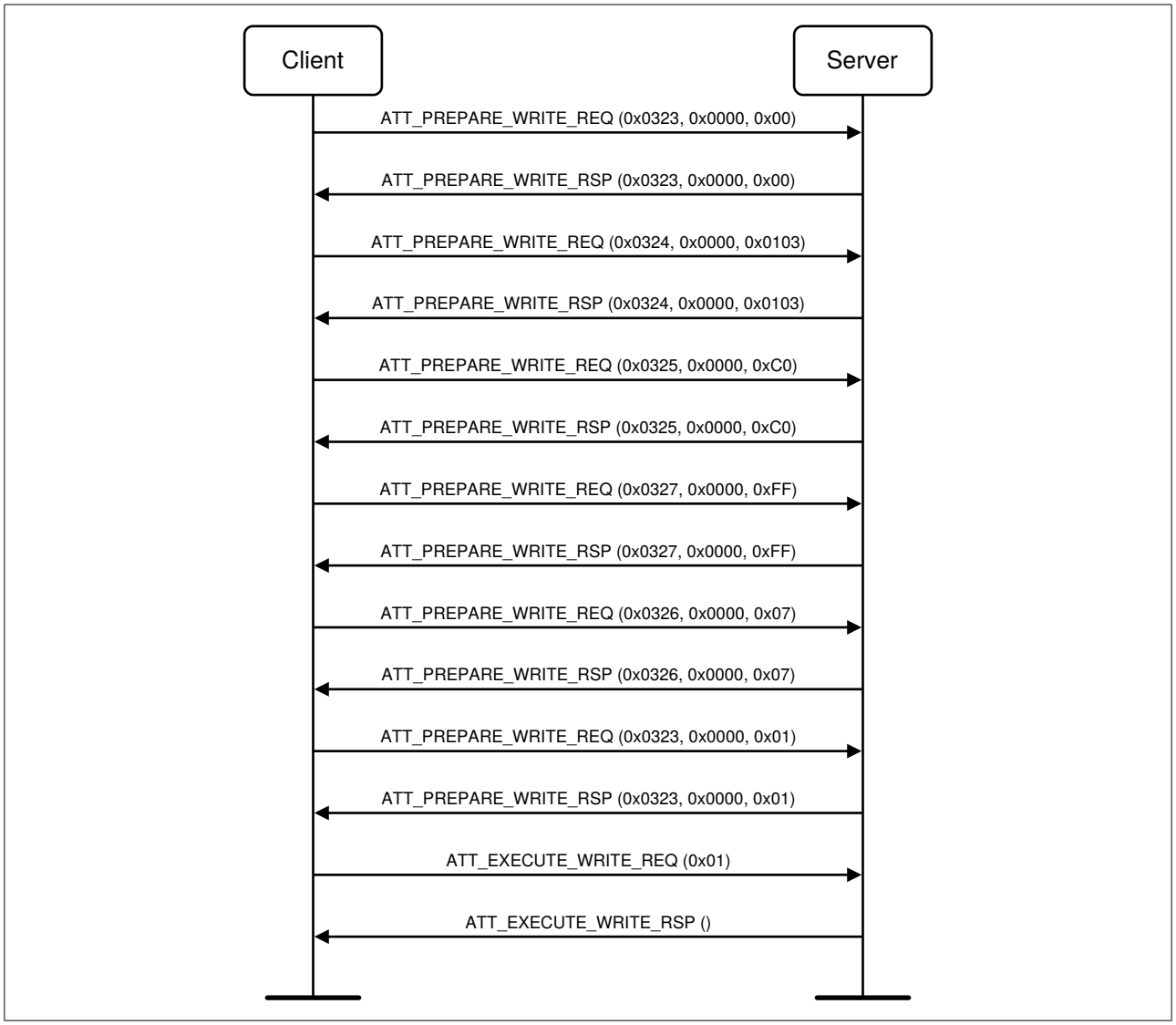



*Figure 4.16: Characteristic Value Reliable Writes example*

## 4.10 Characteristic Value Notification

This procedure is used to notify a client of the value of a *Characteristic Value* from a server without expecting any Attribute Protocol layer acknowledgment that the notification was successfully received. There are two sub-procedures that can be used to notify a value: Single Notification[1] and Multiple Variable Length Notifications. Notifications can be configured using the Client Characteristic Configuration descriptor (See Section 3.3.3.3).

A profile defines when to use notifications.

[1]This was previously just "Notifications".

### 4.10.1 Single Notification

This sub-procedure is used when a server is configured to notify a *Characteristic Value* to a client.

The ATT_HANDLE_VALUE_NTF PDU is used to perform this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle* being notified, and the *Attribute Value* parameter shall be set to the *Characteristic Value*.

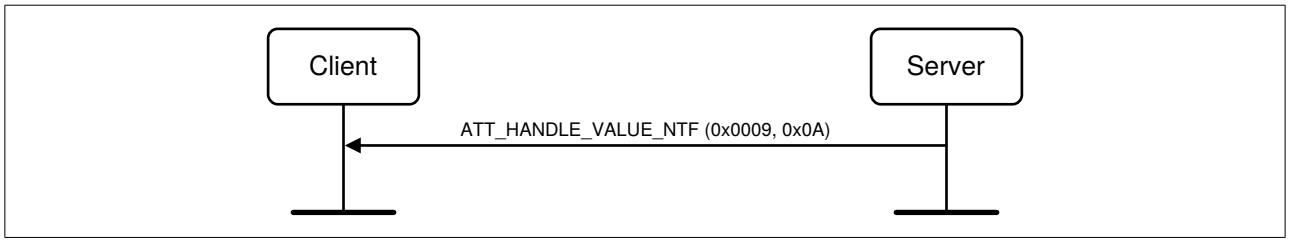



*Figure 4.17: Notifications example*

### 4.10.2 Multiple Variable Length Notifications

This sub-procedure is used when a server is configured to notify multiple Characteristic Values to a client.

The Attribute Protocol ATT_MULTIPLE_HANDLE_VALUE_NTF PDU is used to perform this sub-procedure. The Handle Length Value Tuple List parameter shall include the set of Characteristic Value Handles and associated Attribute Values.

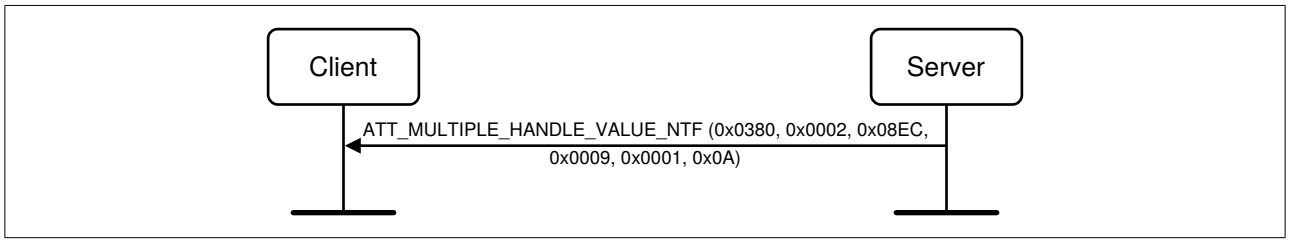



*Figure 4.18: Multiple Variable Length Notifications example*

## 4.11 Characteristic Value Indication

This procedure is used to indicate the *Characteristic Value* from a server to a client. There is one sub-procedure that can be used to indicate a value: Indication. Indications can be configured using the Client Characteristic Configuration descriptor (See Section 3.3.3.3).

A profile defines when to use indications.

### 4.11.1 Indication

This sub-procedure is used when a server is configured to indicate a *Characteristic Value* to a client and expects an Attribute Protocol layer acknowledgment that the indication was successfully received.

The ATT_HANDLE_VALUE_IND PDU is used to perform this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle* being indicated, and the *Attribute Value* parameter shall be set to the *Characteristic Value*. Once the ATT_HANDLE_VALUE_IND PDU is received by the client, the client shall respond with an ATT_HANDLE_VALUE_CFM PDU.

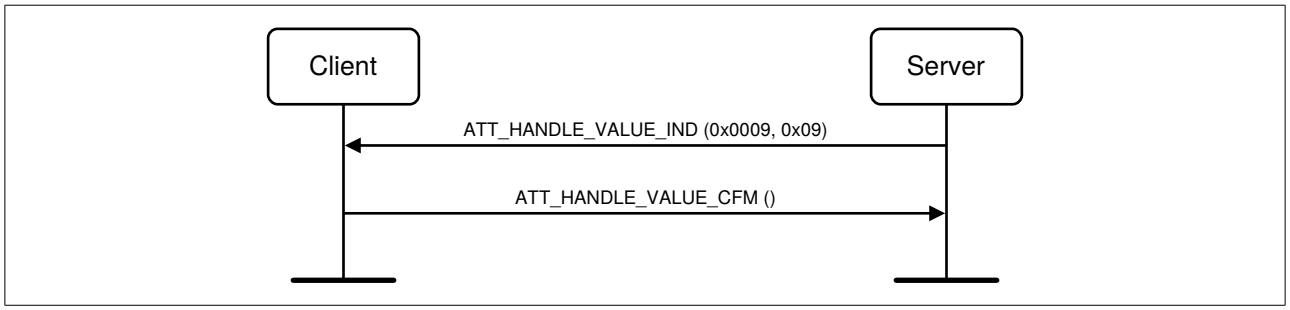



*Figure 4.19: Indication example*

## 4.12 Characteristic Descriptors

This procedure is used to read and write characteristic descriptors on a server. There are four sub-procedures that can be used to read and write characteristic descriptors: Read Characteristic Descriptor, Read Long Characteristic Descriptor, Write Characteristic Descriptor, and Write Long Characteristic Descriptor.

### 4.12.1 Read Characteristic Descriptor

This sub-procedure is used to read a characteristic descriptor from a server when the client knows the characteristic descriptor declaration's Attribute handle.

The ATT_READ_REQ PDU is used for this sub-procedure. The ATT_READ_REQ PDU is used with the *Attribute Handle* parameter set to the characteristic descriptor handle. The ATT_READ_RSP PDU returns the characteristic descriptor value in the *Attribute Value* parameter.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, or if a read operation is not permitted on the *Characteristic Value*. The *Error Code* parameter is set accordingly.

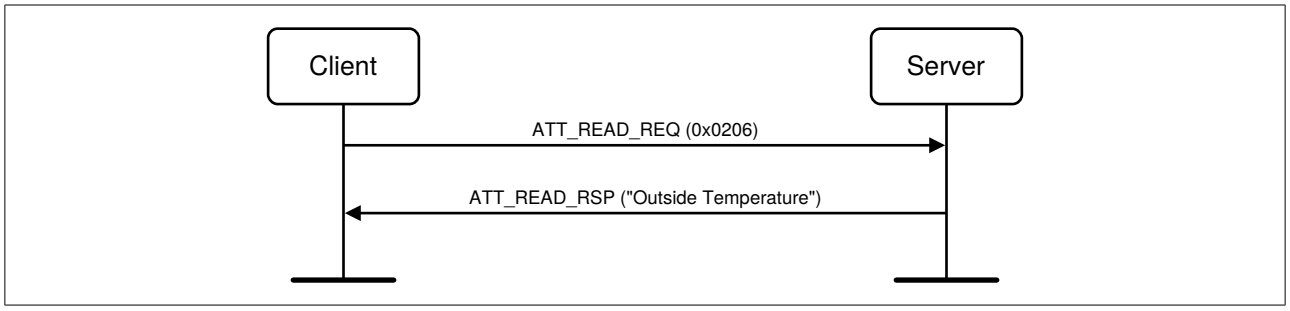



*Figure 4.20: Read Characteristic Descriptor example*

### 4.12.2 Read Long Characteristic Descriptor

This sub-procedure is used to read a characteristic descriptor from a server when the client knows the characteristic descriptor declaration's Attribute handle and the length of the characteristic descriptor declaration is longer than can be sent in a single ATT_READ_RSP PDU.

The ATT_READ_BLOB_REQ PDU is used to perform this sub-procedure. The Attribute Handle parameter shall be set to the characteristic descriptor handle. The Value Offset parameter shall be the offset within the characteristic descriptor to be read. To read the complete characteristic descriptor the offset should be set to 0x00 for the first ATT_READ_BLOB_REQ PDU. The offset for subsequent ATT_READ_BLOB_REQ PDUs is the next octet that has yet to be read. The ATT_READ_BLOB_REQ PDU is repeated until the ATT_READ_BLOB_RSP PDU's Part Attribute Value parameter is zero or an ATT_ERROR_RSP PDU is sent by the server with the Error Code parameter set to *Invalid Offset* (0x07).

For each ATT_READ_BLOB_REQ PDU an ATT_READ_BLOB_RSP PDU is received with a portion of the characteristic descriptor value contained in the Part Attribute Value parameter.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_READ_BLOB_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, or if a read operation is not permitted on the characteristic descriptor. The Error Code parameter is set accordingly.

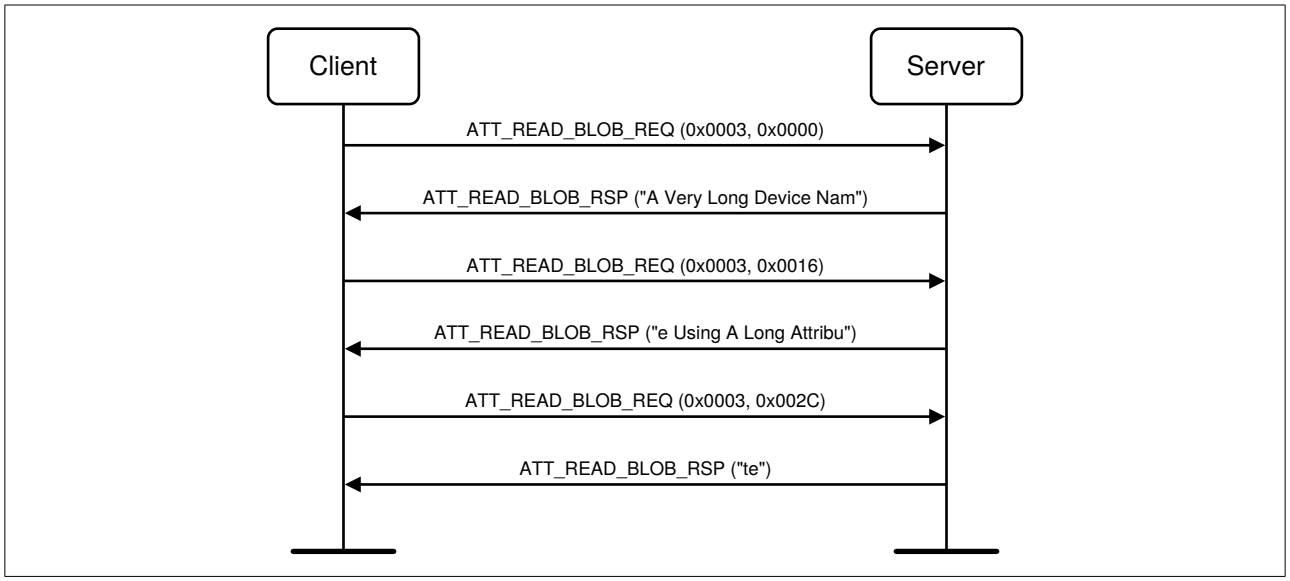



*Figure 4.21: Read Long Characteristic Descriptor example*

Note: The ATT_READ_BLOB_REQ PDU can be used to read the remainder of an characteristic descriptor value where the first part was read using a simple ATT_READ_REQ PDU.

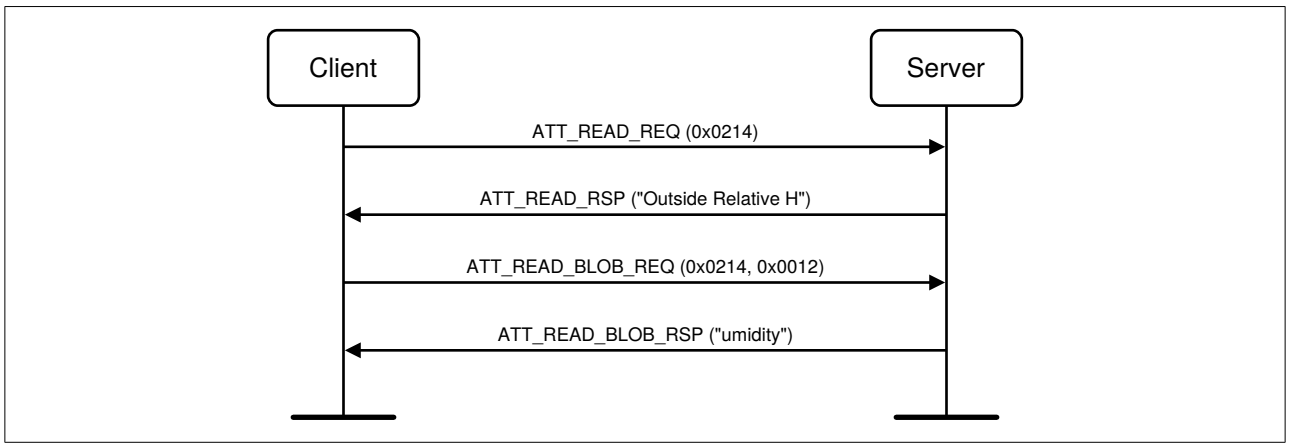



*Figure 4.22: Read Long Characteristic Descriptor (following simple read) example*

### 4.12.3 Write Characteristic Descriptor

This sub-procedure is used to write a characteristic descriptor value to a server when the client knows the characteristic descriptor handle.

The ATT_WRITE_REQ PDU is used for this sub-procedure. The *Attribute Handle* parameter shall be set to the characteristic descriptor handle. The *Attribute Value* parameter shall be set to the new characteristic descriptor value.

An ATT_WRITE_RSP PDU shall be sent by the server if the write of the characteristic descriptor value succeeded.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_WRITE_REQ PDU if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the client, the *Attribute Value* to be written is too long (see [Vol 3] Part F, Section 3.4.5.1), or if a write operation is not permitted on the *Characteristic Value*. The Error Code parameter shall be set as specified in the Attribute Protocol. If the characteristic descriptor value has an invalid value as defined by the profile or the operation is not permitted at this time, then the value shall not be written and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Application Error* (0x80 – 0x9F) by the server.

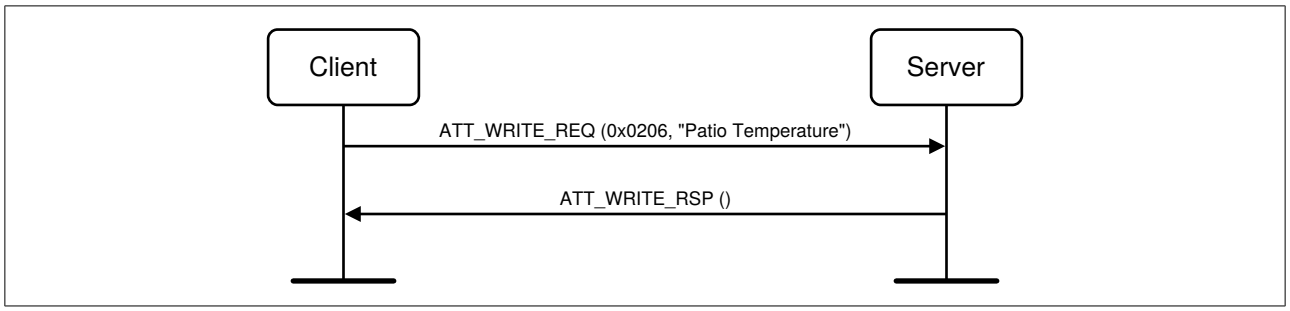



*Figure 4.23: Write Characteristic Descriptor example*

### 4.12.4 Write Long Characteristic Descriptor

This sub-procedure is used to write a characteristic descriptor value to a server when the client knows the characteristic descriptor handle but the length of the characteristic descriptor value is longer than can be sent in a single ATT_WRITE_REQ PDU.

The ATT_PREPARE_WRITE_REQ and ATT_EXECUTE_WRITE_REQ PDUs are used to perform this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Descriptor Handle* of the *Characteristic Value* to be written. The *Part Attribute Value* parameter shall be set to the part of the *Attribute Value* that is being written. The *Value Offset* parameter shall be the offset within the *Characteristic Value* to be written. To write the complete *Characteristic Value* the offset should be set to 0x0000 for the first ATT_PREPARE_WRITE_REQ PDU. The offset for subsequent ATT_PREPARE_WRITE_REQ PDUs is the next octet that has yet to be written. The ATT_PREPARE_WRITE_REQ PDU is repeated until the complete *Characteristic Value* has been transferred, after which an ATT_EXECUTE_WRITE_REQ PDU is used to write the complete value.

Note: The values in the ATT_PREPARE_WRITE_RSP PDU do not need to be verified in this sub-procedure.

An ATT_ERROR_RSP PDU shall be sent by the server in response to the ATT_PREPARE_WRITE_REQ or ATT_EXECUTE_WRITE_REQ PDUs if insufficient authentication, insufficient authorization, a too-short encryption key size is used by the

client, or if a write operation is not permitted on the *Characteristic Value*. The Error Code parameter is set as specified in the Attribute Protocol. If the *Attribute Value* that is written is the wrong size, or has an invalid value as defined by the profile, then the write shall not succeed and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Application Error* (0x80 – 0x9F) by the server.

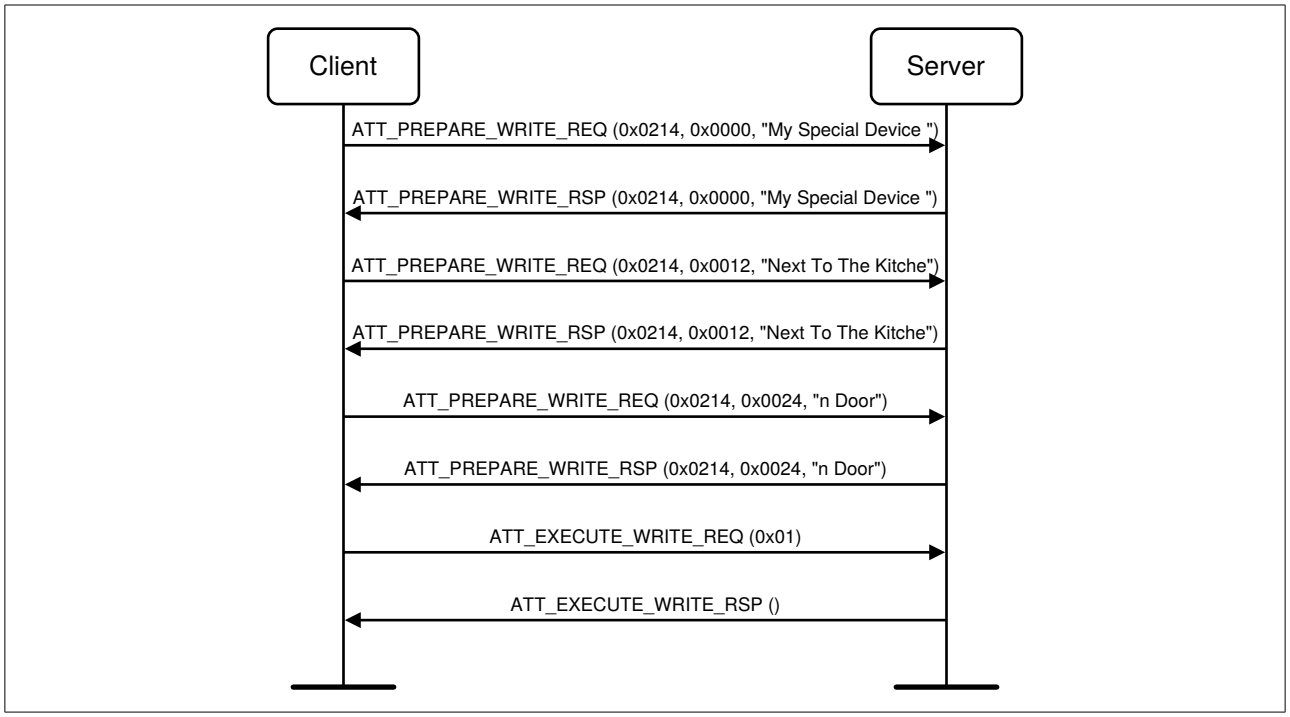



*Figure 4.24: Write Long Characteristic Descriptor example*

## 4.13 GATT procedure mapping to ATT protocol opcodes

Table 4.2 describes the mapping of the ATT protocol opcodes to the GATT procedures and sub-procedures. Only those portions of the ATT protocol requests, responses, notifications or indications necessary to implement the mandatory or supported optional sub-procedures is required.

| Feature | Sub-Procedure | ATT Protocol Opcodes |
|---|---|---|
| **Server Configuration** | Exchange MTU | ATT_EXCHANGE_MTU_REQ<br>ATT_EXCHANGE_MTU_RSP<br>ATT_ERROR_RSP |
| **Primary Service Discovery** | Discover All Primary Services | ATT_READ_BY_GROUP_TYPE_REQ<br>ATT_READ_BY_GROUP_TYPE_RSP<br>ATT_ERROR_RSP |

| Feature | Sub-Procedure | ATT Protocol Opcodes |
|---|---|---|
| | Discover Primary Service By Service UUID | ATT_FIND_BY_TYPE_VALUE_REQ<br>ATT_FIND_BY_TYPE_VALUE_RSP<br>ATT_ERROR_RSP |
| **Relationship Discovery** | Find Included Services | ATT_READ_BY_TYPE_REQ<br>ATT_READ_BY_TYPE_RSP<br>ATT_ERROR_RSP |
| **Characteristic Discovery** | Discover All Characteristics of a Service | ATT_READ_BY_TYPE_REQ<br>ATT_READ_BY_TYPE_RSP<br>ATT_ERROR_RSP |
| | Discover Characteristics by UUID | ATT_READ_BY_TYPE_REQ<br>ATT_READ_BY_TYPE_RSP<br>ATT_ERROR_RSP |
| **Characteristic Descriptor Discovery** | Discover All Characteristic Descriptors | ATT_FIND_INFORMATION_REQ<br>ATT_FIND_INFORMATION_RSP<br>ATT_ERROR_RSP |
| **Characteristic Value Read** | Read Characteristic Value | ATT_READ_REQ<br>ATT_READ_RSP<br>ATT_ERROR_RSP |
| | Read Using Characteristic UUID | ATT_READ_BY_TYPE_REQ<br>ATT_READ_BY_TYPE_RSP<br>ATT_ERROR_RSP |
| | Read Long Characteristic Value | ATT_READ_BLOB_REQ<br>ATT_READ_BLOB_RSP<br>ATT_ERROR_RSP |
| | Read Multiple Characteristic Values | ATT_READ_MULTIPLE_REQ<br>ATT_READ_MULTIPLE_RSP<br>ATT_ERROR_RSP |
| | Read Multiple Variable Length Characteristic Values | ATT_READ_MULTIPLE_VARIABLE_REQ<br>ATT_READ_MULTIPLE_VARIABLE_RSP<br>ATT_ERROR_RSP |

| Feature | Sub-Procedure | ATT Protocol Opcodes |
|---|---|---|
| **Characteristic Value Write** | Write Without Response | ATT_WRITE_CMD |
| | Write Characteristic Value | ATT_WRITE_REQ<br>ATT_WRITE_RSP<br>ATT_ERROR_RSP |
| | Write Long Characteristic Value | ATT_PREPARE_WRITE_REQ<br>ATT_PREPARE_WRITE_RSP<br>ATT_EXECUTE_WRITE_REQ<br>ATT_EXECUTE_WRITE_RSP<br>ATT_ERROR_RSP |
| | Characteristic Value Reliable Writes | ATT_PREPARE_WRITE_REQ<br>ATT_PREPARE_WRITE_RSP<br>ATT_EXECUTE_WRITE_REQ<br>ATT_EXECUTE_WRITE_RSP<br>ATT_ERROR_RSP |
| **Characteristic Value Notification** | Single Notification | ATT_HANDLE_VALUE_NTF |
| | Multiple Variable Length Notifications | ATT_MULTIPLE_HANDLE_VALUE_NTF |
| **Characteristic Value Indication** | Indication | ATT_HANDLE_VALUE_IND<br>ATT_HANDLE_VALUE_CFM |
| **Characteristic Descriptor Value Read** | Read Characteristic Descriptor | ATT_READ_REQ<br>ATT_READ_RSP<br>ATT_ERROR_RSP |
| | Read Long Characteristic Descriptor | ATT_READ_BLOB_REQ<br>ATT_READ_BLOB_RSP<br>ATT_ERROR_RSP |
| **Characteristic Descriptor Value Write** | Write Characteristic Descriptor | ATT_WRITE_REQ<br>ATT_WRITE_RSP<br>ATT_ERROR_RSP |
| | Write Long Characteristic Descriptor | ATT_PREPARE_WRITE_REQ<br>ATT_PREPARE_WRITE_RSP<br>ATT_PREPARE_WRITE_REQ<br>ATT_PREPARE_WRITE_RSP<br>ATT_ERROR_RSP |

*Table 4.2: GATT procedure mapping to ATT protocol opcodes*

## 4.14 Procedure timeouts

GATT procedures are protected from failure with an Attribute Protocol transaction timeout.

If the Attribute Protocol transaction times out, the procedure shall be considered to have failed, and the local higher layer shall be notified. No further GATT procedures shall be performed on that ATT bearer. A new GATT procedure shall only be performed on another ATT bearer.

# 5 L2CAP INTEROPERABILITY REQUIREMENTS

The following default values shall be used by an implementation of this profile. The default values used may be different depending on the physical channel that the Attribute Protocol is being sent over.

## 5.1 BR/EDR L2CAP interoperability requirements

When using an Unenhanced ATT bearer, L2CAP connection-oriented channels over BR/EDR not in Enhanced Flow Control mode can be used to transmit Attribute Protocol PDUs. These channels use the channel establishment procedure from L2CAP using the ATT fixed PSM (see [1]) including the configuration procedure to determine the ATT_MTU (see [Vol 3] Part A, Section 5.1). Therefore, the ATT bearer (or the logical link as referred to in the Attribute Protocol) is, in this case, an established L2CAP connection-oriented channel.

### 5.1.1 ATT_MTU

At the end of the L2CAP configuration phase, upon transition to the OPEN state, the ATT_MTU for this ATT bearer shall be set to the minimum of the negotiated Maximum Transmission Unit configuration options.

Note: The minimum ATT_MTU for BR/EDR is 48 octets, as defined by L2CAP in [Vol 3] Part A, Section 5.1.

### 5.1.2 BR/EDR channel requirements

All Attribute Protocol messages sent by GATT over an L2CAP channel are sent using a dynamic channel ID derived by connecting using a fixed PSM. The use of a fixed PSM allows rapid reconnection of the L2CAP channel for Attribute Protocol as a preliminary SDP query is not required.

All packets sent on this L2CAP channel shall be Attribute PDUs.

PDUs shall be reliably sent.

The flow specification for the Attribute Protocol shall be best effort.

If operating in Basic L2CAP mode, the information payload of the L2CAP B-frame shall be a single Attribute PDU.

The channel shall be encrypted. The Key_Type shall be either an Unauthenticated Combination Key or an Authenticated Combination Key.

The L2CAP connection may be initiated by the client or by the server.

### 5.1.3 [This section is no longer used]

## 5.2 LE L2CAP interoperability requirements

When using an Unenhanced ATT bearer, the channel used to carry Attribute Protocol PDUs over LE is the Attribute L2CAP fixed channel.

To terminate the ATT bearer, the physical channel has to be disconnected.

### 5.2.1 ATT_MTU

Both GATT Client and GATT Server implementations shall support an ATT_MTU not less than the default value.

| Default Value | Value for LE |
|---|---|
| ATT_MTU | 23 |

*Table 5.1: LE L2CAP ATT_MTU*

### 5.2.2 LE channel requirements

L2CAP fixed CID 0x0004 shall be used for the Attribute Protocol. All packets sent on this fixed channel shall be Attribute Protocol PDUs.

The flow specification for the Attribute Protocol shall be best effort.

The device's upper layer may mark any command or notification over the LE Attribute Protocol fixed channel as flushable data (see [Vol 3] Part A, Section 7.12). All other PDUs shall be reliably sent, and not flushed.

The retransmission and flow control mode for this channel shall be Basic L2CAP mode

The default parameters for the payload of the L2CAP B-frame shall be a single Attribute PDU.

| Parameter | Value |
|---|---|
| MTU | 23 |
| Flush Timeout | 0xFFFF (Infinite) |
| QoS | Best Effort |
| Mode | Basic Mode |

*Table 5.2: Attribute Protocol fixed channel configuration parameters*

## 5.3 Enhanced ATT bearer L2CAP interoperability requirements

When using an Enhanced ATT bearer over BR/EDR, L2CAP connection-oriented channels in Enhanced Retransmission mode can be used to transmit Attribute Protocol PDUs. Such channels are established using L2CAP_CONNECTION_REQ packets.

When using an Enhanced ATT bearer over either BR/EDR or LE, L2CAP connection-oriented channels in Enhanced Credit Based Flow Control mode can be used to transmit Attribute Protocol PDUs. Such channels are established using L2CAP_CREDIT_BASED_CONNECTION_REQ packets.

In both cases, the EATT fixed PSM [1] is used and the ATT bearer is the established L2CAP connection-oriented channel.

Multiple L2CAP channels can be established between a client and a server.

### 5.3.1 ATT_MTU

The ATT_MTU for the Enhanced ATT bearer shall be set to the minimum of the MTU field values of the two devices; these values come from the L2CAP_CREDIT_BASED_CONNECTION_REQ and L2CAP_CREDIT_BASED_CONNECTION_RSP signaling packets or the latest L2CAP_CREDIT_BASED_RECONFIGURE_REQ packets.

Note: The minimum ATT_MTU for an Enhanced ATT bearer is 64 octets.

### 5.3.2 Channel Requirements

All Attribute Protocol messages sent by GATT over an L2CAP Enhanced Credit Based Flow Control mode channel are sent using one of the dynamic channel IDs derived by connecting using a fixed PSM.

All packets sent on this L2CAP channel shall be Attribute PDUs.

The flow specification for the Attribute Protocol shall be best effort.

The information payload of the L2CAP K-frame shall be a single Attribute PDU.

The underlying ACL connection shall be encrypted before the channel is created and shall remain encrypted (other than to refresh the key) thereafter.

## 5.4 L2CAP collision mitigation

If both devices request L2CAP connections simultaneously and both devices have limited resources, a device may reject the incoming request and find its own request is also rejected. In this situation, the Central may retry immediately but the Peripheral shall

wait a minimum of 100 ms before retrying; on LE connections, the Peripheral shall wait at least 2 × (*connPeripheralLatency* + 1) × *connInterval* if that is longer.

## 5.5 Bearer support

A GATT implementation supporting bearers over BR/EDR shall support at least one of Unenhanced and Enhanced ATT bearers over BR/EDR and may support both.

A GATT implementation supporting bearers over LE shall support Unenhanced ATT bearers over LE and may support Enhanced ATT bearers over LE.

Note: A GATT implementation supporting bearers over both BR/EDR and LE can therefore support any combination of bearers provided that it supports Unenhanced ATT bearers over LE and at least one type of ATT bearer over BR/EDR.

# 6 GAP INTEROPERABILITY REQUIREMENTS

## 6.1 BR/EDR GAP interoperability requirements

### 6.1.1 Connection Establishment

To establish an Unenhanced ATT bearer, the Channel Establishment procedure (as defined in [Vol 3] Part C, Section 7.2) shall be used with the PSM set to ATT.

Either device may establish an ATT bearer at any time.

To establish an Enhanced ATT bearer, the Section 5.3 procedure shall be used with the fixed PSM set to EATT. A device shall not attempt to establish an Enhanced ATT bearer with a peer unless it is aware that the peer supports Enhanced ATT bearers, for example by using an out of band method, via a higher layer protocol, or because the device has checked the peer's Server Supported Features characteristic (see Section 7.4) or its SDP record.

Either device may terminate an ATT bearer at any time.

No idle mode procedures or modes are defined by this profile.

## 6.2 LE GAP interoperability requirements

### 6.2.1 Connection Establishment

To establish an Unenhanced ATT bearer, the Connection Establishment procedure (as defined in [Vol 3] Part C, Section 9.3.5 to Section 9.3.8) shall be used.

To establish an Enhanced ATT bearer, the Section 5.3 procedure shall be used with the fixed PSM set to EATT.

Either device may terminate an ATT bearer at any time.

No idle mode procedures or modes are defined by this profile.

Note: Unlike BR/EDR, it is not necessary to check the Server Supported Features characteristic before attempting to establish an Enhanced ATT bearer.

### 6.2.2 Profile roles

This profile can be used in the following profile roles (as defined in [Vol 3] Part C, Section 2.2.2):

- Central
- Peripheral

## 6.3 Disconnected events

### 6.3.1 Notifications and indications while disconnected

If a client has configured the server to send a notification or indication to the client, it shall be configured to allow re-establishment of the connection when it is disconnected.

If the client is disconnected, but intends to become a Central in the connection it shall perform a GAP connection establishment procedure. If the client is disconnected, but intends to become a Peripheral in the connection it shall go into a GAP connectable mode.

A server shall re-establish a connection with a client when an event or trigger operation causes a notification or indication to a client.

If the server is disconnected, but intends to become a Peripheral in the connection it shall go into a GAP connectable mode. If the server is disconnected, but intends to become a Central in the connection it shall perform a GAP connection establishment procedure.

If the server cannot re-establish a connection, then the notification or indication for this event shall be discarded and no further connection re-establishment shall occur, until another event occurs.

# 7 DEFINED GATT SERVICE

All characteristics defined within this section shall be contained in a primary service with the service UUID set to «GATT Service» as defined in Section 3.1. Only one instance of the GATT service shall be exposed on a GATT Server.

Table 7.1 lists characteristics that may be present in the server and the characteristics that may be supported by the client.

| Characteristic | Ref. | Support in Client | Support in Server |
|---|---|---|---|
| Service Changed | 7.1 | M | C.1 |
| Client Supported Features | 7.2 | O | C.2 |
| Database Hash | 7.3 | O | O |
| Server Supported Features | 7.4 | O | C.3 |
| C.1: Mandatory if service definitions on the server can be added, changed, or removed; otherwise excluded | | | |
| C.2: Mandatory if the *Database Hash* and *Service Changed* characteristics are supported or if Enhanced ATT Bearer or Multiple Variable Length Notifications are supported; otherwise excluded | | | |
| C.3: Mandatory if any of the features in Table 7.11 are supported, otherwise optional | | | |

*Table 7.1: GATT service characteristic support*

The assigned UUIDs for these characteristics are defined in Assigned Numbers [1].

## 7.1 Service Changed

The «Service Changed» characteristic is a control-point attribute (as defined in [Vol 3] Part F, Section 3.2.6) that shall be used to indicate to connected devices that services have changed (i.e., added, removed or modified). The characteristic shall be used to indicate to clients that have a trusted relationship (i.e. bond) with the server when GATT based services have changed when they re-connect to the server. See Section 2.5.2.

This *Characteristic Value* shall be configured to be indicated using the Client Characteristic Configuration descriptor by a client. Indications caused by changes to the Service Changed Characteristic Value shall be considered lost if the client has erroneously not enabled indications in the Client Characteristic Configuration descriptor (see [Vol 3] Part F, Section 3.3.3).

| Attribute Handle | Attribute Type | Attribute Value | | | Attribute Permission |
|---|---|---|---|---|---|
| 0xNNNN | 0x2803 – UUID for «Characteristic» | Characteristic Properties = 0x20 | 0xMMMM = Handle of Characteristic Value | 0x2A05 – UUID for «Service Changed» | No Authentication,<br>No Authorization |

*Table 7.2: Service Changed Characteristic declaration*

The *Service Changed Characteristic Value* is two 16-bit *Attribute Handles* concatenated together indicating the beginning and ending *Attribute Handles* affected by an addition, removal, or modification to a GATT-based service on the server. A change to a characteristic value is not considered a modification of the service. If a change has been made to any of the GATT service definition characteristic values other than the *Service Changed* characteristic value and the *Client Supported Features* characteristic value, the range shall also include the beginning and ending *Attribute Handle* for the GATT service definition.

| Attribute Handle | Attribute Type | Attribute Value | | Attribute Permission |
|---|---|---|---|---|
| 0xMMMM | 0x2A05 – UUID for «Service Changed» | 0xSSSS – Start of Affected Attribute Handle Range | 0xTTTT – End of Affected Attribute Handle Range | No Authentication,<br>No Authorization, Not Readable, Not Writable |

*Table 7.3: Service Changed Characteristic Value declaration*

There shall be only one instance of the *Service Changed* characteristic within the GATT service definition. A *Service Changed* characteristic value shall exist for each client with a trusted relationship.

If the list of GATT based services and the service definitions cannot change for the lifetime of the device then this characteristic shall not exist, otherwise this characteristic shall exist.

If the *Service Changed* characteristic exists on the server, the *Characteristic Value Indication* support on the server is mandatory.

The client shall support *Characteristic Value Indication* of the *Service Changed* characteristic.

The *Service Changed* characteristic *Attribute Handle* on the server shall not change if the server has a trusted relationship with any client.

## 7.2 Client Supported Features

The *Client Supported Features* characteristic is used by the client to inform the server which features are supported by the client. If the characteristic exists on the server, the

client may update the *Client Supported Features* bit field. If a client feature bit is set by a client and the server supports that feature, the server shall fulfill all requirements associated with this feature when communicating with this client. If a client feature bit is not set by a client, then the server shall not use any of the features associated with that bit when communicating with this client.

| Attribute Handle | Attribute Type | Attribute Value | | | Attribute Permission |
|---|---|---|---|---|---|
| 0xNNNN | 0x2803 – UUID for «Characteristic» | Characteristic Properties = 0x0A | 0xMMMM = Handle of Characteristic Value | 0x2B29 – UUID for «Client Supported Features» | Read only,<br>No Authentication,<br>No Authorization |

*Table 7.4: Client Supported Features characteristic declaration*

The format of the *Client Supported Features* characteristic is defined in Table 7.5.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permission |
|---|---|---|---|
| 0xMMMM | 0x2B29 - UUID for «Client Supported Features» | 0xXX...XX (variable length) - Client Features | Readable,<br>Writable,<br>No Authentication,<br>No Authorization |

*Table 7.5: Client Supported Features value declaration*

The *Client Supported Features* characteristic value is an array of octets, each of which is a bit field. The allocation of these bits is specified in Table 7.6. All bits not listed are reserved for future use. The array should not have any trailing all-zero octets.

If any octet number in Table 7.6 does not appear in the attribute value because it is too short, the server shall behave as if that octet were present with a value of zero.

| Client Features | Octet | Bit | Ref. | Description |
|---|---|---|---|---|
| Robust Caching | 0 | 0 | 2.5.2.1 | The client supports robust caching |
| Enhanced ATT bearer | 0 | 1 | 5.3 | The client supports Enhanced ATT bearer |
| Multiple Handle Value Notifications | 0 | 2 | 4.10.2 | The client supports receiving ATT_MULTIPLE_HANDLE_VALUE_NTF PDUs |

*Table 7.6: Client Supported Features bit assignments*

The default value for the *Client Supported Features* characteristic value shall be all bits set to zero.

There shall be only one instance of the *Client Supported Features* characteristic within the GATT service definition.

A *Client Supported Features* characteristic value shall exist for each connected client. For clients with a trusted relationship, the characteristic value shall be persistent across connections. For clients without a trusted relationship the characteristic value shall be set to the default value at each connection.

The Attribute Handle of the *Client Supported Features* characteristic on the server shall not change during a connection or if the server has a trusted relationship with any client.

A client shall not clear any bits it has set. The server shall respond to any such request with the Error Code parameter set to *Value Not Allowed* (0x13).

## 7.3 Database Hash

The *Database Hash* characteristic contains the result of a hash function applied to the service definitions in the GATT database. The client may read the characteristic at any time to determine if services have been added, removed, or modified. If any of the input fields to the hash function (as listed in Section 7.3.1) have changed, the server shall calculate a new Database Hash and update the characteristic value.

The *Database Hash* characteristic is a read-only attribute.

| Attribute Handle | Attribute Type | Attribute Value | | | Attribute Permission |
|---|---|---|---|---|---|
| 0xNNNN | 0x2803 – UUID for «Characteristic» | Characteristic Properties = 0x02 | 0xMMMM = Handle of Characteristic Value | 0x2B2A – UUID for «Database Hash» | Read only,<br>No Authentication,<br>No Authorization |

*Table 7.7: Database Hash characteristic declaration*

The characteristic value is a 128-bit unsigned integer number. The calculation of the Database Hash is specified in Section 7.3.1.

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permission |
|---|---|---|---|
| 0xMMMM | 0x2B2A - UUID for «Database Hash» | uint128 - Database Hash | Read only,<br>No Authentication,<br>No Authorization |

*Table 7.8: Database Hash characteristic value declaration*

There is only one instance of the *Database Hash* characteristic within the GATT service definition. The same *Database Hash* value is used for all clients, whether a trusted relationship exists or not.

In order to read the value of this characteristic the client shall always use the *GATT Read Using Characteristic UUID* sub-procedure. The *Starting Handle* should be set to 0x0001 and the *Ending Handle* should be set to 0xFFFF.

If a client reads the value of this characteristic while the server is re-calculating the hash following a change to the database, the server shall return the new hash, delaying its response until it is available.

### 7.3.1 Database Hash calculation

The Database Hash shall be calculated according to RFC-4493 [2]. This RFC defines the Cipher-based Message Authentication Code (CMAC) using AES-128 as the block cipher function, also known as AES-CMAC.

The inputs to AES-CMAC are:

*m* is the variable length data to be hashed
*k* is the 128-bit key, which shall be all zero
(0x00000000_00000000_00000000_00000000)

The 128-bit Database Hash is generated as follows:

Database Hash = AES-CMAC$_k$(m), where *m* is calculated as follows:

In ascending order of attribute handles, starting with the first handle, concatenate the fields *Attribute Handle*, *Attribute Type*, and *Attribute Value* if the attribute has one of the following types: «Primary Service», «Secondary Service», «Included Service», «Characteristic», or «Characteristic Extended Properties», concatenate the fields *Attribute Handle* and *Attribute Type* if the attribute has one of the following types: «Characteristic User Description», «Client Characteristic Configuration», «Server Characteristic Configuration», «Characteristic Presentation Format», or «Characteristic Aggregate Format», and ignore the attribute if it has any other type (such attributes are not part of the concatenation).
For each *Attribute Handle*, the fields shall be concatenated in the order given above. The byte order used for each field or subfield value shall be little-endian. If a field contains subfields, the subfields shall be concatenated in the order they appear in Section 3 (Service Interoperability Requirements). For instance, a characteristic declaration value of {0x02, 0x0005, 0x2A00} is represented after concatenation as 02 05 00 00 2A.

The formats of the fields *Attribute Handle*, *Attribute Type*, and *Attribute Value* for the Attribute Types listed above are defined in Section 3 (Service Interoperability Requirements).

If the length of *m* is not a multiple of the AES-CMAC block length of 128 bits, padding shall be applied as specified in RFC-4493 Section 2.4.

## 7.4 Server Supported Features

The *Server Supported Features* characteristic is a read-only characteristic that shall be used to indicate support for server features. The server shall set a bit only if the corresponding feature is supported.

| Attribute Handle | Attribute Type | Attribute Value | | | Attribute Permission |
|---|---|---|---|---|---|
| 0xNNNN | 0x2803 – UUID for «Characteristic» | Characteristic Properties = 0x02 | 0xMMMM = Handle of Characteristic Value | 0x2B3A – UUID for «Server Supported Features» | Read Only,<br>No Authentication,<br>No Authorization |

*Table 7.9: Server Supported Features characteristic declaration*

| Attribute Handle | Attribute Type | Attribute Value | Attribute Permission |
|---|---|---|---|
| 0xMMMM | 0x2B3A – UUID for «Server Supported Features» | 0xuu - Server Supported Features | Readable |

*Table 7.10: Server Supported Features value declaration*

The *Server Supported Features* characteristic is an array of octets, each of which is a bit field. The allocation of these bits is specified in Table 7.11. All bits not listed are reserved for future use. The array should not have any trailing all-zero octets.

| Server Supported Features | Octet | Bit | Ref. | Description |
|---|---|---|---|---|
| EATT Supported | 0 | 0 | 5.3 | Enhanced ATT bearer supported |

*Table 7.11: Server Supported Features bit assignments*

If any octet number in Table 7.11 does not appear in the attribute value because it is too short, the client shall behave as if that octet were present with the value of zero.

There shall be only one instance of the *Server Supported Features* characteristic within the GATT service definition.

# 8 SECURITY CONSIDERATIONS

## 8.1 Authentication requirements

Authentication in the GATT Profile is applied to each characteristic independently. Authentication requirements are specified in this profile, related higher layer specifications or are implementation specific if not specified otherwise.

The GATT Profile procedures are used to access information that may require the client to be authenticated and have an encrypted connection before a characteristic can be read or written.

If such a request is issued when the physical link is unauthenticated or unencrypted, the server shall send an ATT_ERROR_RSP PDU. To read or write this characteristic, the client can then request that the physical link be authenticated using the GAP authentication procedure, and once this has been completed, send the request again.

The list of services and characteristics that a device supports is not considered private or confidential information, and therefore the Service and Characteristic Discovery procedures shall always be permitted. This implies that an Error Code parameter set to *Insufficient Authentication* (0x05) shall not be used in an ATT_ERROR_RSP PDU for a *Find Information Request*.

Note: A characteristic can be allowed to be read by any device but only written by an authenticated device. Similarly, a characteristic can be allowed to be written by any device but only read by an authenticated device. Each individual characteristic can have different security properties.

Once sufficient authentication of the client has been established to allow access to one characteristic within a service definition, a server may also allow access to other characteristics within the service definition depending on the higher level or implementation specific requirements.

A server may allow access to most characteristics within a service definition once sufficient authentication has been performed, but restrict access to other characteristics within the same service definition. This may result due to some characteristics requiring stronger authentication requirements than currently enabled.

Once a server has authenticated a client for access to characteristics in one service definition, it may automatically allow access to characteristics in other service definitions.

## 8.2 Authorization requirements

Authorization in the GATT Profile is applied to each characteristic independently. Authorization requirements may be specified in this profile, related higher layer specifications or are implementation specific if not specified otherwise.

The GATT Profile can be used to access information that may require authorization before a characteristic can be read or written.

If such a request is issued to a characteristic contained in a service definition that is not authorized, the responder shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authorization* (0x08).

Once a server has authorized a client for access to characteristics in one group or service definition, it may automatically allow access to characteristics in other service definitions.

# 9 SDP INTEROPERABILITY REQUIREMENTS

A device that supports GATT over BR/EDR and only one of ATT or EATT shall publish the SDP record shown in Table 9.1; if both ATT and EATT are supported, the device shall publish the SDP record shown in Table 9.2. The GATT Service Start Handle shall be set to the attribute handle of the «GATT Service» service declaration. The GATT Service End Handle shall be set to the attribute handle of the last attribute within the «GATT Service» service definition group.

| Item | Type | Value | Meaning |
|---|---|---|---|
| Attribute ID | uint16 | 0x0001 | ServiceClassIDList |
| Attribute Value | Data element sequence (1 item) | | |
| Service Class | UUID | «GATT Service» | |
| Attribute ID | uint16 | 0x0004 | ProtocolDescriptorList |
| Attribute Value | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x001F<br>*or* 0x0027 | PSM = ATT<br>*or* PSM = EATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GATT service start handle |
| Parameter 1 | uint16 | 0xHHHH | GATT service end handle |
| Attribute ID | uint16 | 0x0005 | BrowseGroupList |
| Attribute Value | Data element sequence (1 item) | | |
| Group | UUID | «PublicBrowseRoot» | |

*Table 9.1: SDP record for GATT if only one of ATT or EATT is supported*

| Item | Type | Value | Meaning |
|---|---|---|---|
| Attribute ID | uint16 | 0x0001 | ServiceClassIDList |
| Attribute Value | Data element sequence (1 item) | | |
| Service class | UUID | «GATT Service» | |
| Attribute ID | uint16 | 0x0004 | ProtocolDescriptorList |
| Attribute Value | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |

| Item | Type | Value | Meaning |
|---|---|---|---|
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x001F | PSM = ATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GATT service start handle |
| Parameter 1 | uint16 | 0xHHHH | GATT service end handle |
| Attribute ID | uint16 | 0x000D | AdditionalProtocolDescriptorLists |
| Attribute Value | Data element sequence (1 item) | | |
| Protocol Descriptor List | Data element sequence (2 items) | | |
| Protocol Descriptor | Data element sequence (2 items) | | |
| Protocol | UUID | «L2CAP» | |
| Parameter 0 | uint16 | 0x0027 | PSM = EATT |
| Protocol Descriptor | Data element sequence (3 items) | | |
| Protocol | UUID | «ATT» | |
| Parameter 0 | uint16 | 0xHHHH | GATT service start handle |
| Parameter 1 | uint16 | 0xHHHH | GATT service end handle |
| Attribute ID | uint16 | 0x0005 | BrowseGroupList |
| Attribute Value | Data element sequence (1 item) | | |
| Group | UUID | «PublicBrowseRoot» | |

*Table 9.2: SDP record for GATT if both ATT and EATT are supported*

# 10 REFERENCES

[1] Assigned Numbers Specification: https://www.bluetooth.com/specifications/assigned-numbers

[2] RFC-4493: http://www.ietf.org/rfc/rfc4493.txt

[3] Core Specification Supplement, Part A, Data Types Specification

# Appendix A Example ATT Server contents

Table A.1 shows an example ATT Server and the attributes contained on the server.

Note: This example does not necessarily use UUIDs or services defined by the Bluetooth SIG or in adopted profiles.

| Handle | Attribute Type | Attribute Value |
|---|---|---|
| 0x0001 | «Primary Service» | «GAP Service» |
| 0x0004 | «Characteristic» | {0x02, 0x0006, «Device Name»} |
| 0x0006 | «Device Name» | “Example Device” |
| 0x0010 | «Primary Service» | «GATT Service» |
| 0x0011 | «Characteristic» | {0x26, 0x0012, «Service Changed»} |
| 0x0012 | «Service Changed» | 0x0000, 0x0000 |
| 0x0100 | «Primary Service» | «Battery State Service» |
| 0x0106 | «Characteristic» | {0x02, 0x0110, «Battery State»} |
| 0x0110 | «Battery State» | 0x04 |
| 0x0200 | «Primary Service» | «Thermometer Humidity Service» |
| 0x0201 | «Include» | {0x0500, 0x0504, «Manufacturer Service»} |
| 0x0202 | «Include» | {0x0550,0x0568} |
| 0x0203 | «Characteristic» | {0x02, 0x0204, «Temperature»} |
| 0x0204 | «Temperature» | 0x028A |
| 0x0205 | «Characteristic Presentation Format» | {0x0E, 0xFE, «degrees Celsius», 0x01, «Outside»} |
| 0x0206 | «Characteristic User Description» | “Outside Temperature” |
| 0x0210 | «Characteristic» | {0x02, 0x0212, «Relative Humidity»} |
| 0x0212 | «Relative Humidity» | 0x27 |
| 0x0213 | «Characteristic Presentation Format» | {0x04, 0x00, «Percent», «Bluetooth SIG», «Outside»} |
| 0x0214 | «Characteristic User Description» | “Outside Relative Humidity” |
| 0x0280 | «Primary Service» | «Weight Service» |
| 0x0281 | «Include» | 0x0505, 0x0509, «Manufacturer Service»} |
| 0x0282 | «Characteristic» | {0x02, 0x0283, «Weight kg»} |
| 0x0283 | «Weight kg» | 0x00005582 |

| Handle | Attribute Type | Attribute Value |
|---|---|---|
| 0x0284 | «Characteristic Presentation Format» | {0x08, 0xFD, «kilogram», «Bluetooth SIG», «Hanging»} |
| 0x0285 | «Characteristic User Description» | “Rucksack Weight” |
| 0x0300 | «Primary Service» | «Position Service» |
| 0x0301 | «Characteristic» | {0x02, 0x0302, «Latitude Longitude»} |
| 0x0302 | «Latitude Longitude» | 0x28BEAFA40B320FCE |
| 0x0304 | «Characteristic» | {0x02, 0x0305, «Latitude Longitude Elevation»} |
| 0x0305 | «Latitude Longitude Elevation» | 0x28BEAFA40B320FCE0176 |
| 0x0400 | «Primary Service» | «Alert Service» |
| 0x0401 | «Characteristic» | {0x0E, 0x0402, «Alert Enumeration»} |
| 0x0402 | «Alert Enumeration» | 0x00 |
| 0x0500 | «Secondary Service» | «Manufacturer Service» |
| 0x0501 | «Characteristic» | {0x02, 0x0502, «Manufacturer Name»} |
| 0x0502 | «Manufacturer Name» | “ACME Temperature Sensor” |
| 0x0503 | «Characteristic» | {0x02, 0x0504, «Serial Number»} |
| 0x0504 | «Serial Number» | “237495-3282-A” |
| 0x0505 | «Secondary Service» | «Manufacturer Service» |
| 0x0506 | «Characteristic» | {0x02, 0x0507, «Manufacturer Name»} |
| 0x0507 | «Manufacturer Name» | “ACME Weighing Scales” |
| 0x0508 | «Characteristic» | {0x02, 0x0509, «Serial Number»} |
| 0x0509 | «Serial Number» | “11267-2327A00239” |
| 0x0550 | «Secondary Service» | «Vendor Specific Service» |
| 0x0560 | «Characteristic» | {0x02, 0x0568, «Vendor Specific Type»} |
| 0x0568 | «Vendor Specific Type» | 0x56656E646F72 |

*Table A.1: Examples of ATT Server contents*

As can be seen, the ATT Server indicates support for ten services: GAP Service, GATT Service, Battery State Service, Thermometer Humidity Service, Weight Service, Position Service, Alert Service, two Manufacturer Services, and a Vendor Specific Service.

The server contains the following information about each of the services:

- The characteristic containing the name of the device is “Example Device”.
- The characteristic indicating the server supports all the attribute opcodes, and supports two prepared write values.

- The characteristic containing the battery state with a value of 0x04, meaning it is discharging.
- The characteristic containing the outside temperature with a value of 6.5 °C.
- The characteristic containing the outside relative humidity with a value of 39%.
- The characteristic containing the weight hanging off the device with a value of 21.89 kg.
- The characteristic containing the position of this device with the value of 68.3585444 degrees north, 18.7830222 degrees east, with an elevation of 374 meters.
- The characteristic containing the temperature sensor manufacturer with the value of ACME Temperature Sensor.
- The characteristic containing the serial number for the temperature sensor with a value of 237495-3282-A.
- The characteristic containing the weighing sensor is manufacturer with a value of ACME Weight Scales.
- The characteristic containing the serial number for the weighing sensor with a value of 11267-2327A00239.

The device is therefore on the side of the Abisko Turiststation, Norrbottens Län, Sweden, with a battery in good state, measuring a relatively warm day, with low humidity, and a heavy rucksack.

# Appendix B Example Database Hash

Table B.1 shows how the variable length data *m* for calculating the Database Hash is constructed from an example GATT database. The column *Included in Hash* indicates whether the *Attribute Handle* (H), *Attribute Type* (T), or *Attribute Value* (V) are included in *m*.

| Attribute Handle | Attribute Type | Attribute Value | Notes | Included in Hash | Data Included in m |
|---|---|---|---|---|---|
| 0x0001 | 0x2800 | 0x1800 | Primary Service: GAP Service | HTV | 01 00 00 28 00 18 |
| 0x0002 | 0x2803 | {0x0A, 0x0003, 0x2A00} | Characteristic (Read, Write): Device Name | HTV | 02 00 03 28 0A 03 00 00 2A |
| 0x0003 | 0x2A00 | *any* | Characteristic Value: Device Name | No | *none* |
| 0x0004 | 0x2803 | {0x02, 0x0005, 0x2A01} | Characteristic (Read): Appearance | HTV | 04 00 03 28 02 05 00 01 2A |
| 0x0005 | 0x2A01 | *any* | Characteristic Value: Appearance | No | *none* |
| 0x0006 | 0x2800 | 0x1801 | Primary Service: GATT Service | HTV | 06 00 00 28 01 18 |
| 0x0007 | 0x2803 | {0x20, 0x0008, 0x2A05} | Characteristic (Indicate): Service Changed | HTV | 07 00 03 28 20 08 00 05 2A |
| 0x0008 | 0x2A05 | *any* | Characteristic Value: Service Changed | No | *none* |
| 0x0009 | 0x2902 | 0x0002 | Client Characteristic Configuration Descriptor | HT | 09 00 02 29 |
| 0x000A | 0x2803 | {0x0A, 0x000B, 0x2B29} | Characteristic (Read, Write): Client Supported Features | HTV | 0A 00 03 28 0A 0B 00 29 2B |
| 0x000B | 0x2B29 | *any* | Characteristic Value: Client Supported Features | No | *none* |
| 0x000C | 0x2803 | {0x02, 0x000D, 0x2B2A} | Characteristic (Read): Database Hash | HTV | 0C 00 03 28 02 0D 00 2A 2B |
| 0x000D | 0x2B2A | *any* | Characteristic Value: Database Hash | No | *none* |

| Attribute Handle | Attribute Type | Attribute Value | Notes | Included in Hash | Data Included in m |
|---|---|---|---|---|---|
| 0x000E | 0x2800 | 0x1808 | Primary Service: Glucose Service | HTV | 0E 00 00 28 08 18 |
| 0x000F | 0x2802 | {0x0014, 0x0016, 0x180F} | Included Service: Battery Service | HTV | 0F 00 02 28 14 00 16 00 0F 18 |
| 0x0010 | 0x2803 | {0xA2, 0x0011, 0x2A18} | Characteristic (Read, Indicate, Extended Properties): Glucose Measurement | HTV | 10 00 03 28 A2 11 00 18 2A |
| 0x0011 | 0x2A18 | *any* | Characteristic Value: Glucose Measurement | No | *none* |
| 0x0012 | 0x2902 | 0x0002 | Client Characteristic Configuration Descriptor | HT | 12 00 02 29 |
| 0x0013 | 0x2900 | 0x0000 | Extended Properties | HTV | 13 00 00 29 00 00 |
| 0x0014 | 0x2801 | 0x180F | Secondary Service: Battery Service | HTV | 14 00 01 28 0F 18 |
| 0x0015 | 0x2803 | {0x02, 0x0016, 0x2A19} | Characteristic (Read): Battery Level | HTV | 15 00 03 28 02 16 00 19 2A |
| 0x0016 | 0x2A19 | *any* | Characteristic Value: Battery Level | No | *none* |

*Table B.1: Example database*

The resulting variable length data m divided into blocks M0 to M6 is:

M0: 01000028 00180200 03280A03 00002A04
M1: 00032802 0500012A 06000028 01180700
M2: 03282008 00052A09 0002290A 0003280A
M3: 0B00292B 0C000328 020D002A 2B0E0000
M4: 2808180F 00022814 0016000F 18100003
M5: 28A21100 182A1200 02291300 00290000
M6: 14000128 0F181500 03280216 00192A

The resulting Database Hash is (MSB to LSB):

Database Hash = AES-CMAC$_k$(m) = F1 CA 2D 48 EC F5 8B AC 8A 88 30 BB B9 FB A9 90

Note: The bytes in M0 to M6 and the Database Hash are ordered from the most significant on the left to the least significant on the right.

Host
Part H

# SECURITY MANAGER SPECIFICATION

*The Security Manager (SM) defines the protocol and behavior to manage pairing, authentication, and encryption between LE-only or BR/EDR/LE devices.*

Bluetooth SIG Proprietary

# CONTENTS

**1 Introduction ..... 1635**
1.1 Scope ..... 1635
1.2 Conventions ..... 1635
1.2.1 Bit and byte ordering conventions ..... 1635
1.2.2 Random numbers ..... 1636
1.3 Requirements ..... 1636

**2 Security Manager ..... 1637**
2.1 Introduction ..... 1637
2.2 Cryptographic toolbox ..... 1639
2.2.1 Security function *e* ..... 1640
2.2.2 Random address hash function *ah* ..... 1640
2.2.3 Confirm value generation function *c1* for LE legacy pairing ..... 1641
2.2.4 Key generation function *s1* for LE legacy pairing ..... 1642
2.2.5 Function AES-CMAC ..... 1643
2.2.6 LE Secure Connections confirm value generation function *f4* ..... 1643
2.2.7 LE Secure Connections key generation function *f5* ... 1644
2.2.8 LE Secure Connections check value generation function *f6* ..... 1646
2.2.9 LE Secure Connections numeric comparison value generation function *g2* ..... 1647
2.2.10 Link key conversion function *h6* ..... 1648
2.2.11 Link key conversion function *h7* ..... 1648
2.3 Pairing methods ..... 1649
2.3.1 Security Properties ..... 1650
2.3.2 IO capabilities ..... 1651
2.3.3 OOB authentication data ..... 1652
2.3.4 Encryption key size ..... 1652
2.3.5 Pairing algorithms ..... 1653
2.3.5.1 Selecting key generation method ..... 1653
2.3.5.2 LE legacy pairing - Just Works ..... 1656
2.3.5.3 LE legacy pairing - Passkey Entry ..... 1657
2.3.5.4 Out of band ..... 1658
2.3.5.5 LE legacy pairing phase 2 ..... 1658
2.3.5.6 LE Secure Connections pairing phase 2 .... 1660
2.3.5.7 Cross-transport key derivation ..... 1668
2.3.6 Repeated attempts ..... 1669

2.4 Security in Bluetooth Low Energy ..... 1669
2.4.1 Definition of keys and values ..... 1669
2.4.2 Generation of distributed keys ..... 1670
2.4.2.1 Generation of IRK ..... 1670
2.4.2.2 [This section is no longer used] ..... 1670
2.4.2.3 LE legacy pairing - generation of LTK, EDIV and Rand ..... 1670
2.4.2.4 Derivation of BR/EDR link key from LE LTK ..... 1671
2.4.2.5 Derivation of LE LTK from BR/EDR link key ..... 1671
2.4.3 Distribution of keys ..... 1672
2.4.3.1 LE legacy pairing key distribution ..... 1672
2.4.3.2 LE Secure Connections key distribution ..... 1672
2.4.4 Encrypted session setup ..... 1673
2.4.4.1 Encryption setup using STK ..... 1673
2.4.4.2 Encryption setup using LTK ..... 1673
2.4.5 [This section is no longer used] ..... 1674
2.4.6 Peripheral Security Request ..... 1674

**3 Security Manager Protocol ..... 1677**
3.1 Introduction ..... 1677
3.2 Security Manager Channel over L2CAP ..... 1677
3.3 Command format ..... 1677
3.4 SMP timeout ..... 1678
3.5 Pairing methods ..... 1679
3.5.1 Pairing Request ..... 1679
3.5.2 Pairing Response ..... 1682
3.5.3 Pairing Confirm ..... 1684
3.5.4 Pairing Random ..... 1685
3.5.5 Pairing Failed ..... 1686
3.5.6 Pairing Public Key ..... 1687
3.5.7 Pairing DHKey Check ..... 1688
3.5.8 Keypress Notification ..... 1689
3.6 Security in Bluetooth Low Energy ..... 1689
3.6.1 Key distribution and generation ..... 1689
3.6.2 Encryption Information ..... 1692
3.6.3 Central Identification ..... 1693
3.6.4 Identity Information ..... 1694
3.6.5 Identity Address Information ..... 1694
3.6.6 [This section is no longer used] ..... 1695
3.6.7 Security Request ..... 1695

**4 References ..... 1697**

**Appendix A EDIV and Rand Generation ..... 1698**
A.1 EDIV masking ..... 1698
A.1.1 DIV mask generation function *dm* ..... 1698
A.1.2 EDIV generation ..... 1699
A.1.3 DIV recovery ..... 1699

**Appendix B Key Management ..... 1700**
B.1 Database lookup ..... 1700
B.2 Key hierarchy ..... 1700
B.2.1 Diversifying function *d1* ..... 1701
B.2.2 Generating keys from ER ..... 1702
B.2.3 Generating keys from IR ..... 1702

**Appendix C Message sequence charts ..... 1704**
C.1 Phase 1: Pairing feature exchange ..... 1704
C.1.1 Peripheral security request – Central requests pairing ..... 1705
C.2 Phase 2: Authenticating and encrypting ..... 1705
C.2.1 LE legacy pairing ..... 1705
C.2.1.1 Legacy Phase 2: Short Term Key generation – Just Works ..... 1706
C.2.1.2 Legacy Phase 2: Short Term Key generation – Passkey Entry ..... 1707
C.2.1.3 Legacy Phase 2: Short Term Key generation – Out of Band ..... 1708
C.2.2 LE Secure Connections ..... 1708
C.2.2.1 Public key exchange ..... 1708
C.2.2.2 Authentication stage 1 ..... 1709
C.2.2.3 Long Term Key calculation ..... 1720
C.2.2.4 Authentication stage 2 (DHKey checks) ..... 1720
C.3 Phase 3: Transport specific key distribution ..... 1722
C.4 Security re-established using previously distributed LTK ..... 1722
C.4.1 Central initiated security - Central initiated Link Layer encryption ..... 1722
C.4.2 Peripheral security request - Central initiated Link Layer encryption ..... 1722
C.5 Failure conditions ..... 1723
C.5.1 Pairing not supported by Peripheral ..... 1723
C.5.2 Central rejects pairing because of key size ..... 1723
C.5.3 Peripheral rejects pairing because of key size ..... 1724
C.5.4 Passkey Entry failure on Central ..... 1725

C.5.5 Passkey Entry failure on Peripheral ..... 1726
C.5.6 Peripheral rejects Central's confirm value ..... 1726
C.5.7 Central rejects Peripheral's confirm value ..... 1727
C.6 Cross-transport key derivation ..... 1728
C.6.1 LE derivation from BR/EDR ..... 1728
C.6.2 BR/EDR derivation from LE ..... 1729

**Appendix D Sample data ..... 1731**
D.1 AES-CMAC RFC4493 test vectors ..... 1731
D.1.1 Example 1: Len = 0 ..... 1731
D.1.2 Example 2: Len = 16 ..... 1731
D.1.3 Example 3: Len = 40 ..... 1731
D.1.4 Example 4: Len = 64 ..... 1731
D.2 *f4* LE SC confirm value generation function ..... 1731
D.3 *f5* LE SC key generation function ..... 1732
D.4 *f6* LE SC check value generation function ..... 1732
D.5 *g2* LE SC numeric comparison generation function ..... 1733
D.6 *h6* LE SC link key conversion function ..... 1733
D.7 *ah* random address hash functions ..... 1733
D.8 *h7* LE SC link key conversion function ..... 1733
D.9 LTK to link key conversion using CT2=1 ..... 1733
D.10 LTK to link key conversion using CT2=0 ..... 1733
D.11 Link key to LTK conversion using CT2=1 ..... 1734
D.12 Link key to LTK conversion using CT2=0 ..... 1734

# 1 INTRODUCTION

## 1.1 Scope

The Security Manager defines methods of pairing and key distribution, a protocol for those methods and a security toolbox to be used by those methods and other parts of an LE-only or BR/EDR/LE device.

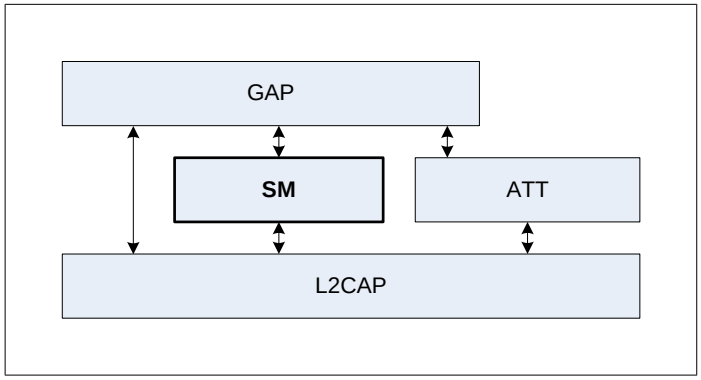



*Figure 1.1: Relationship of the Security Manager to the rest of the LE Bluetooth architecture*

The document describes Central and Peripheral roles in terms of protocol and requirements; these have the same meaning and are mapped to the LE device roles described in [Vol 1] Part A, Section 1.2 or BR/EDR device roles (see [Vol 1] Part A, Section 1.1).

## 1.2 Conventions

### 1.2.1 Bit and byte ordering conventions

When multiple bit fields are contained in a single octet and represented in a drawing in this Part, the least significant (low-order) bits are shown toward the left and most significant (high-order) bits toward the right.

Multiple-octet fields are drawn with the least significant octets toward the left and the most significant octets toward the right. Multiple-octet fields shall be transmitted with the least significant octet first.

Multiple-octet values written in hexadecimal notation have the most significant octet towards the left and the least significant octet towards the right, for example if ‘12’ is the most significant octet and ‘34’ is the least significant octet it would be written as 0x1234.

### 1.2.2 Random numbers

In this Part, the term "random number" includes pseudo-random numbers. Random number generators shall conform to the requirements of [Vol 2] Part H, Section 2.

## 1.3 Requirements

A BR/EDR/LE device that supports SM may support cross-transport key derivation (see Section 2.3.5.7).

# 2 SECURITY MANAGER

## 2.1 Introduction

The Security Manager (SM) uses a key distribution approach to perform identity and encryption functionalities in radio communication. This means that each device generates and controls the keys it distributes and no other device affects the generation of these keys. The strength of a key is as strong as the algorithms implemented inside the distributing device.

The security architecture is designed such that memory and processing requirements for a responding device are lower than the memory and processing requirement for an initiating device.

Pairing is performed to establish keys which can then be used to encrypt a link. A transport-specific key distribution is then performed to share the keys, which can be used to encrypt a link in future reconnections and to resolve private addresses.

Pairing is a three-phase process. The first two phases are always used and may be followed by an optional transport specific key distribution phase (see Figure 2.1):

- Phase 1: Pairing Feature Exchange
- Phase 2 (LE legacy pairing): Short Term Key (STK) Generation
- Phase 2 (LE Secure Connections): Long Term Key (LTK) Generation
- Phase 3: Transport Specific Key Distribution

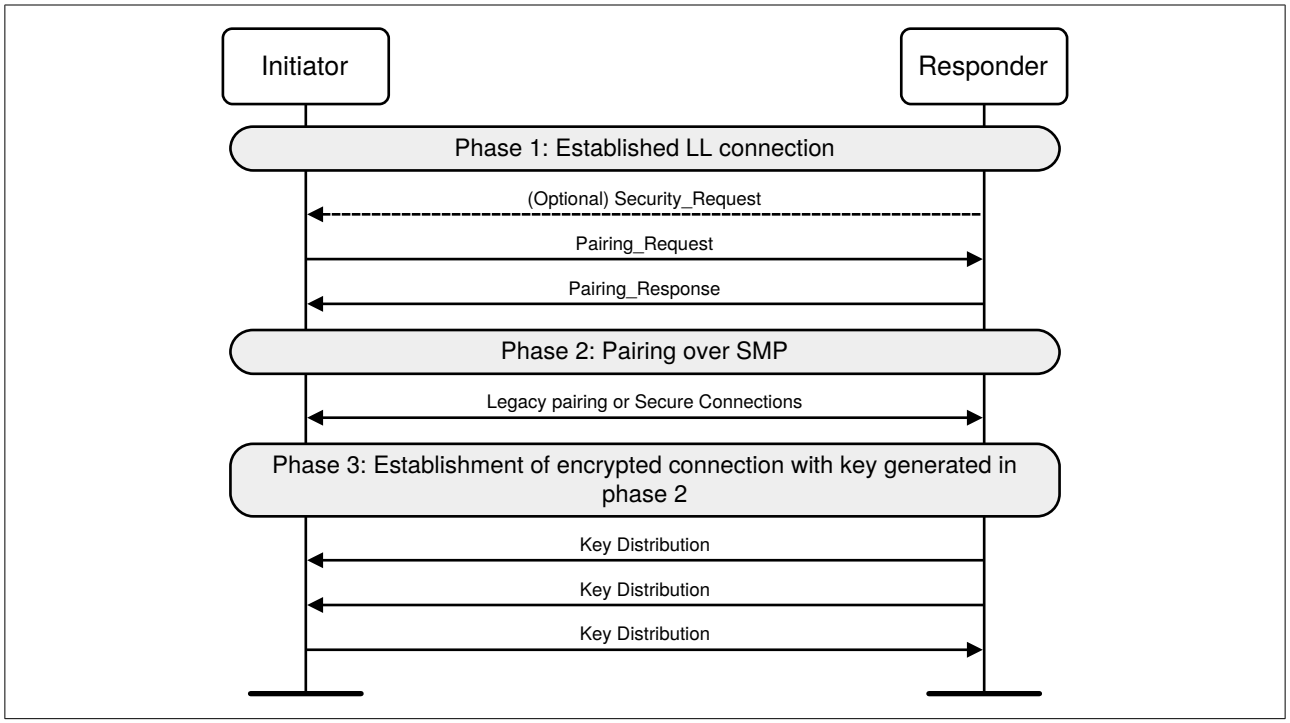



*Figure 2.1: LE pairing phases*

The devices shall first exchange authentication requirements and IO capabilities in the Pairing Feature Exchange to determine which of the following methods shall be used in Phase 2:

- Just Works
- Numeric Comparison (Only for LE Secure Connections)
- Passkey Entry
- Out Of Band (OOB)

Authentication requirements retrieved from the Pairing Feature Exchange also determine whether LE Secure Connections or LE legacy pairing is used.

Phase 3 may then be performed to distribute transport specific keys, for example the Identity Resolving Key (IRK) value and Identity Address information. Phase 1 and Phase 3 are identical regardless of the method used in Phase 2.

Phase 3 shall only be performed on a link which is encrypted using:

- The STK generated in Phase 2 when using LE legacy pairing or
- The LTK generated in Phase 2 when using LE Secure Connections or
- The shared Link Key generated using BR/EDR pairing (see Section 2.3.5.7).

Phase 1 and Phase 2 may be performed on a link which is either encrypted or not encrypted.

## 2.2 Cryptographic toolbox

In order to support random addressing, pairing and other operations SM provides a toolbox of cryptographic functions. The following cryptographic functions are defined:

- *ah* is used to create a 24-bit hash used in random address creation and resolution.

The following cryptographic functions are defined to support the LE legacy pairing process:

- *c1* is used to generate confirm values used during the pairing process.
- *s1* is used to generate the STK during the pairing process.

The following cryptographic functions are defined to support the LE Secure Connections pairing process:

- *f4* is used to generate confirm values during the pairing process.
- *f5* is used to generate the LTK and the MacKey during the pairing process.
- *f6* is used to generate the check values during authentication stage 2 in the pairing process.
- *g2* is used to generate the 6-digit numeric comparison values during authentication stage 1 in the pairing process.
- *h6* is used to generate the LE LTK from a BR/EDR link key derived from Secure Connections and is used to generate the BR/EDR link key from an LE LTK derived from Secure Connections.
- *h7* is used to generate intermediate keys while generating the LE LTK from a BR/EDR link key derived from Secure Connections and the BR/EDR link key from an LE LTK derived from Secure Connections.

The building block for the cryptographic functions *ah*, *c1* and *s1* is the security function *e*.

The building block for the cryptographic functions *f4, f5, f6, g2, h6,* and *h7* is the security function AES-CMAC.

Inside the *f4, f5, f6, g2, h6,* and *h7* functions when a multi-octet integer parameter is used as input to AES-CMAC the most significant octet of the integer shall be the first octet of the stream and the least significant octet of the integer shall be the last octet of the stream. The output of AES-CMAC inside these functions is a multi-octet integer where the first octet is MSB and the last octet is LSB of this integer.

### 2.2.1 Security function *e*

Security function *e* generates 128-bit *encryptedData* from a 128-bit key and 128-bit *plaintextData* using the AES-128-bit block cypher as defined in FIPS-197[1]:

*encryptedData* = *e*(*key*, *plaintextData*)

The most significant octet of *key* corresponds to key[0], the most significant octet of *plaintextData* corresponds to in[0] and the most significant octet of *encryptedData* corresponds to out[0] using the notation specified in FIPS-197[1].

Note: The security function *e* can be implemented in a Host or be implemented using the HCI_LE_Encrypt command (see [Vol 4] Part E, Section 7.8.22).

### 2.2.2 Random address hash function *ah*

The random address hash function *ah* is used to generate a hash value that is used in resolvable private addresses, see [Vol 3] Part C, Section 10.8.2.

The following are inputs to the random address hash function *ah*:

*k* is 128 bits
*r* is 24 bits
*padding* is 104 bits

*r* is concatenated with *padding* to generate *r'* which is used as the 128-bit input parameter *plaintextData* to security function *e*:

r' = padding || r

The least significant octet of *r* becomes the least significant octet of *r'* and the most significant octet of *padding* becomes the most significant octet of *r'*.

For example, if the 24-bit value *r* is 0x423456 then *r'* is 0x000000000000000000000000423456.

The output of the random address function *ah* is:

*ah*(k, r) = *e*(k, r') *mod* $2^{24}$

The output of the security function *e* is then truncated to 24 bits by taking the least significant 24 bits of the output of *e* as the result of *ah*.

[1]NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf)

### 2.2.3 Confirm value generation function *c1* for LE legacy pairing

During the LE legacy pairing process confirm values are exchanged. This confirm value generation function *c1* is used to generate the confirm values.

The following are inputs to the confirm value generation function *c1*:

*k* is 128 bits
*r* is 128 bits
*pres* is 56 bits
*preq* is 56 bits
*iat* is 1 bit
*ia* is 48 bits
*rat* is 1 bit
*ra* is 48 bits
*padding* is 32 zero bits

*iat* is concatenated with 7 zero bits to create *iat'* which is 8 bits in length. *iat* is the least significant bit of *iat'*.

*rat* is concatenated with 7 zero bits to create *rat'* which is 8 bits in length. *rat* is the least significant bit of *rat'*.

*pres*, *preq*, *rat'* and *iat'* are concatenated to generate *p1* which is XORed with *r* and used as 128-bit input parameter *plaintextData* to security function *e*:

p1 = pres || preq || rat' || iat'

The octet of *iat'* becomes the least significant octet of *p1* and the most significant octet of *pres* becomes the most significant octet of *p1*.

For example, if the 8-bit *iat'* is 0x01, the 8-bit *rat'* is 0x00, the 56-bit *preq* is 0x07071000000101 and the 56 bit *pres* is 0x05000800000302 then *p1* is 0x05000800000302070710000001010001.

*ra* is concatenated with *ia* and *padding* to generate *p2* which is XORed with the result of the security function *e* using *p1* as the input parameter *plaintextData* and is then used as the 128-bit input parameter *plaintextData* to security function *e*:

p2 = padding || ia || ra

The least significant octet of *ra* becomes the least significant octet of *p2* and the most significant octet of *padding* becomes the most significant octet of *p2*.

For example, if 48-bit *ia* is 0xA1A2A3A4A5A6 and the 48-bit *ra* is 0xB1B2B3B4B5B6 then *p2* is 0x00000000A1A2A3A4A5A6B1B2B3B4B5B6.

The output of the confirm value generation function *c1* is:

c1 (k, r, preq, pres, iat, rat, ia, ra) = e(k, e(k, r ⊕ p1) ⊕ p2)

The 128-bit output of the security function *e* is used as the result of confirm value generation function *c1*.

For example, if the 128-bit *k* is 0x00000000000000000000000000000000, the 128-bit value *r* is 0x5783D52156AD6F0E6388274EC6702EE0, the 128-bit value *p1* is 0x05000800000302070710000001010001 and the 128-bit value *p2* is 0x00000000A1A2A3A4A5A6B1B2B3B4B5B6 then the 128-bit output from the *c1* function is 0x1E1E3FEF878988EAD2A74DC5BEF13B86.

### 2.2.4 Key generation function *s1* for LE legacy pairing

The key generation function *s1* is used to generate the STK during the LE legacy pairing process.

The following are inputs to the key generation function *s1*:

*k* is 128 bits
*r1* is 128 bits
*r2* is 128 bits

The most significant 64-bits of r1 are discarded to generate r1' and the most significant 64-bits of r2 are discarded to generate r2'.

For example if the 128-bit value r1 is 0x000F0E0D0C0B0A091122334455667788 then r1' is 0x1122334455667788. If the 128-bit value r2 is 0x010203040506070899AABBCCDDEEFF00 then r2' is 0x99AABBCCDDEEFF00.

*r1'* is concatenated with *r2'* to generate *r'* which is used as the 128-bit input parameter *plaintextData* to security function *e*:

*r'* = *r1'* || *r2'*

The least significant octet of *r2'* becomes the least significant octet of r' and the most significant octet of *r1'* becomes the most significant octet of *r'*.

For example, if the 64-bit value *r1'* is 0x1122334455667788 and *r2'* is 0x99AABBCCDDEEFF00 then *r'* is 0x11223344556778899AABBCCDDEEFF00.

The output of the key generation function *s1* is:

*s1*(k, *r1*, *r2*) = e(k, r')

The 128-bit output of the security function e is used as the result of key generation function *s1*.

For example if the 128-bit value *k* is

0x00000000000000000000000000000000

and the 128-bit value *r'* is

0x11223344556677889AABBCCDDEEFF00

then the output from the key generation function *s1* is

0x9a1fe1f0e8b0f49b5b4216ae796da062.

### 2.2.5 Function AES-CMAC

RFC-4493[1] defines the Cipher-based Message Authentication Code (CMAC) that uses AES-128 as the block cipher function, also known as AES-CMAC.

The inputs to AES-CMAC are:

*m* is the variable length data to be authenticated
*k* is the 128-bit key

The 128-bit message authentication code (MAC) is generated as follows:[2]

$MAC = AES\text{-}CMAC_k(m)$

A device can implement AES functions in the Host or can use the HCI_LE_Encrypt command (see [Vol 4] Part E, Section 7.8.22) in order to use the AES function in the Controller.

### 2.2.6 LE Secure Connections confirm value generation function *f4*

During the LE Secure Connections pairing process, confirm values are exchanged. These confirm values are computed using the confirm value generation function *f4*.

This confirm value generation function makes use of the MAC function $AES\text{-}CMAC_X$, with a 128-bit key X.

The inputs to function f4 are:

U is 256 bits
V is 256 bits

[1] http://www.ietf.org/rfc/rfc4493.txt

[2] RFC4493 uses the notation MAC = AES-CMAC(k,m) where k is the key. This is functionally the same as the notation used in this specification $MAC = AES\text{-}CMAC_k(m)$

X is 128 bits
Z is 8 bits

Z is zero (i.e. 8 bits of zeros) for Numeric Comparison and OOB protocol. In the Passkey Entry protocol, the most significant bit of Z is set equal to one and the least significant bit is made up from one bit of the passkey e.g. if the passkey bit is 1, then Z = 0x81 and if the passkey bit is 0, then Z = 0x80.

U, V and Z are concatenated and used as input *m* to the function AES-CMAC and X is used as the key *k*.

The inputs to *f4* are different depending on different association models:

| **Numeric Comparison/ Just Works** | **Out-Of-Band** | **Passkey Entry** |
|---|---|---|
| Ca = *f4*(PKax, PKbx, Na, 0)<br>Cb = *f4*(PKbx, PKax, Nb, 0) | Ca = *f4*(PKax, PKax, ra, 0)<br>Cb = *f4*(PKbx, PKbx, rb, 0) | Cai = *f4*(PKax, PKbx, Nai, rai)<br>Cbi = *f4*(PKbx, PKax, Nbi, rbi) |

*Table 2.1: Inputs to f4 for the different protocols*

PKax denotes the x-coordinate of the public key PKa of A.

Similarly, PKbx denotes the x-coordinate of the public key PKb of B.

Nai is the nonce value of $i^{th}$ round. For each round Nai value is a new 128-bit number. Similarly, rai is a one bit value of the passkey expanded to 8 bits (either 0x80 or 0x81).

Na and Nb are nonces from Devices A and B. ra and rb are random values generated by devices A and B.

The output of the confirm value generation function *f4* is as follows:

$f4(U, V, X, Z) = \text{AES-CMAC}_X (U \,||\, V \,||\, Z)$

The least significant octet of Z becomes the least significant octet of the AES-CMAC input message *m* and the most significant octet of U becomes the most significant octet of the AES-CMAC input message *m*.

### 2.2.7 LE Secure Connections key generation function *f5*

The LE Secure Connections key generation function *f5* is used to generate derived keying material in order to create the LTK and keys for the commitment function f6 during the LE Secure Connections pairing process.

The definition of this key generation function makes use of the MAC function $\text{AES-CMAC}_T$ with a 128-bit key T.

The inputs to function *f5* are:

W is 256 bits
$N_1$ is 128 bits
$N_2$ is 128 bits
$A_1$ is 56 bits
$A_2$ is 56 bits

The key (T) is computed as follows:

$T = AES\text{-}CMAC_{SALT}(W)$

SALT is the 128-bit value:

`0x6C888391_AAF5A538_60370BDB_5A6083BE`

Counter, keyID, N1, N2, A1, A2, and Length are concatenated and used as input *m* to the function AES-CMAC and T is used as the key *k*.

Counter is one octet. Length is two octets.

The string “btle” is mapped into a keyID using ASCII as 0x62746C65.

The output of the key generation function f5 is as follows:

$f5(W, N1, N2, A1, A2) = AES\text{-}CMAC_T$ (Counter = 0 || keyID || N1 || N2 || A1 || A2 || Length = 256) || $AES\text{-}CMAC_T$ (Counter = 1 || keyID || N1 || N2 || A1 || A2 || Length = 256)

The least significant octet of Length becomes the least significant octet of the AES-CMAC input message *m* and the most significant octet of Counter becomes the most significant octet of the AES-CMAC input message *m*.

The LTK and MacKey are calculated as:

MacKey || LTK = f5(DHKey, Nc, Np, BD_ADDR_C, BD_ADDR_P)

DHKey is the shared secret Diffie-Hellman key generated during LE Secure Connections pairing phase 2.

Nc is whichever of N1 and N2 was generated by the Central and sent to the Peripheral; Np is the other.

BD_ADDR_C is the device address of the Central and BD_ADDR_P is the device address of the Peripheral. The device addresses are the values used during connection setup. The least significant bit in the most significant octet in both BD_ADDR_C and

BD_ADDR_P is set to 1 if the address is a random address and set to 0 if the address is a public address. The 7 most significant bits of the most significant octet in both BD_ADDR_C and BD_ADDR_P are set to 0.

The LTK is the least significant 128 bits (Counter = 1) of *f5*. The MacKey (see Section 2.2.8) is the most significant 128 bits (Counter = 0) of *f5*.

A device can implement Diffie-Hellman key generation in the Host or can use the HCI_LE_Generate_DHKey command (see [Vol 4] Part E, Section 7.8.37) to generate the key in the Controller.

Note: When using the HCI_LE_Generate_DHKey command, the device can only pair one remote device at a time.

### 2.2.8 LE Secure Connections check value generation function *f6*

The LE Secure Connections check value generation function *f6* is used to generate check values during authentication stage 2 of the LE Secure Connections pairing process.

The definition of the *f6* function makes use of the MAC function AES-CMAC$_W$ with a 128-bit key W.

The inputs to function *f6* are:

W is 128 bits
N1 is 128 bits
N2 is 128 bits
R is 128 bits
IOcap is 24 bits
A1 is 56 bits
A2 is 56 bits

N1, N2, R, IOcap, A1 and A2 are concatenated and used as input *m* to the function AES-CMAC and W is used as the key *k*.

The inputs to f6 are different depending on different association models:

| **Numeric Comparison/ Just Works** | **Out-Of-Band** | **Passkey Entry** |
|---|---|---|
| Ea = *f6*(MacKey, Na, Nb, 0, IOcapA, A, B) | Ea = *f6*(MacKey, Na, Nb, rb, IOcapA, A, B) | Ea = *f6*(MacKey, Na20, Nb20, rb, IOcapA, A, B) |
| Eb = *f6*(MacKey, Nb, Na, 0, IOcapB, B, A) | Eb = *f6*(MacKey, Nb, Na, ra, IOcapB, B, A) | Eb = *f6*(MacKey, Nb20, Na20, ra, IOcapB, B, A) |

*Table 2.2: Inputs to f6 for the different protocols*

MacKey is the 128-bit MSBs of the output of f5.

Na is the random number sent by the Central to the Peripheral and Nb is the random number sent by the Peripheral to the Central.

IOcapA is the capabilities of the Central and IOcapB is the capabilities of the Peripheral. IOcapA and IOcapB are both three octets with the most significant octet as the AuthReq parameter, the middle octet as the OOB data flag and the least significant octet as the IO capability parameter. The AuthReq, OOB data flag and IO capability parameters are present in the Pairing Request and Pairing Response SMP packets.

In Passkey Entry, ra and rb are 6-digit passkey values expressed as a 128-bit integer. For instance, if the 6-digit value of ra is 131313 then

ra = 0x 00 00 00 00 00 00 00 00 00 00 00 00 02 00 f1

A is the device address of the Central and B is the device address of the Peripheral. The least significant bit in the most significant octet in both A and B is set to 1 if the address is a random address and set to 0 if the address is a public address. The 7 most significant bits of the most significant octet in both A and B are set to 0.

The output of the check value generation function *f6* is as follows:

f6(W, N1, N2, R, IOcap, A1, A2) =
$AES\text{-}CMAC_W$ (N1 || N2 || R || IOcap || A1 || A2)

The least significant octet of A2 becomes the least significant octet of the AES-CMAC input message *m* and the most significant octet of N1 becomes the most significant octet of the AES-CMAC input message *m*.

### 2.2.9 LE Secure Connections numeric comparison value generation function *g2*

The LE Secure Connections numeric comparison value generation function *g2* is used to generate the numeric comparison values during authentication stage 1 of the LE Secure Connections pairing process.

The definition of *g2* makes use of the MAC function $AES\text{-}CMAC_X$ with 128-bit key X.

The inputs to function *g2* are:

U is 256 bits
V is 256 bits
X is 128 bits
Y is 128 bits

U, V, and Y are concatenated and used as input *m* to the function AES-CMAC and X is used as the key *k*.

The output of the numeric comparison value generation function *g2* is as follows:

$$g2(U, V, X, Y) = \text{AES-CMAC}_X(U \,||\, V \,||\, Y) \; mod \; 2^{32}$$

The least significant octet of Y becomes the least significant octet of the AES-CMAC input message *m* and the most significant octet of U becomes the most significant octet of the AES-CMAC input message *m*.

The numeric verification value is taken as the six least significant digits of the 32-bit integer *g2*(PKax, PKbx, Na, Nb) where PKax denotes the x-coordinate of the public key PKa of A and PKbx denotes the x-coordinate of the public key PKb of B. Na and Nb are nonces from devices A and B. The value is then converted to decimal numeric value. The checksum used for numeric comparison is the least significant six digits.

$$\text{Compare Value} = g2\,(U, V, X, Y) \; mod \; 10^6$$

For example, if output = 0x 01 2e b7 2a then decimal value = 19838762 and the checksum used for numeric comparison is 838762.

### 2.2.10 Link key conversion function *h6*

The function *h6* is used to convert keys of a given size from one key type to another key type with equivalent strength.

The definition of the *h6* function makes use of the hashing function $\text{AES-CMAC}_W$ with 128-bit key W.

The inputs to function *h6* are:

W is 128 bits
keyID is 32 bits

keyID is used as input *m* to the hashing function AES-CMAC and the most significant 128-bits of W are used as the key *k*.

The output of *h6* is as follows:

$$h6(W, \text{keyID}) = \text{AES-CMAC}_W(\text{keyID})$$

### 2.2.11 Link key conversion function *h7*

The function *h7* is used to convert keys of a given size from one key type to another key type with equivalent strength.

The definition of the *h7* function makes use of the hashing function $\text{AES-CMAC}_{SALT}$ with 128-bit key SALT.

The inputs to function *h7* are:

SALT is 128 bits
W is 128 bits

W is used as input *m* to the hashing function AES-CMAC and SALT is used as the key *k*.

The output of *h7* is as follows:

$h7(SALT, W) = AES\text{-}CMAC_{SALT}(W)$

## 2.3 Pairing methods

There are two types of pairing: LE legacy pairing and LE Secure Connections pairing. All Security Manager implementations shall support one of these and may support both.

When pairing is started, the Pairing Feature Exchange shall be initiated by the initiating device. If the responding device does not support pairing or pairing cannot be performed then the responding device shall respond using the Pairing Failed message with the error code “Pairing Not Supported” otherwise it responds with a Pairing Response message.

The Pairing Feature Exchange is used to exchange IO capabilities, OOB authentication data availability, authentication requirements, key size requirements and which transport specific keys to distribute. The IO capabilities, OOB authentication data availability and authentication requirements are used to determine the key generation method used in Phase 2.

All of the LE legacy pairing methods use and generate 2 keys:

1. Temporary Key (TK): a 128-bit temporary key used in the pairing process which is used to generate STK (see Section 2.3.5.5).
2. Short Term Key (STK): a 128-bit temporary key used to encrypt a connection following pairing.

The LE Secure Connections pairing methods use and generate 1 key:

1. Long Term Key (LTK): a 128-bit key used to encrypt the connection following pairing and subsequent connections.

Authentication requirements are set by GAP, (see [Vol 3] Part C, Section 10.3). The authentication requirements include the type of bonding and man-in-the-middle protection (MITM) requirements.

The initiating device indicates to the responding device which transport specific keys it would like to send to the responding device and which keys it would like the responding device to send to the initiator. The responding device replies with the keys that the initiating device shall send and the keys that the responding device shall send. The keys that can be distributed are defined in Section 2.4.3. If the device receives a command with invalid parameters, it shall respond with Pairing Failed command with the error code “Invalid Parameters.”

### 2.3.1 Security Properties

Security Properties provided by SM are classified into the following categories:

- LE Secure Connections pairing
- Authenticated MITM protection
- Unauthenticated no MITM protection
- No security requirements

LE Secure Connections pairing uses the P-256 elliptic curve (see [Vol 2] Part H, Section 7.6).

In LE legacy pairing, Authenticated man-in-the-middle (MITM) protection is obtained by using the out of band pairing method with 128 bits of OOB data entropy and an OOB channel resistant to eavesdropping and MITM attacks. In LE Secure Connections pairing, Authenticated man-in-the-middle (MITM) protection is obtained by using the passkey entry pairing method or the numeric comparison method or may be obtained using the out of band pairing method. To ensure that Authenticated MITM Protection is generated, the selected Authentication Requirements option must have MITM protection specified.

Unauthenticated no MITM Protection does not have protection against MITM attacks.

For LE Legacy Pairing, none of the pairing methods provide protection against a passive eavesdropper during the pairing process as predictable or easily established values for *TK* are used. If the pairing information is distributed without an eavesdropper being present then all the pairing methods provide confidentiality.

An initiating device shall maintain a record of the Security Properties for the distributed keys in a security database.

A responding device may maintain a record of the distributed key sizes and Security Properties for the distributed keys in a security database. Depending upon the key generation method and negotiated key size a responding device may have to shorten the key length (see Section 2.3.4) so that the initiator and responder are using identical keys.

Security Properties of the key generated in phase 2 under which the keys are distributed shall be stored in the security database.

### 2.3.2 IO capabilities

Input and output capabilities of a device are combined to generate its IO capabilities. The input capabilities are described in Table 2.3. The output capabilities are described in Table 2.4.

| Capability | Description |
|---|---|
| No input | Device does not have the ability to indicate ‘yes’ or ‘no’ |
| Yes / No | Device has at least two buttons that can be easily mapped to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note below). |
| Keyboard | Device has a numeric keyboard that can input the numbers '0' to '9' and a confirmation. Device also has at least two buttons that can be easily mapped to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note below). |

*Table 2.3: User input capabilities*

Note: 'yes' could be indicated by pressing a button within a certain time limit otherwise 'no' would be assumed.

| Capability | Description |
|---|---|
| No output | Device does not have the ability to display or communicate a 6 digit decimal number |
| Numeric output | Device has the ability to display or communicate a 6 digit decimal number |

*Table 2.4: User output capabilities*

The individual input and output capabilities are mapped to a single IO capability for that device which is used in the pairing feature exchange. The mapping is described in Table 2.5.

| | | **Local output capacity** | |
|---|---|---|---|
| | | **No output** | **Numeric output** |
| **Local input capacity** | **No input** | NoInputNoOutput | DisplayOnly |
| | **Yes/No** | NoInputNoOutput[1] | DisplayYesNo |
| | **Keyboard** | KeyboardOnly | KeyboardDisplay |

*Table 2.5: IO capabilities mapping*

[1]None of the pairing algorithms can use Yes/No input and no output, therefore NoInputNoOutput is used as the resulting IO capability.

### 2.3.3 OOB authentication data

An out of band mechanism may be used to communicate information which is used during the pairing process. The information shall be a sequence of AD structures (see [Vol 3] Part C, Section 11).

The OOB data flag shall be set if a device has the peer device's out of band authentication data. A device uses the peer device's out of band authentication data to authenticate the peer device. In LE legacy pairing, the out of band method is used if both the devices have the other device's out of band authentication data available. In LE Secure Connections pairing, the out of band method is used if at least one device has the peer device's out of band authentication data available.

### 2.3.4 Encryption key size

Each device shall have maximum and minimum encryption key length parameters which defines the maximum and minimum size of the encryption key allowed in octets. The maximum and minimum encryption key length parameters shall be between 7 octets (56 bits) and 16 octets (128 bits), in 1 octet (8 bit) steps. This is defined by a profile or device application.

The shorter of the initiating and responding devices' maximum encryption key length parameters shall be used as the encryption key size.

Both the initiating and responding devices shall check that the resultant encryption key size is not shorter than the minimum key size parameter for that device and if it is, the device shall send the Pairing Failed command with error code "Encryption Key Size".

The encryption key size may be stored so it can be checked by any service that has minimum encryption key length requirements.

If a key has an encryption key size that is shorter than 16 octets (128 bits), it shall be created by masking the appropriate number of most significant octets of the generated key to provide a resulting key that has the agreed encryption key size. The key shall be masked after generation and, if required, after the key is used to derive a BR/EDR link key. The key shall be masked before the key is distributed, used for encryption, or stored.

For example, if a 128-bit encryption key is

0x12345678_9ABCDEF0_12345678_9ABCDEF0

and it is reduced to 7 octets (56 bits), then the resulting key is

0x00000000_00000000_00345678_9ABCDEF0.

### 2.3.5 Pairing algorithms

The information exchanged in Phase 1 is used to select which key generation method is used in Phase 2.

When LE legacy pairing is used, the pairing is performed by each device generating a Temporary Key (*TK*). The method to generate *TK* depends upon the pairing method chosen using the algorithm described in Section 2.3.5.1. If Just Works is used then *TK* shall be generated as defined in Section 2.3.5.2. If Passkey Entry is used then *TK* shall be generated as defined in Section 2.3.5.3. If Out Of Band is used then *TK* shall be generated as defined in Section 2.3.5.4. The *TK* value shall be used in the authentication mechanism defined in Section 2.3.5.5 to generate the STK and encrypt the link.

#### 2.3.5.1 Selecting key generation method

If both devices have not set the MITM option in the Authentication Requirements Flags, then the IO capabilities shall be ignored and the Just Works association model shall be used.

In LE legacy pairing, if both devices have Out of Band authentication data, then the Authentication Requirements Flags shall be ignored when selecting the pairing method and the Out of Band pairing method shall be used. Otherwise, the IO capabilities of the device shall be used to determine the pairing method as defined in Table 2.8.

In LE Secure Connections pairing, if one or both devices have out of band authentication data, then the Authentication Requirements Flags shall be ignored when selecting the pairing method and the Out of Band pairing method shall be used. Otherwise, the IO capabilities of the device shall be used to determine the pairing method as defined in Table 2.8.

Table 2.6 defines the STK generation method when at least one of the devices does not support LE Secure Connections.

| | | | Initiator | | | |
|---|---|---|---|---|---|---|
| | | | OOB Set | | OOB Not Set | |
| | | | MITM Set | MITM Not Set | MITM Set | MITM Not Set |
| Responder | OOB Set | MITM Set | Use OOB | Use OOB | Use IO Capabilities | Use IO Capabilities |
| | | MITM Not Set | Use OOB | Use OOB | Use IO Capabilities | Use Just Works |
| | OOB Not Set | MITM Set | Use IO Capabilities | Use IO Capabilities | Use IO Capabilities | Use IO Capabilities |
| | | MITM Not Set | Use IO Capabilities | Use Just Works | Use IO Capabilities | Use Just Works |

*Table 2.6: Rules for using Out-of-Band and MITM flags for LE legacy pairing*

Table 2.7 defines the LTK generation method when both devices support LE Secure Connections.

| | | | Initiator | | | |
|---|---|---|---|---|---|---|
| | | | OOB Set | | OOB Not Set | |
| | | | MITM Set | MITM Not Set | MITM Set | MITM Not Set |
| Responder | OOB Set | MITM Set | Use OOB | Use OOB | Use OOB | Use OOB |
| | | MITM Not Set | Use OOB | Use OOB | Use OOB | Use OOB |
| | OOB Not Set | MITM Set | Use OOB | Use OOB | Use IO Capabilities | Use IO Capabilities |
| | | MITM Not Set | Use OOB | Use OOB | Use IO Capabilities | Use Just Works |

*Table 2.7: Rules for using Out-of-Band and MITM flags for LE Secure Connections pairing*

| | **Initiator** | | | | |
|---|---|---|---|---|---|
| **Responder** | **DisplayOnly** | **Display YesNo** | **Keyboard Only** | **NoInput NoOutput** | **Keyboard Display** |
| **Display Only** | Just Works<br>Unauthenticated | Just Works<br>Unauthenticated | Passkey Entry: responder displays, initiator inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Just Works<br>Unauthenticated | Passkey Entry: responder displays, initiator inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated |
| **Display YesNo** | Just Works<br>Unauthenticated | Just Works (For LE Legacy Pairing)<br>Unauthenticated | Passkey Entry: responder displays, initiator inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Just Works<br>Unauthenticated | Passkey Entry (For LE Legacy Pairing): responder displays, initiator inputs<br>Unauthenticated |
| | | Numeric Comparison (For LE Secure Connections)<br>Authenticated | | | Numeric Comparison (For LE Secure Connections)<br>Authenticated |
| **Keyboard Only** | Passkey Entry: initiator displays, responder inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Passkey Entry: initiator displays, responder inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Passkey Entry: initiator and responder input<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Just Works<br>Unauthenticated | Passkey Entry: initiator displays, responder inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated |
| **NoInput NoOutput** | Just Works<br>Unauthenticated | Just Works<br>Unauthenticated | Just Works<br>Unauthenticated | Just Works<br>Unauthenticated | Just Works<br>Unauthenticated |

| | Initiator | | | | |
|---|---|---|---|---|---|
| **Responder** | **DisplayOnly** | **Display YesNo** | **Keyboard Only** | **NoInput NoOutput** | **Keyboard Display** |
| **Keyboard Display** | Passkey Entry: initiator displays, responder inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections): Authenticated | Passkey Entry (For LE Legacy Pairing):<br>initiator displays, responder inputs<br>Unauthenticated | Passkey Entry: responder displays, initiator inputs<br>(For LE Legacy Pairing): Unauthenticated<br>(For LE Secure Connections):<br>Authenticated | Just Works<br>Unauthenticated | Passkey Entry (For LE Legacy Pairing):<br>initiator displays, responder inputs<br>Unauthenticated |
| | | Numeric Comparison (For LE Secure Connections)<br>Authenticated | | | Numeric Comparison (For LE Secure Connections)<br>Authenticated |

*Table 2.8: Mapping of IO capabilities to key generation method*

The generated key will either be an Authenticated or Unauthenticated key. If the out of band authentication method is used and the Out of Band mechanism is known to be secure from eavesdropping the key is assumed to be Authenticated; however, the exact strength depends upon the method used to transfer the out of band information. If the Out of Band method is used and the Out of Band mechanism is not secure from eavesdropping or the level of eavesdropping protection is unknown, the key shall be Unauthenticated. The mapping of IO capabilities to an authenticated or unauthenticated key is described in Table 2.8.

In LE legacy pairing, if the initiating device has Out of Band data and the responding device does not have Out of Band data then the responding device may send the Pairing Failed command with the error code “OOB Not Available” instead of the Pairing Response command.

If the key generation method does not result in a key that provides sufficient Security Properties (see Section 2.3.1) then the device shall send the Pairing Failed command with the error code “Authentication Requirements”.

#### 2.3.5.2 LE legacy pairing - Just Works

The Just Works STK generation method provides no protection against eavesdroppers or man in the middle attacks during the pairing process. If the attacker is not present

during the pairing process then confidentiality can be established by using encryption on a future connection.

Both devices set the TK value used in the authentication mechanism defined in Section 2.3.5.5 to zero.

#### 2.3.5.3 LE legacy pairing - Passkey Entry

The Passkey Entry STK generation method uses 6 numeric digits passed out of band by the user between the devices. A 6 digit numeric randomly generated passkey achieves approximately 20 bits of entropy.

If the IO capabilities of a device are DisplayOnly or if Table 2.8 defines that the device displays the passkey, then that device shall display a randomly generated passkey value between 000,000 and 999,999. The display shall ensure that all 6 digits are displayed – including zeros. The other device shall allow the user to input a value between 000,000 and 999,999.

If the IO capabilities of both devices are KeyboardOnly then the user generates a random 6-digit passkey value and enters it into both devices. Both devices shall allow the user to input a value between 000,000 and 999,999.

A new passkey shall be generated randomly for each pairing procedure.

If entry of passkey in UI fails to occur or is cancelled then the device shall send Pairing Failed command with reason code “Passkey Entry Failed”.

For example, if the user entered passkey is ‘019655’ then *TK* shall be 0x00000000000000000000000000004CC7.

The passkey Entry method does not provide protection against active “man-in-the-middle” (MITM) attacks.

The Passkey Entry STK generation method provides very limited protection against eavesdroppers during the pairing process because of the limited range of possible TK values which STK is dependent upon. If the attacker is not present during the pairing process then confidentiality and authentication can be established by using encryption on a future connection.

The TK value shall then be used in the authentication mechanism defined in Section 2.3.5.5.

#### 2.3.5.4 Out of band

An out of band mechanism may be used to communicate information to help with device discovery, for example device address, and the 128-bit *TK* value used in the pairing process. The *TK* value shall be a 128-bit random number.

If the OOB communication is resistant to MITM attacks, then this association method is also resistant to MITM attacks. Also, in the Out of Band method, the size of authentication parameter (*TK*) need not be restricted by what the user can comfortably read or type. For that reason, the Out of Band method can be more secure than using the Passkey Entry or Just Works methods. However, both devices need to have matching OOB interfaces.

MITM protection is only provided if an active man-in-the-middle chance of a successful attack has a probability of 0.000001 or less in succeeding.

#### 2.3.5.5 LE legacy pairing phase 2

The initiating device generates a 128-bit random number (LP_RAND_I).

The initiating device calculates the 128-bit confirm value (LP_CONFIRM_I) using the confirm value generation function *c1* (see Section 2.2.3) with the input parameter *k* set to *TK*, the input parameter *r* set to LP_RAND_I, the input parameter *preq* set to Pairing Request command as exchanged with the peer device (i.e. without any modifications), the input parameter *pres* set to the Pairing Response command as exchanged with the peer device (i.e. without any modifications), the input parameter *iat* set to the initiating device address type, *ia* set to the initiating device address, *rat* set to the responding device address type and *ra* set to the responding device address:

LP_CONFIRM_I = *c1*(*TK*, LP_RAND_I,
Pairing Request command, Pairing Response command,
initiating device address type, initiating device address,
responding device address type, responding device address)

Initiating and responding device addresses used for confirmation generation shall be device addresses used during connection setup, see [Vol 3] Part C, Section 9.3

The responding device generates a 128-bit random number (LP_RAND_R).

The responding device calculates the 128-bit confirm value (LP_CONFIRM_R) using the confirm value generation function c1 (see Section 2.2.3) with the input parameter *k* set to *TK*, the input parameter *r* set to LP_RAND_R, the input parameter *preq* set to Pairing Request command, the input parameter *pres* set to the Pairing Response command, the input parameter *iat* set to the initiating device address type, *ia* set to the

initiating device address, *rat* set to the responding device address type and *ra* set to the responding device address:

LP_CONFIRM_R = *c1*(*TK*, LP_RAND_R,
Pairing Request command, Pairing Response command,
initiating device address type, initiating device address,
responding device address type, responding device address)

The initiating device transmits LP_CONFIRM_I to the responding device. When the responding device receives LP_CONFIRM_I it transmits LP_CONFIRM_R to the initiating device. When the initiating device receives LP_CONFIRM_R it transmits LP_RAND_I to the responding device.

The responding device verifies the LP_CONFIRM_I value by repeating the calculation the initiating device performed, using the LP_RAND_I value received.

If the responding device's calculated LP_CONFIRM_I value does not match the received LP_CONFIRM_I value from the initiating device then the pairing process shall be aborted and the responding device shall send the Pairing Failed command with reason code "Confirm Value Failed".

If the responding device's calculated LP_CONFIRM_I value matches the received LP_CONFIRM_I value from the initiating device the responding device transmits LP_RAND_R to the initiating device.

The initiating device verifies the received LP_CONFIRM_R value by repeating the calculation the responding device performed, using the LP_RAND_R value received.

If the initiating device receives an LP_CONFIRM_R value that is equal to the LP_CONFIRM_I value or that is not equal to the calculated LP_CONFIRM_R value, then the pairing process shall be aborted and the initiating device shall send the Pairing Failed command with the reason code "Confirm Value Failed".

If the initiating device's calculated LP_CONFIRM_R value matches the received LP_CONFIRM_R value from the responding device the initiating device then calculates STK and tells the Controller to enable encryption.

STK is generated using the key generation function *s1* defined in Section 2.2.4 with the input parameter *k* set to *TK*, the input parameter r1 set to LP_RAND_R, and the input parameter *r2* set to LP_RAND_I:

STK = *s1*(*TK*, LP_RAND_R, LP_RAND_I)

If the encryption key size is shorter than 128 bits then the STK shall be masked to the correct key size as described in Section 2.3.4.

The initiator shall use the generated STK to either enable encryption on the link or if encryption has already been enabled, perform the encryption pause procedure (see Section 2.4.4.1).

#### 2.3.5.6 LE Secure Connections pairing phase 2

The Long Term Key is generated in LE Secure Connections pairing phase 2.

##### 2.3.5.6.1 Public key exchange

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (phase 1). The public-private key pair contains a private (secret) key, and a public key. The private keys of devices A and B are denoted as SKa and SKb respectively. The public keys of devices A and B are denoted as PKa and PKb respectively. See Section 2.3.6 for recommendations on how frequently this key pair should be changed.

Pairing is initiated by the initiating device sending its public key to the receiving device (phase 1a). The responding device replies with its own public key (phase 1b). If the two public keys have the same X coordinate and neither is the debug key, then each device shall fail pairing by sending the Pairing Failed message (not necessarily immediately) with the reason code "DHKey Check Failed". These public keys are not regarded as secret although they may identify the devices.

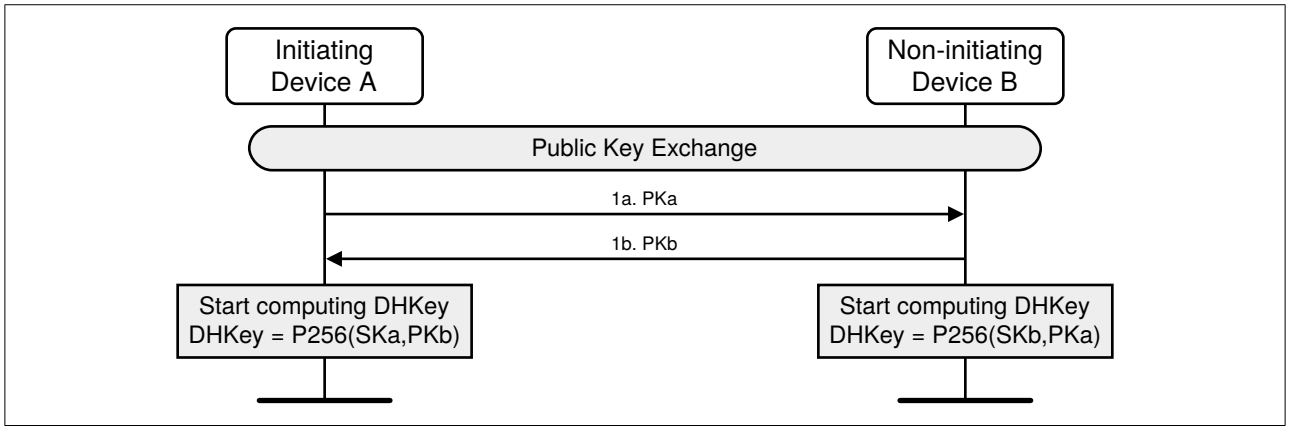



*Figure 2.2: Public key exchange*

A device shall validate that any public key received from any BD_ADDR is on the correct curve (P-256).

A valid public key Q = ($X_Q$, $Y_Q$) is one where $X_Q$ and $Y_Q$ are both in the range 0 to p - 1 and satisfy the equation $(Y_Q)^2 = (X_Q)^3 + aX_Q + b$ (*mod* p) in the relevant curve's finite field. See [Vol 2] Part H, Section 7.6 for the values of a, b, and p.

A device can validate a public key by directly checking the curve equation, by implementing elliptic curve point addition and doubling using formulas that are valid only on the correct curve, or by other means.

A device that detects an invalid public key from the peer at any point during the LE Secure Connections pairing process shall fail pairing by sending the Pairing Failed message (not necessarily immediately) with the reason code "DHKey Check Failed" and shall not use the resulting LTK, if any.

After the public keys have been exchanged, the device can then start computing the Diffie-Hellman Key.

When the Security Manager is placed in a Debug mode it shall use the following Diffie-Hellman private / public key pair:

```
Private key:   3f49f6d4 a3c55f38 74c9b3e3 d2103f50 4aff607b eb40b799 5899b8a6 cd3c1abd
Public key (X): 20b003d2 f297be2c 5e2c83a7 e9f9a5b9 eff49111 acf4fddb cc030148 0e359de6
Public key (Y): dc809c49 652aeb6d 63329abf 5a52155c 766345c2 8fed3024 741c8ed0 1589d28b
```

If a device receives this debug public key and it is in a mode in which it cannot accept the debug key then it may send the Pairing Failed command with the reason set to "Invalid Parameters".

Note: Only one side (initiator or responder) needs to set Secure Connections debug mode in order for debug equipment to be able to determine the LTK and, therefore, be able to monitor the encrypted connection.

#### 2.3.5.6.2 Authentication stage 1 – Just Works or Numeric Comparison

The Numeric Comparison association model will be used during pairing if the MITM bit is set to 1 in the Authentication Requirements in the Pairing Request PDU and/or Pairing Response PDU and both devices have IO capabilities set to either DisplayYesNo or KeyboardDisplay.

The sequence diagram of Authentication stage 1 for the Just Works or Numeric Comparison protocol from the cryptographic point of view is shown in Figure 2.3.

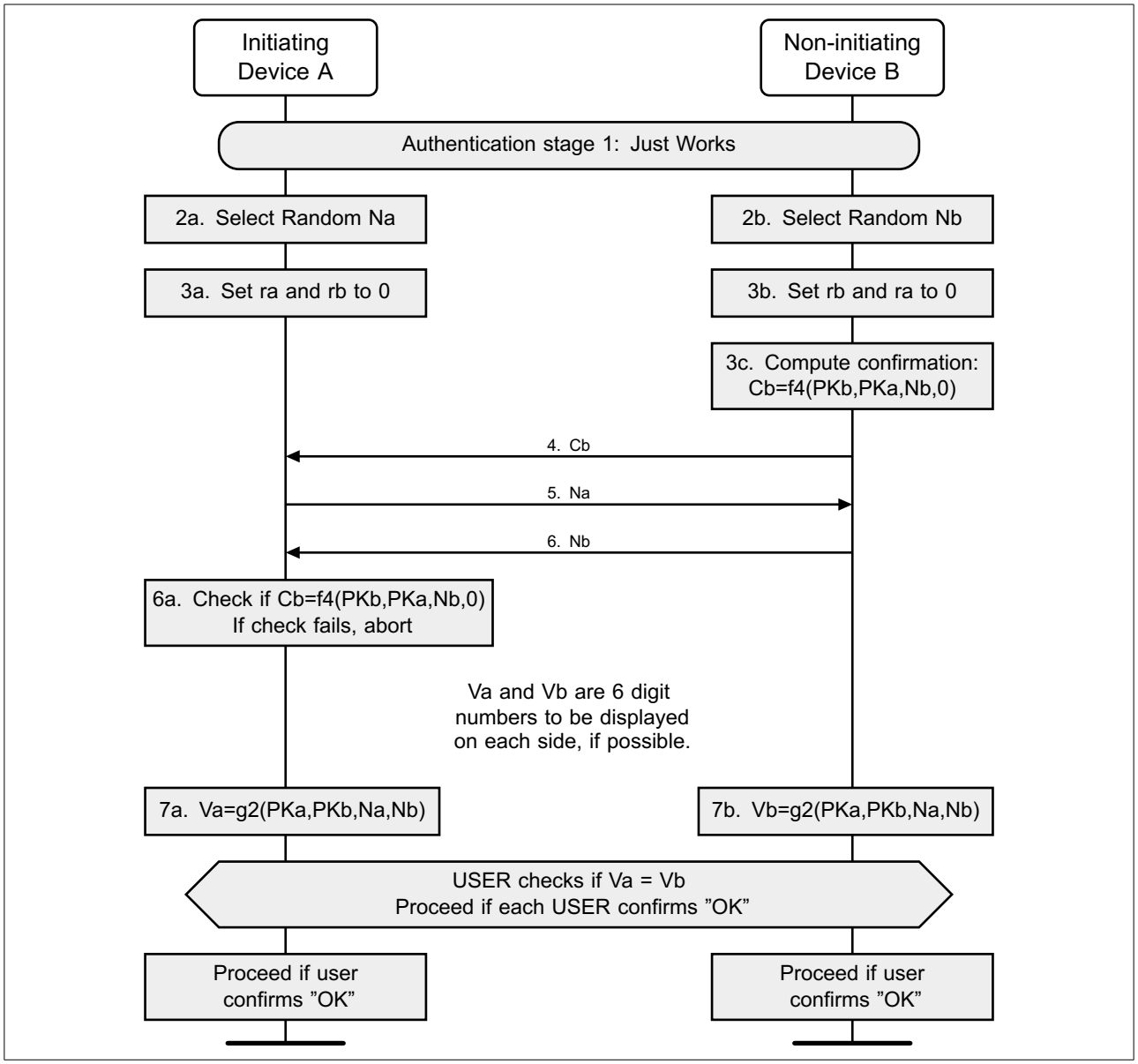



*Figure 2.3: "Authentication stage 1: Just Works or Numeric Comparison, LE Secure Connections*

After the public keys have been exchanged, each device selects a random 128-bit nonce (step 2). This value is used to mitigate replay attacks and shall be freshly generated with each instantiation of the pairing protocol.

Following this the responding device then computes a commitment to the two public keys that have been exchanged and its own nonce value (step 3c). This commitment is computed as a one-way function of these values and is transmitted to the initiating device (step 4). The commitment prevents an attacker from changing these values at a later time.

The initiating and responding devices then exchange their respective nonce values (steps 5 and 6) and the initiating device confirms the commitment (step 6a). A failure at this point indicates the presence of an attacker or other transmission error and the initiating device shall abort the process by sending a Pairing Failed message with the reason code "Confirm Value Failed". The protocol may be repeated with or without the generation of new public-private key pairs, but new nonces shall be generated if the protocol is repeated.

When Just Works is used, the commitment checks (steps 7a and 7b) are not performed and the user is not shown the 6-digit values.

When Numeric Comparison is used, assuming that the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices (steps 7a, 7b, and 8). The user is expected to check that these 6-digit values match and to confirm if there is a match. If there is no match, then at least one device shall abort the process by sending a Pairing Failed message with the reason code "Numeric Comparison Failed" and, as before, new nonces shall be generated if the protocol is to be repeated.

An active MITM must inject its own key material into this process to have any effect other than denial-of-service. A simple MITM attack will result in the two 6-digit display values being different with probability 0.999999. A more sophisticated attack may attempt to engineer the display values to match, but this is thwarted by the commitment sequence. If the attacker first exchanges nonces with the responding device, it must commit to the nonce that it will use with the initiating device before it sees the nonce from the initiating device. If the attacker first exchanges nonces with the initiating device, it must send a nonce to the responding device before seeing the nonce from the responding device. In each case, the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices. It therefore cannot choose its own nonces in such a way as to cause the display values to match.

#### 2.3.5.6.3 Authentication stage 1 – Passkey Entry

The Passkey Entry protocol is used when SMP IO capability exchange sequence indicates that Passkey Entry shall be used.

The sequence diagram for Authentication stage 1 for Passkey Entry from the cryptographic point of view is shown in Figure 2.4.

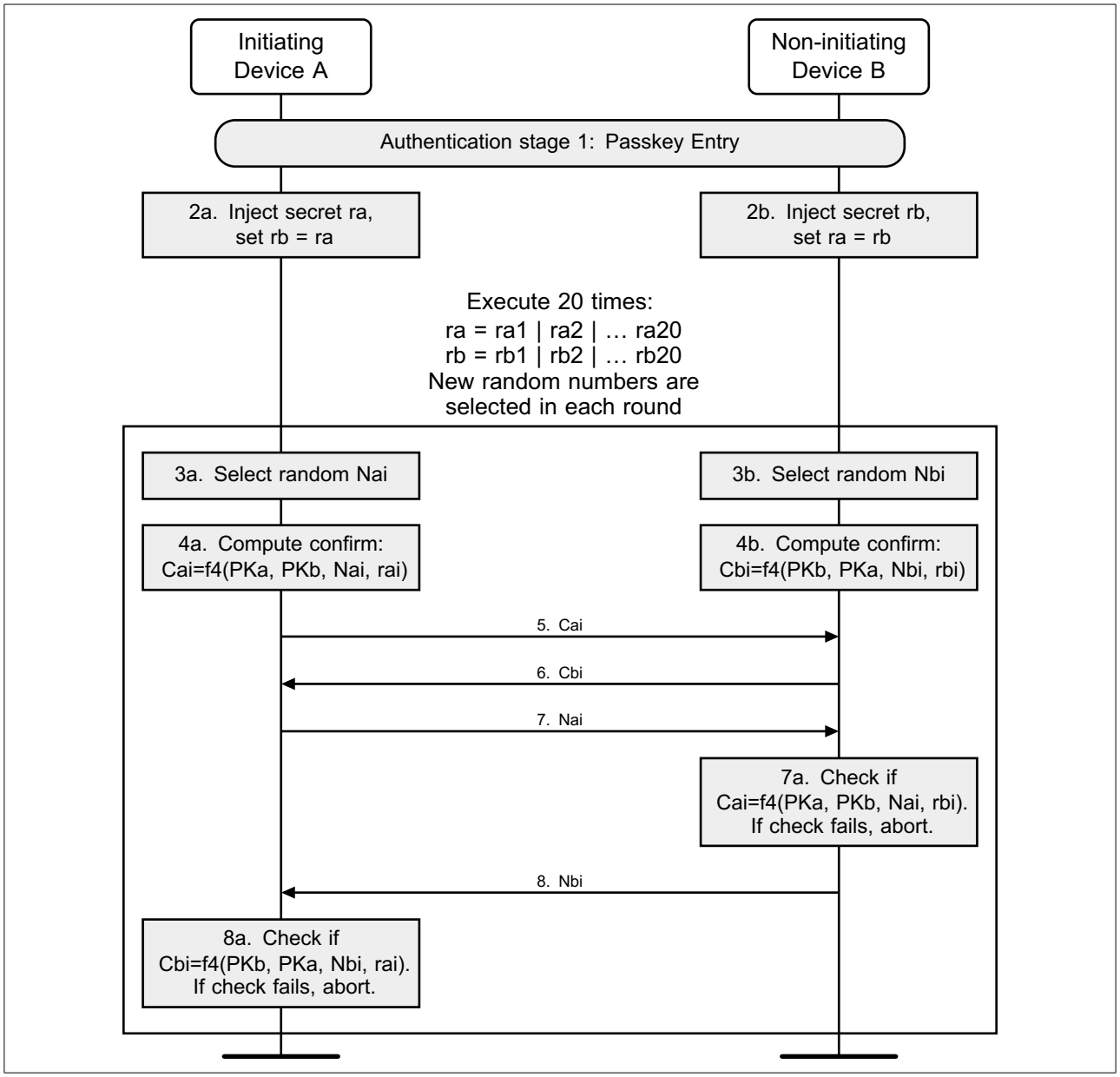



*Figure 2.4: Authentication stage 1: Passkey Entry, LE Secure Connections*

The user inputs an identical Passkey into both devices. Alternately, the Passkey may be generated and displayed on one device, and the user then inputs it into the other (step 2). This short shared key will be the basis of the mutual authentication of the devices. A new Passkey shall be generated randomly for each pairing procedure.

Steps 3 to 8 are repeated 20 times since a 6-digit Passkey is 20 bits (999999=0xF423F). If the device allows a shorter passkey to be entered, it shall be prefixed with zeros (e.g. “1234” is equivalent to “001234”).

In Steps 3 to 8, each side commits to each bit of the Passkey, using a long nonce (128 bits), and sending the hash of the nonce, the bit of the Passkey, and both public keys to the other party. The parties then take turns revealing their commitments until the entire Passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the Passkey effectively reveals that bit of the Passkey in the process, but the other party then has to reveal the corresponding commitment to show the same bit value for that bit of the Passkey, or else the first party shall send a Pairing Failed message with the reason code "Confirm Value Failed", after which no more bits of the Passkey are revealed.

This "gradual disclosure" prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a MITM attack. A MITM attacker with only partial knowledge of the Passkey will only receive one incorrectly-guessed bit of the Passkey before the protocol fails. Hence, a MITM attacker who engages first one side, then the other will only gain an advantage of at most two bits over a simple brute-force guesser, making the probability of success 0.000004 instead of 0.000001.

The long nonce is included in the commitment hash to make it difficult to brute force even after the protocol has failed. The public Diffie-Hellman values are included to tie the Passkey protocol to the original ECDH key exchange, to prevent a MITM from substituting the attacker's public key on both sides of the ECDH exchange in standard MITM fashion.

At the end of this stage, Na is set to Na20 and Nb is set to Nb20 for use in Authentication stage 2.

#### 2.3.5.6.4 Authentication stage 1 – Out of Band

The Out-of-Band protocol is used when authentication information has been received by at least one of the devices and indicated in the OOB data flag parameter included in the SMP Pairing Request and SMP Pairing Response PDU. The mode in which the discovery of the peer device is first done in-band and then followed by the transmission of authentication parameters through OOB interface is not supported. The sequence diagram for Authentication stage 1 for Out of Band from the cryptographic point of view is shown in Figure 2.5.

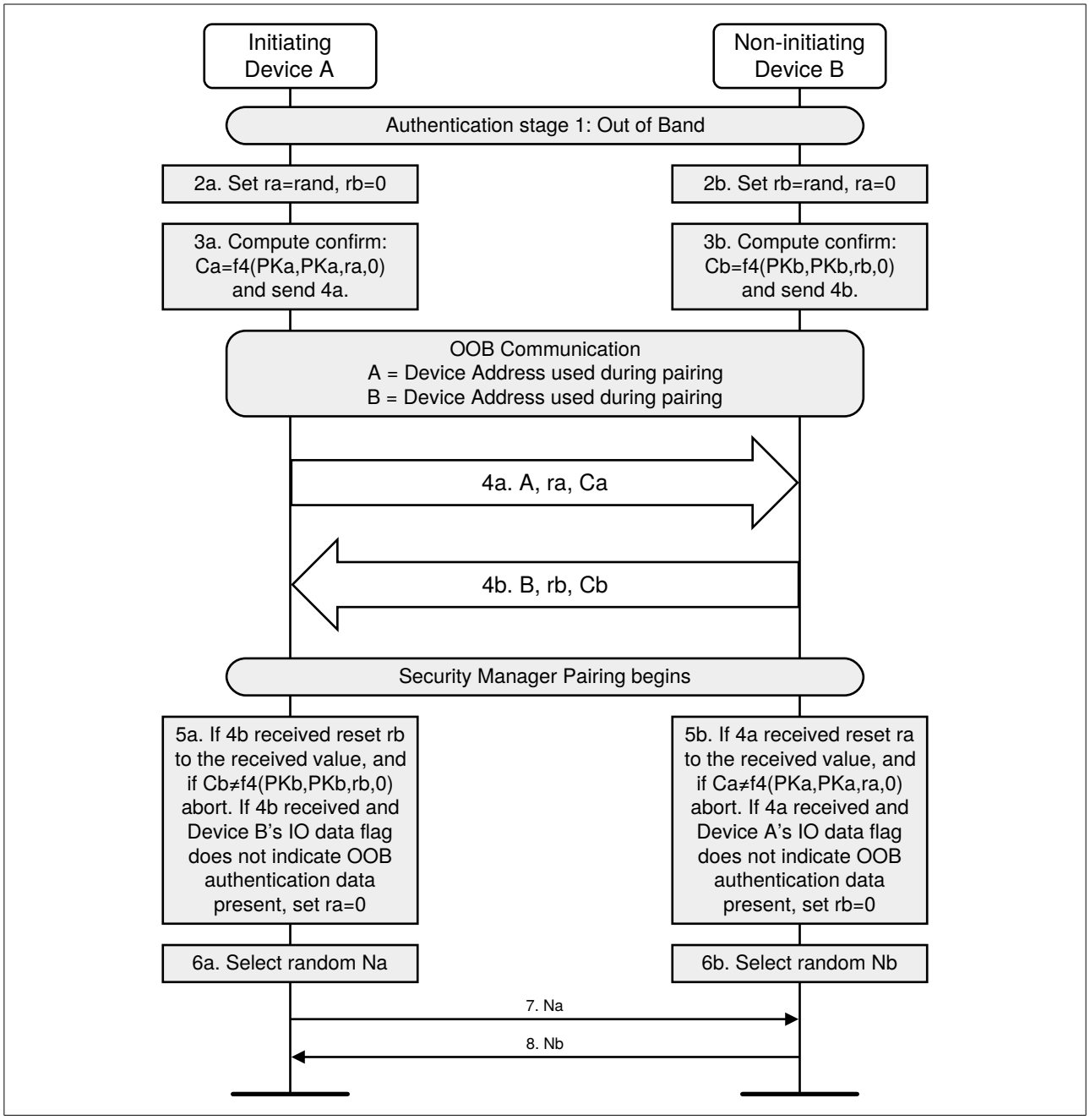



*Figure 2.5: Authentication stage 1: Out of Band, LE Secure Connections*

**Principle of operation.** If both devices can transmit and/or receive data over an out-of-band channel, then mutual authentication will be based on the commitments of the public keys (Ca and Cb) exchanged OOB in Authentication stage 1.

If OOB communication is possible only in one direction, then authentication of the device sending the OOB communication will be based on the commitment of its public key (Ca or Cb) while authentication of the device receiving the OOB communication will be based on that device knowing a random number r sent via OOB. In this case,

mutual authentication can only be obtained if r is secret: r can be created afresh every time, or access to the device sending r must be restricted. If r is not secret, then mutual authentication cannot be achieved; only the device receiving the OOB information in step 4a or 4b can authenticate the peer. If r is not sent by a device, it is assumed to be 0 by the device receiving the OOB information in step 4a or 4b.

**Roles of A and B.** The OOB Authentication stage 1 protocol is symmetric with respect to the roles of A and B. It does not require that device A always will initiate pairing and it automatically resolves asymmetry in the OOB communication.

**Order of steps.** The public key exchange must happen before the verification step 5. In the diagram the in-band public key exchange between the devices (step 1) is done before the OOB communication (step 4). But when the pairing is initiated by an OOB interface, public key exchange will happen after the OOB communication (step 1 will be between steps 4 and 5).

Values of ra and rb: Since the direction of the peer's OOB interface cannot be verified before the OOB communication takes place, a device shall always generate and if possible transmit through its OOB interface a random number r to the peer. Each device applies the following rules locally to set the values of its own r and the value of the peer's r:

1. Initially, r of the device is set to a random number and r of the peer is set to 0 (step 2).
2. If a device has received OOB, it sets the peer's r value to what was sent by the peer (Step 5).
3. If the remote device's OOB data flag sent in the SMP Pairing Request or SMP Pairing Response is set to “OOB Authentication data not present”, it sets its own r value to 0 (Step 5)

#### 2.3.5.6.5 Authentication stage 2 and long term key calculation

The second stage of authentication then confirms that both devices have successfully completed the exchange. This stage is identical in all three protocols and is shown in Figure 2.6.

Each device computes the MacKey and the LTK using the previously exchanged values and the newly derived shared key (step 9). Each device then computes a new key confirmation value that includes the previously exchanged values and the newly derived MacKey (step 10a and 10b). The initiating device then transmits its key confirmation value, which is checked by the responding device (step 11). If this check fails, then it indicates that the initiating device has not confirmed the pairing, and the responding device shall abort pairing by sending a Pairing Failed message with the reason code "DHKey Check Failed". The responding device then transmits its key confirmation

value, which is checked by the initiating device (step 12). A failure indicates that the responding device has not confirmed the pairing and the initiating device shall abort pairing by sending a Pairing Failed message with the reason code "DHKey Check Failed".

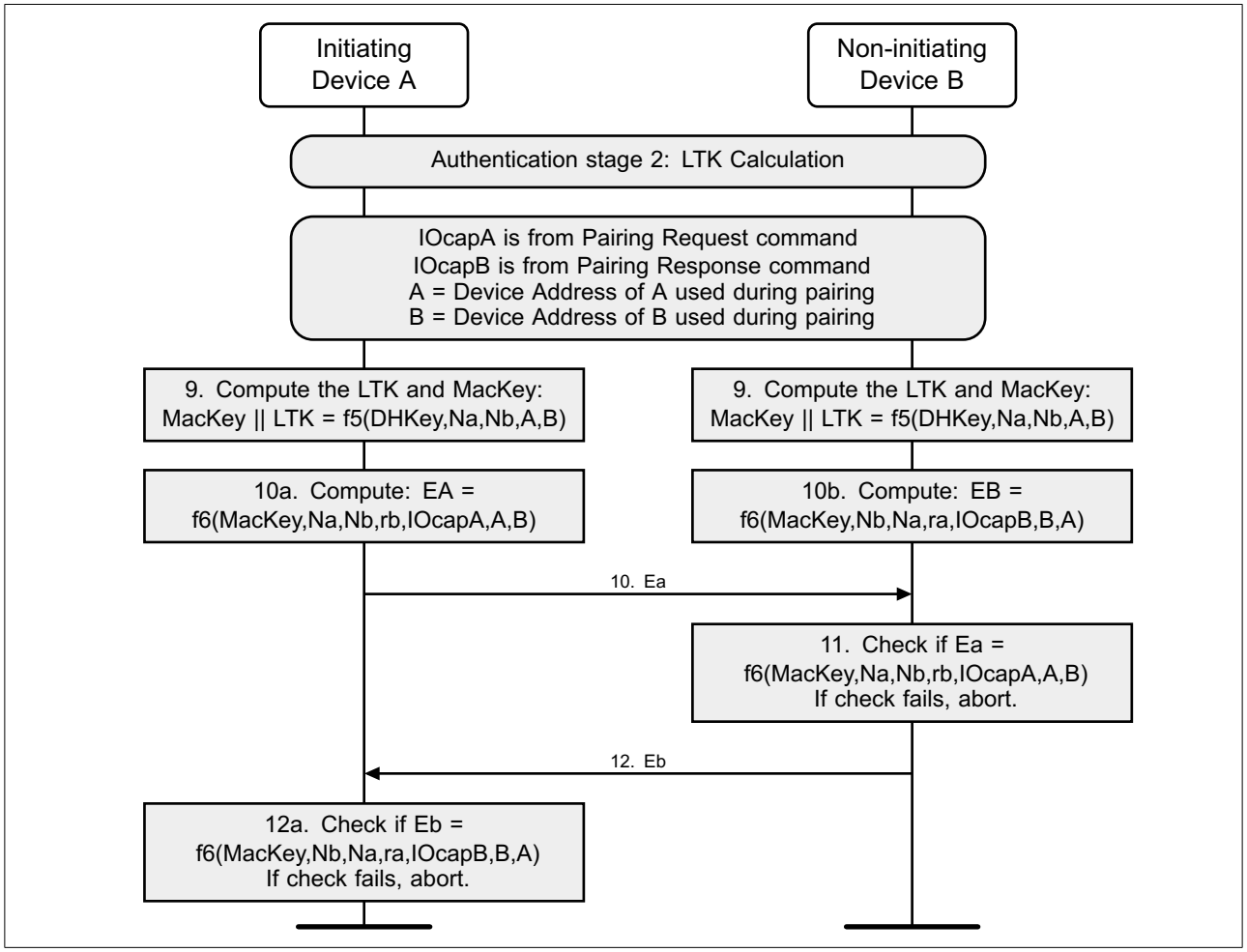



*Figure 2.6: Authentication stage 2 and long term key calculation*

### 2.3.5.7 Cross-transport key derivation

When a pair of BR/EDR/LE devices support Secure Connections on a transport, the devices may generate a key of identical strength for the other transport. There are two sequences:

- If Secure Connections pairing occurs on the LE transport, the procedures in Section 2.4.2.4 may be used. Additional keys for BR/EDR may be distributed over the LE transport.
- If Secure Connections pairing occurs on the BR/EDR transport, the procedures in Section 2.4.2.5 may be used. Additional keys for LE may be distributed over the BR/EDR transport.

### 2.3.6 Repeated attempts

When a pairing procedure fails a waiting interval shall pass before the verifier will initiate a new Pairing Request command or Security Request command to the same claimant, or before it will respond to a Pairing Request command or Security Request command initiated by a device claiming the same identity as the failed device. For each subsequent failure, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval should be limited to a maximum.

The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure restricts the rate at which an intruder can repeat the pairing procedure with different keys.

A device shall generate a new public/private key-pair after every pairing (successful or failed).

## 2.4 Security in Bluetooth Low Energy

Security shall be initiated by the Security Manager in the device in the Central role. The device in the Peripheral role shall be the responding device. The Peripheral may request the Central to initiate pairing or other security procedures, see Section 2.4.6.

The Peripheral in the key distribution phase gives keys to the Central so a reconnection can be encrypted or its random addresses can be resolved.

The Central may also provide keys to the Peripheral so a reconnection can be encrypted if the roles are reversed or the Central's random addresses can be resolved.

### 2.4.1 Definition of keys and values

LE security uses the following keys and values for encryption and random addressing:

1. Identity Resolving Key (IRK) is a 128-bit key used to generate and resolve random addresses.
2. Long Term Key (LTK) is a 128-bit key used to generate the contributory session key for an encrypted connection. Link Layer encryption is described in [Vol 6] Part B, Section 5.1.3.
3. Encrypted Diversifier (EDIV) is a 16-bit stored value used to identify the LTK distributed during LE legacy pairing. A new EDIV is generated each time a unique LTK is distributed.

[1] Another appropriate integer value larger than 1 may be used.

4. Random Number (Rand) is a 64-bit stored valued used to identify the LTK distributed during LE legacy pairing. A new Rand is generated each time a unique LTK is distributed.

### 2.4.2 Generation of distributed keys

Any method of generation of keys that are being distributed that results in the keys having 128 bits of entropy can be used, as the generation method is not visible outside the Peripheral (see Appendix B). The keys shall not be generated only from information that is distributed to the Central or only from information that is visible outside of the Peripheral.

#### 2.4.2.1 Generation of IRK

The Identity Resolving Key (IRK) is used for resolvable private address construction (see [Vol 3] Part C, Section 10.8.2). A Central that has received IRK from a Peripheral can resolve that Peripheral's random device addresses. A Peripheral that has received IRK from a Central can resolve that Central's random device addresses. The privacy concept only protects against devices that are not part of the set to which the IRK has been given.

IRK can be assigned, or randomly generated by the device during manufacturing, or some other method could be used.

The encryption key size does not apply to IRK; therefore, its size does not need to be shortened before distribution.

#### 2.4.2.2 [This section is no longer used]

#### 2.4.2.3 LE legacy pairing - generation of LTK, EDIV and Rand

Devices which support encryption in the Link Layer Connection state in the Peripheral Role shall be capable of generating LTK, EDIV, and Rand.

The EDIV and Rand are used by the Peripheral to establish a previously shared LTK in order to start an encrypted connection with a previously paired Central.

The generated LTK size shall not be longer than the negotiated encryption key size, so its size may need to be shortened (see Section 2.3.4).

New values of LTK, EDIV, and Rand shall be generated each time they are distributed.

The Peripheral may store the mapping between EDIV, Rand and LTK in a security database so the correct LTK value is used when the Central requests encryption. Depending upon the LTK generation method additional information may be stored, for example the size of the distributed LTK.

The Central may also distribute EDIV, Rand, and LTK to the Peripheral which can be used to encrypt a reconnection if the device roles are reversed in a future connection.

#### 2.4.2.4 Derivation of BR/EDR link key from LE LTK

The LTK from the LE physical transport can be converted to the BR/EDR link key for the BR/EDR transport as follows, using intermediate link key (ILK) as an intermediate value:

If at least one device sets CT2 = 0 then

1. ILK = *h6*(LTK, “tmp1”)
2. BR/EDR link key = *h6*(ILK, “lebr”)

If both devices set CT2 = 1 then

1. ILK = *h7*(SALT, LTK)
2. BR/EDR link key = *h6*(ILK, “lebr”)

The string “lebr” is mapped into a keyID using ASCII as 0x6C656272.

The string “tmp1” is mapped into a keyID using ASCII as 0x746D7031 and into a SALT as 0x00000000_00000000_00000000_746D7031.

Note: If the LTK has an encryption key size that is shorter than 16 octets (128 bits), then the BR/EDR link key is derived before the LTK gets masked.

#### 2.4.2.5 Derivation of LE LTK from BR/EDR link key

The BR/EDR Link Key from the BR/EDR physical transport can be converted to the LTK for the LE transport as follows, using intermediate long term key (ILTK) as an intermediate value:

If at least one device sets CT2 = 0 then

1. ILTK = *h6*(Link Key, “tmp2”)
2. LTK = *h6*(ILTK, “brle”)

If both devices set CT2 = 1 then

1. ILTK = *h7*(SALT, Link Key)
2. LTK = *h6*(ILTK, “brle”)

The string “brle” is mapped into a keyID using ASCII as 0x62726C65.

The string “tmp2” is mapped into a keyID using ASCII as 0x746D7032 and into a SALT as 0x00000000_00000000_00000000_746D7032.

### 2.4.3 Distribution of keys

Key distribution for LE Legacy Pairing and LE Secure Connections is described in the following sections.

#### 2.4.3.1 LE legacy pairing key distribution

The Peripheral may distribute to the Central the following keys:

- LTK, EDIV, and Rand
- IRK

The Central may distribute to the Peripheral the following keys:

- LTK, EDIV, and Rand
- IRK

The security properties of the distributed keys shall be set to the security properties of the STK that was used to distribute them. For example if STK has Unauthenticated no MITM Protection security properties then the distributed keys shall have Unauthenticated no MITM Protection security properties.

The link shall be encrypted or re-encrypted using STK generated in Phase 2 (see Section 2.4.4.1) before any keys are distributed.

Note: The distributed EDIV and Rand values are transmitted in clear text by the Central to the Peripheral during encrypted session setup.

The BD_ADDR that is received in the Identity Address Information command shall only be considered valid once a reconnection has occurred using the BD_ADDR and LTK distributed during that pairing. Once this is successful the BD_ADDR and the distributed keys shall be associated with that device in the security database.

A device may request encrypted session setup to use the LTK, EDIV, and Rand values distributed by the Peripheral when the key distribution phase has completed; however, this does not provide any additional security benefit. If an attacker has established the distributed LTK value then performing encrypted session setup to use the distributed values does not provide any protection against that attacker.

#### 2.4.3.2 LE Secure Connections key distribution

The Central and Peripheral may distribute the following key:

- IRK

The security properties of the distributed keys shall be set to the security properties of the LTK that was used to distribute them. For example if LTK has Unauthenticated

no MITM Protection security properties then the distributed keys shall have Unauthenticated no MITM Protection security properties.

The link shall be encrypted or re-encrypted using LTK generated in Phase 2 (see Section 2.4.4.1) before any keys are distributed.

The BD_ADDR that is received in the Identity Address Information command shall only be considered valid once a reconnection has occurred using the BD_ADDR and LTK generated during that pairing. Once this is successful the BD_ADDR and the distributed keys shall be associated with that device in the security database.

### 2.4.4 Encrypted session setup

During the encrypted session setup the Central sends a 16-bit Encrypted Diversifier value, *EDIV,* and a 64-bit Random Number, *Rand*, distributed by the Peripheral during pairing, to the Peripheral. The Central's Host provides the Link Layer with the Long Term Key to use when setting up the encrypted session. The Peripheral's Host receives the *EDIV* and *Rand* values and provides a Long Term Key to the Peripheral's Link Layer to use when setting up the encrypted link.

When both devices support LE Secure Connections, the *EDIV* and *Rand* are set to zero.

#### 2.4.4.1 Encryption setup using STK

To distribute LTK and other keys in pairing Phase 3 an encrypted session needs to be established (see Section 2.3.5.5).

The encrypted session is setup using the STK generated in Phase 2 (see Section 2.3.5.5) as the Long Term Key provided to the Link Layer (see [Vol 6] Part B, Section 5.1.3.1) *EDIV*, and *Rand* values shall be set to zero.

If the link is already encrypted then the encryption pause procedure is performed using the STK generated in Phase 2 as the Long Term Key provided to the Link Layer (see [Vol 6] Part B, Section 5.1.3.2). *EDIV* and *Rand* values shall be set to zero.

#### 2.4.4.2 Encryption setup using LTK

The Central must have the security information (*LTK*, *EDIV*, and *Rand*) distributed by the Peripheral in LE legacy pairing or the LTK generated in LE Secure Connections to setup an encrypted session.

The Central initiates the encrypted session using the security information; see [Vol 6] Part B, Section 5.1.3.1. If the link is already encrypted the encryption pause procedure is performed using the security information; see [Vol 6] Part B, Section 5.1.3.2.

In LE legacy pairing, the *EDIV* and *Rand* values are used to establish *LTK* which is used as the Long Term Key on the Peripheral. If LTK cannot be established from EDIV and Rand values then the Peripheral shall reject the request to encrypt the link and may disconnect the link.

The LTK size shall not be longer than the negotiated encryption key size, so its size may need to be shortened (see Section 2.3.4).

When the security information is stored, subsequent encryption setups may fail if the remote device has deleted the security information. Table 2.9 defines what shall be done depending on the type of the security properties and whether or not bonding was performed when subsequent encryption setup fails.

| Security Properties | Devices Bonded | Action to take when enabling encryption fails |
|---|---|---|
| Unauthenticated, no MITM protection | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate pairing.<br>• Option 2: Notify user and ask if pairing is ok.<br>Option 1 is recommended. |
| Unauthenticated, no MITM protection | Yes | Notify user of security failure. |
| Authenticated MITM protection | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate pairing.<br>• Option 2: Notify user and ask if pairing is ok.<br>Option 2 is recommended. |
| Authenticated MITM protection | Yes | Notify user of security failure. |

*Table 2.9: Action after encryption setup failure*

### 2.4.5 [This section is no longer used]

### 2.4.6 Peripheral Security Request

The Peripheral may request security by transmitting a Security Request command to the Central. When a Central receives a Security Request command it may encrypt the link, initiate the pairing procedure, or reject the request.

The Peripheral shall not send the Security Request command if the pairing procedure is in progress, or if the encryption procedure is in progress.

The Security Request command includes the required security properties. A security property of MITM protection required shall only be set if the Peripheral's IO capabilities

would allow the Passkey Entry association model to be used or out of band authentication data is available.

The Central shall ignore the Peripheral's Security Request if the Central has sent a Pairing Request without receiving a Pairing Response from the Peripheral or if the Central has initiated encryption mode setup.

If pairing or encryption mode is not supported or cannot be initiated at the time when the Peripheral's Security Request command is received, then the Central shall respond with a Pairing Failed command with the reason set to "Pairing Not Supported."

After receiving a Security Request, the Central shall first check whether it has the required security information to enable encryption; see Section 2.4.4.2. If this information is missing or does not meet the security properties requested by the Peripheral, then the Central shall initiate the pairing procedure. If the pairing procedure is successful, the Central's security database is updated with the keys and security properties are distributed during the pairing procedure.

If the Central has the required security information to enable encryption and it meets the security properties request by the Peripheral, it shall perform encryption setup using LTK, see Section 2.4.4.2.

Figure 2.7 shows a summary of the actions and decisions that a Central shall take when receiving a Security Request.

The Peripheral shall check that any Pairing Request command received from the Central after sending a Security Request command contains Authentication Requirements that meet the requested security properties.

If the Peripheral requests a security property that is not Just Works and receives an encryption procedure request after sending a Security Request command then it shall check that any existing Security Information is of sufficient security properties.

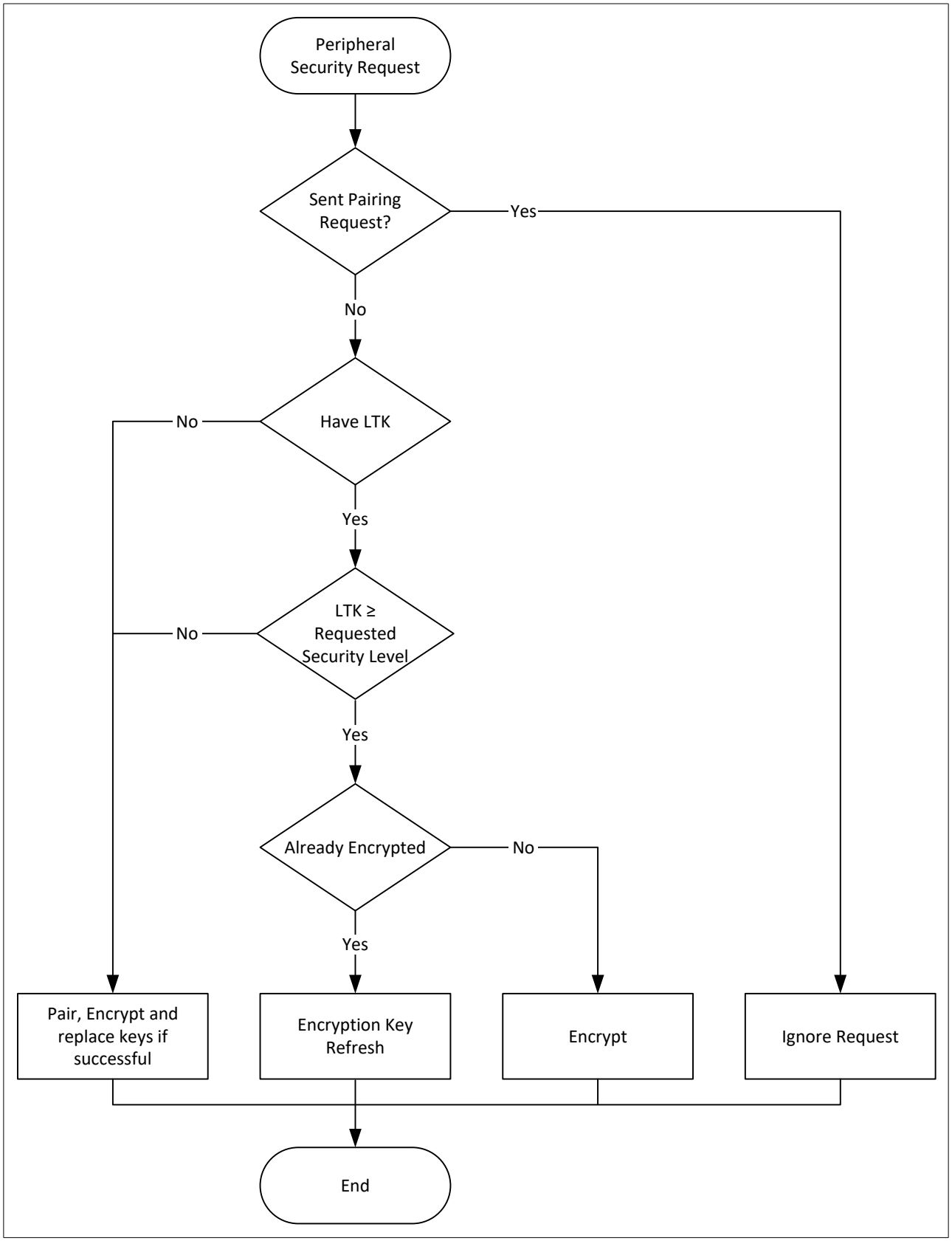



*Figure 2.7: Central actions after receiving Security Request command*

# 3 SECURITY MANAGER PROTOCOL

## 3.1 Introduction

The Security Manager Protocol (SMP) is used for pairing and transport specific key distribution.

## 3.2 Security Manager Channel over L2CAP

All SMP commands are sent over the Security Manager Channel which is an L2CAP fixed channel (see [Vol 3] Part A, Section 2.1). The configuration parameters for the Security Manager Channel when LE Secure Connections is not supported shall be as shown below in Table 3.1.

| Parameter | Value |
|---|---|
| MTU | 23 |
| Flush Timeout | 0xFFFF (Infinite) |
| QoS | Best Effort |
| Mode | Basic Mode |

*Table 3.1: Security Manager Channel configuration parameters without LE Secure Connections*

The configuration parameters for the Security Manager Channel when LE Secure Connections is supported shall be as shown below in Table 3.2.

| Parameter | Value |
|---|---|
| MTU | 65 |
| Flush Timeout | 0xFFFF (Infinite) |
| QoS | Best Effort |
| Mode | Basic Mode |

*Table 3.2: Security Manager Channel configuration parameters with LE Secure Connections*

## 3.3 Command format

The general format for all SMP commands is shown in Figure 3.1.

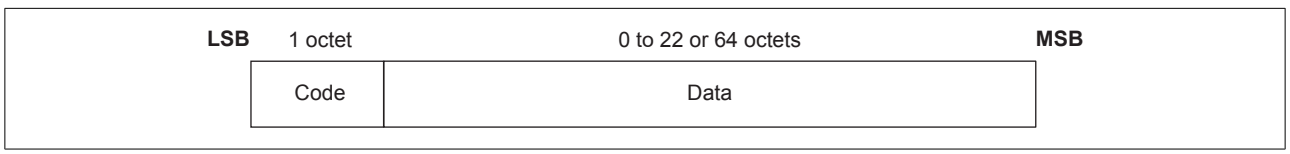



*Figure 3.1: SMP command format*

The following are the fields shown:

- *Code (1 octet)*

  The Code field is one octet long and identifies the type of command. Table 3.3 lists the codes defined by this document. If a packet is received with a Code that is reserved for future use it shall be ignored.

| Code | Description | Logical Link Supported |
|---|---|---|
| 0x01 | Pairing Request | LE-U, ACL-U |
| 0x02 | Pairing Response | LE-U, ACL-U |
| 0x03 | Pairing Confirm | LE-U |
| 0x04 | Pairing Random | LE-U |
| 0x05 | Pairing Failed | LE-U, ACL-U |
| 0x06 | Encryption Information | LE-U |
| 0x07 | Central Identification | LE-U |
| 0x08 | Identity Information | LE-U, ACL-U |
| 0x09 | Identity Address Information | LE-U, ACL-U |
| 0x0A | *Previously used* | |
| 0x0B | Security Request | LE-U |
| 0x0C | Pairing Public Key | LE-U |
| 0x0D | Pairing DHKey Check | LE-U |
| 0x0E | Pairing Keypress Notification | LE-U |
| All other values | *Reserved for future use* | |

*Table 3.3: SMP command codes*

- *Data (0 or more octets)*

  The Data field is variable in length. The Code field determines the format of the Data field.

If a device does not support pairing then it shall respond with a Pairing Failed command with the reason set to “Pairing Not Supported” (see Section 3.5.5) when any command is received. If pairing is supported then all commands shall be supported.

## 3.4 SMP timeout

To protect the Security Manager protocol from stalling, a Security Manager Timer is used. Upon transmission of the Security Request command or reception of the Security Request command, the Security Manager Timer shall be reset and restarted. Upon transmission of the Pairing Request command or reception of the Pairing Request command, the Security Manager Timer shall be reset and started.

The Security Manager Timer shall be reset when an L2CAP SMP command is queued for transmission. The Security Manager Timer should be reset upon reception of a Keypress Notification (if the timer is not reset on receipt of a Keypress Notification, the Security Manager can time out before the peer's Security Manager because there is no response to a Keypress Notification).

When a Pairing process completes (whether successfully or not), the Security Manager Timer shall be stopped.

If the Security Manager Timer reaches 30 seconds, the procedure shall be considered to have failed, and the local higher layer shall be notified. No further SMP commands shall be sent over the L2CAP Security Manager Channel. A new Pairing process shall only be performed when a new physical link has been established.

## 3.5 Pairing methods

The SMP commands defined in this section are used to perform Pairing Feature Exchange and key generation (see Section 2.1).

### 3.5.1 Pairing Request

The initiator starts the Pairing Feature Exchange by sending a Pairing Request command to the responding device. The Pairing Request command is defined in Figure 3.2.

The rules for handing a collision between a pairing procedure on the LE transport and a pairing procedure on the BR/EDR transport are defined in [Vol 3] Part C, Section 14.2.

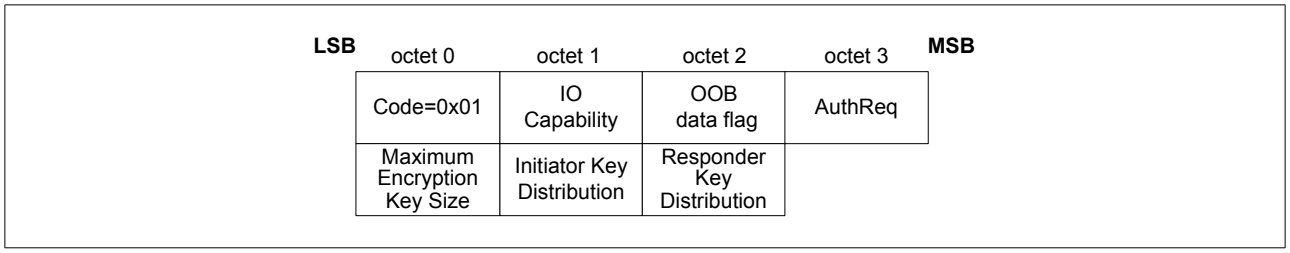



*Figure 3.2: Pairing Request packet*

The following data fields are used:

- *IO Capability (1 octet)*

  Table 3.4 defines the values which are used when exchanging IO capabilities (see Section 2.3.2).

| Value | Description |
|---|---|
| 0x00 | DisplayOnly |
| 0x01 | DisplayYesNo |
| 0x02 | KeyboardOnly |
| 0x03 | NoInputNoOutput |
| 0x04 | KeyboardDisplay |
| 0x05 to 0xFF | Reserved for future use |

*Table 3.4: IO capability values*

- *OOB data flag (1 octet)*

  Table 3.5 defines the values which are used when indicating whether OOB authentication data is available (see Section 2.3.3).

| Value | Description |
|---|---|
| 0x00 | OOB Authentication data not present |
| 0x01 | OOB Authentication data from remote device present |
| 0x02 to 0xFF | Reserved for future use |

*Table 3.5: OOB data present values*

- *AuthReq (1 octet)*

  The AuthReq field is a bit field that indicates the requested security properties (see Section 2.3.1) for the STK and LTK and GAP bonding information (see [Vol 3] Part C, Section 9.4).

  Figure 3.3 defines the authentication requirements bit field.

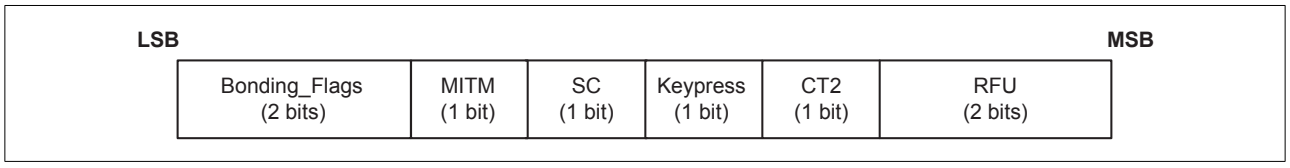



*Figure 3.3: Authentication requirements flags*

The Bonding_Flags field is a 2-bit field that indicates the type of bonding being requested by the initiating device as defined in Table 3.6.

| Bonding_Flags $b_1b_0$ | Bonding Type |
|---|---|
| 00 | No Bonding |
| 01 | Bonding |

| Bonding_Flags $b_1b_0$ | Bonding Type |
|---|---|
| 10 | Reserved for future use |
| 11 | Reserved for future use |

*Table 3.6: Bonding flags*

The MITM field is a 1-bit flag that is set to one if the device is requesting MITM protection, otherwise it shall be set to 0. A device sets the MITM flag to one to request an Authenticated security property for the STK when using LE legacy pairing and the LTK when using LE Secure Connections.

The SC field is a 1 bit flag. If LE Secure Connections pairing is supported by the device, then the SC field shall be set to 1, otherwise it shall be set to 0. If both devices support LE Secure Connections pairing, then LE Secure Connections pairing shall be used, otherwise LE Legacy pairing shall be used.

The keypress field is a 1-bit flag that is used only in the Passkey Entry protocol and shall be ignored in other protocols. When both sides set that field to one, Keypress notifications shall be generated and sent using SMP Pairing Keypress Notification PDUs.

The CT2 field is a 1-bit flag that shall be set to 1 upon transmission to indicate support for the *h7* function. See Section 2.4.2.4 and Section 2.4.2.5.

- *Maximum Encryption Key Size (1 octet)*

  This value defines the maximum encryption key size in octets that the device can support. The maximum key size shall be in the range 7 to 16 octets.

- *Initiator Key Distribution / Generation (1 octet)*

  The Initiator Key Distribution / Generation field indicates which keys the initiator is requesting to distribute / generate or use during the Transport Specific Key Distribution phase (see Section 2.4.3). The Initiator Key Distribution / Generation field format and usage is defined in Section 3.6.1.

- *Responder Key Distribution / Generation (1 octet)*

  The Responder Key Distribution / Generation field indicates which keys the initiator is requesting the responder to distribute / generate or use during the Transport Specific Key Distribution phase (see Section 2.4.3). The Responder Key Distribution / Generation field format and usage is defined in Section 3.6.1.

If Secure Connections pairing has been initiated over BR/EDR, the following fields of the SM Pairing Request PDU are reserved for future use:

- the IO Capability field,
- the OOB data flag field, and
- all bits in the Auth Req field except the CT2 bit.

### 3.5.2 Pairing Response

This command is used by the responding device to complete the Pairing Feature Exchange after it has received a Pairing Request command from the initiating device, if the responding device allows pairing. The Pairing Response command is defined in Figure 3.4.

The rules for handing a collision between a pairing procedure on the LE transport and a pairing procedure on the BR/EDR transport are defined in [Vol 3] Part C, Section 14.2.

If a Pairing Request is received over the BR/EDR transport when either cross-transport key derivation/generation is not supported or the BR/EDR transport is not encrypted using a Link Key generated using P256, a Pairing Failed shall be sent with the error code *Cross-Transport Key Derivation/Generation Not Allowed* (0x0E).

If a Pairing Request is received and the device is not ready to perform the pairing procedure, a Pairing Failed may be sent with the error code *Busy* (0x10).

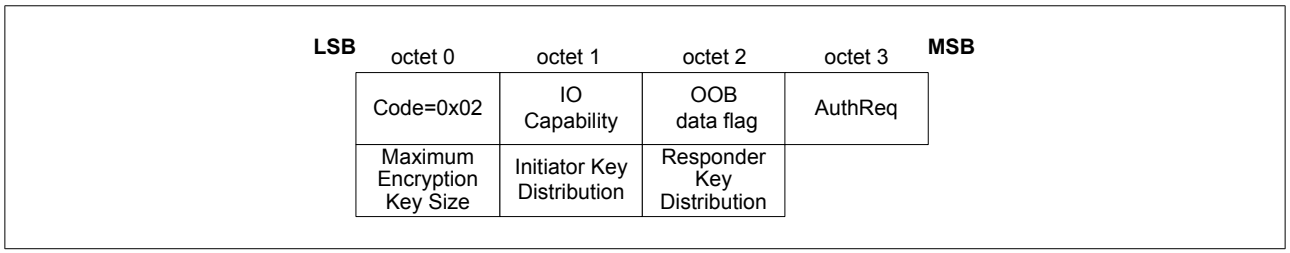



*Figure 3.4: Pairing Response packet*

The following data fields are used:

- *IO Capability (1 octet)*

  Table 3.4 defines the values which are used when exchanging IO capabilities (see Section 2.3.2).

- *OOB data flag (1 octet)*

  Table 3.5 defines the values which are used when indicating whether OOB authentication data is available (see Section 2.3.3).

- *AuthReq (1 octet)*

  The AuthReq field is a bit field that indicates the requested security properties (see Section 2.3.1) for the STK or LTK and GAP bonding information (see [Vol 3] Part C, Section 9.4).

  Figure 3.3 defines the authentication requirements bit field.

  The Bonding_Flags field is a 2-bit field that indicates the type of bonding being requested by the responding device as defined in Table 3.6.

  The MITM field is a 1-bit flag that is set to one if the device is requesting MITM protection, otherwise it shall be set to 0. A device sets the MITM flag to one to request an Authenticated security property for the STK when using LE legacy pairing and the LTK when using LE Secure Connections.

  The SC field is a 1 bit flag. If LE Secure Connections pairing is supported by the device, then the SC field shall be set to 1, otherwise it shall be set to 0. If both devices support LE Secure Connections pairing, then LE Secure Connections pairing shall be used, otherwise LE Legacy pairing shall be used.

  The keypress field is a 1-bit flag that is used only in the Passkey Entry protocol and shall be ignored in other protocols. When both sides set that field to one, Keypress notifications shall be generated and sent using SMP Pairing Keypress Notification PDUs.

  The CT2 field is a 1-bit flag that shall be set to 1 upon transmission to indicate support for the *h7* function. See Section 2.4.2.4 and Section 2.4.2.5.

- *Maximum Encryption Key Size (1 octet)*

  This value defines the maximum encryption key size in octets that the device can support. The maximum key size shall be in the range 7 to 16 octets.

- *Initiator Key Distribution (1 octet)*

  The Initiator Key Distribution field defines which keys the initiator shall distribute and use during the Transport Specific Key Distribution phase (see Section 2.4.3). The Initiator Key Distribution field format and usage are defined in Section 3.6.1.

- *Responder Key Distribution (1 octet)*

  The Responder Key Distribution field defines which keys the responder shall distribute and use during the Transport Specific Key Distribution phase (see Section 2.4.3). The Responder Key Distribution field format and usage are defined in Section 3.6.1.

If Secure Connections pairing has been initiated over BR/EDR, the following fields of the SM Pairing Response PDU are reserved for future use:

- the IO Capability field,
- the OOB data flag field, and
- all bits in the Auth Req field except the CT2 bit.

### 3.5.3 Pairing Confirm

This is used following a successful Pairing Feature Exchange to start STK Generation for LE legacy pairing and LTK Generation for LE Secure Connections pairing. The Pairing Confirm command is defined in Figure 3.5.

This command is used by both devices to send the confirm value to the peer device, see Section 2.3.5.5 for LE legacy pairing and Section 2.3.5.6 for LE Secure Connections pairing.

The initiating device starts key generation by sending the Pairing Confirm command to the responding device. The initiating device can transmit a Pairing Failed command instead to abort pairing.

The responding device sends the Pairing Confirm command after it has received a Pairing Confirm command from the initiating device.

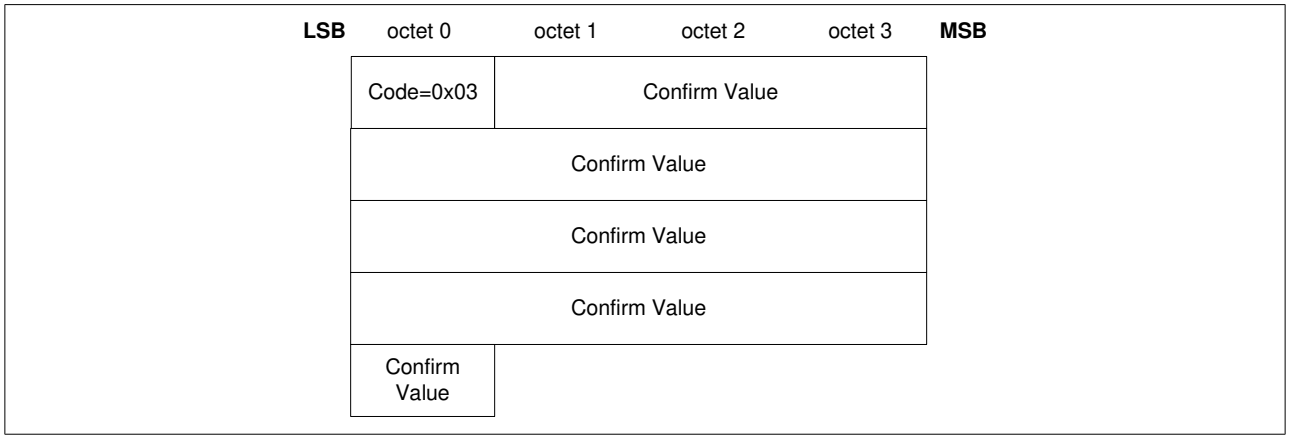



*Figure 3.5: Pairing Confirm packet*

The following data field is used:

- *Confirm value (16 octets)*

  In LE legacy pairing, the initiating device sends LP_CONFIRM_I and the responding device sends LP_CONFIRM_R as defined in Section 2.3.5.5. In LE Secure Connections, Ca and Cb are defined in Section 2.2.6.

### 3.5.4 Pairing Random

This command is used by the initiating and responding device to send the random number used to calculate the Confirm value sent in the Pairing Confirm command. The Pairing Random command is defined in Figure 3.6.

The initiating device sends a Pairing Random command after it has received a Pairing Confirm command from the responding device.

In LE legacy pairing, the responding device shall send a Pairing Random command after it has received a Pairing Random command from the initiating device if the Confirm value calculated on the responding device matches the Confirm value received from the initiating device. If the calculated Confirm value does not match then the responding device shall respond with the Pairing Failed command.

In LE Secure Connections, the responding device shall send a Pairing Random command after it has received a Pairing Random command from the initiating device. If the calculated Confirm value does not match then the responding device shall respond with the Pairing Failed command.

The initiating device shall encrypt the link using the generated key (STK in LE legacy pairing or LTK in LE Secure Connections) if the Confirm value calculated on the initiating device matches the Confirm value received from the responding device. The successful encryption or re-encryption of the link is the signal to the responding device that key generation has completed successfully. If the calculated Confirm value does not match then the initiating device shall respond with the Pairing Failed command.

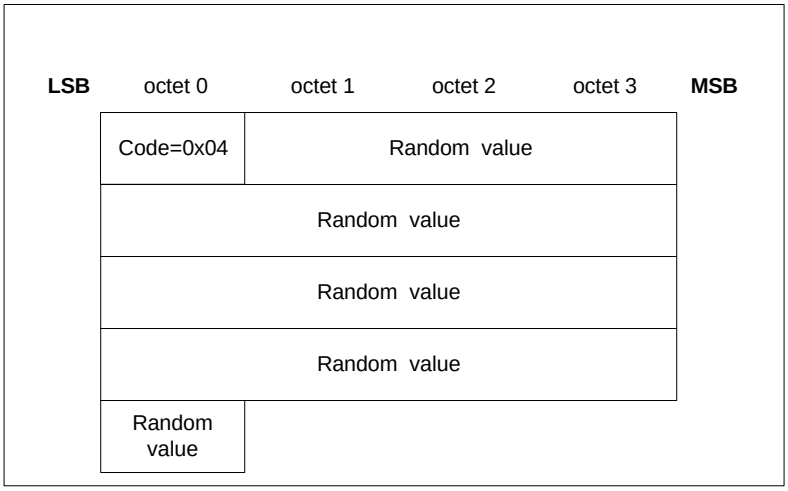



*Figure 3.6: Pairing Random packet*

The following are the data fields:

- *Random value (16 octets)*

  In LE legacy pairing, the initiating device sends LP_RAND_I and the responding device sends LP_RAND_R as defined in Section 2.3.5.5. In LE Secure Connections, the initiating device sends Na and the responding device sends Nb.

### 3.5.5 Pairing Failed

This is used when there has been a failure during pairing and reports that the pairing procedure has been stopped and no further communication for the current pairing procedure is to occur. The Pairing Failed command is defined in Figure 3.7.

Any subsequent pairing procedure shall restart from the Pairing Feature Exchange phase.

This command may be sent at any time during the pairing process by either device in response to a message from the remote device.

During LE Secure Connections pairing, this command shall be sent if the remote device's public key is invalid (see Section 2.3.5.6.1). The Reason field shall be set to "DHKey Check Failed".

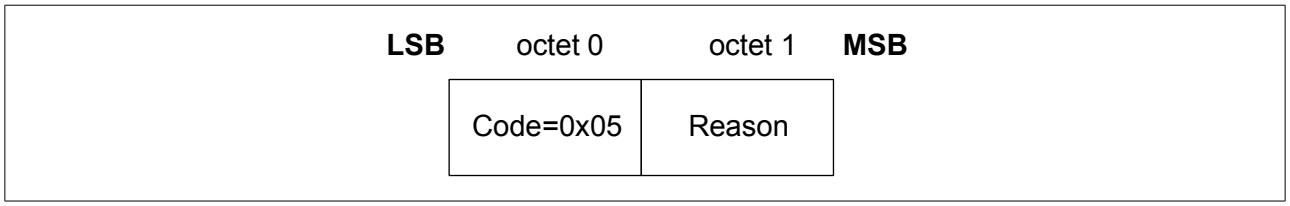



*Figure 3.7: Pairing Failed packet*

The following data field is used:

- *Reason (1 octets)*

The Reason field indicates why the pairing failed. The reason codes are defined in Table 3.7.

| Value | Name | Description |
|---|---|---|
| 0x01 | Passkey Entry Failed | The user input of passkey failed, for example, the user cancelled the operation. |
| 0x02 | OOB Not Available | The OOB data is not available. |
| 0x03 | Authentication Requirements | The pairing procedure cannot be performed as authentication requirements cannot be met due to IO capabilities of one or both devices. |

| Value | Name | Description |
|---|---|---|
| 0x04 | Confirm Value Failed | The confirm value does not match the calculated compare value. |
| 0x05 | Pairing Not Supported | Pairing is not supported by the device. |
| 0x06 | Encryption Key Size | The resultant encryption key size is not long enough for the security requirements of this device. |
| 0x07 | Command Not Supported | The SMP command received is not supported on this device. |
| 0x08 | Unspecified Reason | Pairing failed due to an unspecified reason. |
| 0x09 | Repeated Attempts | Pairing or authentication procedure is disallowed because too little time has elapsed since last pairing request or security request. |
| 0x0A | Invalid Parameters | The Invalid Parameters error code indicates that the command length is invalid or that a parameter is outside of the specified range. |
| 0x0B | DHKey Check Failed | Indicates to the remote device that the DHKey Check value received doesn't match the one calculated by the local device. |
| 0x0C | Numeric Comparison Failed | Indicates that the confirm values in the numeric comparison protocol do not match. |
| 0x0D | BR/EDR pairing in progress | Indicates that the pairing over the LE transport failed due to a Pairing Request sent over the BR/EDR transport in progress. |
| 0x0E | Cross-transport Key Derivation/Generation not allowed | Indicates that the BR/EDR Link Key generated on the BR/EDR transport cannot be used to derive and distribute keys for the LE transport or the LE LTK generated on the LE transport cannot be used to derive a key for the BR/EDR transport. |
| 0x0F | Key Rejected | Indicates that the device chose not to accept a distributed key. |
| 0x10 | Busy | Indicates that the device is not ready to perform a pairing procedure. |
| All other values | | Reserved for future use. The recipient should treat these values as 0x08 (Unspecified Reason). |

*Table 3.7: Pairing Failed reason codes*

### 3.5.6 Pairing Public Key

This message is used to transfer the device's local public key (X and Y co-ordinates) to the remote device. This message is used by both the initiator and responder. This PDU is only used for Secure Connections. Its format is specified in Figure 3.8.

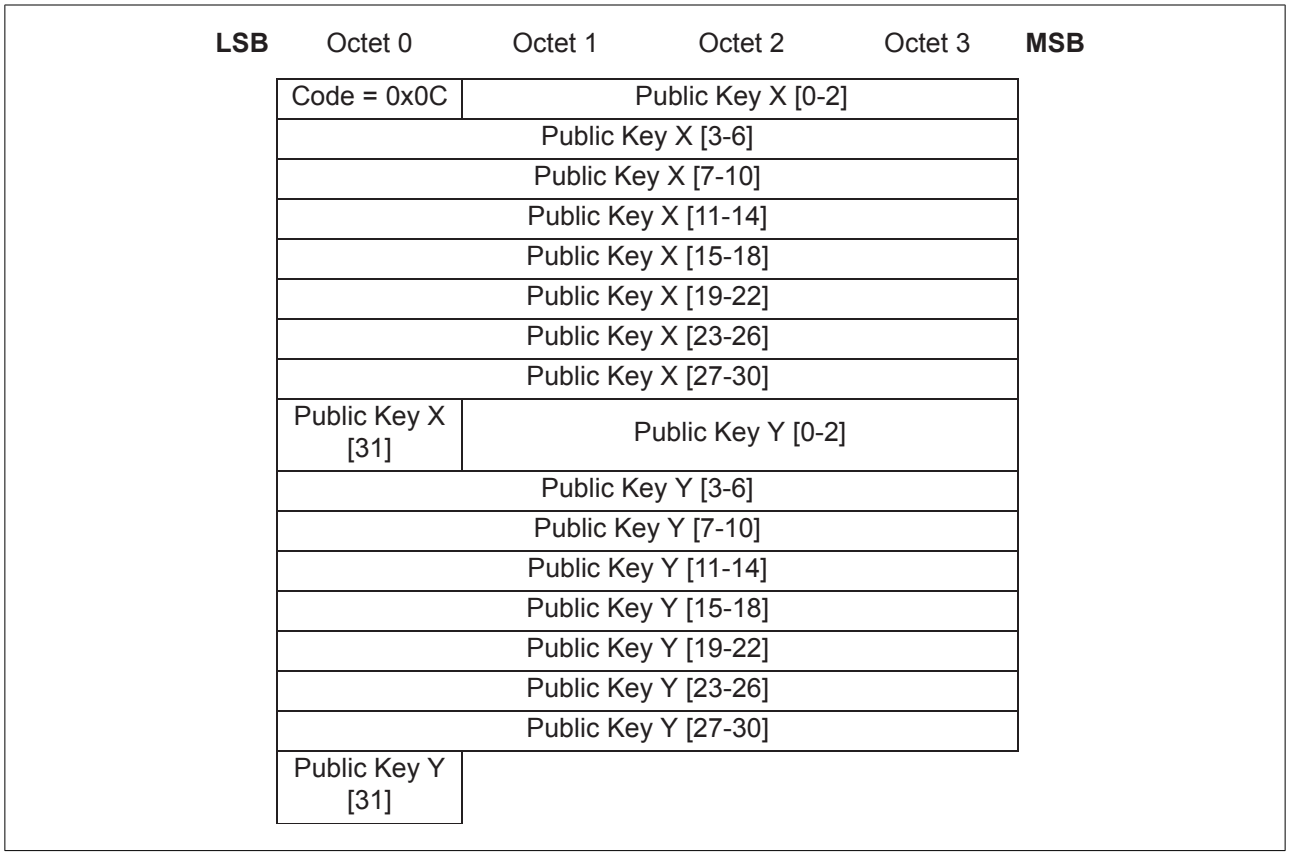



*Figure 3.8: Pairing Public Key PDU*

### 3.5.7 Pairing DHKey Check

This message is used to transmit the 128-bit DHKey Check values (Ea and Eb) generated using f6. These are confirmation values generated using the DHKey. This message is used by both initiator and responder. This PDU is only used for LE Secure Connections. Its format is specified in Figure 3.9.

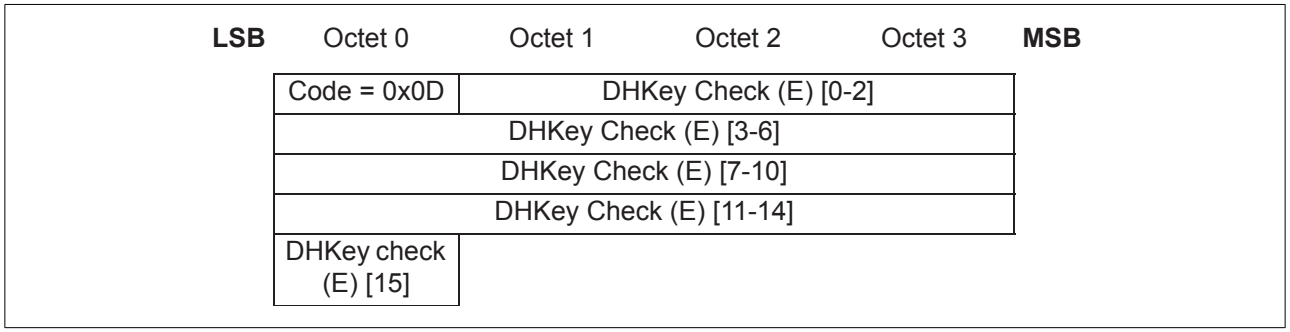



*Figure 3.9: Pairing DHKey Check PDU*

### 3.5.8 Keypress Notification

This message is used during the Passkey Entry protocol by a device with KeyboardOnly IO capabilities to inform the remote device when keys have been entered or erased. Its format is specified in Figure 3.10.

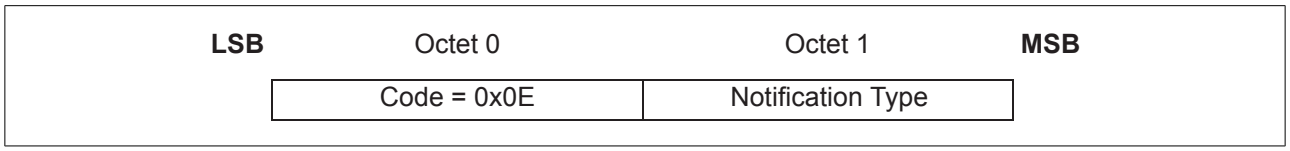



*Figure 3.10: Pairing Keypress Notification PDU*

Notification Type can take one of the following values:

| Value | Parameter Description |
|---|---|
| 0 | Passkey entry started |
| 1 | Passkey digit entered |
| 2 | Passkey digit erased |
| 3 | Passkey cleared |
| 4 | Passkey entry completed |
| 5 to 255 | Reserved for future use |

*Table 3.8: Notification Type*

## 3.6 Security in Bluetooth Low Energy

### 3.6.1 Key distribution and generation

Bluetooth Low Energy devices can distribute keys from the Peripheral to the Central and from the Central to the Peripheral. When using LE legacy pairing, the following keys may be distributed from the Peripheral to the Central:

- LTK using Encryption Information command
- EDIV and Rand using Central Identification command
- IRK using Identity Information command
- Public device or static random address using Identity Address Information command

When using LE Secure Connections, the following keys may be distributed from the Peripheral to the Central:

- IRK using Identity Information command
- Public device or static random address using Identity Address Information command

When using LE legacy pairing, the Central may distribute to the Peripheral the following key:

- LTK using Encryption Information command
- EDIV and Rand using Central Identification command
- IRK using Identity Information command
- Public device or static random address using Identity Address Information command

When using LE Secure Connections, the Central may distribute to the Peripheral the following key:

- IRK using Identity Information command
- Public device or static random address using Identity Address Information command

The keys which are to be distributed in the Transport Specific Key Distribution phase are indicated in the Key Distribution field of the Pairing Request and Pairing Response commands see Section 3.5.1 and Section 3.5.2.

The format of the Initiator Key Distribution / Generation field and Responder Key Distribution / Generation field in the Pairing Request and Pairing Response commands for LE is defined in Figure 3.11.

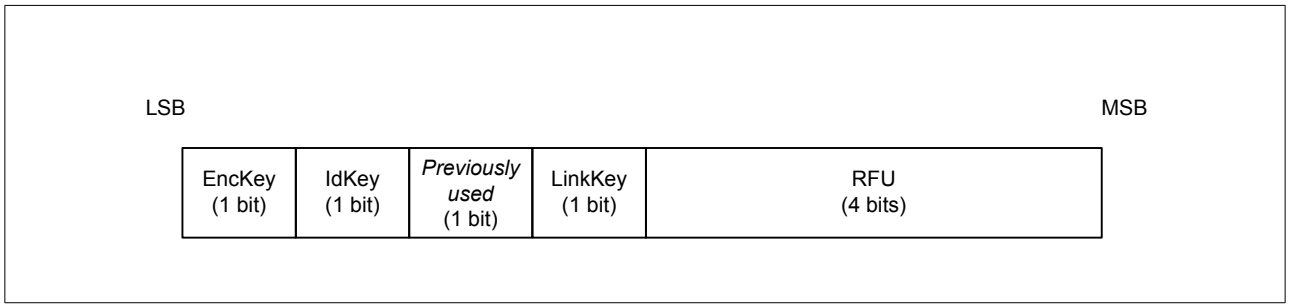



*Figure 3.11: LE Key Distribution format*

The Key Distribution / Generation field has the following flags:

- In LE legacy pairing, EncKey is a 1-bit field that is set to one to indicate that the device shall distribute LTK using the Encryption Information command followed by EDIV and Rand using the Central Identification command.

  In LE Secure Connections pairing, when SMP is running on the LE transport, then the EncKey field shall be ignored. EDIV and Rand shall be set to zero and shall not be distributed.

  When SMP is running on the BR/EDR transport, the EncKey field is set to one to indicate that the device would like to derive the LTK from the BR/EDR Link Key. When

EncKey is set to 1 by both devices in the initiator and responder Key Distribution / Generation fields, the procedures for calculating the LTK from the BR/EDR Link Key shall be used.

- IdKey is a 1-bit field that is set to one to indicate that the device shall distribute IRK using the Identity Information command followed by its public device or static random address using Identity Address Information.
- LinkKey is a 1-bit field. When SMP is running on the LE transport, the LinkKey field is set to one to indicate that the device would like to derive the Link Key from the LTK. When LinkKey is set to 1 by both devices in the initiator and responder Key Distribution / Generation fields, the procedures for calculating the BR/EDR link key from the LTK shall be used. Devices not supporting LE Secure Connections shall set this bit to zero and ignore it on reception. When SMP is running on the BR/EDR transport, the LinkKey field is reserved for future use.

The Initiator Key Distribution / Generation field in the Pairing Request command is used by the Central to request which keys are distributed or generated by the initiator to the responder. The Responder Key Distribution / Generation field in the Pairing Request command is used by the Central to request which keys are distributed or generated by the responder to the initiator. The Initiator Key Distribution / Generation field in the Pairing Response command from the Peripheral defines the keys that shall be distributed or generated by the initiator to the responder. The Responder Key Distribution / Generation field in the Pairing Response command from the Peripheral defines the keys that shall be distributed or generated by the responder to the initiator. The Peripheral shall not set to one any flag in the Initiator Key Distribution / Generation or Responder Key Distribution / Generation field of the Pairing Response command that the Central has set to zero in the Initiator Key Distribution / Generation and Responder Key Distribution / Generation fields of the Pairing Request command.

When using LE legacy pairing, the keys shall be distributed in the following order:

1. LTK by the Peripheral
2. EDIV and Rand by the Peripheral
3. IRK by the Peripheral
4. BD_ADDR by the Peripheral
5. LTK by the Central
6. EDIV and Rand by the Central
7. IRK by the Central
8. BD_ADDR by the Central

When using LE Secure Connections, the keys shall be distributed in the following order:

1. IRK by the Peripheral
2. BD_ADDR by the Peripheral
3. IRK by the Central
4. BD_ADDR by the Central

If a key is not being distributed then the command to distribute that key shall not be sent.

Note: If a key is not distributed, then the capabilities that use this key will not be available. For example, if an LTK is not distributed from the Peripheral to the Central, then the Central cannot encrypt a future link with that Peripheral, therefore pairing would have to be performed again.

The initiator should determine the keys needed based on the capabilities that are required by higher layer specifications. For example, if the initiator determines that encryption is required in a future link with that Peripheral, then the initiator must request that Peripheral's LTK is distributed by setting the EncKey bit to one in the Responder Key Distribution / Generation field of the Pairing Request command.

A device may reject a distributed key by sending the Pairing Failed command with the reason set to "Key Rejected".

If EncKey and IdKey are set to zero in the Initiator Key Distribution / Generation and Responder Key Distribution / Generation fields, then no keys shall be distributed or generated and the link will be encrypted using the generated STK when using LE legacy pairing and LTK when using LE Secure Connections pairing.

Key distribution is complete in the device sending the final key when it receives the Baseband acknowledgment for that key and is complete in the receiving device when it receives the final key being distributed.

### 3.6.2 Encryption Information

Encryption Information is used in the LE legacy pairing Transport Specific Key Distribution to distribute LTK that is used when encrypting future connections. The Encryption Information command is defined in Figure 3.12.

The Encryption Information command shall only be sent when the link has been encrypted or re-encrypted using the generated STK.

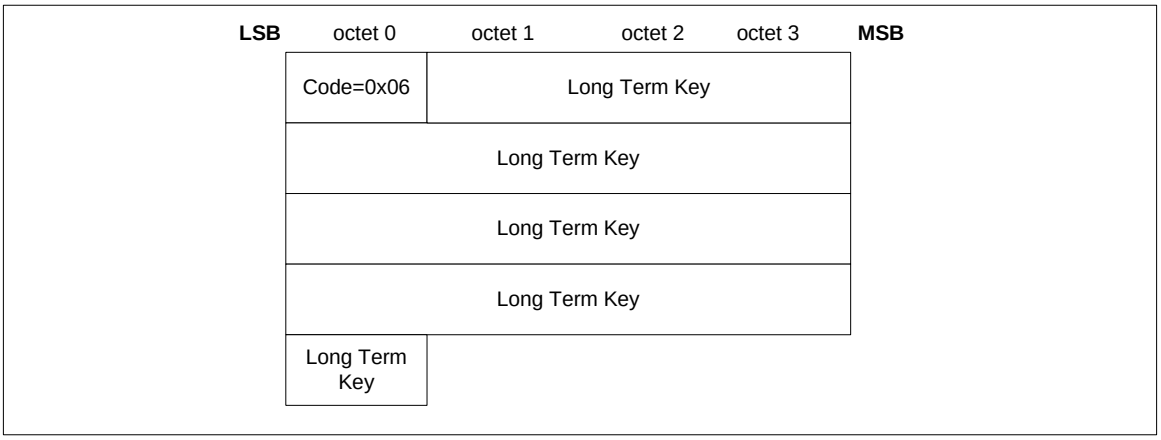



*Figure 3.12: Encryption Information packet*

The following is the data field:

- *Long Term Key (16 octets)*

  The generated LTK value being distributed, see Section 2.4.2.3.

### 3.6.3 Central Identification

Central Identification is used in the LE legacy pairing Transport Specific Key Distribution phase to distribute EDIV and Rand which are used when encrypting future connections. The Central Identification command is defined in Figure 3.13.

The Central Identification command shall only be sent when the link has been encrypted or re-encrypted using the generated STK.

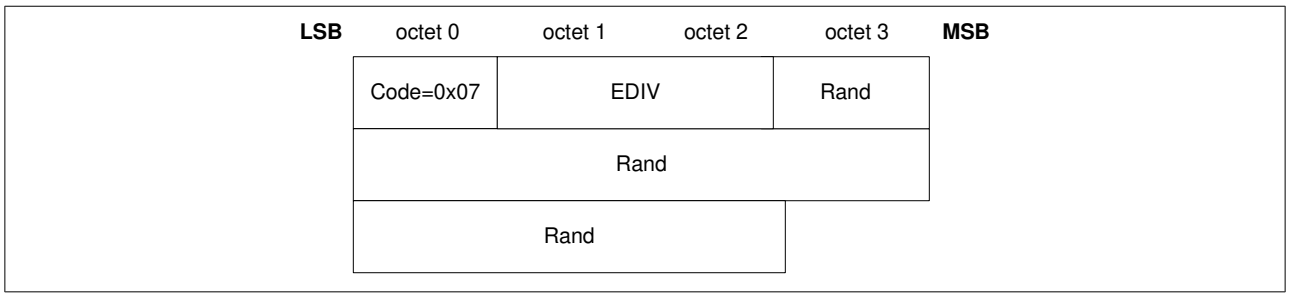



*Figure 3.13: Central Identification packet*

The following data fields are used:

- *EDIV (2 octets)*

  The EDIV value being distributed (see Section 2.4.2.3).

- *Rand (8 octets)*

  64-bit Rand value being distributed (see Section 2.4.2.3).

### 3.6.4 Identity Information

Identity Information is used in the Transport Specific Key Distribution phase to distribute the IRK. The Identity Information command is defined in Figure 3.14.

The Identity Information command shall only be sent when the link has been encrypted or re-encrypted using the generated key.

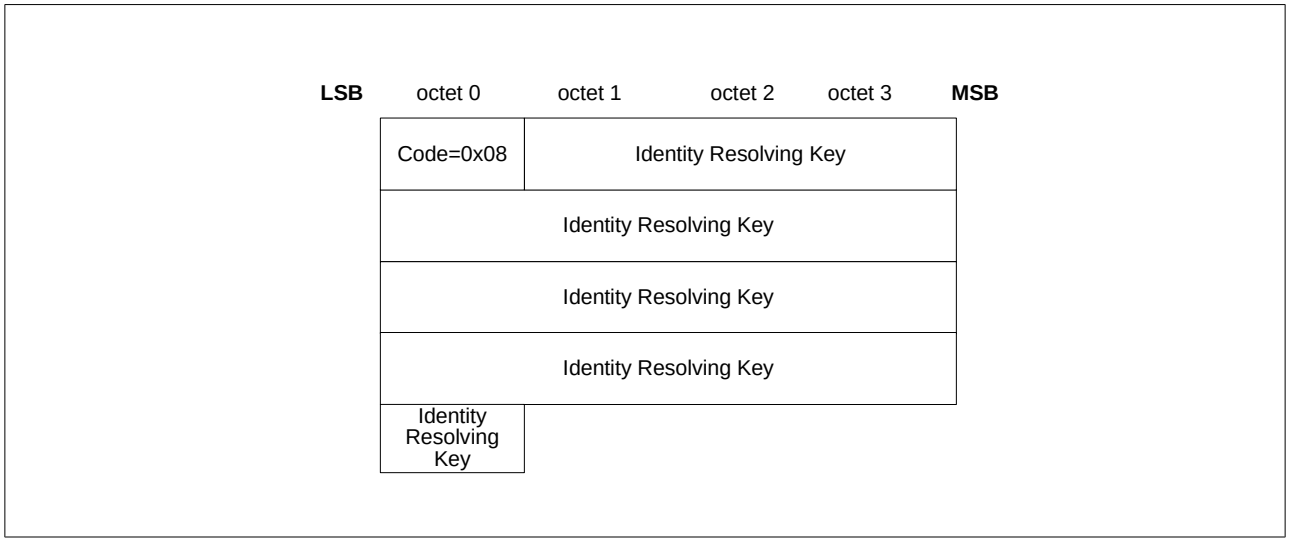



*Figure 3.14: Identity Information packet*

The following are the data fields:

- *Identity Resolving Key (16 octets)*

  128-bit IRK value being distributed (see Section 2.4.2.1).

An all zero Identity Resolving Key data field indicates that a device does not have a valid resolvable private address.

### 3.6.5 Identity Address Information

Identity Address Information is used in the Transport Specific Key Distribution phase to distribute its public device address or static random address. The Identity Address Information command is defined in Figure 3.15.

The Identity Address Information command shall only be sent when the link has been encrypted or re-encrypted using the generated key.

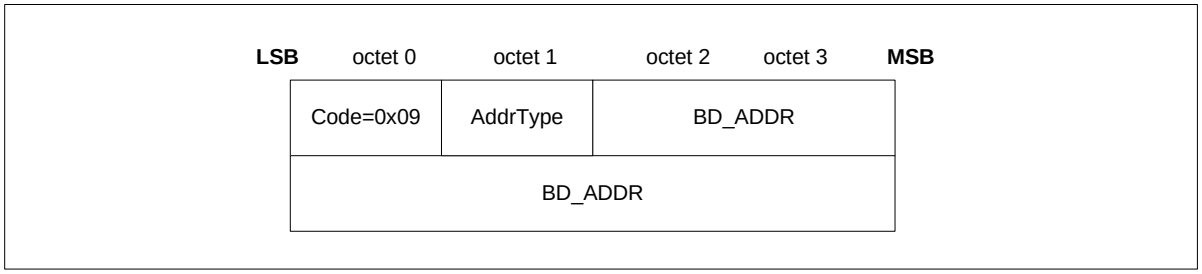



*Figure 3.15: Identity Address Information packet*

The data fields are:

- *AddrType (1 octet)*

  If BD_ADDR is a public device address, then AddrType shall be set to 0x00. If BD_ADDR is a static random device address then AddrType shall be set to 0x01.

- *BD_ADDR (6 octets)*

  This field is set to the distributing device's public device address or static random address.

### 3.6.6 [This section is no longer used]

### 3.6.7 Security Request

The Security Request command is used by the Peripheral to request that the Central initiates security with the requested security properties, see Section 2.4.6. The Security Request command is defined in Figure 3.16.

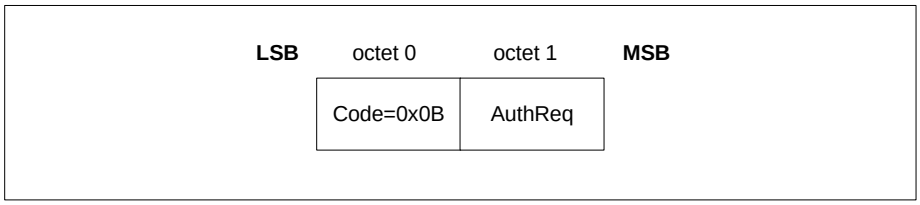



*Figure 3.16: Security Request packet*

The following data field is used:

- *AuthReq (1 octet)*

  The AuthReq field is a bit field that indicates the requested security properties (see Section 2.3.1) for the STK or LTK and GAP bonding information (see [Vol 3] Part C, Section 9.4).

  Figure 3.3 defines the authentication requirements bit field.

The Bonding_Flags field is a 2-bit field that indicates the type of bonding being requested by the responding device as defined in Table 3.6.

The MITM field is a 1-bit flag that is set to one if the device is requesting MITM protection, otherwise it shall be set to 0. A device sets the MITM flag to one to request an Authenticated security property for the STK when using LE legacy pairing and the LTK when using LE Secure Connections.

The SC field is a 1 bit flag. If LE Secure Connections pairing is supported by the device, then the SC field shall be set to 1, otherwise it shall be set to 0. If both devices support LE Secure Connections pairing, then LE Secure Connections pairing shall be used, otherwise LE Legacy pairing shall be used.

The keypress field is a 1-bit flag that is used only in the Passkey Entry protocol and shall be ignored in other protocols. When both sides set that field to one, Keypress notifications shall be generated and sent using SMP Pairing Keypress Notification PDUs.

# 4 REFERENCES

[1] NIST Special Publication 800-38B: http://dx.doi.org/10.6028/NIST.SP.800-38B

# Appendix A EDIV and Rand Generation

EDIV and Rand are used by the responding device to identify an initiator and recover LTK. This section provides an example of how the distributed EDIV value is a masked version of the real value (DIV) which is used to recover LTK. Other methods can be used that provide equal or higher levels of confidentially for DIV.

## A.1 EDIV masking

The masking process uses a Diversifier Hiding Key (DHK) which is a 128-bit key that is never distributed.

DHK can be assigned, randomly generated by the device during manufacturing, part of a key hierarchy (see Appendix B, Section B.2.3) or some other method could be used, that results in DHK having 128 bits of entropy.

If DHK is changed then DIV values cannot be recovered from previously distributed EDIV values.

Section A.1.1 defines a cryptographic function that is used by the responding device when generating EDIV and recovering DIV.

Section A.1.2 describes how a responding device generates an EDIV value to be distributed to an initiating device and Section A.1.3 describes how the responding device recovers DIV from a distributed EDIV value.

### A.1.1 DIV mask generation function *dm*

DIV is masked before distribution and unmasked during the encryption session setup using the output of the DIV mask generation function *dm*.

The following are inputs to the DIV mask generation function *dm*:

*k* is 128 bits
*r* is 64 bits
*padding* is 64 bits

*r* is concatenated with padding to generate *r'* which is used as the 128-bit input parameter *plaintextData* to security function *e*:

r' = padding || r

The least significant octet of *r* becomes the least significant octet of *r'* and the most significant octet of *padding* becomes the most significant octet of *r'*.

For example, if the 64-bit value *r* is 0x123456789ABCDEF0 then *r'* is 0x0000000000000000123456789ABCDEF0.

The output of the DIV mask generation function *dm* is

$$dm(k, r) = e(k, r') \ mod \ 2^{16}$$

The output of the security function e is then truncated to 16 bits by taking the least significant 16 bits of the output of *e* as the result of *dm*.

### A.1.2 EDIV generation

The responding device generates a 64-bit random value, *Rand.* The *Rand* value is used to generate 16-bit Y using the DIV mask generation function *dm* with the input parameter *k* set to DHK and the input parameter *r* set to *Rand*.

$$Y = dm(DHK, Rand)$$

The responding device then masks the DIV value to be distributed by bitwise XORing it with Y to generate EDIV.

$$EDIV = Y \text{ xor } DIV$$

EDIV and Rand are distributed to an initiating device during the transport specific key distribution phase using the Central Identification command.

### A.1.3 DIV recovery

When the responding device receives a request to encrypt a session it calculates Y using the DIV mask generation function *dm* with the input parameter *k* set to DHK and the input parameter *r* set to *Rand.* The Y value is bitwise XORed with *EDIV* from the initiator to recover DIV.

$$DIV = Y \text{ xor } EDIV$$

The recovered DIV value can then used to recover LTK which is used to enable encryption on the link.

# Appendix B Key Management

The security provided by different methods can vary and care should be taken to ensure that a chosen method is suitable for a device's requirements.

Section B.1 uses a database for managing the keys. Section B.2 uses a key hierarchy to manage the keys.

## B.1 Database lookup

The LTK which is distributed is a 128-bit random number which is stored in a database, using EDIV as an index. There is no direct relationship between LTK and EDIV.

IRK and other keys shall also be stored in the database. There is no relationship between the keys stored in the database or distributed LTKs, EDIVs, or Rands.

If the example EDIV and Rand generation method described in Section A.1 is used then the database shall be used to store DHK. DIV should be used as the index to recover LTK.

## B.2 Key hierarchy

A key hierarchy can be used to generate the keys.

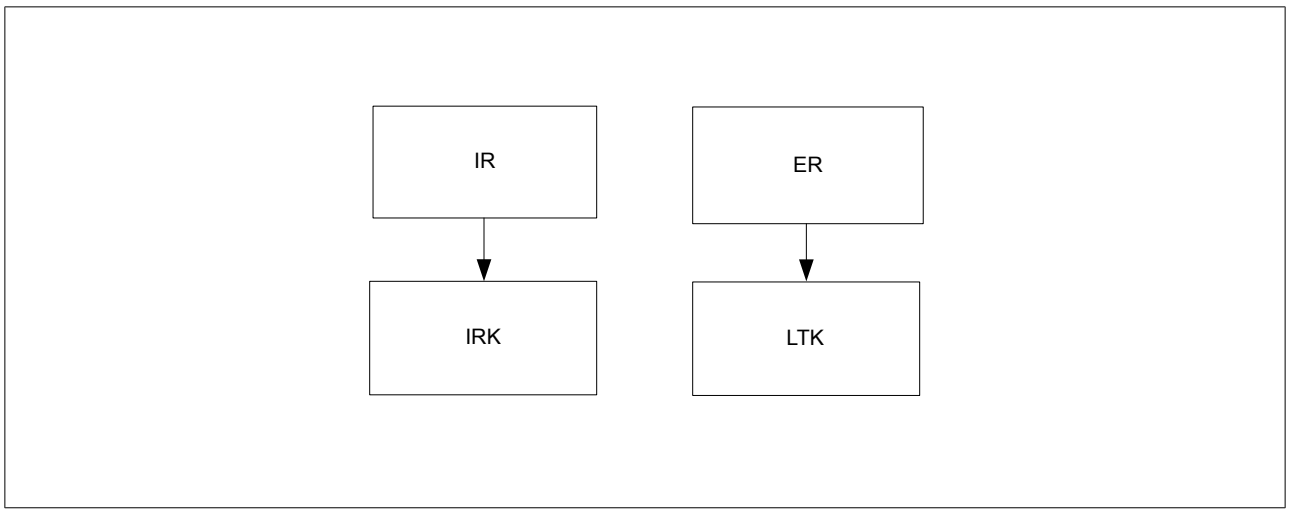



*Figure B.1: Example key hierarchy*

Figure B.1 is an example key hierarchy where LTK is generated from a common ER key and IRK is generated from a common IR key.

1. ER is a 128-bit key generated for each LE device that supports encrypted connections. It is used to generate LTK using EDIV; see Section B.2.2.
2. IR is a 128-bit key generated for each LE device that supports encrypted connections or uses random addresses. IR is used to generate IRK; see Section B.2.3. It can also be used to generate DHK; see Section A.1.

New LTK, EDIV, and Rand values shall be generated each time they are distributed. If ER is changed then any previously distributed LTK keys will no longer be valid.

The distributed IRK shall be the same for all devices it is distributed to. If IR is changed then any previously distributed IRK keys will no longer be valid.

The distributing device only needs to store IR and ER. LTK and IRK can be regenerated when they are required. This reduces the storage requirements on the distributing device.

Section B.2.1 defines an example of the key diversifying function which can be used to generate LTK, IRK, and other keys. Other implementations of this function can be used depending upon the exact security requirements of the device.

The NIST Special Publication 800-108 (http://csrc.nist.gov/publications/PubsSPs.html) defines key derivation functions which could be used instead of the example diversifying function *d1*.

### B.2.1 Diversifying function *d1*

Diversified keys are generated with function *d1*. The diversifying function *d1* makes use of the security function *e*.

The following are inputs to diversifying function *d1*:

*k* is 128 bits
*d* is 16 bits
*r* is 16 bits
*padding* is 96 bits

*d* is concatenated with *r* and *padding* to generate *d',* which is used as the 128-bit input parameter *plaintextData* to security function *e*:

d' = padding || r || d

The least significant octet of *d* becomes the least significant octet of *d'* and the most significant octet of *padding* becomes the most significant octet of *d'*.

For example, if the 16-bit value *d* is 0x1234 and the 16-bit value *r* is 0xABCD, then *d'* is 0x0000000000000000000000ABCD1234.

The output diversifying function *d1* is:

*d1*(k, d, r) = e(k, d')

The 128-bit output of the security function *e* is used as the result of diversifying function *d1*.

### B.2.2 Generating keys from ER

ER is used to generate LTK. ER can be assigned, randomly generated by the device during manufacturing or some other method could be used, that results in ER having 128 bits of entropy.

The EDIV and Rand generation method described in Appendix A shall be used. LTK is the result of the diversifying function *d1* with the ER as the input parameter *k*, the DIV as the input parameter *d*, and the value 0 as the input parameter *r*; see Section B.2.1.

LTK = *d1*(ER, DIV, 0)

LTK can be recovered from ER and DIV by repeating the calculation when LTK is required.

This method provides an LTK with limited amount of entropy because LTK is directly related to EDIV and may be less secure than other generation methods.

To reduce the probability of the same LTK value being generated, the DIV values must be unique for each LTK, EDIV, and Rand set that is distributed.

A method for preventing a malicious device from repeatedly pairing and collecting LTK and DIV information, which could be used in a known plain text attack in ER, should be implemented.

Note: The generation of LTK using ER is only applicable when doing LE Legacy Pairing.

### B.2.3 Generating keys from IR

IR can be used to generate IRK and other required keys. IR can be assigned, randomly generated by the device during manufacturing or some other method could be used, that results in IR having 128 bits of entropy.

IRK is the result of the diversifying function *d1* with IR as the input parameter *k* and the value 1 as the input parameter *d* and the value 0 as the input parameter r; see Section B.2.1.

IRK = *d1*(IR, 1, 0)

If the example EDIV and Rand generation method described in Appendix A, Section A.1 is used then DHK can be the result of diversifying function *d1* with IR as the input parameter *k* and the value 3 as the input parameter *d* and the value 0 as the input parameter r.

DHK = *d1*(IR, 3, 0)

Other keys can be generated by using different values for *k* as the input to the diversifying function *d1*. If the value of *k* is reused for a given IR then the resulting key will be the same.

Note: The generation of DHK using IR is only applicable when doing LE Legacy Pairing. The generation of IRK using IR is applicable both when doing LE Legacy Pairing and LE Secure Connections Pairing.

# Appendix C Message sequence charts

This appendix illustrates only the most common scenarios; it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or Host interface. If any of these charts differ with text elsewhere in this Part, then that text overrides these charts.

A flow diagram of pairing is shown in Figure C.1. The process has 4 steps. Step 2 has a number of different options.

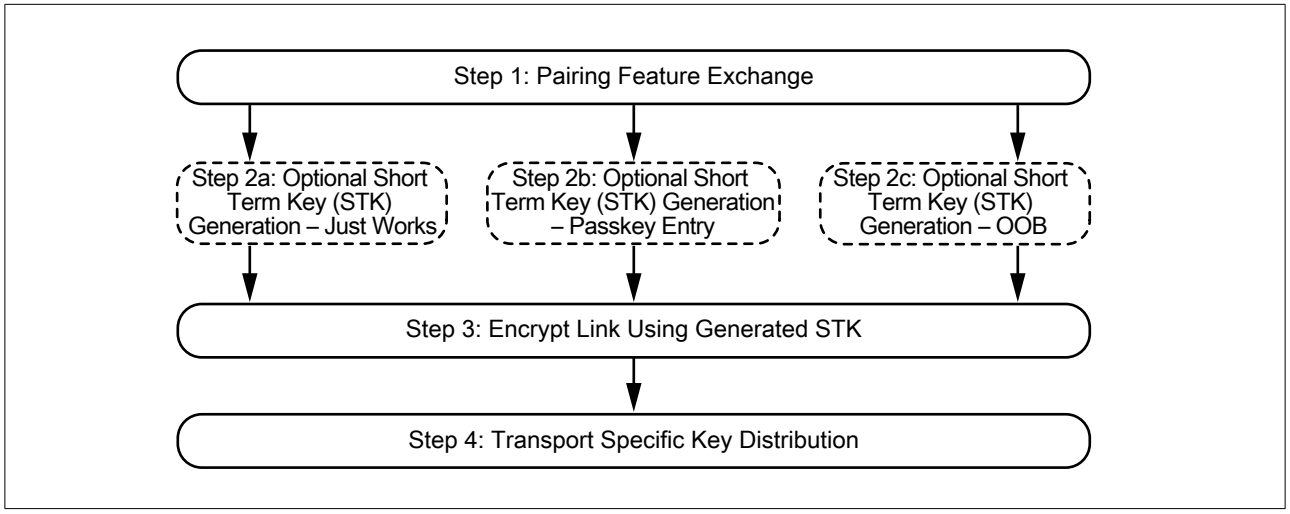



*Figure C.1: Pairing process overview*

Note: In all the MSCs, the Central is also the Initiating Device and the Peripheral is also the Responding (non-Initiating) Device.

## C.1 Phase 1: Pairing feature exchange

The Central initiates the pairing procedure using Pairing Request command as shown in Figure C.2.

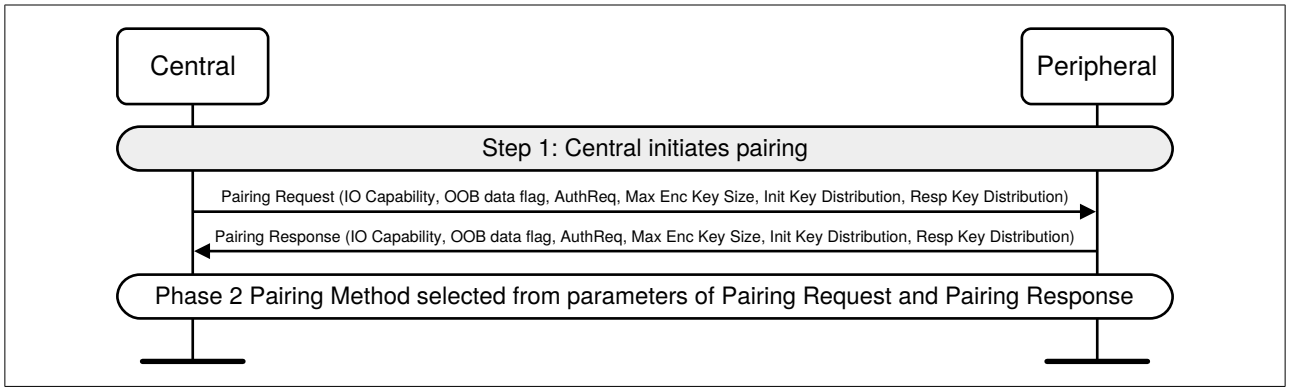



*Figure C.2: Pairing initiated by Central*

### C.1.1 Peripheral security request – Central requests pairing

The Peripheral may request the Central initiates security procedures. Figure C.3 shows an example where the Peripheral requests security and the Central initiates pairing in response.

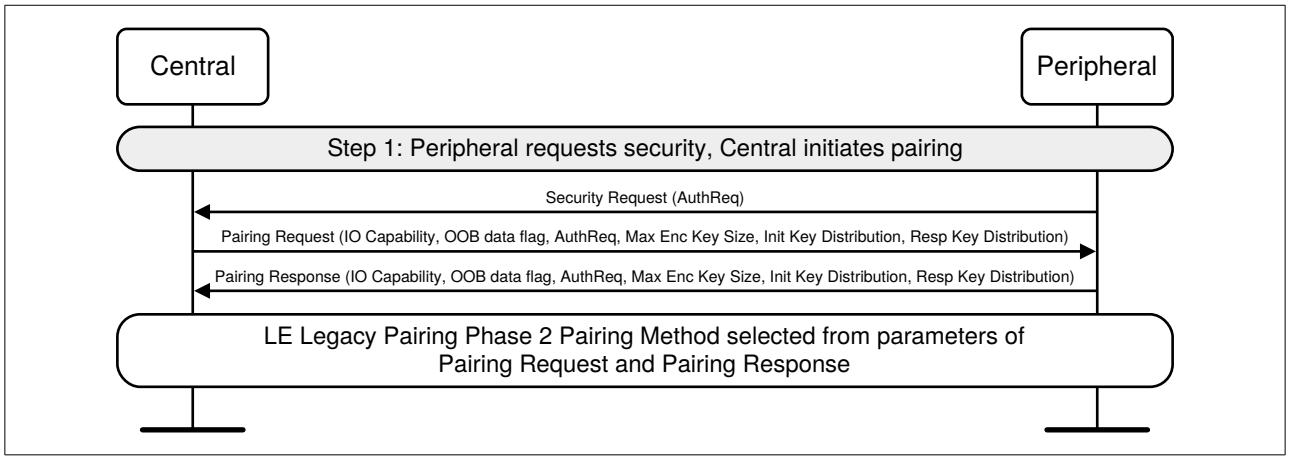



*Figure C.3: Peripheral security request, Central initiated pairing*

## C.2 Phase 2: Authenticating and encrypting

After Pairing Feature Exchange has completed a pairing method is selected one of the possible short term key generation sequences are used. This can be Just Works, Passkey Entry or Out of Band pairing method.

### C.2.1 LE legacy pairing

The following subsections include message sequence charts for LE legacy pairing.

### C.2.1.1 Legacy Phase 2: Short Term Key generation – Just Works

After Pairing Feature Exchange has completed a pairing method is selected. Figure C.4 shows the Just Works pairing method.

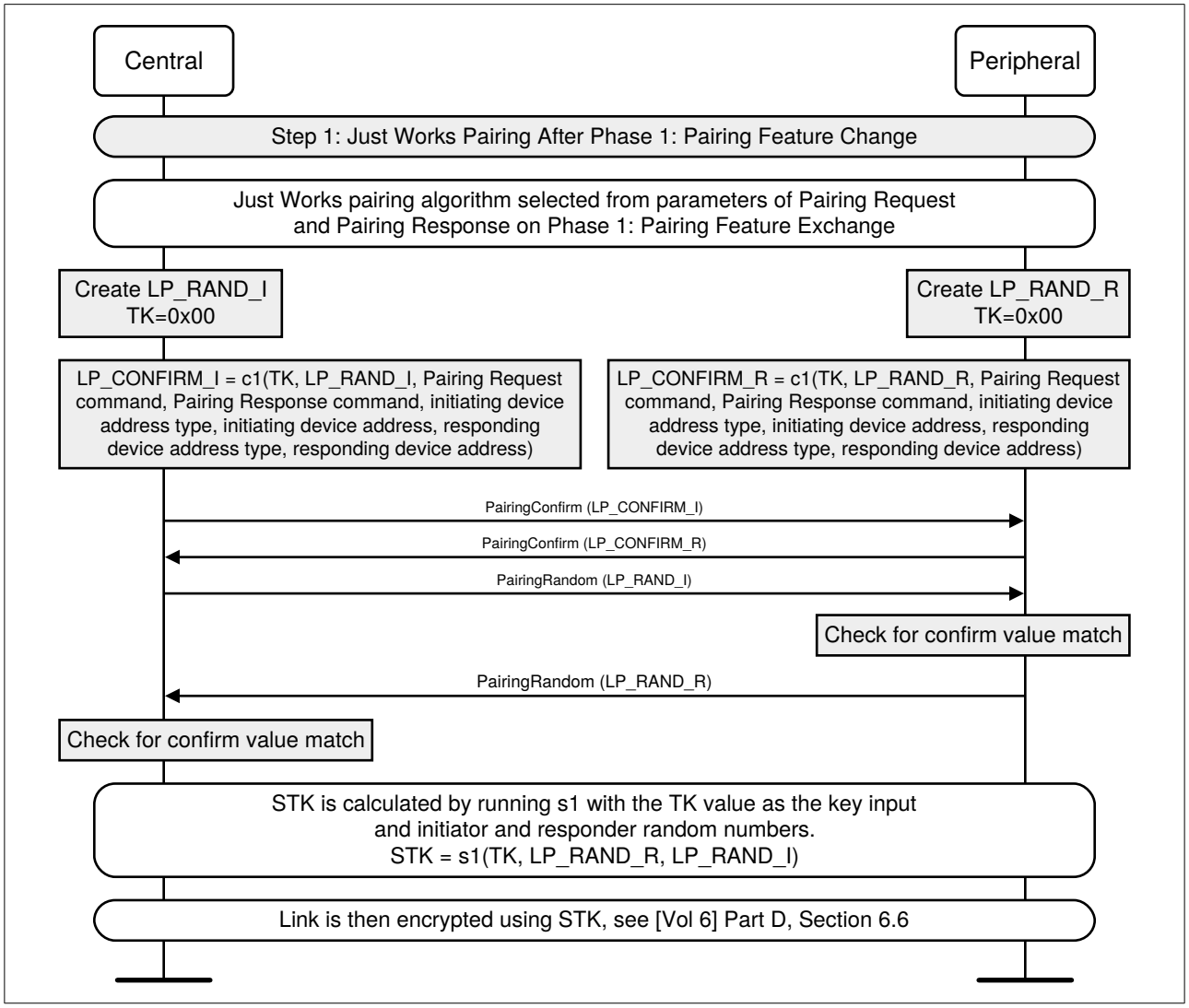



*Figure C.4: Legacy Just Works pairing method*

### C.2.1.2 Legacy Phase 2: Short Term Key generation – Passkey Entry

After Pairing Feature Exchange has completed, a pairing method is selected. Figure C.5 shows the Passkey Entry pairing method.

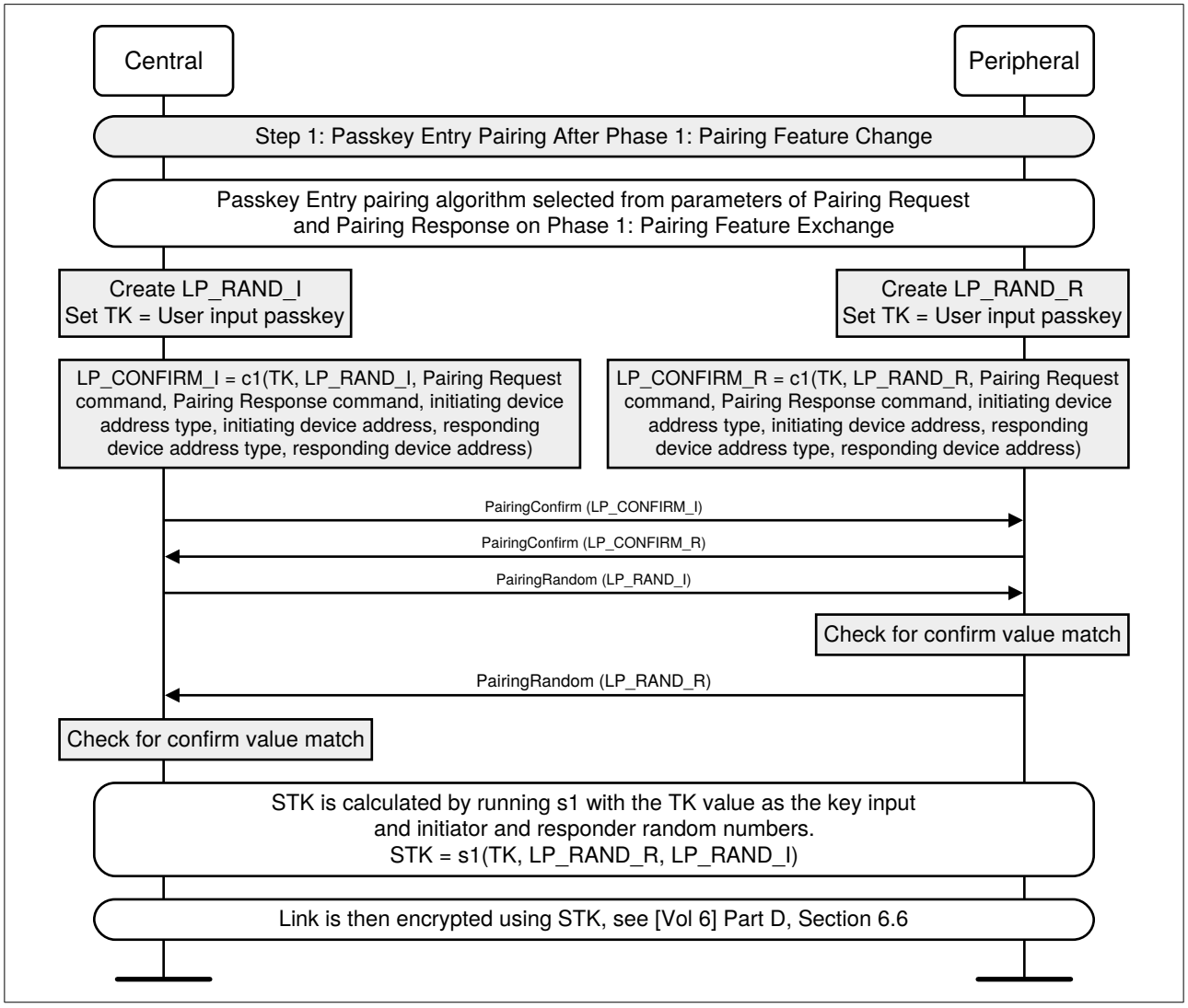



*Figure C.5: Legacy Passkey Entry pairing method*

### C.2.1.3 Legacy Phase 2: Short Term Key generation – Out of Band

After Pairing Feature Exchange has completed, a pairing method is selected. Figure C.6 shows the Out of Band pairing method.

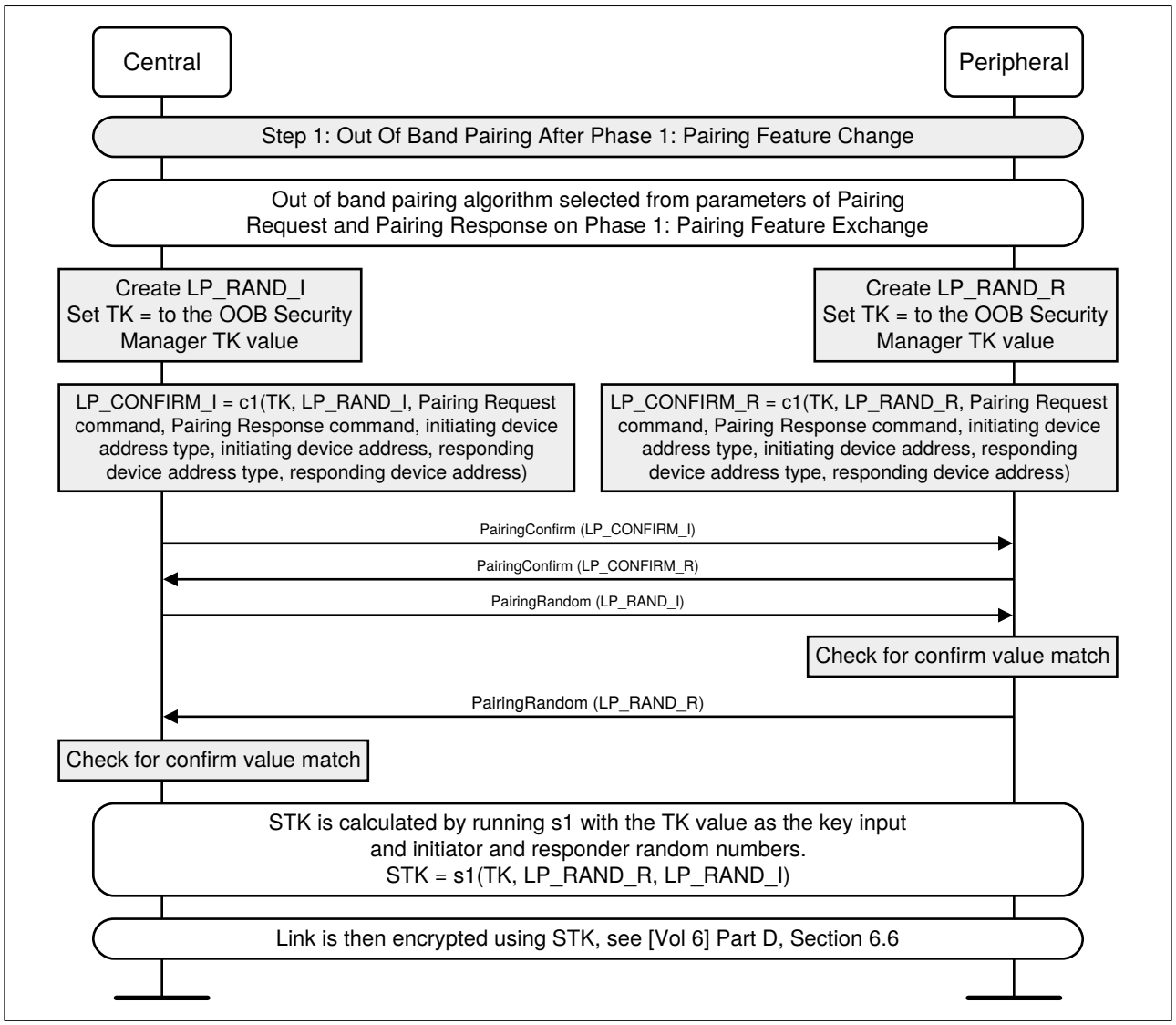



*Figure C.6: Legacy OOB pairing method*

## C.2.2 LE Secure Connections

The following subsections include message sequence charts for LE Secure Connections.

### C.2.2.1 Public key exchange

In Step 1a and 1b, the two devices exchange public keys. The Central sends its public key to the Peripheral followed by Peripheral sending its public key to the Central.

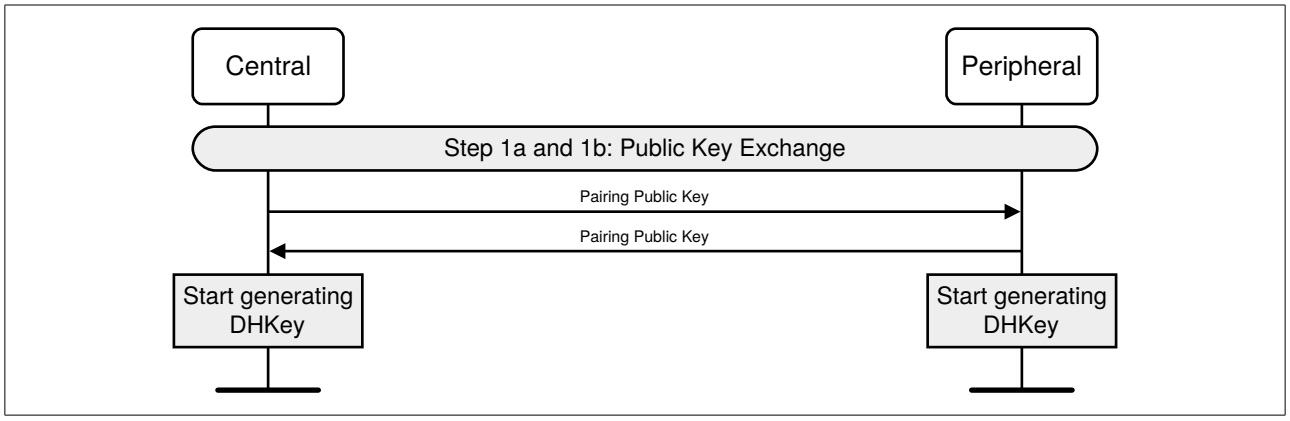



*Figure C.7: Pairing Phase 2 – Public Key Exchange*

### C.2.2.2 Authentication stage 1

The following subsections include message sequence charts for the success and failure cases in each association model.

#### C.2.2.2.1 Successful Numeric Comparison (or Just Works)

The numeric comparison step will be done when both devices have output capabilities, or if one of the devices has no input or output capabilities. If both devices have output capabilities, this step requires the displaying of a user confirmation value. This value should be displayed until the end of step 2. If one or both devices do not have output capabilities, the same protocol is used but the Hosts will skip the step asking for the user confirmation.

Note: The sequence for Just Works is identical to that of Numeric Comparison with the exception that the Host will not show the numbers to the user.

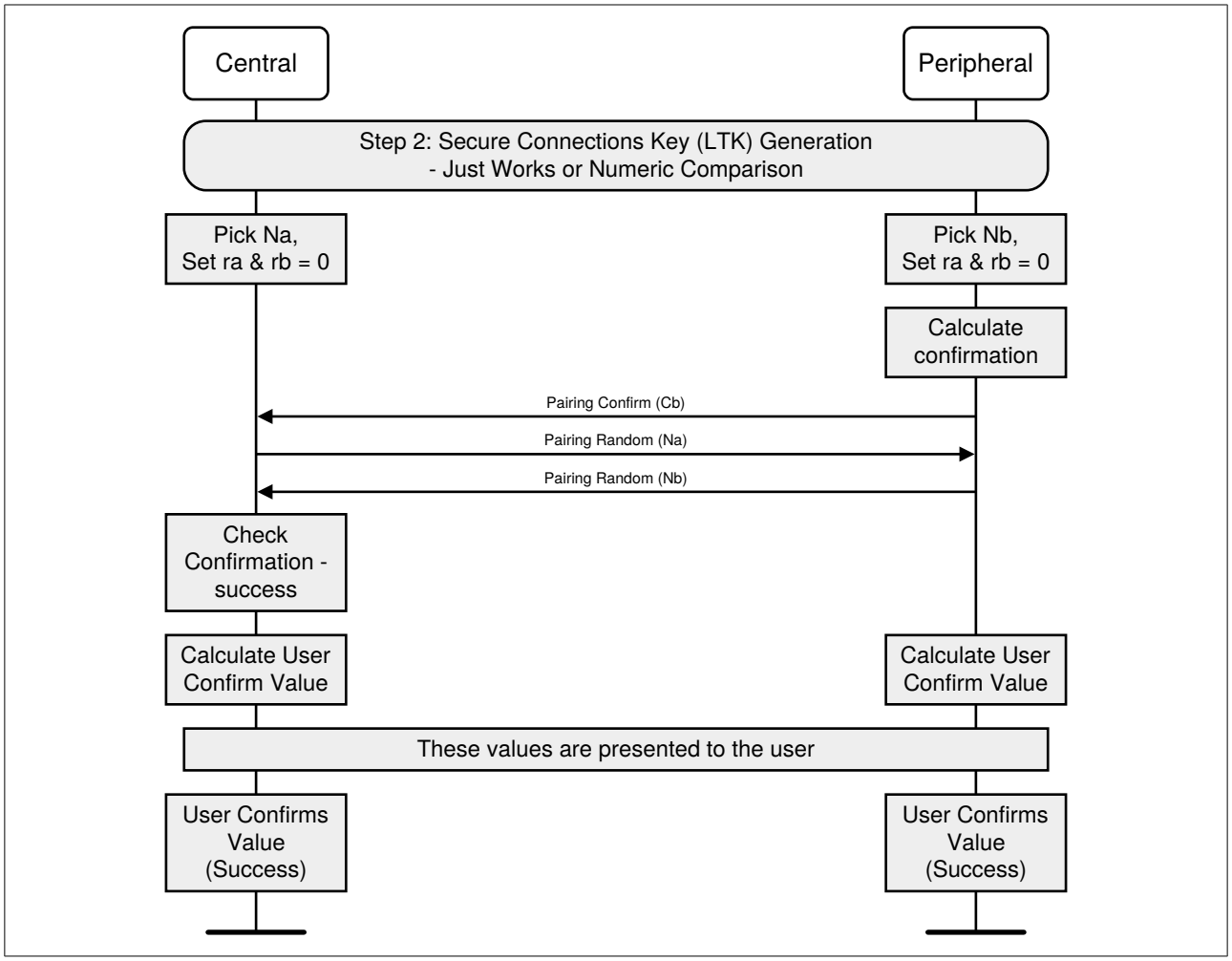



*Figure C.8: Pairing Phase 2, authentication stage 1, successful Numeric Comparison*

#### C.2.2.2.2 Numeric Comparison – Confirm Check failure on the Initiator side

If the Confirm value calculated by the Initiator is not equal to the Commitment value received from the Responder, the Initiator will abort Pairing process by sending Pairing Failed with reason “Confirm Value Failed”.

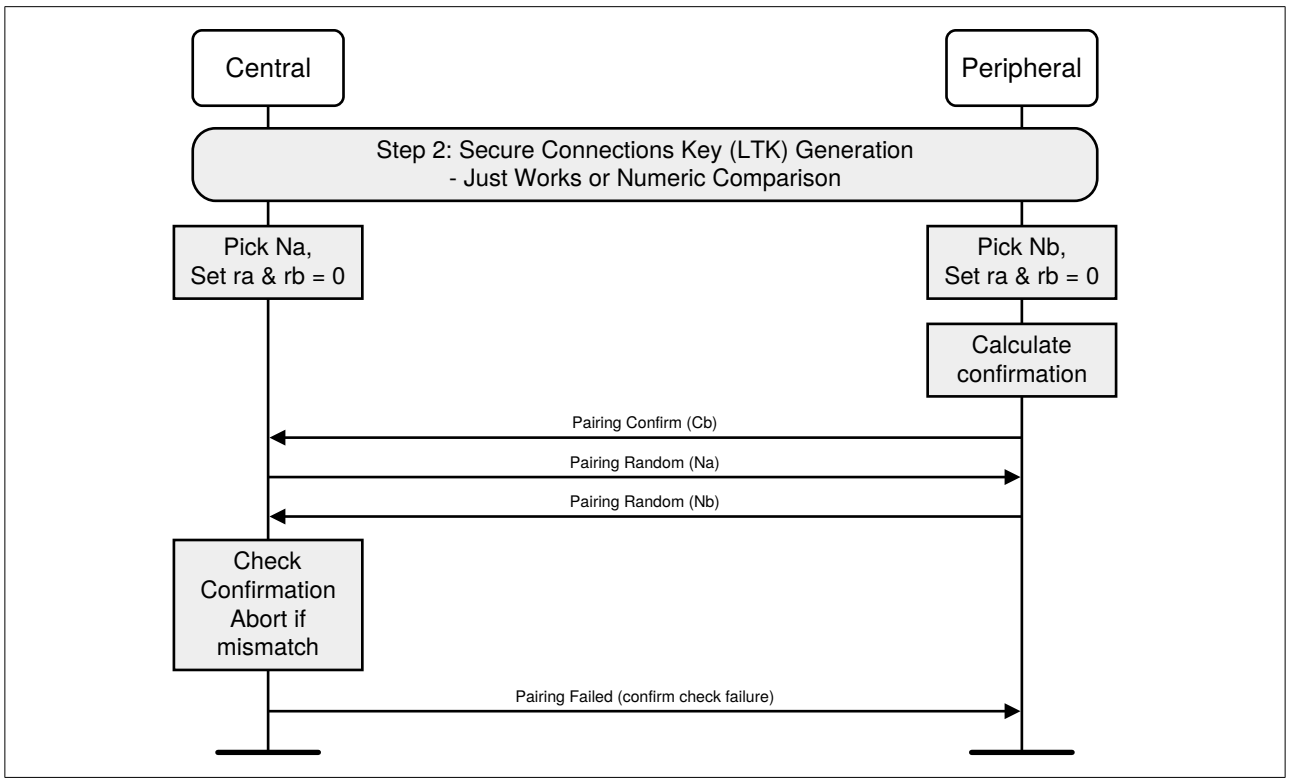



*Figure C.9: Pairing Phase 2, authentication stage 1, Numeric Comparison – Confirm Check failure on Initiator side*

#### C.2.2.2.3 Numeric Comparison failure on the Initiator side

If the numeric comparison fails on the initiating side due to the user indicating that the confirmation values do not match, Pairing is terminated.

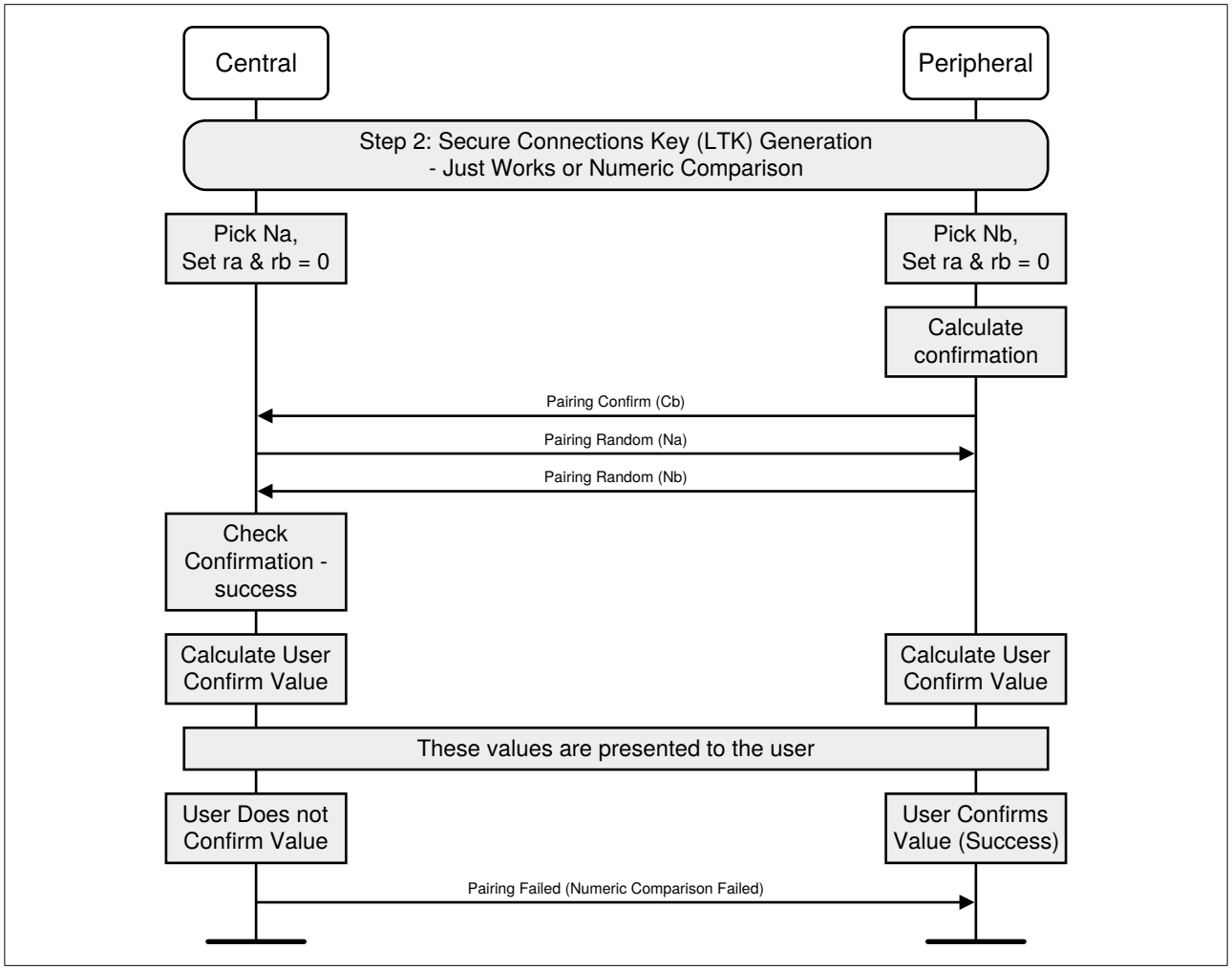



*Figure C.10: Pairing Phase 2, authentication stage 1, Numeric Comparison failure on Initiator side*

#### C.2.2.2.4 Numeric Comparison failure on the Responding side

If the numeric comparison fails on the responding side due to the user indicating that the confirmation values do not match, Pairing is terminated.

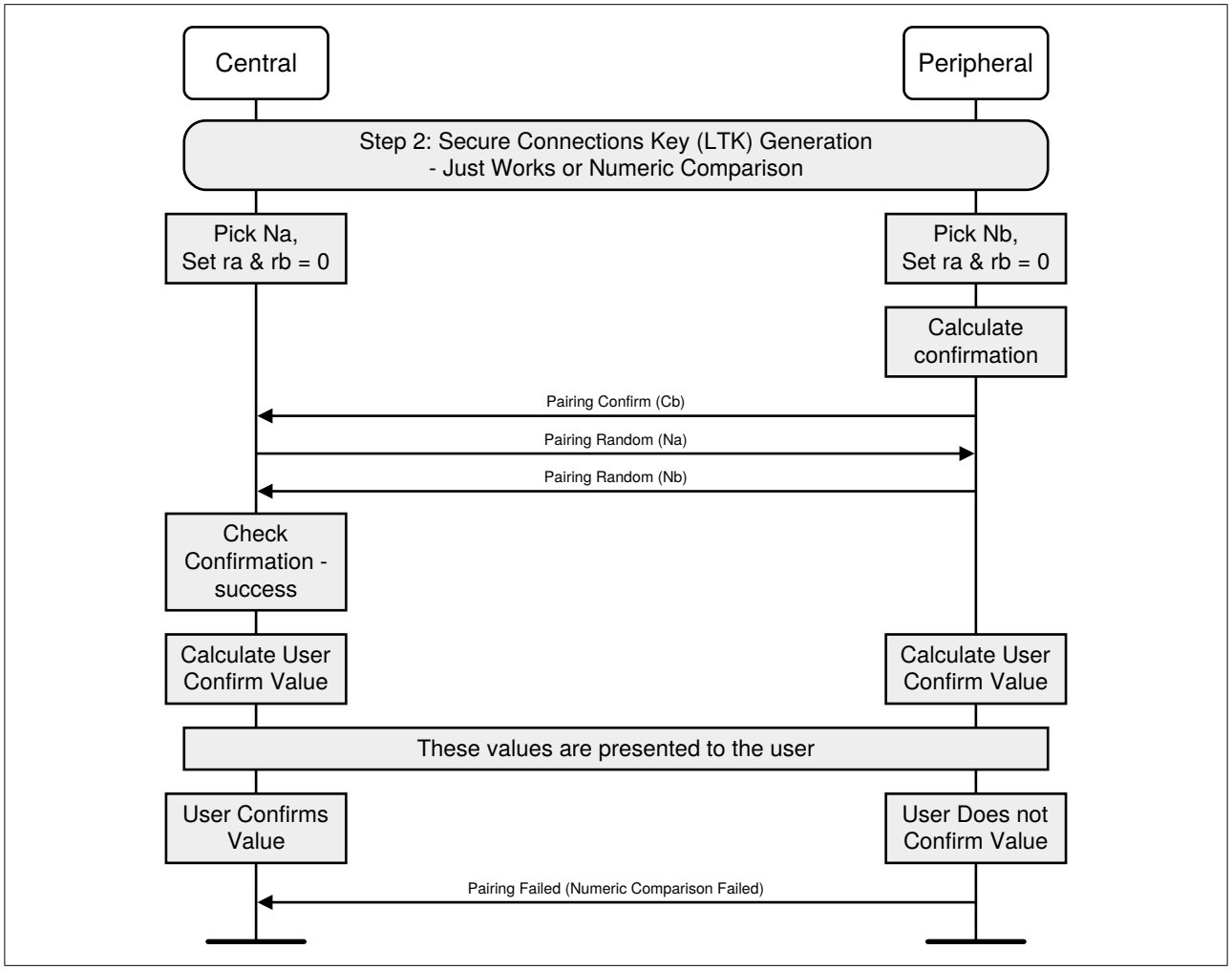



*Figure C.11: Pairing Phase 2, authentication stage 1, Numeric Comparison failure on Responding side*

#### C.2.2.2.5 Successful Passkey Entry

The Passkey Entry step is used in two cases: when one device has numeric input only and the other device has either a display or numeric input capability. In this step, one device display a number to be entered by the other device or the user enters a

number on both devices. Key press notification messages are shown during the user input phase.

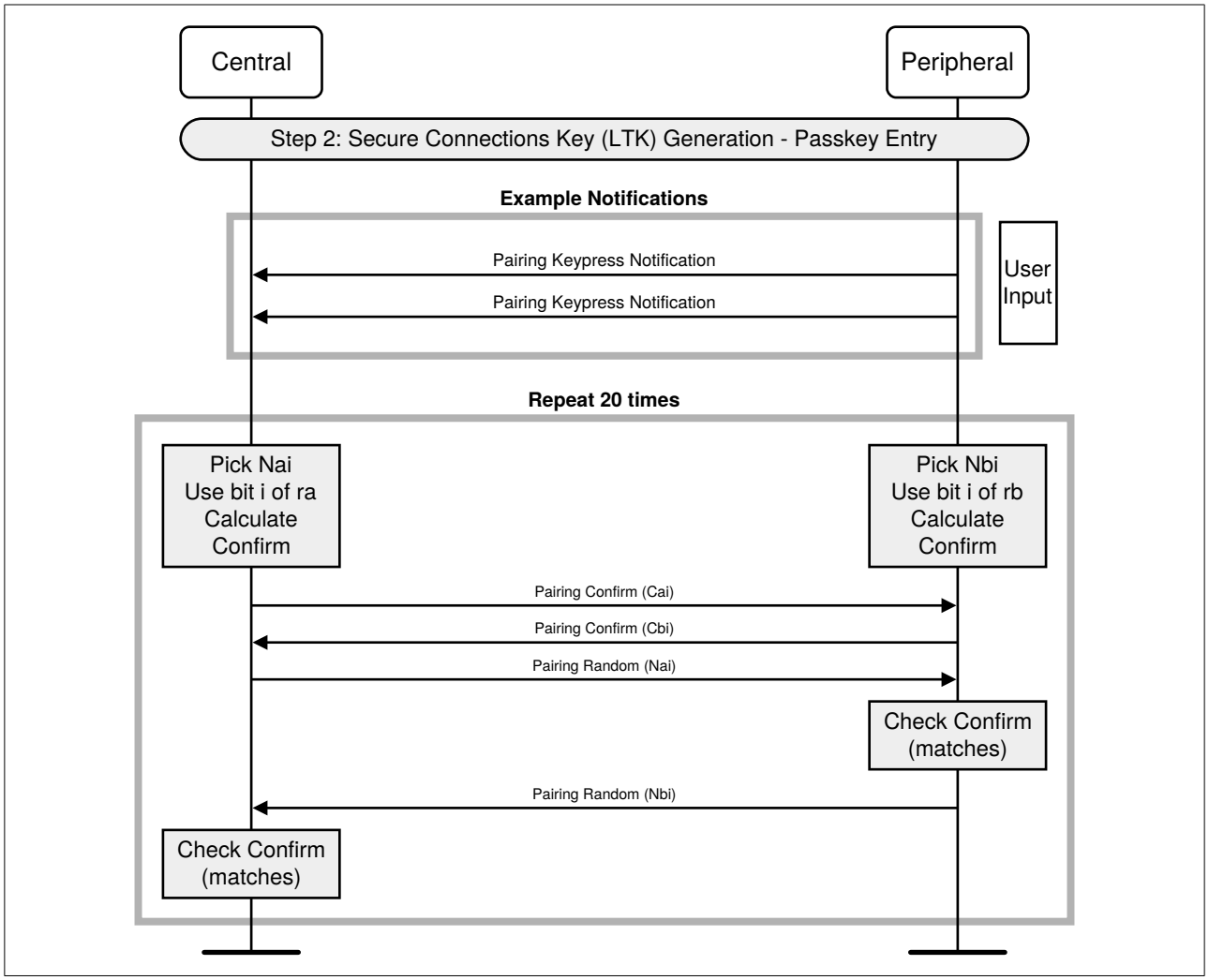



*Figure C.12: Pairing Phase 2, authentication stage 1, Successful Passkey Entry*

Note: Passkey Entry can prolong the pairing experience because of the time required to execute 20 repetitions over SMP.

#### C.2.2.2.6 Passkey Entry – Confirm Check failure on the Responder side

If during one of the 20 repetitions, the Confirm calculated by the Responder is not equal to the one received from the Initiator, the Responder will abort the Pairing process by sending Pairing Failed with reason “Confirm Value Failed.”

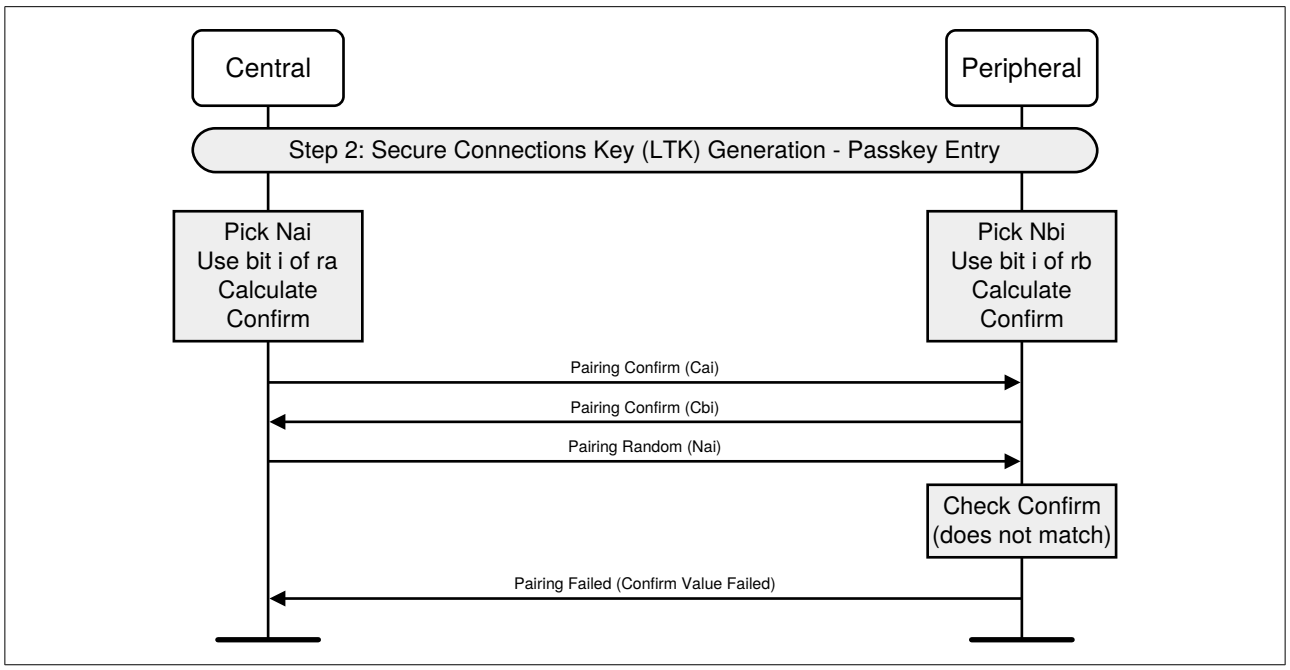



*Figure C.13: Pairing Phase 2, authentication stage 1, Passkey Entry – Confirm Check failure on Responder side*

#### C.2.2.2.7 Passkey Entry – Confirm Check failure on the Initiator side

If during one of the 20 repetitions, the Confirm calculated by the Initiator is not equal to the one received from the Responder, the Initiator will abort the Pairing process by sending Pairing Failed with reason “Confirm Value Failed”.

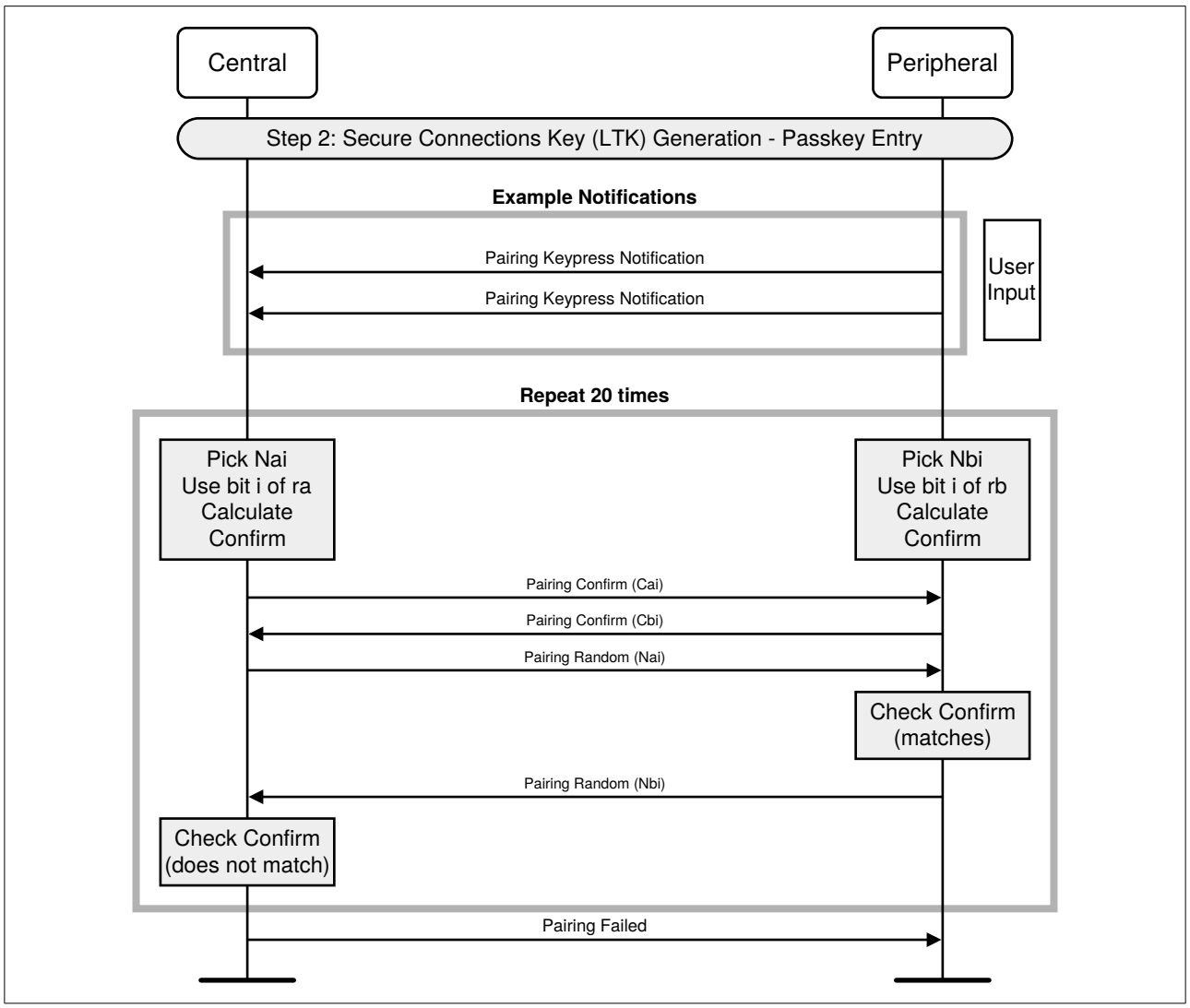



*Figure C.14: Pairing Phase 2, authentication stage 1, Passkey Entry – Confirm Check failure on Initiator side*

#### C.2.2.2.8 Passkey Entry failure on the Responding side

If the passkey entry fails on the responding side, Pairing is terminated.

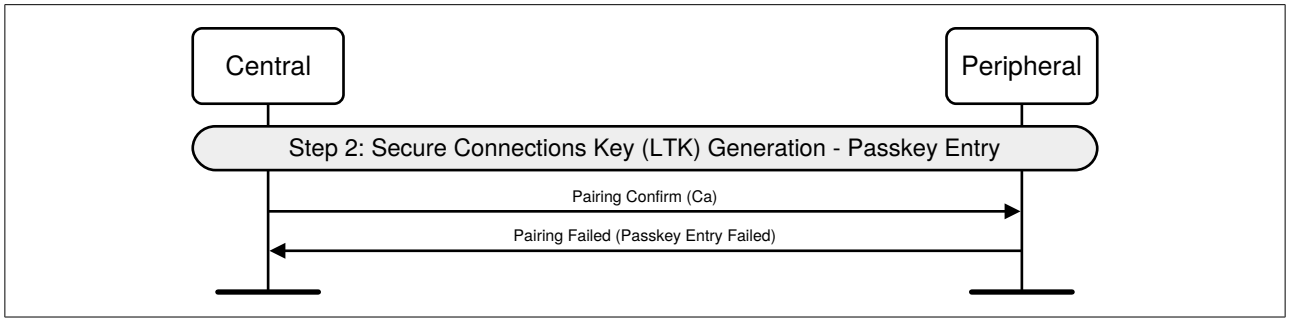



*Figure C.15: Pairing Phase 2, authentication stage 1, Passkey Entry failure on Responding side*

#### C.2.2.2.9 Passkey Entry Failure on the Initiator Side

If the passkey entry fails on the initiating side, Pairing is terminated.

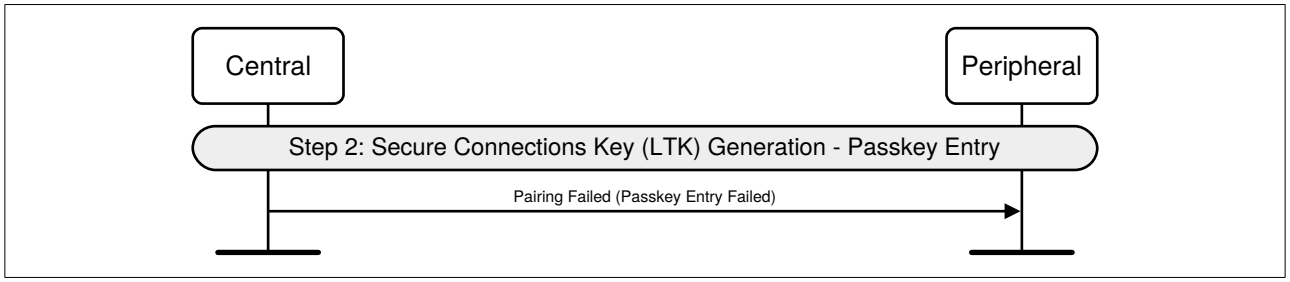



*Figure C.16: Pairing Phase 2, authentication stage 1, Passkey Entry failure on Initiator side*

#### C.2.2.2.10 Successful Out of Band

The OOB authentication will only be done when at least one device has some OOB information to use. This step requires no user interaction.

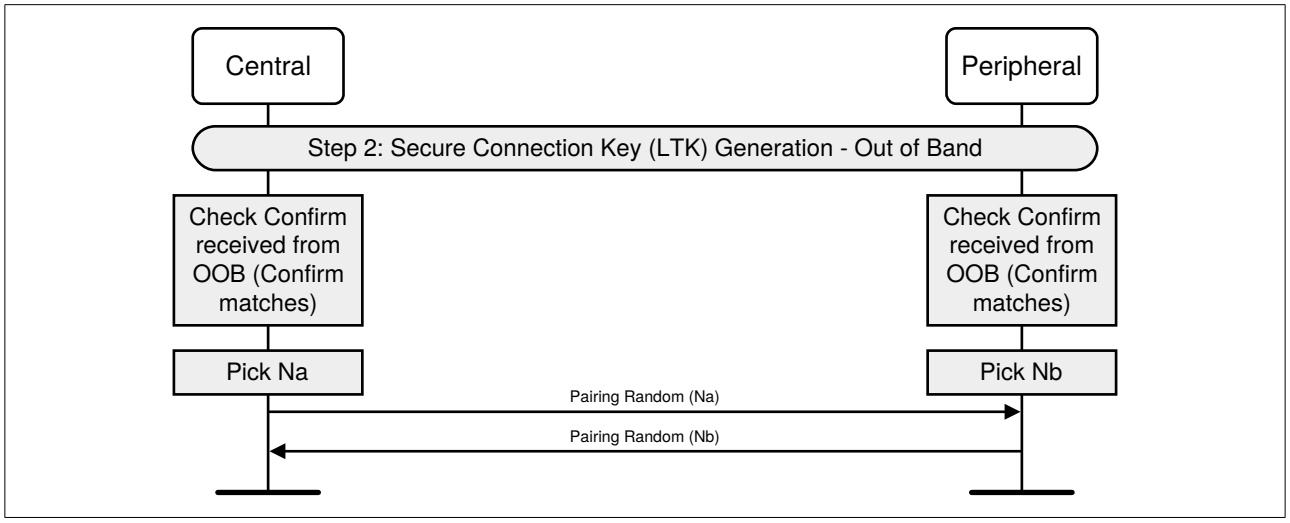



*Figure C.17: Pairing Phase 2, authentication stage 1, successful Out of Band*

#### C.2.2.2.11 Out of Band – Confirm Check failure on the Responder side

If the Confirm value received from OOB is not equal to the calculated Confirm value, the Responder will abort the Pairing process by sending Pairing Failed with reason “Confirm Value Failed”.

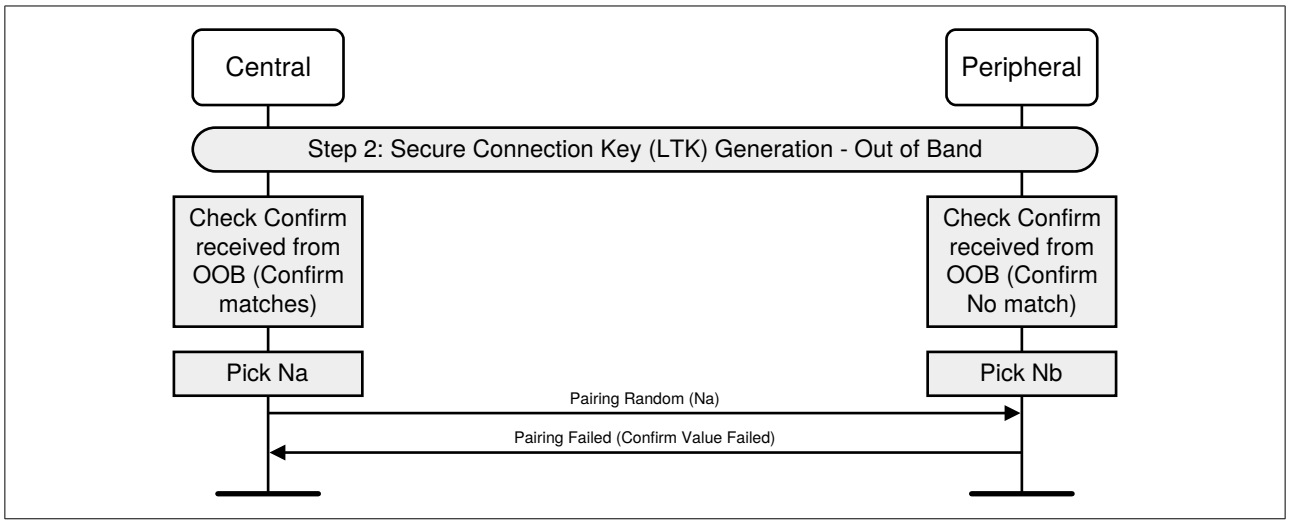



*Figure C.18: Pairing Phase 2, authentication stage 1, Out of Band – Confirm Check failure on Responder side*

#### C.2.2.2.12 Out of Band - Confirm Check failure on the Initiating side

If the Confirm value received from OOB is not equal to the calculated Confirm value, the Initiator will abort the Pairing process by sending Pairing Failed with reason "Confirm Value Failed".

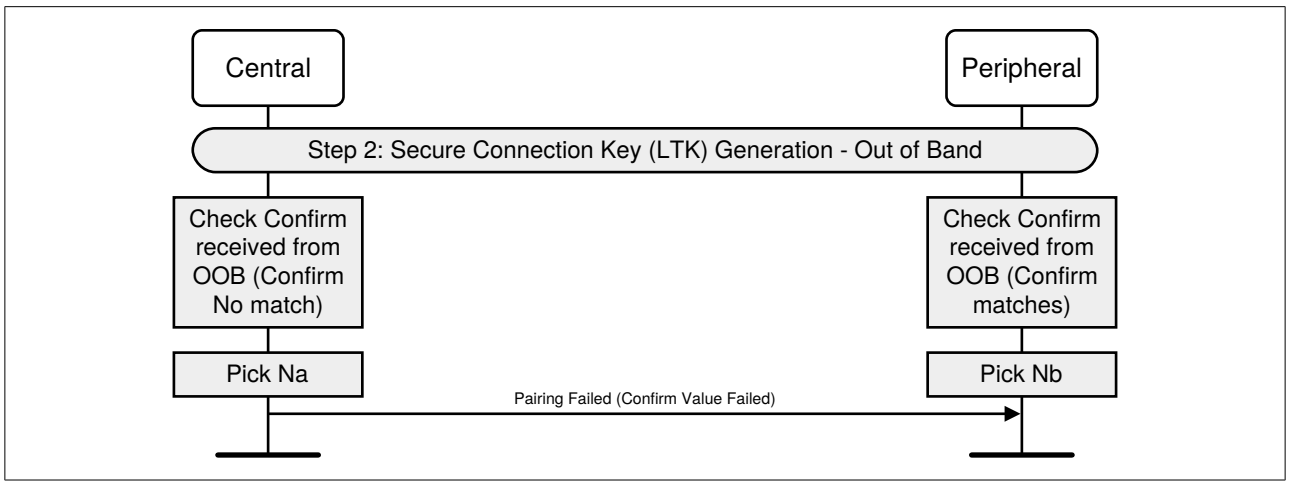



*Figure C.19: Pairing Phase 2: authentication stage 1, Out of Band – Confirm Check failure on Initiator side*

#### C.2.2.2.13 Out of Band Failure on the Initiator side (OOB information not available)

If the initiating side does not have the responder's OOB information, Pairing is terminated.

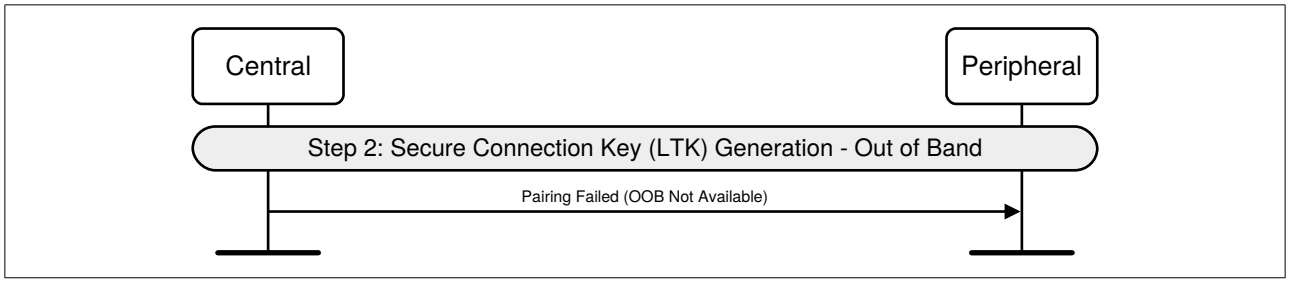



*Figure C.20: Pairing Phase 2, authentication stage 1, Out of Band failure on Initiator side*

#### C.2.2.2.14 Out of Band Failure on the Responding side (OOB information not available)

If the responding side does not have the initiator's OOB information, Pairing is terminated.

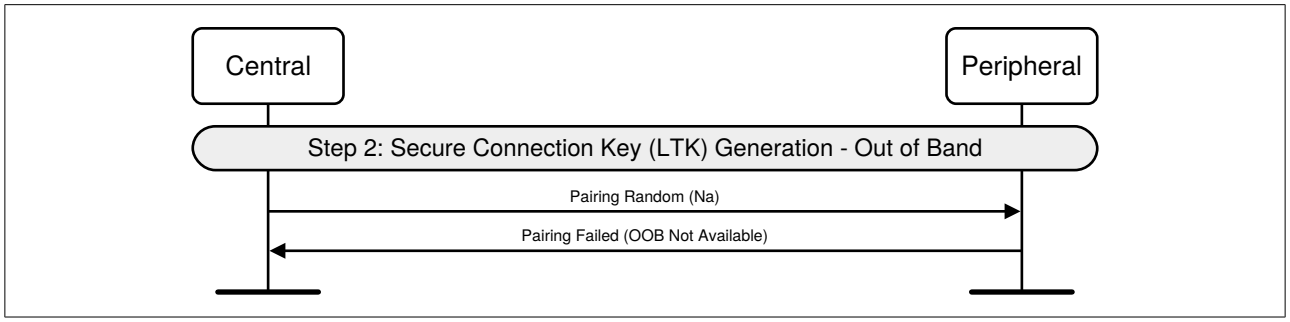



*Figure C.21: Pairing Phase 2, authentication stage 1, Out of Band failure on Responding side*

### C.2.2.3 Long Term Key calculation

Once the DHKey generation is complete, the Long Term Key (LTK) is calculated from the DHKey.

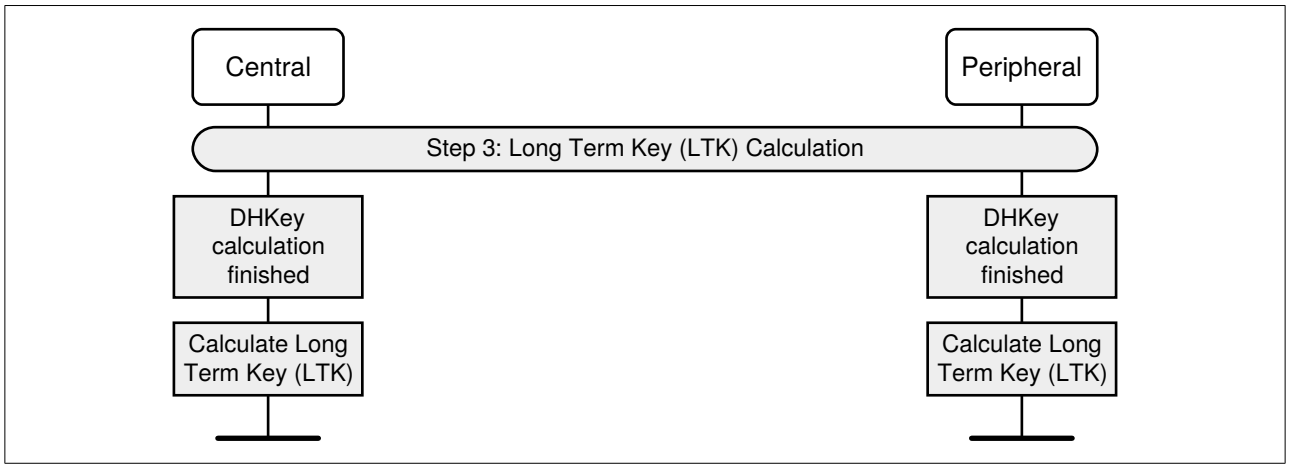



*Figure C.22: Long Term Key calculation*

### C.2.2.4 Authentication stage 2 (DHKey checks)

Once the LTK calculation and authentication stage 1 have completed, the DHKey value generated is checked by exchanging DHKey Check values generated using the DHKey.

If this succeeds, then both devices would have finished displaying information to the user about the process, and therefore the Host can stop displaying this information.

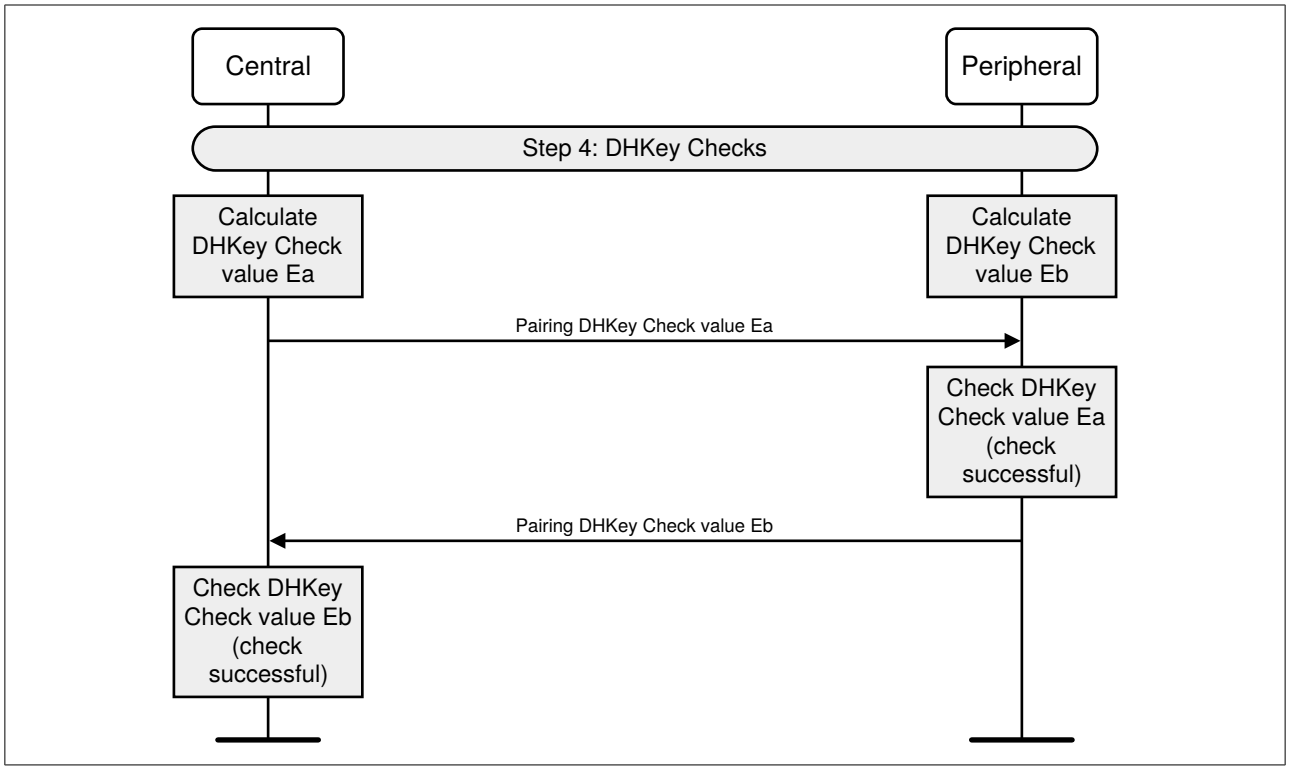



*Figure C.23: Pairing Phase 2, authentication stage 2, DHKey checks*

## C.3 Phase 3: Transport specific key distribution

After short term key generation and the link has been encrypted, transport specific keys are distributed. Figure C.24 shows an example of all keys and values being distributed by Central and Peripheral.

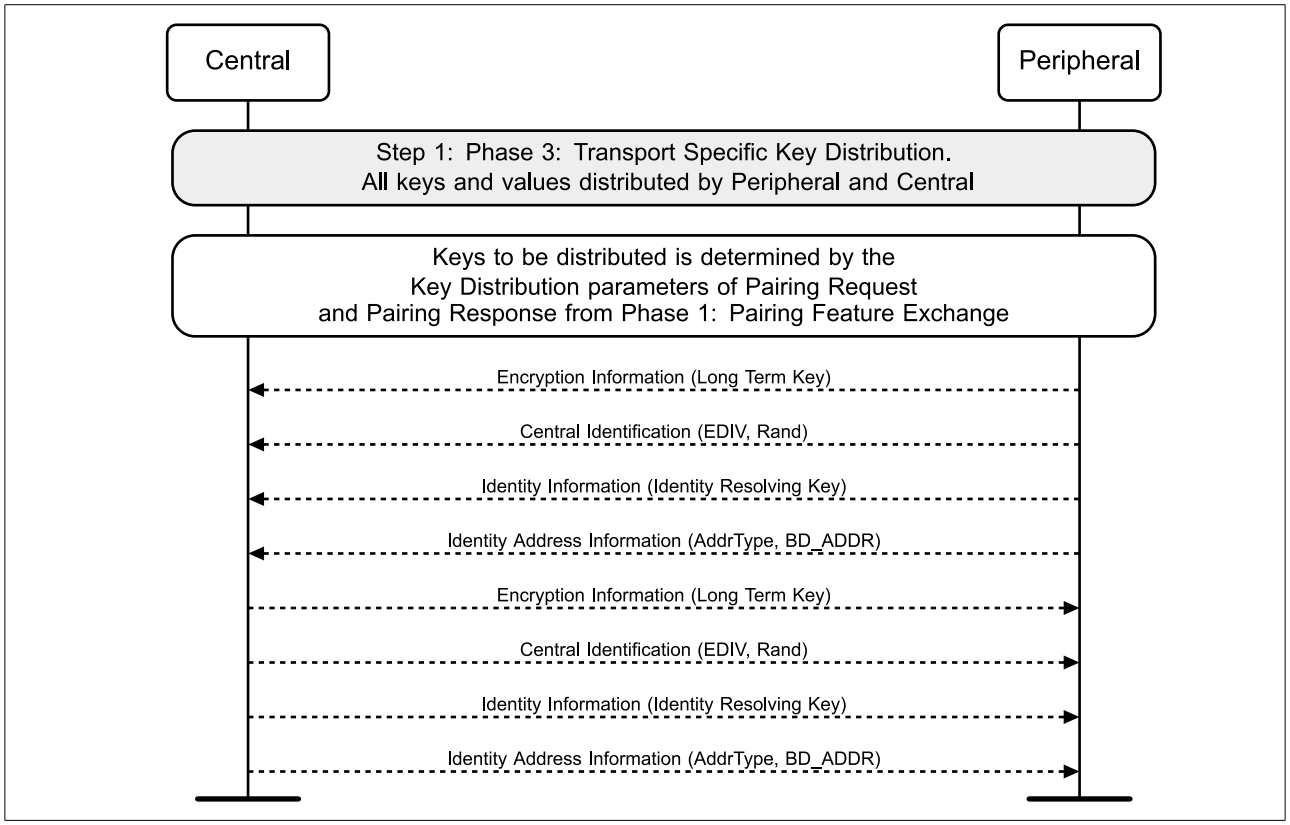



*Figure C.24: Transport specific key distribution*

## C.4 Security re-established using previously distributed LTK

Devices may re-establish security using a previously distributed LTK. The Central always initiates the encryption procedures, and therefore there are two possible sequences: Central initiated and Peripheral requested.

### C.4.1 Central initiated security - Central initiated Link Layer encryption

The Central initiates encryption procedures. There is no SM signaling to enable this; the Central initiates Link Layer encryption only. See [Vol 6] Part D, Section 6.6.

### C.4.2 Peripheral security request - Central initiated Link Layer encryption

The Peripheral may request the Central initiates security procedures. Figure C.25 shows an example where the Peripheral requests security and the Central initiates Link Layer encryption.

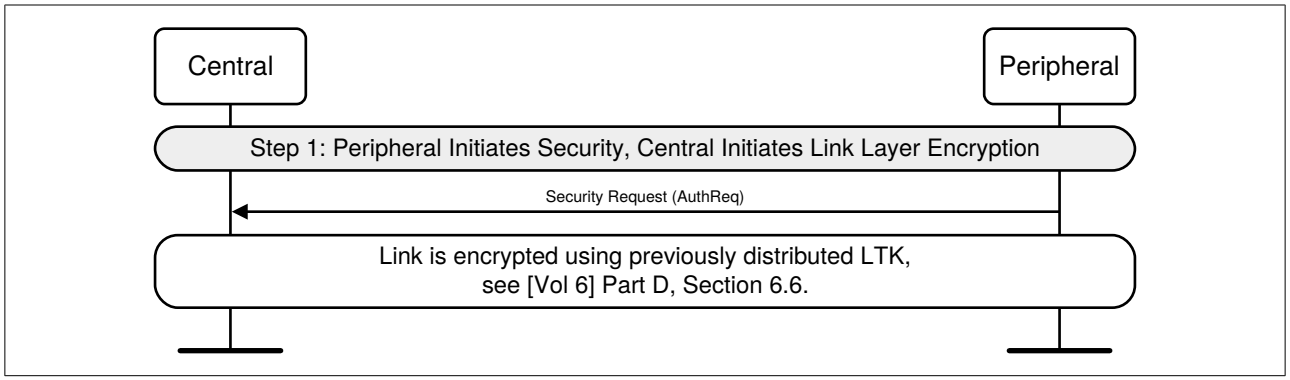



*Figure C.25: Peripheral security request, Central initiates Link Layer encryption*

## C.5 Failure conditions

The following sequences show possible failure conditions and their associated signaling.

### C.5.1 Pairing not supported by Peripheral

If the Peripheral does not support pairing or pairing cannot be performed the Peripheral can reject the request from the Central. Figure C.26 shows the Peripheral rejecting a Pairing Request command from the Central.

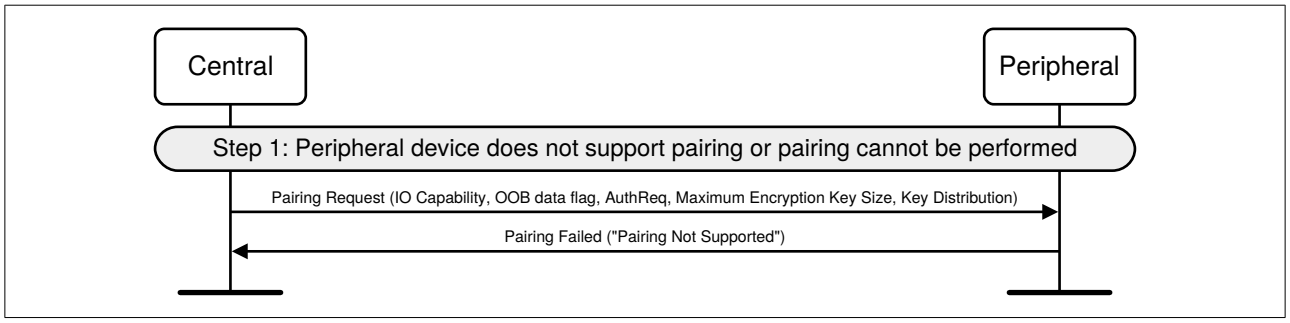



*Figure C.26: Peripheral rejects pairing attempt*

### C.5.2 Central rejects pairing because of key size

During Pairing Feature Exchange the size of the Encryption Key is negotiated. Figure C.27 shows an example where the Central terminates the pairing procedure because the resulting key size is not acceptable.

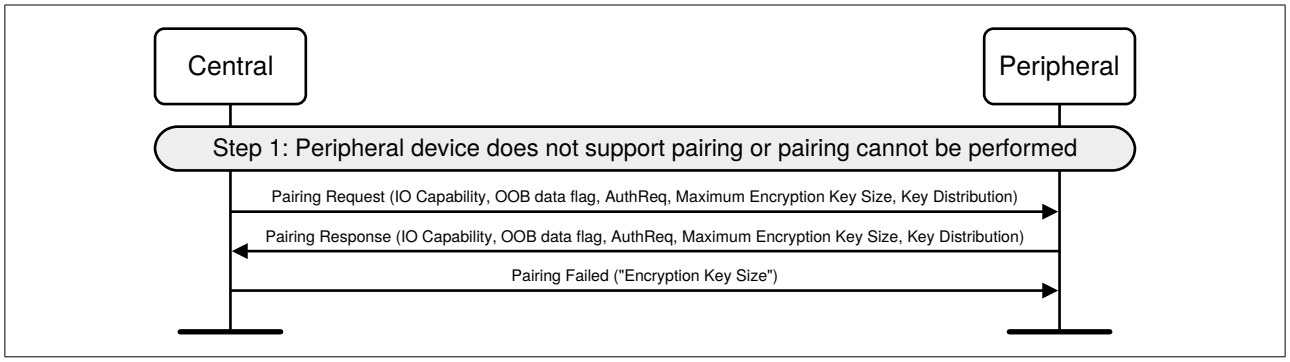



*Figure C.27: Central rejects pairing because of key size*

### C.5.3 Peripheral rejects pairing because of key size

During Pairing Feature Exchange the size of the Encryption Key is negotiated. Figure C.28 shows an example where the Peripheral terminates the pairing procedure because the resulting key size is not acceptable.

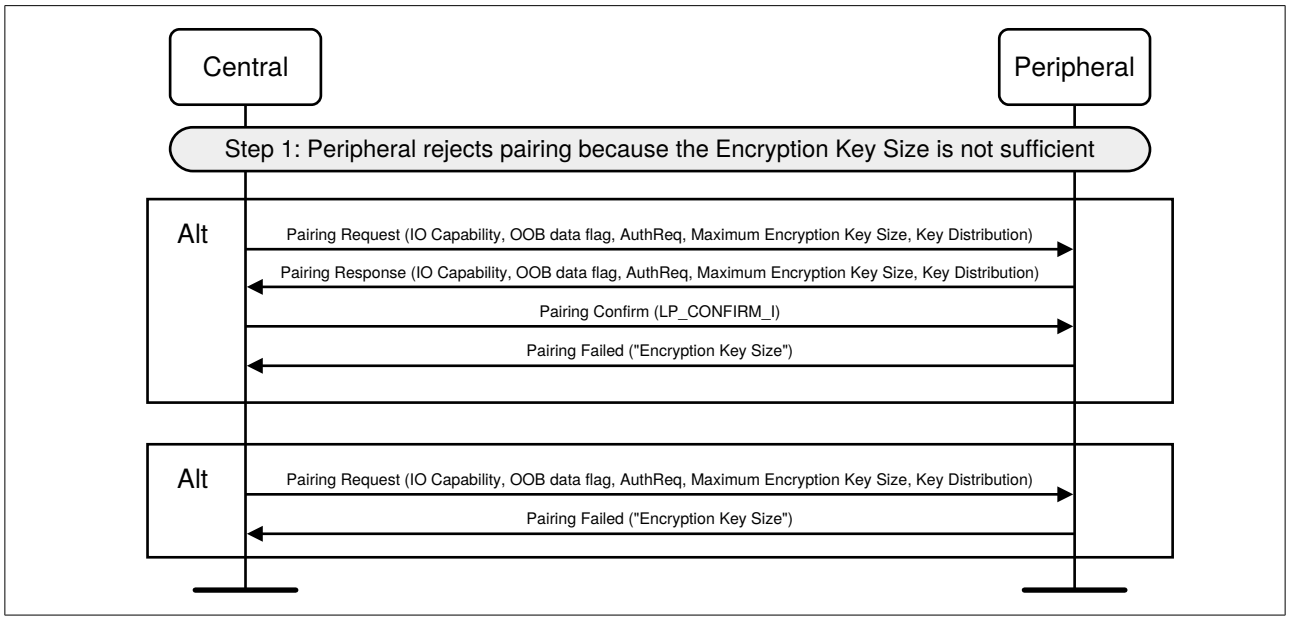



*Figure C.28: Peripheral rejects pairing because of key size*

### C.5.4 Passkey Entry failure on Central

During Passkey Entry pairing the user enters a passkey on both devices. Figure C.29 shows an example where the passkey entry fails on the Central.

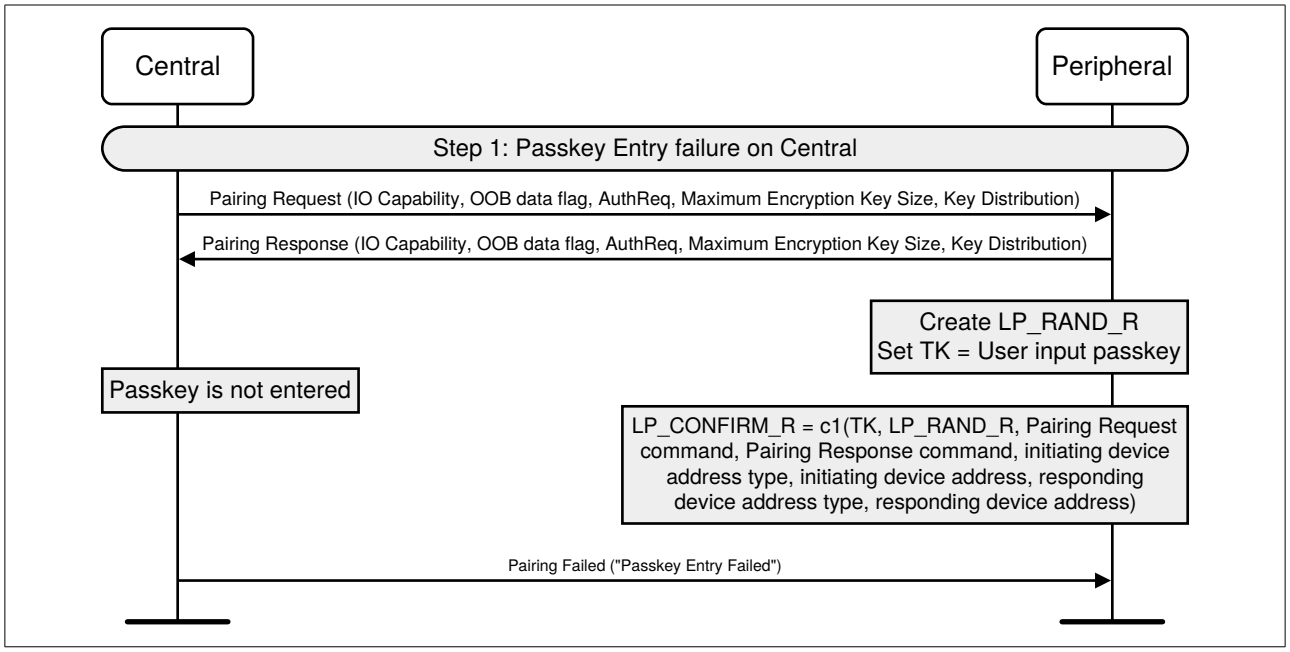



*Figure C.29: Passkey Entry failure on Central*

### C.5.5 Passkey Entry failure on Peripheral

During Passkey Entry pairing the user enters a passkey on both devices. Figure C.30 shows an example where the passkey entry fails on the Peripheral.

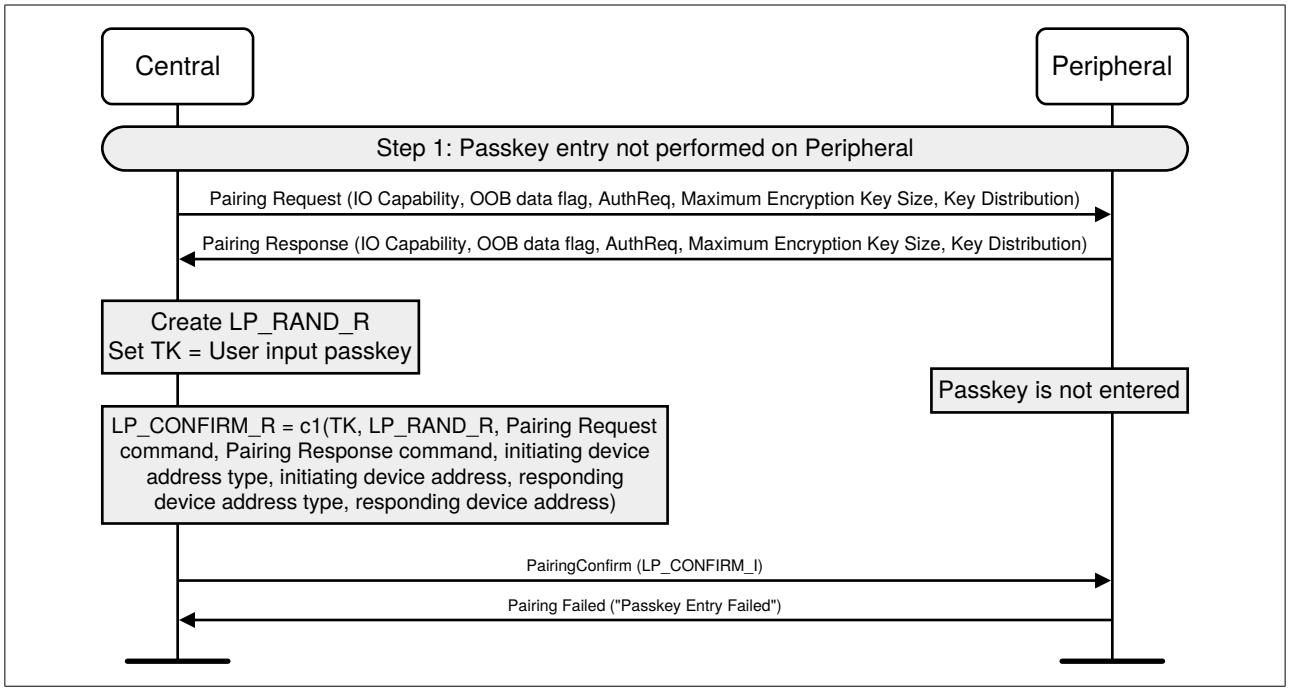



*Figure C.30: Passkey Entry failure on Peripheral*

### C.5.6 Peripheral rejects Central's confirm value

During Passkey Entry pairing the user enters a passkey on both devices. Figure C.31 shows an example where a different passkey is entered on both devices. This sequence could also occur if any of the inputs to c1 (Passkey, LP_RAND_I, LP_RAND_R, Pairing

Request command, Pairing Response command, address types or addresses) are incorrect or altered.

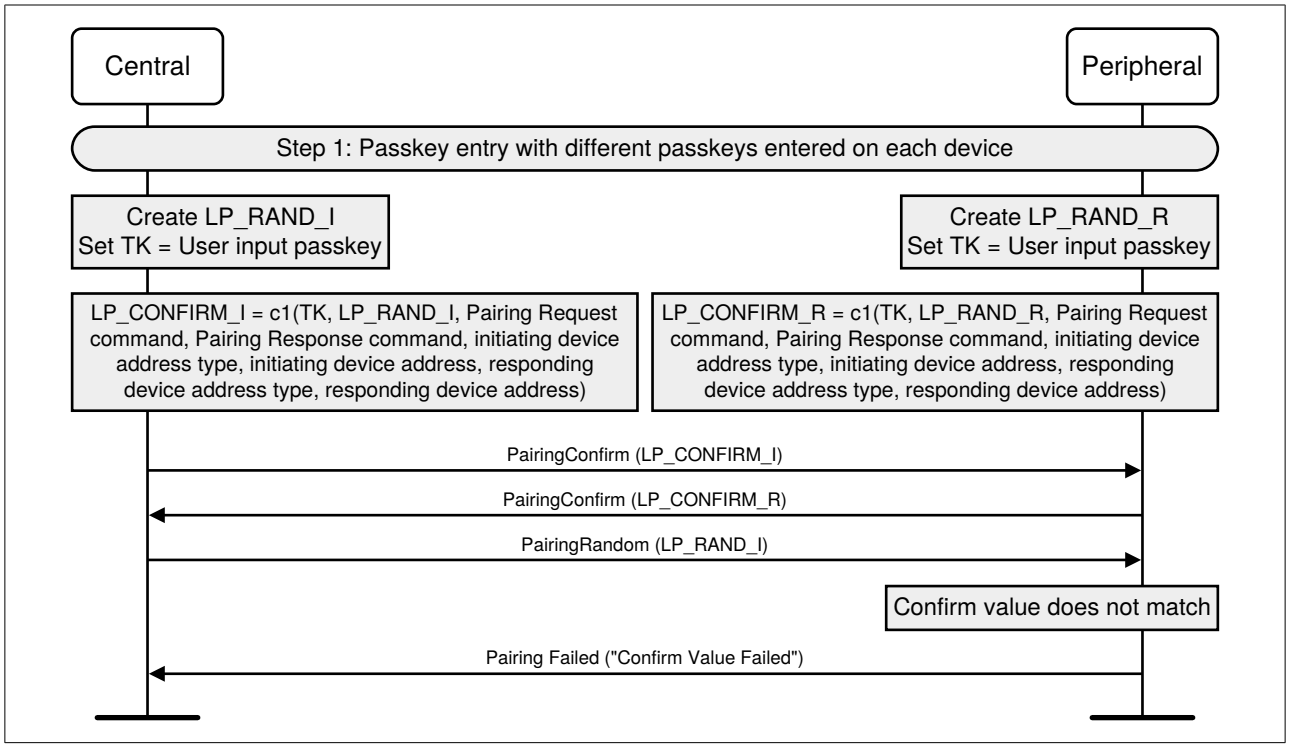



*Figure C.31: Different passkeys entered*

### C.5.7 Central rejects Peripheral's confirm value

During all the pairing methods the Central and Peripheral send random numbers and confirm values. Figure C.32 shows an example where the Central rejects pairing because it cannot verify the confirm value from the Peripheral because any of the

inputs to c1 (Passkey, LP_RAND_I, LP_RAND_R, Pairing Request command, Pairing Response command, address types or addresses) are incorrect or altered.

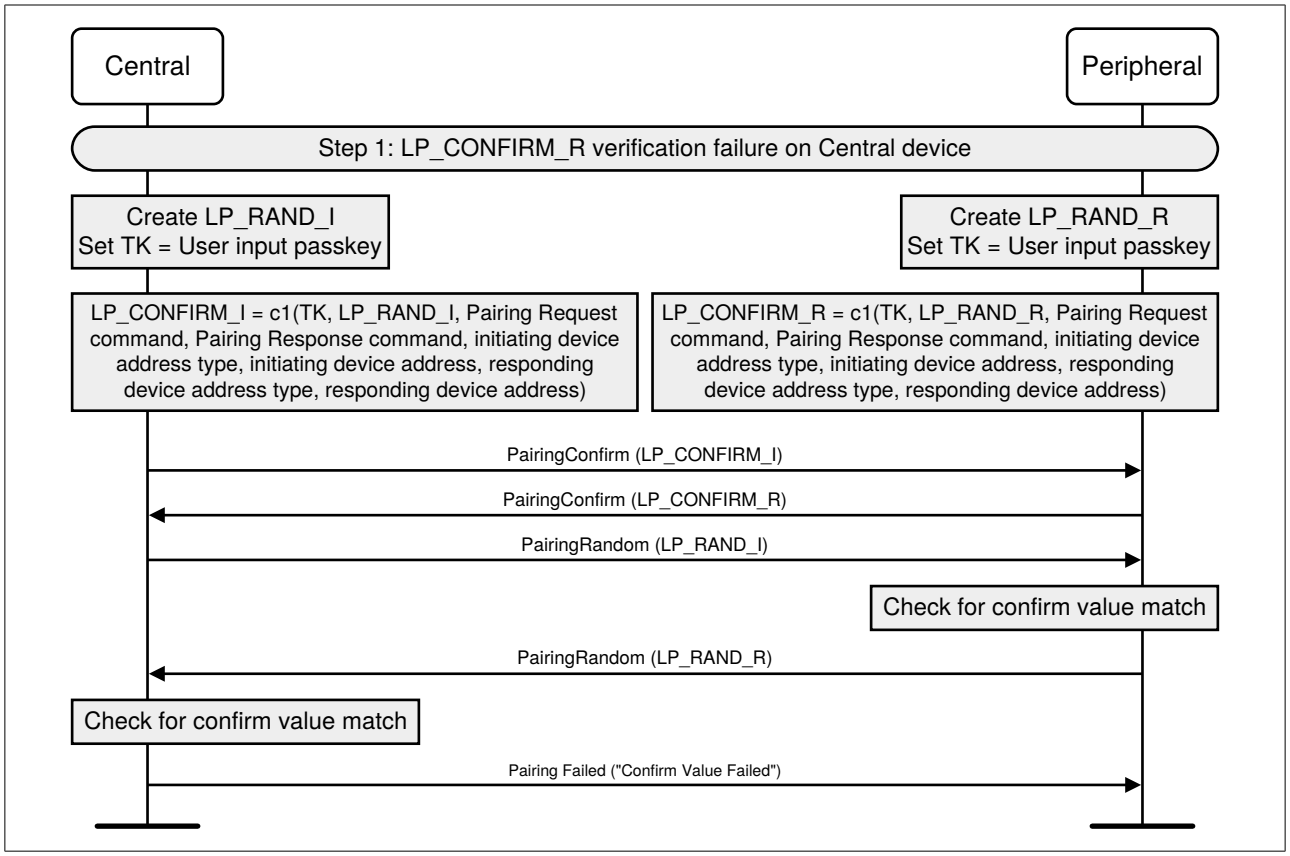



*Figure C.32: Central rejects LP_CONFIRM_R value from Peripheral*

## C.6 Cross-transport key derivation

A pair of BR/EDR/LE devices can generate a Link Key or LTK on one transport and then derive the key for the other transport.

### C.6.1 LE derivation from BR/EDR

Figure C.33 shows an example where the devices generate the Link Key over BR/EDR and then derive an LTK for use on LE.

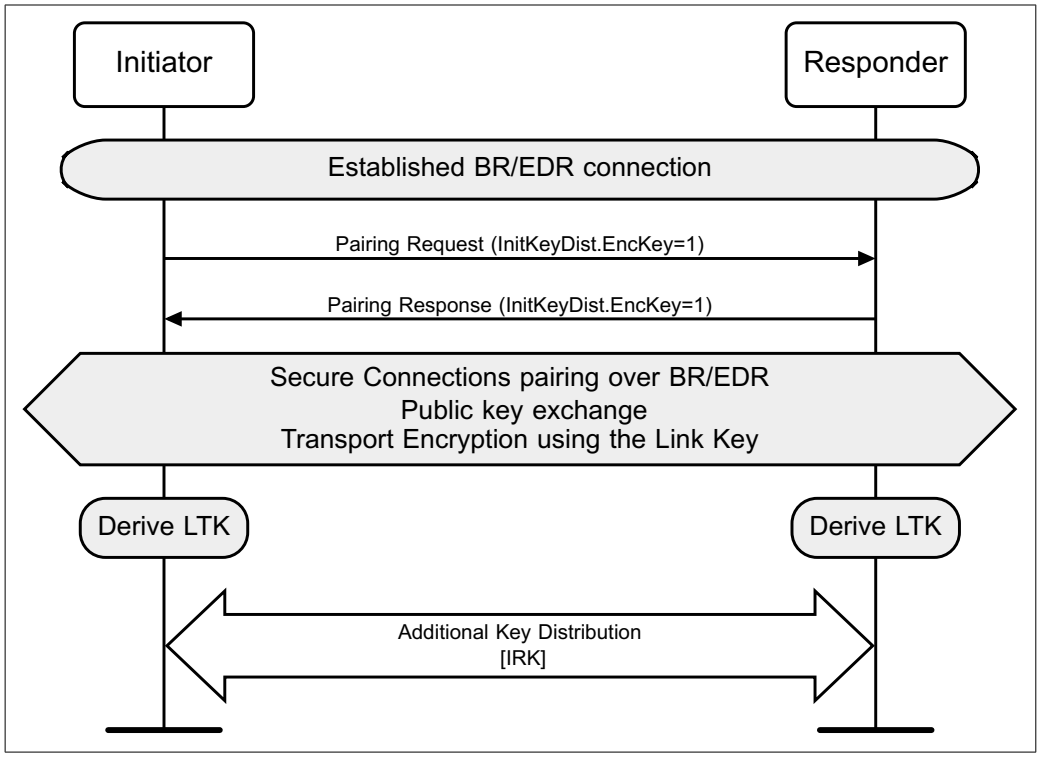



*Figure C.33: LE LTK derived from BR/EDR Link Key*

### C.6.2 BR/EDR derivation from LE

Figure C.34 shows an example where the devices generate the LTK over LE and then derive an Link Key for use on BR/EDR.

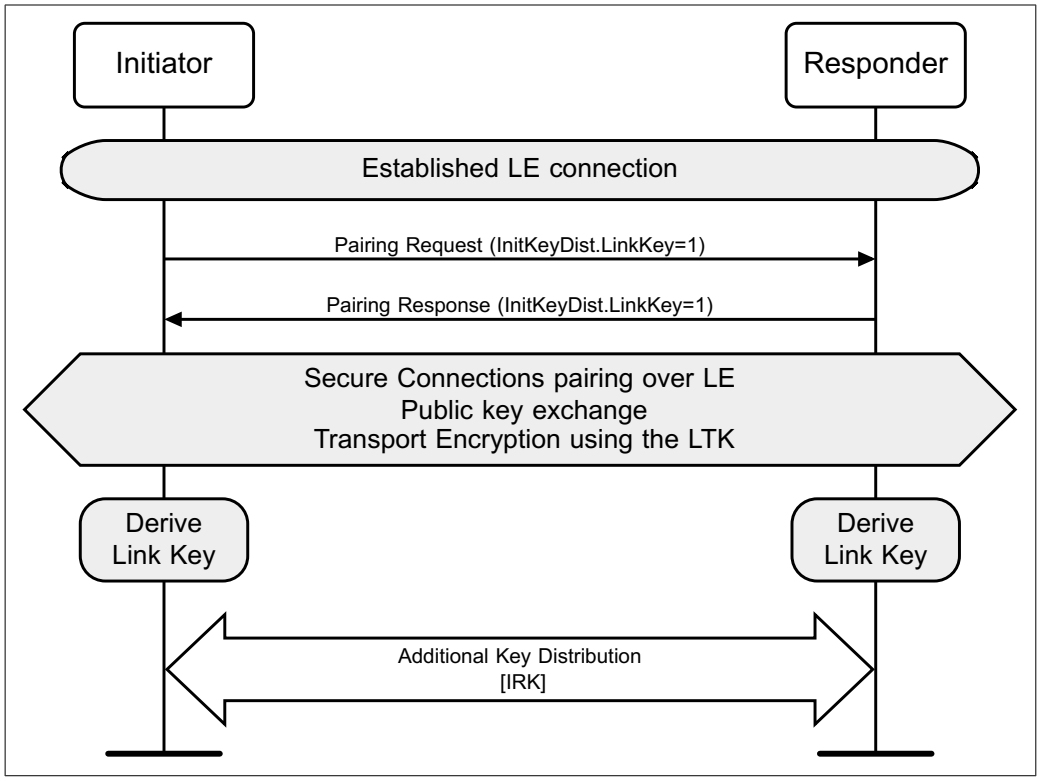



*Figure C.34: BR/EDR Link Key derived from LE LTK*

# Appendix D Sample data

This appendix contains sample data for the Security Manager; see [Vol 1] Part E, Section 4.

In each data set in this section, the bytes are ordered from most significant on the left to least significant on the right. ‘M’ represents the message byte array for which the AES CMAC is calculated.

## D.1 AES-CMAC RFC4493 test vectors

The following test vectors are referenced from RFC4493.

```
K                       2b7e1516 28aed2a6 abf71588 09cf4f3c
Subkey Generation
AES_128(key,0)          7df76b0c 1ab899b3 3e42f047 b91b546f
K1                      fbeed618 35713366 7c85e08f 7236a8de
K2                      f7ddac30 6ae266cc f90bc11e e46d513b
```

### D.1.1 Example 1: Len = 0

```
M               <empty string>
AES_CMAC        bb1d6929 e9593728 7fa37d12 9b756746
```

### D.1.2 Example 2: Len = 16

```
M               6bc1bee2 2e409f96 e93d7e11 7393172a
AES_CMAC        070a16b4 6b4d4144 f79bdd9d d04a287c
```

### D.1.3 Example 3: Len = 40

```
M0              6bc1bee2 2e409f96 e93d7e11 7393172a
M1              ae2d8a57 1e03ac9c 9eb76fac 45af8e51
M2              30c81c46 a35ce411
AES_CMAC        dfa66747 de9ae630 30ca3261 1497c827
```

### D.1.4 Example 4: Len = 64

```
M0              6bc1bee2 2e409f96 e93d7e11 7393172a
M1              ae2d8a57 1e03ac9c 9eb76fac 45af8e51
M2              30c81c46 a35ce411 e5fbc119 1a0a52ef
M3              f69f2445 df4f9b17 ad2b417b e66c3710
AES_CMAC        51f0bebf 7e3b9d92 fc497417 79363cfe
```

## D.2 *f4* LE SC confirm value generation function

```
U               20b003d2 f297be2c 5e2c83a7 e9f9a5b9
```

```
                eff49111 acf4fddb cc030148 0e359de6
V               55188b3d 32f6bb9a 900afcfb eed4e72a
                59cb9ac2 f19d7cfb 6b4fdd49 f47fc5fd
X               d5cb8454 d177733e ffffb2ec 712baeab
Z               0x00
M0              20b003d2 f297be2c 5e2c83a7 e9f9a5b9
M1              eff49111 acf4fddb cc030148 0e359de6
M2              55188b3d 32f6bb9a 900afcfb eed4e72a
M3              59cb9ac2 f19d7cfb 6b4fdd49 f47fc5fd
                00
AES_CMAC        f2c916f1 07a9bd1c f1eda1be a974872d
```

## D.3 *f5* LE SC key generation function

```
DHKey(W)        ec0234a3 57c8ad05 341010a6 0a397d9b
                99796b13 b4f866f1 868d34f3 73bfa698
T               3c128f20 de883288 97624bdb 8dac6989
keyID           62746c65
N1              d5cb8454 d177733e ffffb2ec 712baeab
N2              a6e8e7cc 25a75f6e 216583f7 ff3dc4cf
A1              00561237 37bfce
A2              00a71370 2dcfc1
Length          0100
(LTK)
M0              0162746c 65d5cb84 54d17773 3effffb2
M1              ec712bae aba6e8e7 cc25a75f 6e216583
M2              f7ff3dc4 cf005612 3737bfce 00a71370
M3              2dcfc101 00
AES_CMAC        69867911 69d7cd23 980522b5 94750a38
(MacKey)
M0              0062746c 65d5cb84 54d17773 3effffb2
M1              ec712bae aba6e8e7 cc25a75f 6e216583
M2              f7ff3dc4 cf005612 3737bfce 00a71370
M3              2dcfc101 00
AES_CMAC        2965f176 a1084a02 fd3f6a20 ce636e20
```

## D.4 *f6* LE SC check value generation function

```
N1              d5cb8454 d177733e ffffb2ec 712baeab
N2              a6e8e7cc 25a75f6e 216583f7 ff3dc4cf
MacKey          2965f176 a1084a02 fd3f6a20 ce636e20
R               12a3343b b453bb54 08da42d2 0c2d0fc8
IOcap           010102
A1              00561237 37bfce
A2              00a71370 2dcfc1
M0              d5cb8454 d177733e ffffb2ec 712baeab
M1              a6e8e7cc 25a75f6e 216583f7 ff3dc4cf
M2              12a3343b b453bb54 08da42d2 0c2d0fc8
M3              01010200 56123737 bfce00a7 13702dcf
M4              c1
AES_CMAC        e3c47398 9cd0e8c5 d26c0b09 da958f61
```

## D.5 *g2* LE SC numeric comparison generation function

```
U               20b003d2 f297be2c 5e2c83a7 e9f9a5b9
                eff49111 acf4fddb cc030148 0e359de6
V               55188b3d 32f6bb9a 900afcfb eed4e72a
                59cb9ac2 f19d7cfb 6b4fdd49 f47fc5fd
X               d5cb8454 d177733e ffffb2ec 712baeab
Y               a6e8e7cc 25a75f6e 216583f7 ff3dc4cf
M0              20b003d2 f297be2c 5e2c83a7 e9f9a5b9
M1              eff49111 acf4fddb cc030148 0e359de6
M2              55188b3d 32f6bb9a 900afcfb eed4e72a
M3              59cb9ac2 f19d7cfb 6b4fdd49 f47fc5fd
M4              a6e8e7cc 25a75f6e 216583f7 ff3dc4cf
AES_CMAC        1536d18d e3d20df9 9b7044c1 2f9ed5ba
g2              2f9ed5ba
```

## D.6 *h6* LE SC link key conversion function

```
Key             ec0234a3 57c8ad05 341010a6 0a397d9b
keyID           6c656272
M               6c656272
AES_CMAC        2d9ae102 e76dc91c e8d3a9e2 80b16399
```

## D.7 *ah* random address hash functions

```
IRK             ec0234a3 57c8ad05 341010a6 0a397d9b
prand           00000000 00000000 00000000 00708194
M               00000000 00000000 00000000 00708194
AES_128         159d5fb7 2ebe2311 a48c1bdc c40dfbaa
ah              0dfbaa
```

## D.8 *h7* LE SC link key conversion function

```
Key             ec0234a3 57c8ad05 341010a6 0a397d9b
SALT            00000000 00000000 00000000 746D7031
AES_CMAC        fb173597 c6a3c0ec d2998c2a 75a57011
```

## D.9 LTK to link key conversion using CT2=1

```
LTK             368df9bc e3264b58 bd066c33 334fbf64
Link Key        287ad379 dca40253 0a39f1f4 3047b835
```

## D.10 LTK to link key conversion using CT2=0

```
LTK             368df9bc e3264b58 bd066c33 334fbf64
Link Key        bc1ca4ef 633fc1bd 0d8230af ee388fb0
```

## D.11 Link key to LTK conversion using CT2=1

```
Link Key        05040302 01000908 07060504 03020100
LTK             e85e09eb 5eccb3e2 69418a13 3211bc79
```

## D.12 Link key to LTK conversion using CT2=0

```
Link Key        05040302 01000908 07060504 03020100
LTK             a813fb72 f1a3dfa1 8a2c9a43 f10d0a30
```

# Host Controller Interface

**Specification of the *Bluetooth*® System**

**Volume 4**

**Covered Core Package Version: v6.3**

**Version Date: 2026-05-05**

Bluetooth SIG Proprietary

# UART TRANSPORT LAYER

*This Part describes the UART transport layer (between the Host and the Controller). HCI command, event, and data packets flow through this layer, but the layer does not decode them.*

# CONTENTS

1 General ..... 1738

2 Protocol ..... 1739

3 RS232 settings ..... 1740

4 Error recovery ..... 1741

# 1 GENERAL

The objective of this HCI UART Transport Layer is to make it possible to use the Bluetooth HCI over a serial interface between two UARTs on the same PCB. The HCI UART Transport Layer assumes that the UART communication is free from line errors.

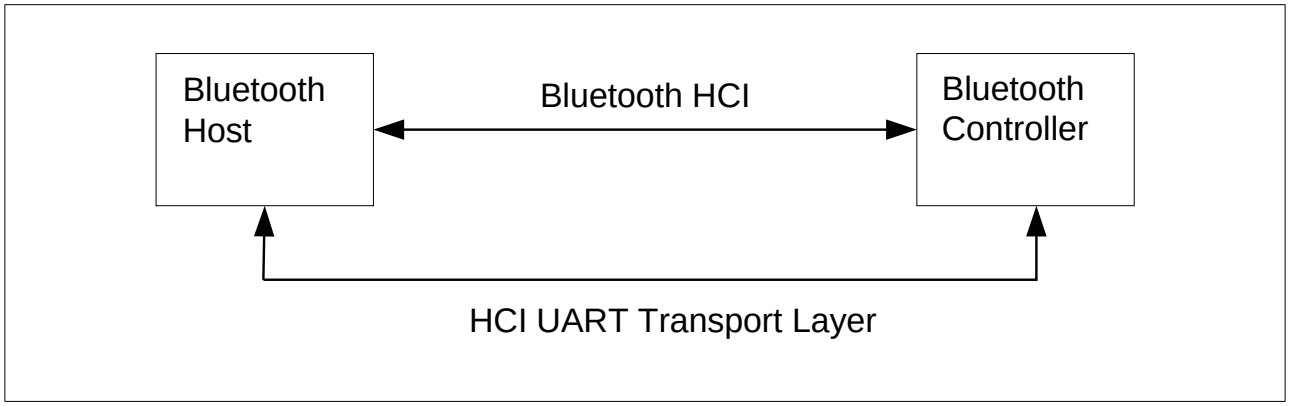



*Figure 1.1: HCI UART Transport Layer*

See [Vol 4] Part D for an alternative UART transport layer for use in the presence of line errors.

# 2 PROTOCOL

There are five kinds of HCI packets that can be sent via the UART Transport Layer; i.e. HCI Command packet, HCI Event packet, HCI ACL Data packet, HCI Synchronous Data packet, and HCI ISO Data packet (see [Vol 4] Part E, Section 5.4). HCI Command packets can only be sent to the Bluetooth Controller, HCI Event packets can only be sent from the Bluetooth Controller, and HCI ACL/Synchronous/ISO Data Packets can be sent both to and from the Bluetooth Controller.

HCI does not provide the ability to differentiate the five HCI packet types. Therefore, if the HCI packets are sent via a common physical interface, an HCI packet indicator has to be added according to Table 2.1 below.

| HCI packet type | HCI packet indicator |
|---|---|
| HCI Command packet | 0x01 |
| HCI ACL Data packet | 0x02 |
| HCI Synchronous Data packet | 0x03 |
| HCI Event packet | 0x04 |
| HCI ISO Data packet | 0x05 |

*Table 2.1: HCI packet indicators*

The HCI packet indicator shall be sent immediately before the HCI packet. All five kinds of HCI packets have a length field, which is used to determine how many bytes are expected for the HCI packet. When an entire HCI packet has been received, the next HCI packet indicator is expected for the next HCI packet. Over the UART Transport Layer, only HCI packet indicators followed by HCI packets are allowed.

# 3 RS232 SETTINGS

The HCI UART Transport Layer uses the following settings for RS232:

| **Baud rate:** | **manufacturer-specific** |
|---|---|
| Number of data bits: | 8 |
| Parity bit: | no parity |
| Stop bit: | 1 stop bit |
| Flow control: | RTS/CTS |
| Flow-off response time: | manufacturer specific |

*Table 3.1: RS232 settings*

Flow control with RTS/CTS is used to prevent temporary UART buffer overrun. It should not be used for flow control of HCI, since HCI has its own flow control mechanisms for HCI commands, HCI events and HCI data.

If CTS is 1, then the Host/Controller is allowed to send.

If CTS is 0, then the Host/Controller is not allowed to send.

The flow-off response time defines the maximum time from setting RTS to 0 until the byte flow actually stops.

The RS232 signals should be connected in a null-modem fashion; i.e. the local TXD should be connected to the remote RXD and the local RTS should be connected to the remote CTS and vice versa.

# 4 ERROR RECOVERY

If the Host or the Controller lose synchronization in the communication over RS232, then a reset is needed. A loss of synchronization means that an incorrect HCI packet indicator has been detected, or that the length field in an HCI packet is out of range.

If the UART synchronization is lost in the communication from Host to Controller, then the Controller shall send an HCI_Hardware_Error event to tell the Host about the synchronization error. The Controller will then expect to receive an HCI_Reset command from the Host in order to perform a reset. The Controller will also use the HCI_Reset command in the byte stream from Host to Controller to re-synchronize.

If the UART synchronization is lost in the communication from Controller to Host, then the Host shall send the HCI_Reset command in order to reset the Controller. The Host shall then re-synchronize by looking for the HCI_Command_Complete event for the HCI_Reset command in the byte stream from Controller to Host.

# USB TRANSPORT LAYER

*This Part describes the USB transport layer (between a Host and the Controller). HCI commands flow through this layer, but the layer does not decode the commands.*

# CONTENTS

**1 Overview ..... 1744**
1.1 Requirements ..... 1745

**2 USB endpoint expectations ..... 1747**
2.1 Descriptor overview ..... 1747
2.1.1 Controller descriptors ..... 1747
2.1.2 [This section is no longer used] ..... 1753
2.2 Control endpoint expectations ..... 1753
2.2.1 Single function Controller ..... 1753
2.2.2 Controller function in a composite device ..... 1753
2.2.3 [This section is no longer used] ..... 1754
2.3 Bulk endpoints expectations ..... 1754
2.4 Interrupt endpoint expectations ..... 1754
2.5 Isochronous endpoints expectations ..... 1754

**3 Class code ..... 1756**
3.1 Bluetooth codes ..... 1756
3.1.1 [This section is no longer used] ..... 1756
3.1.2 [This section is no longer used] ..... 1756
3.1.3 [This section is no longer used] ..... 1756
3.1.4 [This section is no longer used] ..... 1756

**4 Device firmware upgrade ..... 1757**

**5 Limitations ..... 1758**
5.1 Power specific limitations ..... 1758
5.2 Other limitations ..... 1758

**6 Bluetooth Composite Device implementation ..... 1759**
6.1 Configurations ..... 1759
6.2 Using USB Interface Association Descriptors for a Controller function ..... 1759
6.3 [This section is no longer used] ..... 1760

**7 USB Bulk Serialization mode ..... 1761**

**8 References ..... 1762**

# 1 OVERVIEW

This document discusses the requirements of the Universal Serial Bus (USB) interface for Bluetooth hardware. Readers should be familiar with USB, USB design issues, Advanced Configuration Power Interface (ACPI), the overall Bluetooth architecture, and the basics of the radio interface.

The reader should also be familiar with the Bluetooth Host Controller interface.

Referring to Figure 1.1, notice that this document discusses the implementation details of the two-way arrow labeled “USB Function.”

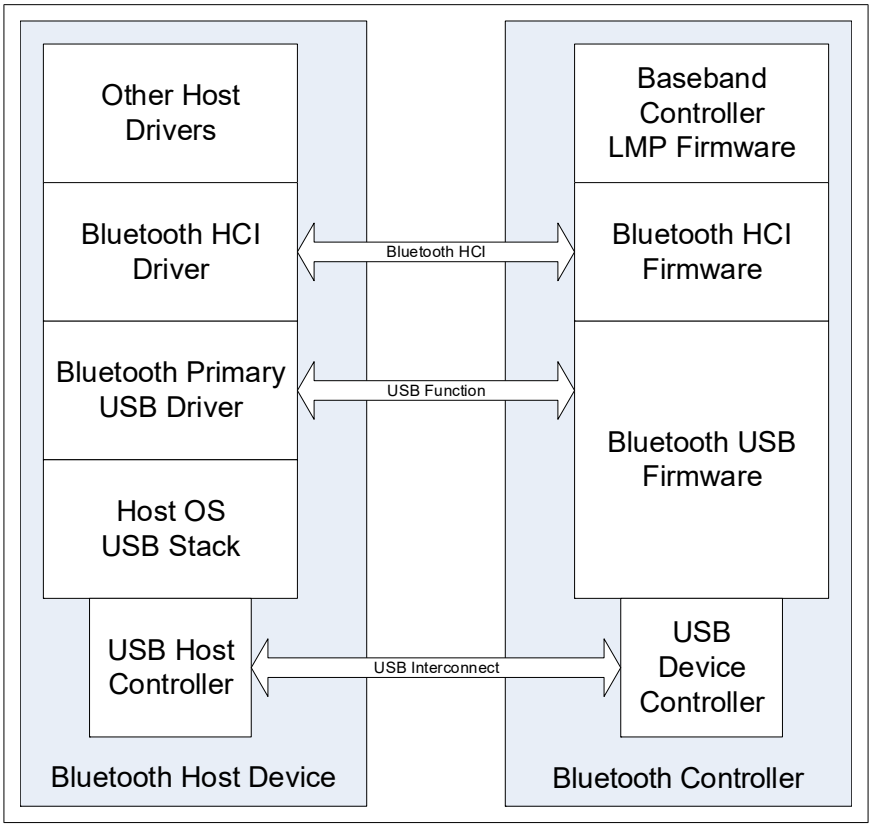



*Figure 1.1: Relationship between the Host and Bluetooth Controller*

The USB hardware can be embodied in one of several ways:

1. As a USB dongle (e.g. cabled USB)
2. As a USB module integrated into the product and connected internally via a cable or connector

3. Integrated onto the motherboard of a notebook PC or other device and connected via circuit board traces with standard USB, Inter-Chip USB or High Speed Inter-Chip USB
4. Integrated as a subsystem on a single-chip System-on-Chip (SoC) design connected on-chip as part of a compound device.

Finally, for an overview of the connection that is established between two Bluetooth devices, reference Figure 1.2.

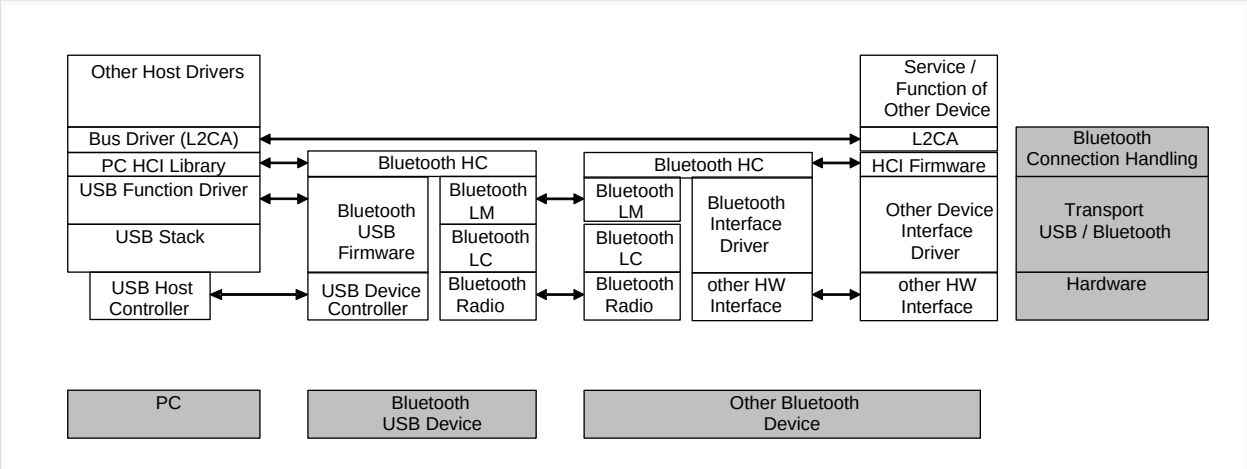



*Figure 1.2: Flow of data from one Bluetooth device to another*

The Bluetooth Controller operates in one of two modes: legacy mode and Bulk Serialization mode. In legacy mode, HCI commands are transmitted over the control endpoint, HCI events are received over the interrupt endpoint, ACL data is exchanged over the bulk endpoints, and SCO data is exchanged over the isochronous endpoints. A mechanism to exchange data for LE Isochronous Channels is not defined in legacy mode. In Bulk Serialization mode, all HCI packets, including HCI ISO Data packets, are exchanged over the bulk endpoints.

## 1.1 Requirements

Implementations of the USB transport layer shall support legacy mode (see Section 2.1) and may support Bulk Serialization mode (see Section 7).

If the Controller is in Bulk Serialization mode, it shall reject any attempt by the Host to change the alternate setting of the second interface by responding to the USB device request with a STALL [1]. Use of the second interface (i.e., the one that contains the isochronous endpoints) is disallowed because SCO data is exchanged over the bulk endpoints while the Controller is in Bulk Serialization mode.

If the Controller is in legacy mode and a non-zero alternate setting has been selected on the second interface, it shall reject any attempt by the Host to change the alternate setting of the first interface by responding to the USB device request with a STALL.

If the Controller is in Bulk Serialization mode, it shall reject any HCI commands received over the control endpoint by responding to the USB device request with a STALL.

If the Controller is instructed by the Host to switch modes while it still has packets pending retrieval by the Host, it shall retain those packets and make them available on the new endpoint(s) after the mode switch is complete.

# 2 USB ENDPOINT EXPECTATIONS

This section outlines specific USB endpoints that are required in order to function properly with the Host. This section assumes a basic familiarity with USB. The endpoint numbers (labeled 'Suggested Endpoint Address' below) may be dynamically recognized upon driver initialization – this depends on the implementation.

## 2.1 Descriptor overview

The Universal Serial Bus is intended for high data rates. USB defines several physical layers, ranging from 1.5 Mb/s to several Gb/s of bus bandwidth. A Bluetooth USB device should provide a USB transport with sufficient bus bandwidth to support the Bluetooth radio transports included in the device.

### 2.1.1 Controller descriptors

The Controller configuration consists of two interfaces. The first interface has a default setting (i.e., alternate setting zero) that contains the bulk and interrupt endpoints. If Bulk Serialization Mode is supported, then it shall have a second alternate setting, alternate setting 1, that contains only the bulk endpoints. The second interface provides scalable isochronous bandwidth. The recommended configuration for the second interface has four alternate settings that provide different bandwidth. The default interface is empty so that the device is capable of scaling down to zero isochronous bandwidth.

An HCI packet consisting of an HCI header and HCI data shall be contained in one USB Transfer. A USB transfer is defined by the USB specification as one or more USB transactions that contain the data from one IO request. For example, an ACL data packet containing 256 bytes (both HCI header and HCI data) would be sent over the bulk endpoint in one IO request. That IO request will require four 64-byte full speed USB Transactions or a single 256-byte High-speed USB Transaction, and forms a Transfer. If the Maximum Packet Size for the endpoint on which the transfer is sent is 64 bytes, then that IO request will require four 64-byte USB transactions.

The endpoints are spread across two interfaces so that when adjusting isochronous bandwidth consumption (via select interface calls), any pending bulk and/or interrupt transactions do not have to be terminated or resubmitted.

Table 2.1 and the following example calculations illustrate recommended endpoint descriptor parameter values and how they are derived. The maximum packet sizes for control endpoints, interrupt endpoints and bulk endpoints may be any value allowed by the relevant USB core specifications. The maximum packet size for isochronous endpoints must be large enough to accommodate the maximum average traffic; they may be set to accommodate the largest HCI transfer, subject to the capabilities of the

Controller. In Table 2.1, the service interval is assumed to be 1 millisecond, for USB Full Speed (FS) frames.

Examples:

1. For a single 8 kHz audio channel with of 64 kb/s CVSD audio the Host may break HCI data into one USB transfers for each USB frame (e.g. 1 ms); in that case, the max packet size must be at least 11 = 3 octet HCI header + 8 octets of data. To reduce HCI header overhead, a common strategy (see Figure 2.1) is to consolidate 3 ms of data into a 27 octet HCI packet of 24 octets of data + 3 octets of HCI header. These HCI packets can be exchanged as a single USB transfer on 3 ms intervals; this requires a max packet size of 27÷3 = 9 octets per 1 millisecond USB Full Speed Frame.
2. For two 8 kHz audio channels of 64 kb/s CVSD audio the Host may double the payload size of each HCI packet, which would be 3 octets HCI header + 48 octets of data = 51 octets. Posting these at 3 ms intervals requires 51÷3 = 17 octets of maximum packet size.
3. For one 16 kHz audio channel the HCI packets need to be large enough to accommodate single octet (128 kb/s) or 2-octet (256 kb/s) encoding. On 3 ms intervals, these would have to be (48+3)÷3 = 17 octets or (96+3)÷3 = 33 octets respectively.
4. For one mSBC[1] compressed wideband audio channel the HCI packets will be 3 octets of HCI header + 60 octets of data. If the Controller can support a maximum packet size of 63 (or 64) octets, an entire mSBC frame may be exchanged in one USB transaction. If the maximum packet size is smaller than 63 octets, additional latency will be introduced. The USB Host Controller will reserve bandwidth that will only be used when the Bluetooth Host or Controller has data to transfer.
5. For combinations of audio channels, if the max packet size can accommodate the largest HCI packets, there is also sufficient bandwidth for the audio channels that have smaller HCI packets. See example 4 above.

Table 2.1 outlines a recommended configuration for a USB Full Speed device.

| Interface Number | Alternate Setting | Suggested Endpoint Address | Endpoint Type | Suggested Max Packet Size | USB Polling Interval/HCI Packet Interval |
|---|---|---|---|---|---|
| **HCI commands** | | | | | |
| *none* | *none* | 0x00 | Control | 8/16/32/64 | *none* |

[1]For information about modified Sub Band Codec (mSBC), see Hands-Free Profile v1.6 or later

| Interface Number | Alternate Setting | Suggested Endpoint Address | Endpoint Type | Suggested Max Packet Size | USB Polling Interval/HCI Packet Interval |
|---|---|---|---|---|---|
| **HCI events** | | | | | |
| 0 | 0 | 0x81 | Interrupt (IN) | 16 | variable |
| **ACL Data** | | | | | |
| 0 | 0 | 0x82 | Bulk (IN) | 32/64 | variable |
| 0 | 0 | 0x02 | Bulk (OUT) | 32/64 | variable |
| **Bulk Serialization mode** | | | | | |
| 0 | 1 | 0x82 | Bulk (IN) | 32/64 | variable |
| 0 | 1 | 0x02 | Bulk (OUT) | 32/64 | variable |
| **No active voice channels (for USB compliance)** | | | | | |
| 1 | 0 | 0x83 | Isoch (IN) | 0 | *none* |
| 1 | 0 | 0x03 | Isoch (OUT) | 0 | *none* |
| **One 8 kHz voice channel with 8-bit encoding** | | | | | |
| 1 | 1 | 0x83 | Isoch (IN) | 9 | 1 ms/3 ms |
| 1 | 1 | 0x03 | Isoch (OUT) | 9 | 1 ms/3 ms |
| **Two 8 kHz voice channels with 8-bit encoding or one 8 kHz voice channel with 16-bit encoding** | | | | | |
| 1 | 2 | 0x83 | Isoch (IN) | 17 | 1 ms/3 ms |
| 1 | 2 | 0x03 | Isoch (OUT) | 17 | 1 ms/3 ms |
| **Three 8 kHz voice channels with 8-bit encoding** | | | | | |
| 1 | 3 | 0x83 | Isoch (IN) | 25 | 1 ms/3 ms |
| 1 | 3 | 0x03 | Isoch (OUT) | 25 | 1 ms/3 ms |
| **Two 8 kHz voice channels with 16-bit encoding or one 16 kHz voice channel with 16-bit encoding** | | | | | |
| 1 | 4 | 0x83 | Isoch (IN) | 33 | 1 ms/3 ms |
| 1 | 4 | 0x03 | Isoch (OUT) | 33 | 1 ms/3 ms |
| **Three 8 kHz voice channels with 16-bit encoding or one 8 kHz voice channel with 16-bit encoding and one 16 kHz voice channel with 16-bit encoding** | | | | | |
| 1 | 5 | 0x83 | Isoch (IN) | 49 | 1 ms/3 ms |
| 1 | 5 | 0x03 | Isoch (OUT) | 49 | 1 ms/3 ms |

| Interface Number | Alternate Setting | Suggested Endpoint Address | Endpoint Type | Suggested Max Packet Size | USB Polling Interval/HCI Packet Interval |
|---|---|---|---|---|---|
| **One mSBC voice channel** | | | | | |
| 1 | 6 | 0x83 | Isoch (IN) | 63 | 1 ms/7.5 ms |
| 1 | 6 | 0x03 | Isoch (OUT) | 63 | 1 ms/7.5 ms |

*Table 2.1: USB firmware interface and endpoint settings*

The following two examples are used to demonstrate the flow of data given the described endpoints. Figure 2.1 shows one voice channel and Figure 2.2 shows two voice channels. In both examples, the duration of the voice data is 3 ms per IO request and the encoding is 8 bits. Convergence is expected because the radio is sending out an average of eight bytes of voice data every millisecond and USB is sending eight bytes of voice data every millisecond.

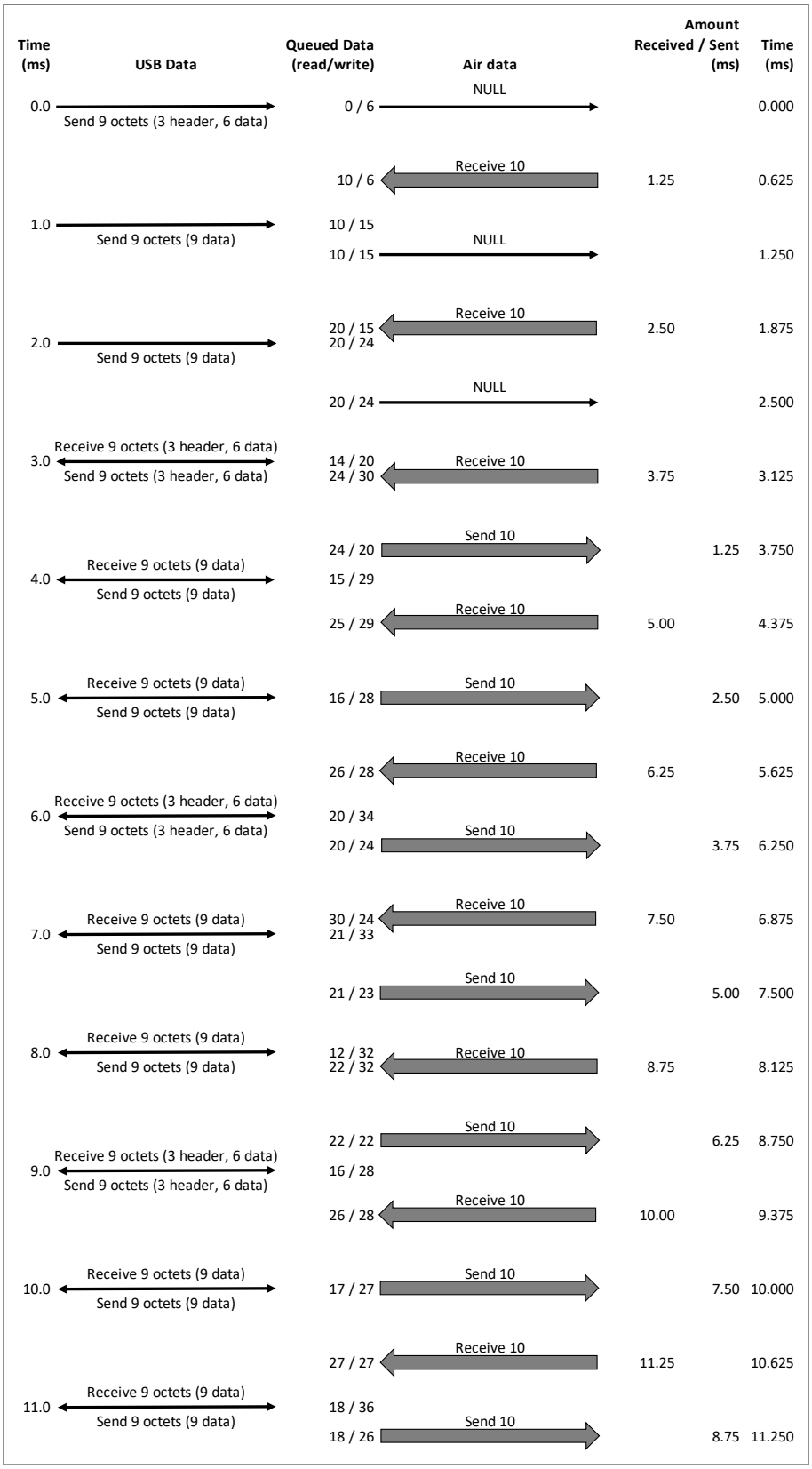



*Figure 2.1: Example USB single-channel voice traffic data flow*

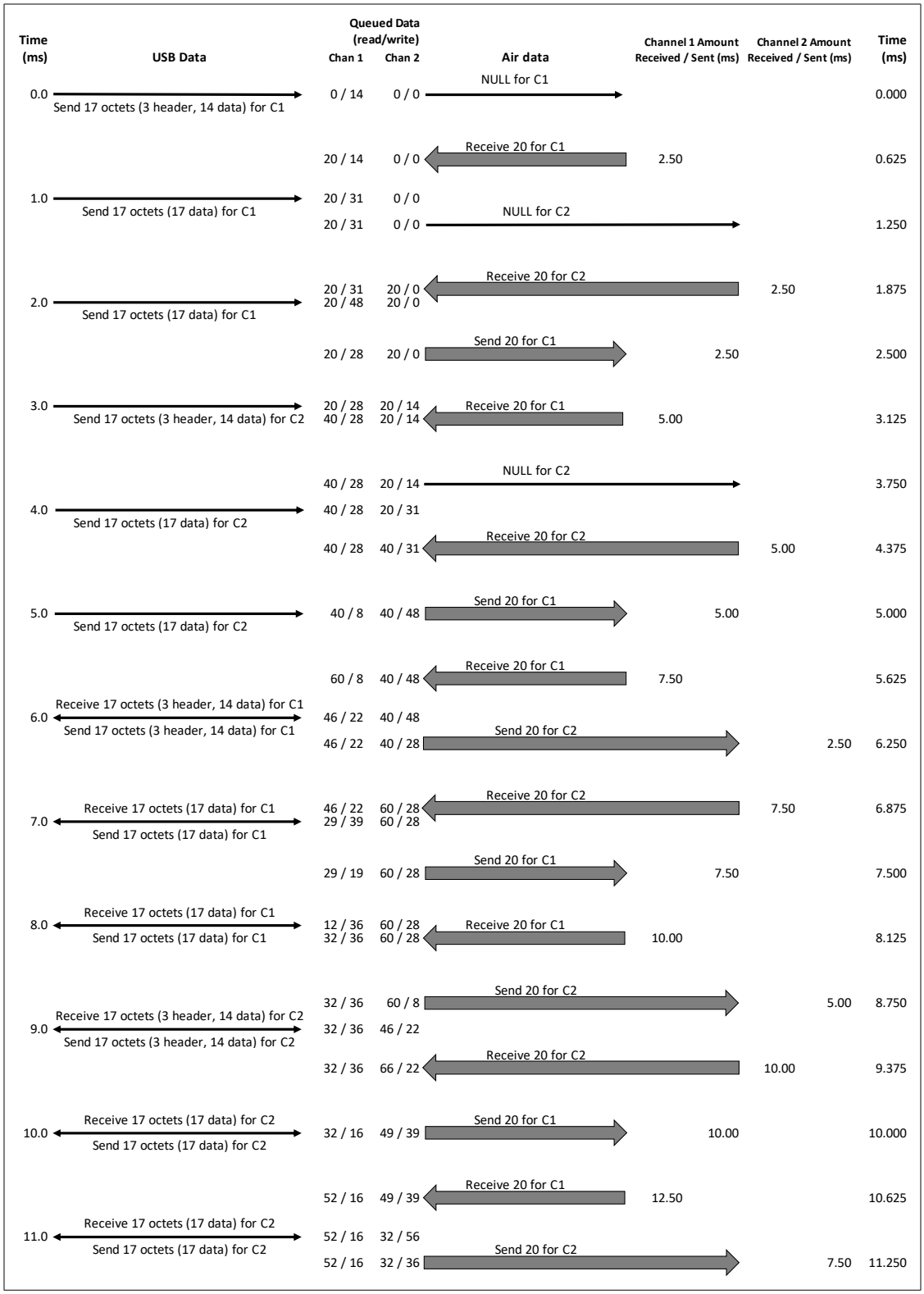



*Figure 2.2: Example USB dual-channel voice traffic data flow*

### 2.1.2 [This section is no longer used]

## 2.2 Control endpoint expectations

Endpoint 0 is used to configure and control the USB device. Endpoint 0 will also be used to allow the Host to send HCI-specific commands to the Controller. HCI command packets should be sent with the following parameters:

bmRequestType = 0x20 (Host-to-device class request, device as target)
bRequest = 0x00
wValue = 0x00
wIndex = 0x00

Some Host devices on the market set bRequest to 0xE0. Hence, for historical reasons, if the Bluetooth Controller firmware receives a class request over this endpoint, it should treat the packet as an HCI command packet regardless of the value of bRequest, wValue and wIndex.

All HCI Control packets delivered to Endpoint 0 are addressed in the Setup Data structure (See 9.3 of [1]). This structure contains fields which determine the destination within the device. The bmRequestType can be used to select the Device or the Interface. If Interface is selected, the wIndex parameter shall select the Index for the targeted Bluetooth Controller.

### 2.2.1 Single function Controller

For a single function Controller, the Host should address HCI command packets to the Device. HCI command packets should be sent with the following parameters:

bmRequestType = 0x20 (Host-to-device class request, device as target)
bRequest = 0x00
wValue = 0x00
wIndex = 0x00

If the Controller firmware receives a packet over this endpoint, it should treat the packet as an HCI command packet regardless of the value of bRequest, wValue, and wIndex. Some Host devices set bRequest to 0xE0.

### 2.2.2 Controller function in a composite device

For a Controller included in a composite (multi-function) device, the Host should address HCI control packets to the Interface of the Controller. HCI command packets should be sent with the following parameters:

bmRequestType = 0x21 (Host-to-Interface class request, interface as target)
bRequest = 0x00

wValue = 0x00
wIndex = the actual Interface number within the composite device

If the Host system driver addresses USB requests containing HCI command packets to the Device (see Section 2.2.1) instead of to the Interface, the device implementation shall recognize these HCI command packets and correctly route them to the Controller function. This allows correct operation of the Controller function and avoids malfunctions in other functions contained in the composite device.

### 2.2.3 [This section is no longer used]

## 2.3 Bulk endpoints expectations

Data integrity is a critical aspect for ACL data. This, in combination with bandwidth requirements, is the reason for using a bulk endpoint. Multiple 64-byte packets can be shipped per USB Frame (1 millisecond, full speed) or 512-byte packets per USB Microframe (125 microseconds, high-speed), across the bus.

Suggested bulk max packet size is 64 bytes for full-speed, or 512 bytes for high speed.

Bulk has the ability to detect errors and correct them. In order to avoid starvation, a flow control model similar to the shared endpoint model is recommended for the Controller.

## 2.4 Interrupt endpoint expectations

An interrupt endpoint is used to deliver events in a predictable and timely manner. Event packets can be sent across USB with a known latency.

The interrupt endpoint should have an interval of 1 ms (full speed). For a Controller using USB high-speed the interrupt interval may have an interval of 125 microseconds.

The USB software and firmware requires no intimate knowledge of the events passed to the Controller.

## 2.5 Isochronous endpoints expectations

These isochronous endpoints transfer synchronous data to and from the Controller of the radio.

Time is the critical aspect for this type of data. The USB firmware should transfer the contents of the data to the Controllers' synchronous FIFOs. If the FIFOs are full, the data should be overwritten with new data.

These endpoints have a one (1) ms interval, as required by Chapter 9 of the USB Specification, Versions 1.0 and 1.1.

The radio is capable of three (3) 64 kb/s voice channels (and can receive the data coded in different ways – 16-bit linear audio coding is the method that requires the most data). A suggested max packet size for this endpoint would be at least 64 bytes. (It is recommended that max packet sizes be on power of 2 boundaries for optimum throughput.) However, if it is not necessary to support three voice channels with 16-bit coding, 32 bytes could also be considered an acceptable max packet size.

# 3 CLASS CODE

A class code will be used that is specific to all USB Bluetooth devices. This will allow the proper driver stack to load, regardless of which vendor built the device.

## 3.1 Bluetooth codes

The values shown in Table 3.1 shall be used in the Device Descriptor for Bluetooth Controller devices with USB HCI transport.

| Code | Label | Value | Description |
|---|---|---|---|
| Class | bDeviceClass | 0xE0 | Wireless Controller |
| Subclass | bDeviceSubClass | 0x01 | RF Controller |
| Protocol | bDeviceProtocol | 0x01 | Bluetooth Controller |

*Table 3.1: USB codes for Controllers*

These values should also be used in the interface descriptors for the interfaces described in Section 2.1 that apply to the Controller.

The bDeviceProtocol value 0x04 is previously used.

### 3.1.1 [This section is no longer used]

### 3.1.2 [This section is no longer used]

### 3.1.3 [This section is no longer used]

### 3.1.4 [This section is no longer used]

# 4 DEVICE FIRMWARE UPGRADE

Firmware upgrade capability is not a required feature. If implemented, the firmware upgrade should be compliant with the “Universal Serial Bus Device Class Specification for Device Firmware Upgrade” (version 1.1 or later) available on the USB Forum web site at http://www.usb.org.

# 5 LIMITATIONS

## 5.1 Power specific limitations

Some USB Host Controllers in portable devices will not receive power while the system is in a sleep mode. For example, many PCs do not supply power to the USB port in system power states S3 or S4, as defined in ACPI. Hence, USB wake-up can only occur when the system is in S1 or S2. Furthermore, all connections and state information of the USB Bluetooth Controller will be lost in the system sleep state if power is lost necessitating re-initialization when the device returns to the active state.

Some USB Host Controllers further continually snoop memory when a device is attached to see if there is any work that needs to be done. The snoop is typically performed every 1 ms for USB full-speed devices. This prevents the processor from dropping into a low power state known as C3. Because the processor is not able to enter the C3 state, significant power consumption may occur. This is a major concern for battery-powered Hosts such notebook computers. Some Host Controllers are capable of scheduling polling of USB devices at short intervals while snooping the Host's memory much less frequently. Systems with such Host Controllers may be able to greatly increase the percentage of time spent in the C3 state even if Bluetooth connections are maintained.

A feature called Link Power Management is also recommended for implementation by Bluetooth devices. It is described in an ECN (Engineering Change Notice) from the USB Implementers' Forum.

## 5.2 Other limitations

Data corruption may occur across isochronous endpoints. Endpoints one and two may suffer from data corruption.

USB provides 16-CRC on all data transfers. The USB has a bit error rate of $10^{-13}$.

Note: When a dongle is removed from the system, the radio will lose power (assuming this is a bus-powered device), which means that devices will lose connection.

# 6 BLUETOOTH COMPOSITE DEVICE IMPLEMENTATION

A USB Composite contains multiple independent functions. This section describes how to implement Bluetooth functions within a USB Composite device. This may require the use of Interface Association Descriptors (IAD) to aggregate multiple Interfaces. This also requires the Host to address USB requests to the specific Interface (see [1]).

## 6.1 Configurations

Bluetooth Controller functions may be included in a USB composite device:

- Controller in a multi-radio device
- Controller in a device also containing non-radio functions (e.g. memory)

## 6.2 Using USB Interface Association Descriptors for a Controller function

A Controller ([Vol 1] Part A, Section 2) shall contain at least two interfaces:

- HCI events and ACL data (3 endpoints)
- HCI SCO data (2 endpoints, multiple alternate settings)

and may also contain

- Device Firmware Upgrade (see [2])

When used in a USB Composite device, a Controller function shall use an IAD descriptor to associate the provided interfaces. The following is an example IAD for a Controller function without Device Firmware Upgrade:

- It would be contained within a Configuration Descriptor set.
- It would be followed by two Interface Descriptors and associated Endpoint Descriptors.

| Offset | Field | Size | Value | Description |
|---|---|---|---|---|
| 0 | bLength | 1 | 0x08 | Size of this descriptor in octets |
| 1 | bDescriptorType | 1 | 0x0B | INTERFACE ASSOCIATION DESCRIPTOR |
| 2 | bFirstInterface | 1 | number | Interface number of the first interface associated with this device |

| Offset | Field | Size | Value | Description |
|---|---|---|---|---|
| 3 | bInterfaceCount | 1 | 0x02 | Number of contiguous interfaces associated with the function |
| 4 | bFunctionClass | 1 | 0xE0 | Wireless Controller |
| 5 | bFunctionSubClass | 1 | 0x01 | RF Controller |
| 6 | bFunctionProtocol | 1 | 0x01 | Bluetooth Controller |
| 7 | iFunction | 1 | Index | Pointer to a name string for this function, if any is provide |

*Table 6.1: Example Interface Association Descriptor used for a Controller function*

## 6.3 [This section is no longer used]

# 7 USB BULK SERIALIZATION MODE

USB Bulk Serialization mode is an optional mode that transmits all HCI traffic using the bulk endpoints. Support for this mode is indicated by the Controller by including in its USB configuration descriptor an alternate setting for the first interface containing only the bulk endpoints. The Host instructs the Controller to switch to USB Bulk Serialization mode by selecting this alternate setting using a USB select interface request.

To distinguish packet types, a 1-byte HCI packet indicator shall be prefixed to each HCI packet according to Table 7.1.

| HCI packet type | HCI packet indicator |
|---|---|
| HCI Command packet | 0x01 |
| HCI ACL Data packet | 0x02 |
| HCI Synchronous Data packet | 0x03 |
| HCI Event packet | 0x04 |
| HCI ISO Data packet | 0x05 |
| Reserved for future use | All other values |

*Table 7.1: USB Bulk Serialization mode HCI packet indicators*

# 8 REFERENCES

[1] Universal Serial Bus specification revision 2.0: http://www.usb.org/developers/docs/usb20_docs/

[2] Universal Serial Bus Device Class specification for Device Firmware Upgrade version 1.1: https://usb.org/sites/default/files/DFU_1.1.pdf

# SECURE DIGITAL (SD) TRANSPORT LAYER

*This Part describes the SD transport layer (between the Host and Controller). HCI command, event and data packets flow through this layer, but the layer does not decode them. The Bluetooth SD transport layer is defined in a document owned and maintained by the Secure Digital Association. Information regarding that document is described herein.*

# CONTENTS

**1 Introduction ..... 1765**

**2 Goals ..... 1766**
2.1 Hardware goals ..... 1766
2.2 Software goals ..... 1766
2.3 Configuration goals ..... 1766
2.4 Configuration for multiple Controllers ..... 1767

**3 Physical interface documents ..... 1768**

**4 Communication ..... 1769**
4.1 Overview ..... 1769

**Appendix A Acronyms and Abbreviations ..... 1770**

**Appendix B Related Documents ..... 1771**

**Appendix C Tests ..... 1772**
C.1 Test suite structure ..... 1772

# 1 INTRODUCTION

This document discusses the requirements of the Secure Digital (SD) interface for Bluetooth hardware. Readers should be familiar with SD, SD design issues, and the overall Bluetooth architecture. The reader should also be familiar with the Bluetooth Host Controller interface.

The SD Bluetooth Protocol is documented in the SDIO Card Type-A Specification for Bluetooth, which is owned and maintained by the Secure Digital Association (SDA). The full specification is available to members of the SDA that have signed all appropriate SD NDA and license requirements. The SDA also makes a Non-NDA version available, the Simplified Version of: SDIO Card Type-A Specification for Bluetooth. There are no changes to the SDA document to comply with the requirements of the Bluetooth SIG.

# 2 GOALS

## 2.1 Hardware goals

The Bluetooth SD transport interface specification is designed to take advantage of both the SD Physical Transport bus and the packet orientation of the Bluetooth HCI protocol. Thus, all data is transferred in blocks as packets. Since the block size used on the SD bus may be smaller than the HCI packet, a segmentation and recombination protocol is defined.

SDIO [2] provides different data rate options, including different bit path widths and clock rates. Systems using SDIO should choose options that provide sufficient bandwidth to support the needs of the Controller, both for device control (HCI commands and events) and for data (ACL, SCO).

The specification supports an SDIO-connected Controller.

## 2.2 Software goals

The Bluetooth SD transport interface specification is designed for non-embedded solutions. It is assumed that the Host software does not necessarily have a priori knowledge of the SD Bluetooth device.

The interface is not designed for embedded applications where much of the information passed via the interface is known in advance.

The SDA also defines a Bluetooth interface for embedded applications where the Controller contains protocol layers above HCI (RFComm, SDP etc.). This specification is called SDIO Card Type-B Specification for Bluetooth. Information about this specification can be obtained from the SDA:

https://www.sdcard.org

## 2.3 Configuration goals

The SDIO Card Specification [2] defines SDIO Standard Function Codes in Table 6-4:

0x2 This function supports the SDIO Type-A for Bluetooth standard interface

0x3 This function supports the SDIO Type-B for Bluetooth standard interface

The SDIO Card Type-A Specification for Bluetooth [3] specifies how to implement a Controller. Table 2.1 defines Service ID codes to route HCI messages (codes 0x01 to 0x05).

| SDIO Type-A service ID | Controller |
|---|---|
| 0x00 | reserved for future use |
| 0x01 | HCI Command packet |
| 0x02 | ACL data |
| 0x03 | SCO data |
| 0x04 | HCI Event packet |
| 0x05 | HCI ISO Data packet |
| All other values | Reserved as per [3] |

*Table 2.1: Bluetooth SDIO Controller Service ID codes*

## 2.4 Configuration for multiple Controllers

An SDIO device may contain one or more Controllers as defined in [3]. These Controller functions shall conform to the requirements of [2] section 6.12.

# 3 PHYSICAL INTERFACE DOCUMENTS

This specification references the SD SDIO Card Type-A Specification for Bluetooth. This SDA document defines the Bluetooth HCI for all SD devices that support an HCI level interface. Any SD Bluetooth device claiming compliance with the SD Bluetooth Transport must support this interface and additionally adhere to its device type specification, which is set by the Secure Digital Association. The SDIO Card Type-A Specification for Bluetooth document is based on the SDIO Card Specification, which in turn is based on the SD Memory Card Specification: Part 1 Physical Layer Specification. All of these documents are copyrighted by the SDA and are available ONLY to SDA member companies that have signed the appropriate NDA documents with the SDA. As an introduction to the SD Bluetooth Type A specification, the SDA has created ‘Simplified’ versions of each of these documents. The simplified versions do not contain enough information to fully implement a device, however they do contain enough information to convey the structure and intent of the specifications.

Applicable SDA Documents available to members of the SDA:

**SD Memory Card Specification: Part 1 Physical Layer Specification**

**SDIO Card Specification**

**SDIO Card Type-A Specification for Bluetooth.**

Applicable Simplified SDA Documents available to non-members and members of the SDA:

**Simplified Version of: SD Memory Card Specification: Part 1 Physical Layer Specification**

**Simplified Version of: SDIO Card Specification:**

**Simplified Version of: SDIO Card Type-A Specification for Bluetooth**

More information on the Secure Digital Association and the SD specifications can be found at the SDA web site at https://www.sdcard.org.

# 4 COMMUNICATION

## 4.1 Overview

Figure 4.1 below is a diagram of the communication interface between a Bluetooth SD device and the Bluetooth Host protocol stack. Modifications to this diagram might be needed for operating systems that do not support a miniport model:

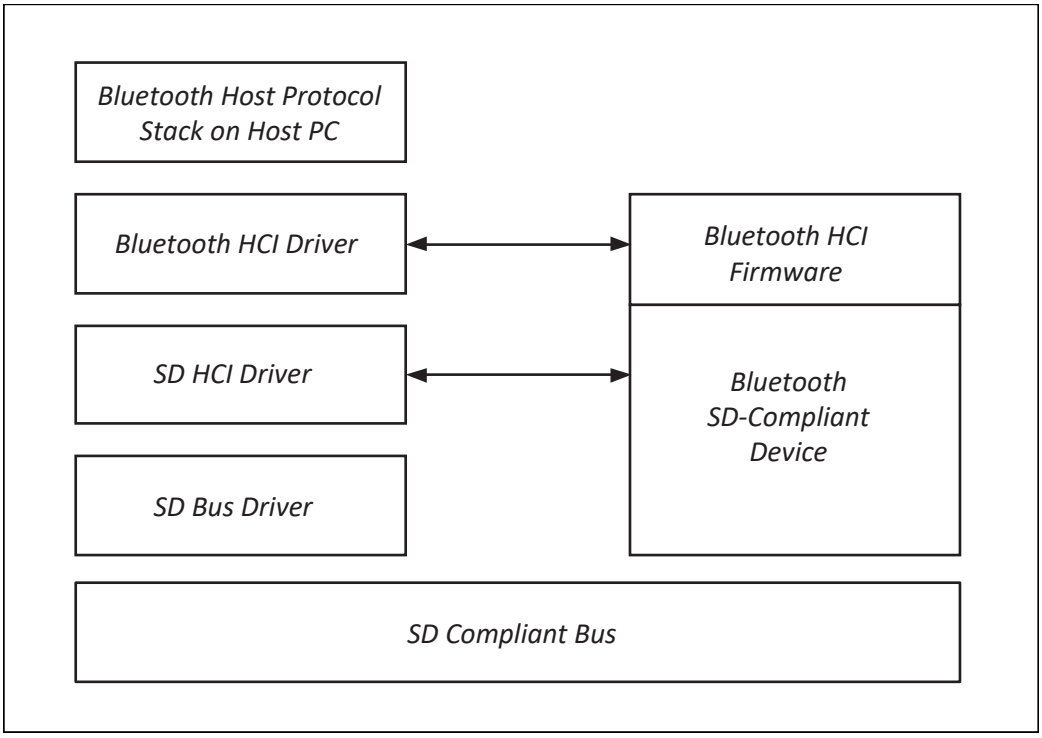



*Figure 4.1: SD communication diagram*

# Appendix A Acronyms and Abbreviations

| Acronym | Description |
|---|---|
| HCI | Host Controller interface |
| NDA | Non-Disclosure Agreement |
| OS | Operating System |
| SD | Secure Digital |
| SDA | Secure Digital Association |
| SDIO | Secure Digital Input/Output |
| SDP | Service Discovery protocol |
| SIG | Special Interest Group |

*Table A.1: Acronyms and abbreviations*

# Appendix B Related Documents

**A)** Applicable SDA Documents available to members of the SDA:

[1] A.1) SD Memory Card Specification: Part 1 Physical Layer Specification

[2] A.2) SDIO Card Specification

[3] A.3) SDIO Card Type-A Specification for Bluetooth

[4] A.4) SDIO Card Type-B Specification for Bluetooth

[5] A.5) SDIO Card Physical Test Specification

[6] A.5) SDIO Host Physical Test Specification

[7] A.6) SD Bluetooth Type A Test Specification

**These documents are available to members of the SDA in the "Members Only" section of the SDA web site** (https://members.sdcard.org/members). See https://www.sdcard.org/join/index.html for information on joining the SDA.

**B)** Applicable Simplified SDA Documents available to non-members and members of the SDA:

B.1) Simplified Version of: SD Memory Card Specification: Part 1 Physical Layer Specification https://www.sdcard.org/downloads/pls/

B.2) Simplified Version of: SDIO Card Specification https://www.sdcard.org/downloads/pls/

B.3) Simplified Version of: SDIO Card Type-A Specification for Bluetooth https://www.sdcard.org/downloads/pls/

# Appendix C Tests

The SDA has defined formal test procedures for SDIO Type A Bluetooth cards (Controller) and Hosts. It is expected that both Controllers and Hosts will comply with all test requirements set forth by the SDA in accordance with the rules of the SDA. The Bluetooth SIG does not require any formal testing to comply with SIG requirements. The test document names are listed in Appendix B.

## C.1 Test suite structure

There are two types of tests defined for the HCI SD Transport Layer:

1. Functional Tests
2. Protocol Tests

Tests of both types are defined for both the Host and Controller.

The purpose of the functional tests is to verify that the SD Bluetooth Type A Specification, SDIO Standard and SD Physical Standard have been implemented according to the specifications. These tests and the test environment for these tests are defined in documents provided by the SDA.

The purpose of the protocol tests are to verify that the Bluetooth Controller SD implementation or the Host implementation are according to the SD Bluetooth Type A specification.

The test environment for the protocol tests consists of the Test System and the implementation under test (IUT) as illustrated in Figure C.1 below.

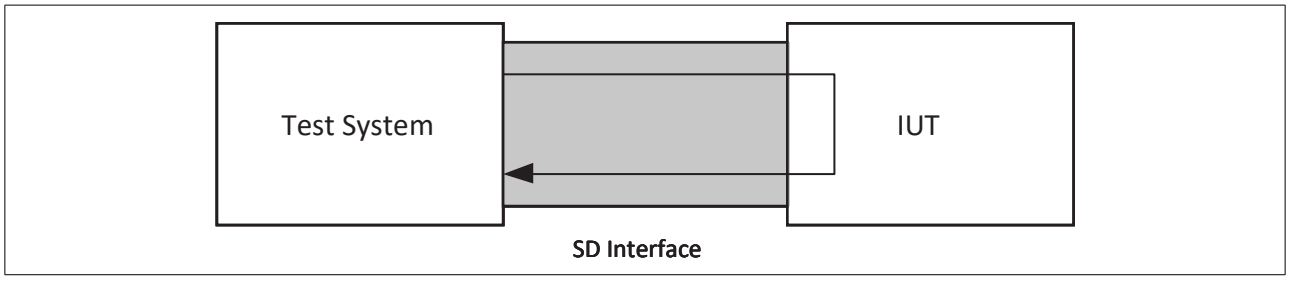



*Figure C.1: Protocol test environment*

The Test System is typically a PC with an SD interface. The IUT is placed into local loopback mode and standard HCI commands are used to drive the tests. The test results are verified in the Test System.

Host Controller Interface
Part D

# THREE-WIRE UART TRANSPORT LAYER

*This Part describes the Three-Wire UART transport layer (between the Host and Controller). HCI command, event, and data packets flow through this layer, but the layer does not decode them.*

# CONTENTS

**1 General ..... 1776**

**2 Overview ..... 1777**

**3 Slip layer ..... 1778**
3.1 Encoding a packet ..... 1778
3.2 Decoding a packet ..... 1778

**4 Packet header ..... 1780**
4.1 Sequence Number ..... 1780
4.2 Acknowledge Number ..... 1780
4.3 Data Integrity Check Present ..... 1781
4.4 Reliable Packet ..... 1781
4.5 Packet Type ..... 1781
4.6 Payload Length ..... 1782
4.7 Packet Header Checksum ..... 1782

**5 Data Integrity Check ..... 1783**
5.1 16-bit CCITT-CRC ..... 1783

**6 Reliable packets ..... 1784**
6.1 Header Checksum error ..... 1784
6.2 Slip Payload Length error ..... 1784
6.3 Data Integrity Check error ..... 1784
6.4 Out Of Sequence Packet error ..... 1784
6.5 Acknowledgment ..... 1785
6.6 Resending packets ..... 1785
6.7 Example reliable packet flow ..... 1785

**7 Unreliable packets ..... 1788**
7.1 Unreliable packet header ..... 1788
7.2 Unreliable packet error ..... 1788

**8 Link Establishment ..... 1789**
8.1 Uninitialized state ..... 1789
8.2 Initialized state ..... 1790
8.3 Active state ..... 1790
8.4 Sync message ..... 1790
8.5 Sync Response message ..... 1791
8.6 Config message ..... 1791
8.7 Config Response message ..... 1792

8.8 Configuration Field ..... 1792
8.8.1 Configuration messages ..... 1793
8.8.2 Sliding window size ..... 1793
8.8.3 Level of Data Integrity Check ..... 1793
8.8.4 Out of Frame Software Flow Control ..... 1794
8.8.5 Version Number ..... 1794

**9 Low power ..... 1795**
9.1 Wakeup message ..... 1795
9.2 Woken message ..... 1795
9.3 Sleep message ..... 1796

**10 Out of Frame Control ..... 1797**
10.1 Software Flow Control ..... 1797

**11 Hardware configuration ..... 1798**
11.1 Wires ..... 1798
11.1.1 Transmit & receive ..... 1798
11.1.2 Ground ..... 1798
11.2 Hardware flow ..... 1798
11.2.1 RTS & CTS ..... 1798

**12 Recommended parameters ..... 1799**
12.1 Timing parameters ..... 1799
12.1.1 Acknowledgment of packets ..... 1799
12.1.2 Resending reliable packets ..... 1799

**13 References ..... 1800**

# 1 GENERAL

The HCI Three-Wire UART Transport Layer makes it possible to use the Bluetooth HCI over a serial interface between two UARTs. The HCI Three-Wire UART Transport Layer assumes that the UART communication may have bit errors, overrun errors or burst errors. See also [Vol 4] Part A, UART Transport Layer.

# 2 OVERVIEW

The HCI Three-Wire UART Transport Layer is a connection based protocol that transports HCI commands, events, ACL and Synchronous packets between the Host and the Controller. Packet construction is in done in two steps. First, it adds a packet header onto the front of every HCI packet which describes the payload. Second, it frames the packets using a SLIP protocol. Finally, it sends this packet over the UART interface.

The SLIP layer converts an unreliable octet stream into an unreliable packet stream. The SLIP layer places start and end octets around the packet. It then changes all occurrences of the frame start or end octet in the packet to an escaped version.

The packet header describes the contents of the packet, and if this packet needs to be reliably transferred, a way of identifying the packet uniquely, allowing for retransmission of erroneous packets.

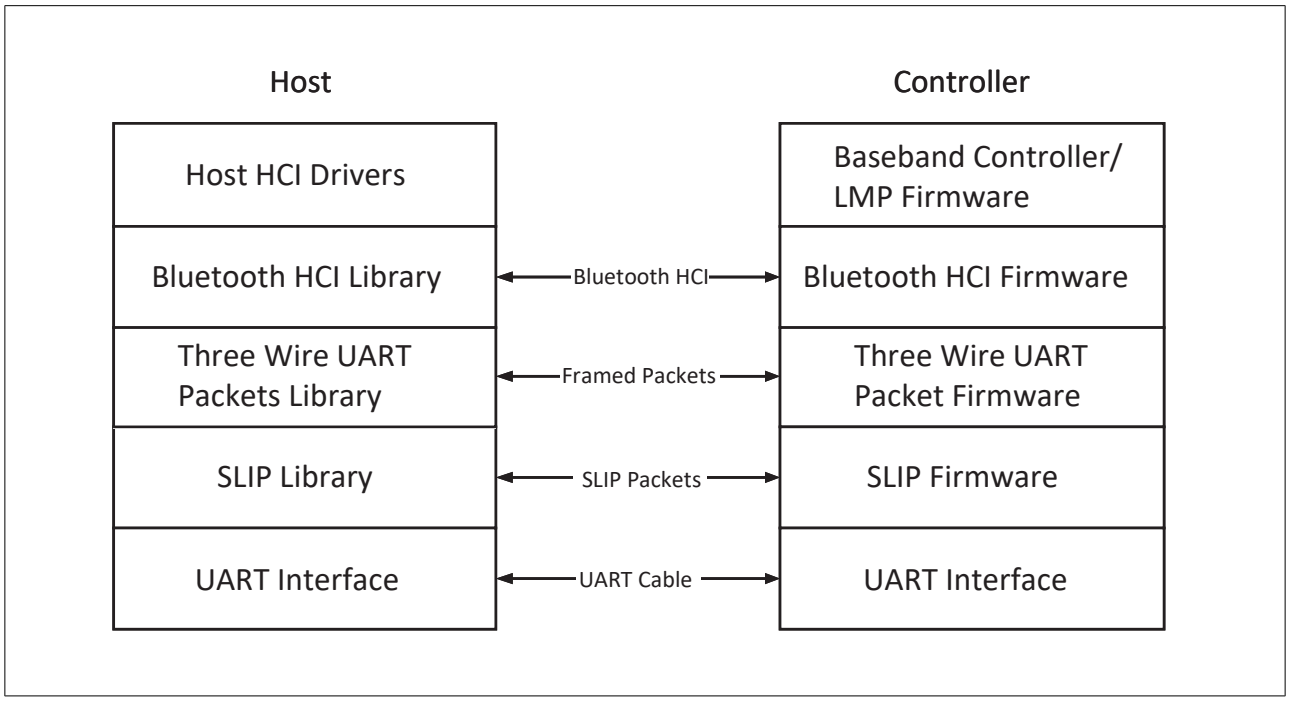



*Figure 2.1: The relationship between the Host and the Controller*

# 3 SLIP LAYER

The SLIP layer places packet framing octets around each packet being transmitted over the Three-Wire UART Transport Layer. This delimits the packets and allows packet boundaries to be detected if the receiver loses synchronization. The SLIP layer is based upon the RFC 1055 Standard [1].

## 3.1 Encoding a packet

The SLIP layer performs octet stuffing on the octets entering the layer so that specific octet codes which may occur in the original data do not occur in the resultant stream.

The SLIP layer places octet 0xC0 at the start and end of every packet it transmits. Any occurrence of 0xC0 in the original packet is changed to the sequence 0xDB 0xDC before being transmitted. Any occurrence of 0xDB in the original packet is changed to the sequence 0xDB 0xDD before being transmitted. These sequences, 0xDB 0xDC and 0xDB 0xDD are SLIP escape sequences. All SLIP escape sequences start with 0xDB. All SLIP escape sequences are listed in Table 3.1.

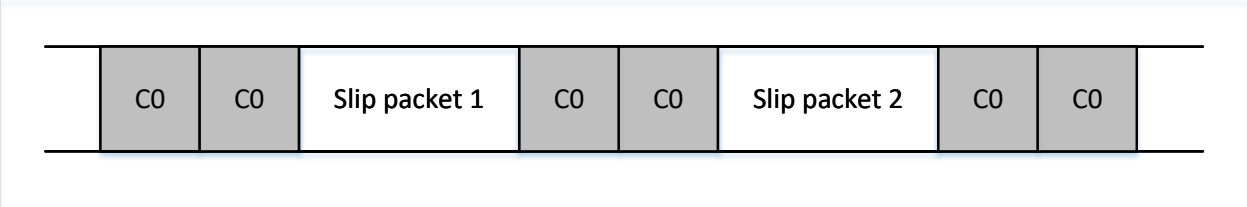



*Figure 3.1: SLIP packets with 0xC0 at the start and end of each packet*

## 3.2 Decoding a packet

When decoding a SLIP stream, a device will first be in an unknown state, not knowing if it is at the start of a packet or in the middle of a packet. The device shall therefore discard all octets until it finds a 0xC0. If the 0xC0 is followed immediately by a second 0xC0, then the device will discard the first 0xC0 as it was presumably the end of the last packet, and the second 0xC0 was the start of the next packet. The device shall then be in the decoding packet state. It can then decode the octets directly changing any SLIP escape sequences back into their unencoded form. When the device decodes the 0xC0 at the end of the packet, it will calculate the length of the SLIP packet, and pass the packet data into the packet decoder. The device will then seek the next packet. If the device does not receive an 0xC0 for the start of the next packet, then all octets up to and including the next 0xC0 will be discarded.

| SLIP Escape Sequence | Unencoded form | Notes |
|---|---|---|
| 0xDB 0xDC | 0xC0 | |
| 0xDB 0xDD | 0xDB | |
| 0xDB 0xDE | 0x11 | Only valid when OOF Software Flow Control is enabled |
| 0xDB 0xDF | 0x13 | Only valid when OOF Software Flow Control is enabled |

*Table 3.1: SLIP escape sequences*

# 4 PACKET HEADER

Every packet that is sent over the Three-Wire UART Transport Layer has a packet header. It also has an optional Data Integrity Check at the end of the payload. The Transport Layer does not support packet segmentation and reassembly. Each transport packet will contain at most one higher layer packet.

A packet consists of a Packet Header of 4 octets, a Payload of 0 to 4095 octets, and an optional Data Integrity Check of 2 octets. See Figure 4.1.

The packet header consists of a Sequence Number of 3 bits, an Acknowledge Number of 3 bits, a Data Integrity Check Present bit, a Reliable Packet bit, a Packet Type of 4 bits, a Payload Length of 12 bits and an 8 bit Header Checksum. See Figure 4.2.

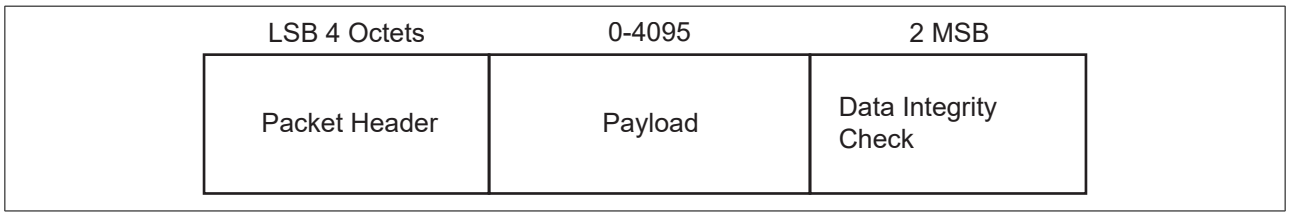



*Figure 4.1: Packet format*

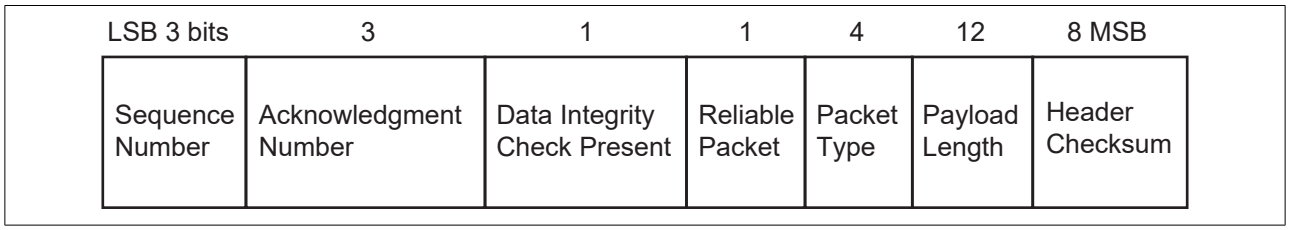



*Figure 4.2: Packet header format*

## 4.1 Sequence Number

For unreliable packets this field shall be set to 0 on transmit and ignored on receive.

Each new reliable packet shall be assigned a sequence number which is equal to the sequence number of the previous reliable packet plus one *mod* eight. A packet shall use the same sequence number each time it is retransmitted.

## 4.2 Acknowledge Number

The acknowledge number shall be set to the sequence number of the next reliable packet this device expects to receive. See Section 6.4.

## 4.3 Data Integrity Check Present

If a 16 bit CCITT-CRC Data Integrity Check is appended to the end of the payload, this bit shall be set to 1.

## 4.4 Reliable Packet

If this bit it set to 1, then this packet is reliable. This means that the sequence number field is valid, and the receiving end shall acknowledge its receipt. If this bit is set to 0, then this packet is unreliable.

## 4.5 Packet Type

There are five kinds of HCI packets that can be sent via the Three-Wire UART Transport Layer; these are HCI Command packet, HCI Event packet, HCI ACL Data packet, HCI Synchronous Data packet, and HCI ISO Data packet (see [Vol 4] Part E, Section 5.4). HCI Command packets can be sent only to the Controller, HCI Event packets can be sent only from the Controller, and HCI ACL/Synchronous/ISO Data packets can be sent both to and from the Controller.

HCI packet coding does not provide the ability to differentiate the five HCI packet types. Therefore, the Packet Type field is used to distinguish the different packets. The acceptable values for this Packet Type field are given in Table 4.1.

| HCI Packet Type | Packet Type |
|---|---|
| Acknowledgment packets | 0 |
| HCI Command packet | 1 |
| HCI ACL Data packet | 2 |
| HCI Synchronous Data packet | 3 |
| HCI Event packet | 4 |
| HCI ISO Data packet | 5 |
| Vendor Specific | 14 |
| Link Control packet | 15 |
| Reserved for future use | All other values |

*Table 4.1: Three-Wire UART packet type*

HCI Command packets, HCI ACL Data packets, HCI Event packets, and HCI ISO Data packets are always sent as reliable packets. HCI Synchronous Data packets are sent as unreliable packets unless HCI Synchronous Flow Control is enabled, in which case they are sent as reliable packets.

In addition to the five HCI packet types, other packet types are defined. One packet type is defined for pure Acknowledgment packets, and one additional packet type is to

support link control. One packet type is made available to vendors for their own use. All other Three-Wire UART Packet Types are reserved for future use.

## 4.6 Payload Length

The payload length is the number of octets in the payload data. This does not include the length of the packet header, or the length of the optional data integrity check.

## 4.7 Packet Header Checksum

The packet header checksum validates the contents of the packet header against corruption. This is calculated by setting the Packet Header Checksum to a value such that the (unsigned) sum *mod* 256 of the four octets of the Packet Header, including the Packet Header Checksum, is 0xFF.

# 5 DATA INTEGRITY CHECK

The Data Integrity Check field is optional. It can be used to ensure that the packet is valid. The Data Integrity Check field is appended onto the end of the packet. Each octet of the Packet Header and Packet Payload is used to compute the Data Integrity Check.

## 5.1 16-bit CCITT-CRC

The CRC is defined using the CRC-CCITT generator polynomial

$g(D) = D^{16} + D^{12} + D^5 + 1$

(see Figure 5.1)

The CRC shift register is filled with 1s before calculating the CRC for each packet. Octets are fed through the CRC generator least significant bit first.

The most significant parity octet is transmitted first (where the CRC shift register is viewed as shifting from the least significant bit towards the most significant bit). Therefore, the transmission order of the parity octets within the CRC shift register is as follows:

x[8] (first), x[9],..., x[15], x[0], x[1],..., x[7] (last)

where x[15] corresponds to the highest power CRC coefficient and x[0] corresponds to the lowest power coefficient.

The switch S shall be set in position 1 while the data is shifted in. After the last bit has entered the LFSR, the switch shall be set in position 2, and the registers contents shall be read out for transmission.

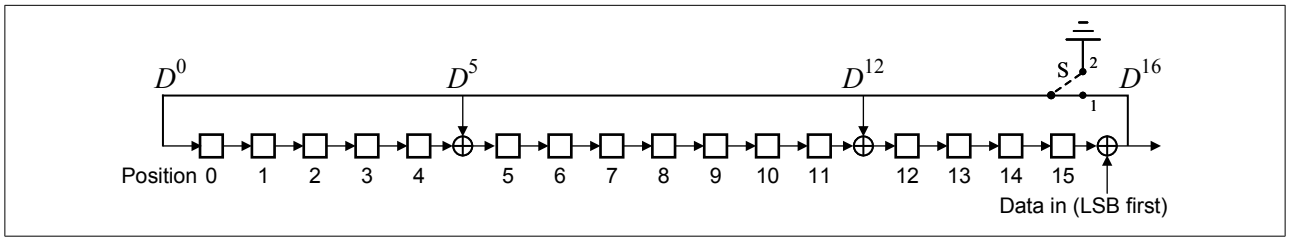



*Figure 5.1: The LFSR circuit generating the CRC*

# 6 RELIABLE PACKETS

To allow the reliable transmission of packets through the transport, a method needs to be defined to recover from packet errors. The Host or Controller can detect a number of different errors in the packet.

## 6.1 Header Checksum error

The header of the packet is protected by a Packet Header Checksum. If the (unsigned) sum *mod* 256 of the four octets of the header is not 0xFF, then the packet has an unrecoverable error and all information contained in the packet shall be discarded.

## 6.2 Slip Payload Length error

The length of the SLIP packet shall be checked against the Packet Payload Length. If the Data Integrity Check Present bit is set to 1, then the SLIP packet length should be 6 + Packet Payload Length. If the Data Integrity Check Present bit is set to 0, then the SLIP packet length should be 4 + Packet Payload Length. If this check fails, then all information contained in the packet shall be discarded. The SLIP packet length is the length of the data received from the SLIP layer after the SLIP framing, and SLIP escape codes have been processed.

## 6.3 Data Integrity Check error

The packet may have a Data Integrity Check at the end of the payload. This is controlled by the Data Integrity Check Present bit in the header. If this is set to 1, then the Data Integrity Check at the end of the payload is checked. If this is different from the value expected, then the packet shall be discarded. If the link is configured to not use data integrity checks, and a packet is received with the Data Integrity Check Present bit set to 1, then the packet shall be discarded.

## 6.4 Out Of Sequence Packet error

Each device keeps track of the sequence number it expects to receive next. This will be one more than the sequence number of the last successfully received reliable packet, *mod* eight. If a reliable packet is received which has the expected sequence number, then this packet shall be accepted.

If a reliable packet is received which does not have the expected sequence number, then the packet shall be discarded.

## 6.5 Acknowledgment

Whenever a reliable packet is received, an acknowledgment shall be generated.

If a packet is available to be sent, the Acknowledgment Number of that packet shall be updated to the latest expected sequence number.

If a requirement to send an acknowledgment value is pending, but there are no other packets available to be sent, the device may send a pure Acknowledgment packet. This is an Unreliable packet, with the Packet Type set to 0, Payload Length set to 0, and the Sequence Number set to 0.

The maximum number of reliable packets that can be sent without acknowledgment defines the sliding window size of the link. This is configured during link establishment. See Section 8.6, Section 8.7, and Section 8.8.

## 6.6 Resending packets

A Reliable packet shall be resent until it is acknowledged. Devices should refrain from resending packets too quickly to avoid saturating the link with retransmits. See Section 12.1.2.

## 6.7 Example reliable packet flow

Figure 6.1 shows the transmission of reliable packets between two devices. Device A sends a packet with a Sequence Number of 6, and an Acknowledgment Number of 3. Device B receives this packet correctly, so needs to generate an acknowledgment. Device B then sends a packet with Sequence Number 3 with its Acknowledgment Number set to the next expected packet Sequence Number from Device A of 7.

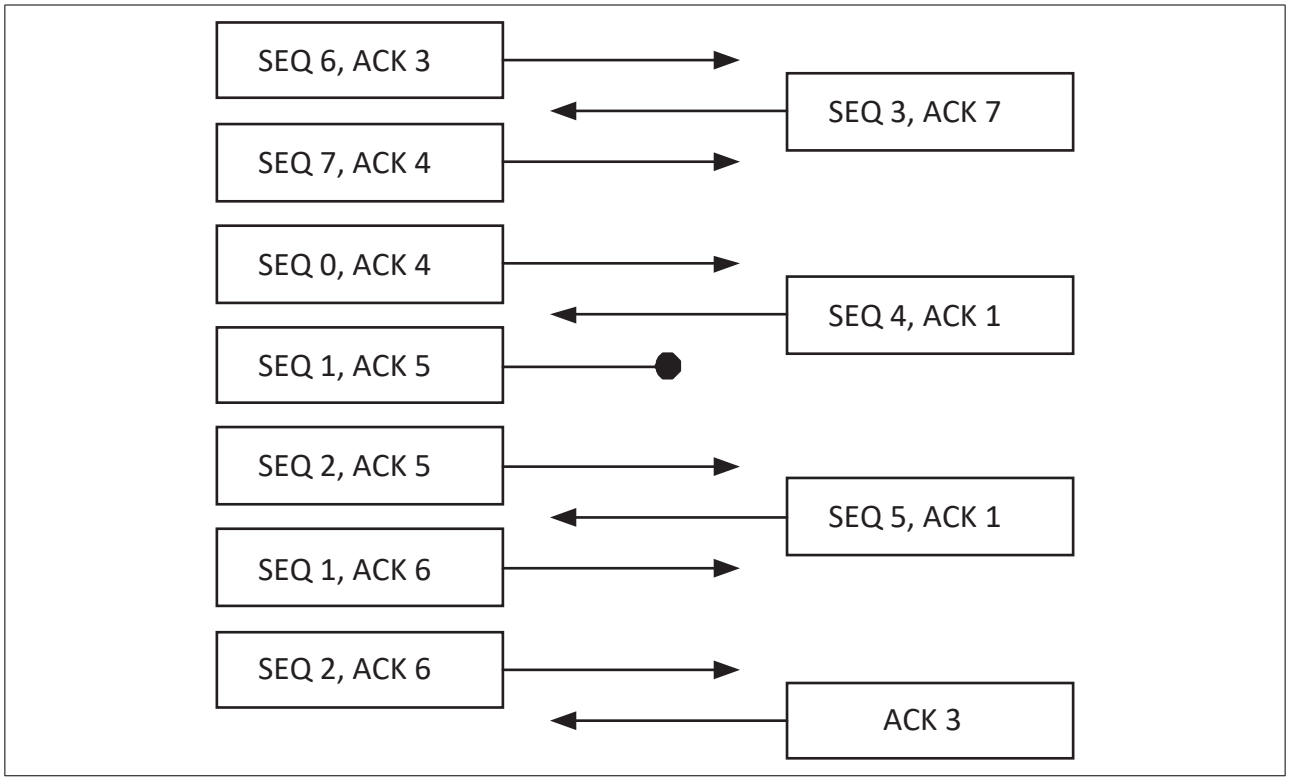



*Figure 6.1: Message diagram showing transmission of reliable packets*

Device A receives a packet with Sequence Number 3 and an Acknowledgment Number of 7. Device A was expecting this sequence number so needs to generate an acknowledgment. The Acknowledgment Number of 7 is one greater than the last Sequence Number that was sent, meaning that this packet was received correctly (see Section 6.6).

Device A sends two packets, Sequence Numbers 7 and 0. Both packets have the Acknowledgment Number of 4, the next sequence number it expects from Device B. Device B receives the first correctly, and increments its next expected sequence number to 0. It then receives the second packet correctly, and increments the next expected sequence number to 1.

Device B sends a packet with Sequence Number 4, and the Acknowledgment Number of 1. This will acknowledge both of the previous two packets sent by Device A.

Device A now sends two more packets, Sequence Numbers 1 and 2. Unfortunately, the first packet is corrupted. Device B receives the first packet, and discovers the error, so discards this packet (see Section 6.1, Section 6.2 or Section 6.3). It generates an acknowledgment of this erroneously received reliable packet. Device B then receives the second packet. This is received out of sequence, as it is currently expecting Sequence Number 1, but has received Sequence Number 2 (see Section 6.4). Again, it generates an acknowledgment.

Device B sends another packet with Sequence Number 5. It is still expecting a packet with Sequence Number 1 next, so the Acknowledgment Number is set to 1. Device A receives this, and accepts this packet.

Device A has not had either of its last two packets acknowledged, so it resends them (see Section 6.6) and updates the Acknowledgment Number of the original packets that were sent (see Section 6.5). The Sequence Numbers of these packets stay the same (see Section 4.1).

Device B receives these packets correctly, and schedules the sending of an acknowledgment. Because Device B doesn't have any data packets that need to be sent, it sends a pure Acknowledgment packet (see Section 6.5).

# 7 UNRELIABLE PACKETS

To allow the transmission of unreliable packets through the transport, the following method shall be used.

## 7.1 Unreliable packet header

An unreliable packet header always has the Reliable Packet bit set to 0. The sequence number shall be set to 0. The Data Integrity Check Present, Acknowledgment Number, Packet Type, Payload Length and Packet Header Checksum shall all be set the same as a Reliable packet.

## 7.2 Unreliable packet error

If a packet that is marked as unreliable and the packet has an error, then the packet shall be discarded.

# 8 LINK ESTABLISHMENT

Before any packets except Link Control packets can be sent, the Link Establishment procedure shall be performed. This ensures that the sequence numbers are initialized correctly and that the two sides are using the same baud rate, allows detection of peer reset, and allows the device to be configured.

Link Establishment is defined by a state machine with three states: Uninitialized, Initialized and Active. When the transport is first started, the link is in the Uninitialized State. There are four messages that are defined: SYNC, SYNC RESPONSE, CONFIG and CONFIG RESPONSE. All four link establishment messages shall be sent with the Data Integrity Present flag set to 0.

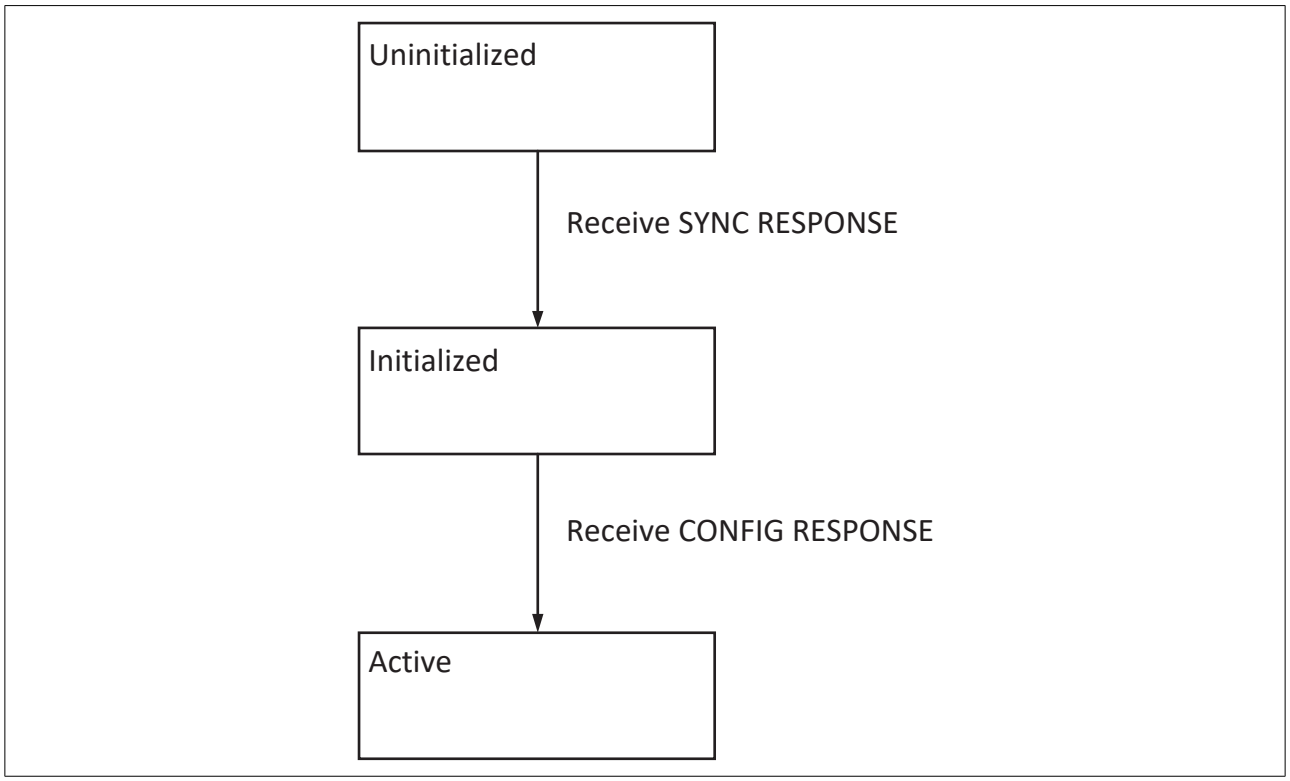



*Figure 8.1: Link Establishment state diagram*

## 8.1 Uninitialized state

In the Uninitialized State a device periodically[1] sends SYNC messages. If a SYNC message is received, the device shall respond with a SYNC RESPONSE message. If a SYNC RESPONSE message is received, the device shall move to the Initialized

[1]During link establishment, various messages are sent periodically. It is suggested to send 4 messages per second.

State. In the Uninitialized State only SYNC and SYNC RESPONSE messages are valid, all other messages that are received shall be discarded. If an invalid packet is received, the device shall respond with a SYNC message. The device shall not send any acknowledgment packets in the Uninitialized State[1].

In the Uninitialized State the Controller may wait until it receives a SYNC message before sending its first SYNC message. This allows the Host to control when the Controller starts to send data.

The SYNC message can be used for automatic baud rate detection. It is assumed that the Controller shall stay on a single baud rate, while the Host could hunt for the baud rate. Upon receipt of a SYNC RESPONSE message, the Host can assume that the correct baud rate has been detected.

## 8.2 Initialized state

In the Initialized State a device periodically sends CONFIG messages. If a SYNC message is received, the device shall respond with a SYNC RESPONSE message. If a CONFIG message is received, the device shall respond with a CONFIG RESPONSE message. If a CONFIG RESPONSE message is received, the device will move to the Active State. All other messages that are received shall be ignored.

## 8.3 Active state

In the Active State, a device can transfer higher layer packets through the transport. If a CONFIG message is received, the device shall respond with a CONFIG RESPONSE message. If a CONFIG RESONSE message is received, the device shall discard this message.

If a SYNC message is received while in the Active State, it is assumed that the peer device has reset. The local device should therefore perform a full reset of the upper stack, and start Link Establishment again at the Uninitialized State.

Upon entering the Active State, the first packet sent shall have its SEQ and ACK numbers set to zero.

## 8.4 Sync message

The SYNC message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2.

[1]Any packet that was erroneous would normally be acknowledged, as the recipient does not know if the packet was a reliable packet or not. The recipient cannot do this in the Uninitialized State, as it is possible to receive corrupt data while in the Uninitialized state.

The payload is composed of the octet pattern 0x01 0x7E[1].

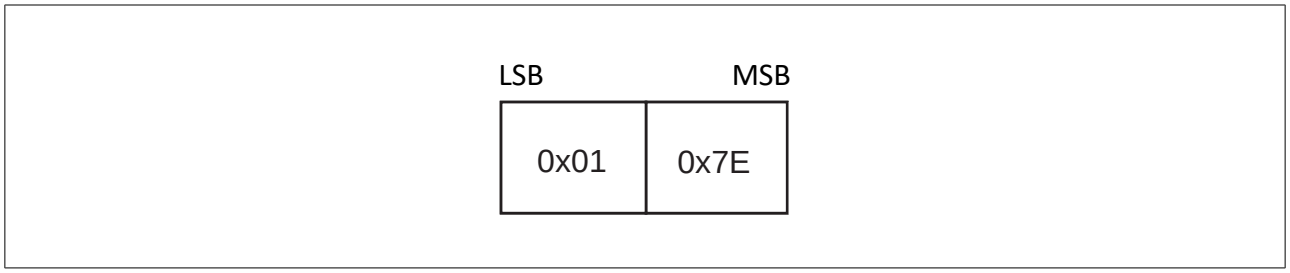



*Figure 8.2: Sync message format*

## 8.5 Sync Response message

The SYNC RESPONSE message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2. The payload is composed of the octet pattern 0x02 0x7D.

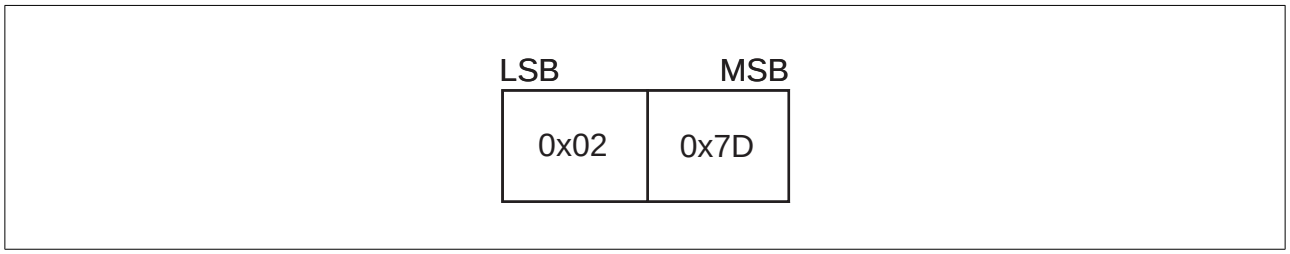



*Figure 8.3: Sync Response message format*

## 8.6 Config message

The CONFIG message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2 plus the size of the Configuration Field. The payload is composed of the octet pattern 0x03 0xFC and the Configuration Field.

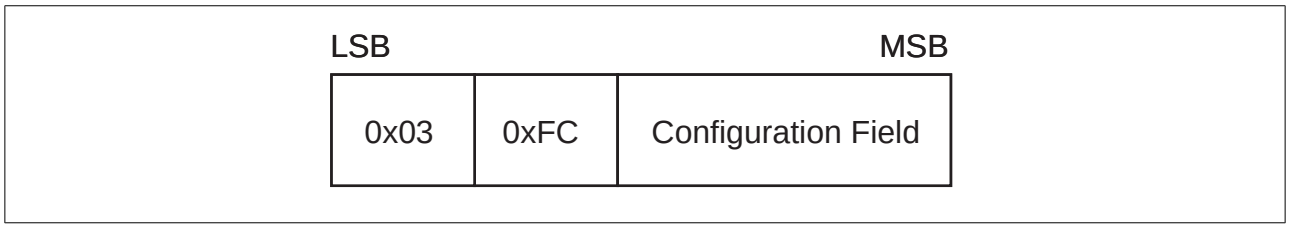



*Figure 8.4: Configuration message format*

[1]The second octet for all Link Control packets equals the least significant 7 bits of the first octet, inverted, with the most significant bit set to ensure even parity.

## 8.7 Config Response message

The CONFIG RESPONSE message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2 plus the size of the Configuration Field. The payload is composed of the octet pattern 0x04 0x7B and the Configuration Field.

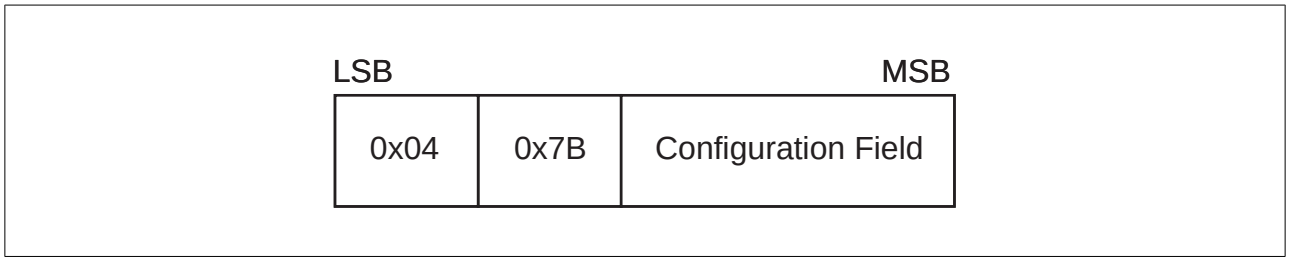



*Figure 8.5: Configuration Response message format*

## 8.8 Configuration Field

The Configuration Field contains the Version Number, Sliding Window Size, the Data Integrity Check Type, and if Out Of Frame (OOF) Software Flow Control is allowed. The format of this field is specified in Figure 8.6.

The Configuration Field in a CONFIG message sent by the Host determines what the Host can transmit and accept. The Configuration Field in a CONFIG RESPONSE message sent by the Controller determines what the Host and Controller shall transmit and can expect to receive.

The Controller sends CONFIG messages without a Configuration Field. The Host sends CONFIG RESPONSE messages without a Configuration Field.

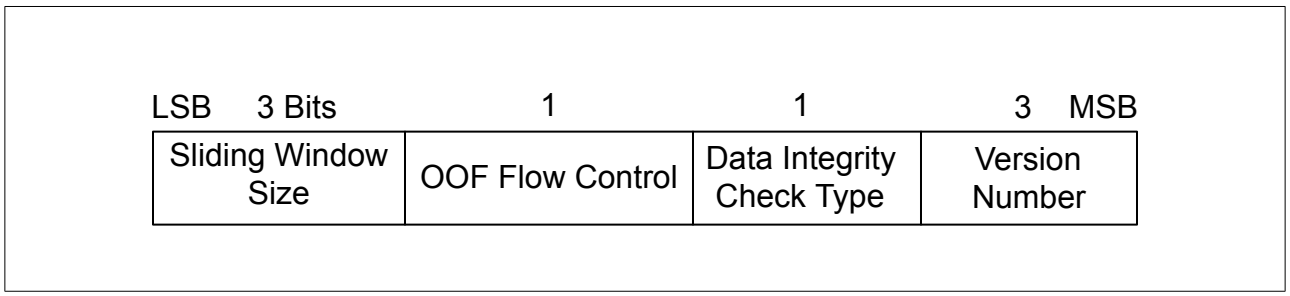



*Figure 8.6: Configuration Field detail*

To allow for future extension of the Configuration Field, the size of the message determines the number of significant Configuration Octets in the payload. Future versions of the specification may use extra octets. Any bits that are not included in the message shall be set to 0. Any bits that are not defined are reserved for future use.

A device shall not change the values if sends in the Configuration Field during Link Establishment.

### 8.8.1 Configuration messages

The CONFIG – CONFIG RESPONSE message sequence configures the link in both directions. Until a CONFIG RESPONSE message is received only unreliable Link Establishment messages may be sent. Once CONFIG RESPONSE message has been received all other packet types may be sent, and received messages passed up to the Host.

The CONFIG and CONFIG RESPONSE messages contain a set of options for both devices on the link. The Host sends a CONFIG message with the set of options that the Host would like to use. The Controller responds with a CONFIG RESPONSE message with the set of options that the Host and the Controller will use. This means that the Controller is in full control of the set of options that will be used for all messages sent by both the Host and Controller.

### 8.8.2 Sliding window size

This is the maximum number of reliable packets a sender of the CONFIG message can send without requiring an acknowledgment. The value of this field shall be in the range one to seven. The value in the CONFIG RESPONSE message shall be less than or equal to the value in the CONFIG message. For example, the Host may suggest a window size of five in its CONFIG message and the Controller may respond with a value of three in its CONFIG RESPONSE message, but not six or seven. Both devices will then use a maximum sliding window size of three.

### 8.8.3 Level of Data Integrity Check

The CONFIG message contains a bit field describing the types of Data Integrity Checks the sender is prepared to transmit. The peer will select the one it is prepared to use and send its choice in the CONFIG RESPONSE message.

If data integrity checks are not required, then the Data Integrity Check Present bit shall be set to 0 by the Host and Controller.

| Level of Data Integrity | Parameter Description for CONFIG Message |
|---|---|
| 0 | No Data Integrity Check is supported. |
| 1 | 16 bit CCITT-CRC may be used. |

*Table 8.1: Data Integrity Check type in the CONFIG message*

| Level of Data Integrity | Parameter Description for CONFIG RESPONSE Message |
|---|---|
| 0 | No Data Integrity Check is used. |
| 1 | 16 bit CCITT-CRC may be used. |

*Table 8.2: Data Integrity Check type in the CONFIG RESPONSE message*

### 8.8.4 Out of Frame Software Flow Control

By default, the transport uses no flow control except that mandated by the HCI Functional Specification and the flow control achieved by not acknowledging reliable Host messages. If Software Flow Control is to be used, this needs to be negotiated.

The CONFIG message specifies whether the sender of the CONFIG message is prepared to receive Out of Frame Software Flow Control messages. The CONFIG RESPONSE message specifies whether the peer can send Out of Frame Software Flow Control messages. The CONFIG RESPONSE message may have the field set to 1 only if the CONFIG message had it set to 1. (See Section 10.1)

### 8.8.5 Version Number

The Version Number of this protocol shall determine which facilities are available to be used.

The CONFIG message specifies the Version Number supported by the Host. The CONFIG RESPONSE message specifies the Version Number that shall be used by the Host and Controller when sent by the Controller. The value in the CONFIG RESPONSE message shall be less than or equal to the value in the CONFIG message. The Version Numbers are enumerated in Table 8.3. This specification defines version 1.0 (Version Number = 0).

| Version Number | Parameter Description for CONFIG and CONFIG RESPONSE Message |
|---|---|
| 0 | Version 1.0 of this Protocol |
| All other values | Reserved for future use |

*Table 8.3: Version number in the CONFIG and CONFIG RESPONSE messages*

# 9 LOW POWER

After a device is in the Active State, either side of the transport link may switch to a low power state. Because recovery from a loss of synchronization is possible, it is allowable to stop listening for incoming packets at any time.

To make the system more responsive after a device has entered a low power state, a system of messages is employed to allow either side to notify the other that they are entering a low power state and to wake a device from that state. These messages are sent as Link Control packets. It is optional for a device to support the Sleep message. The Wakeup and Woken messages are mandatory.

## 9.1 Wakeup message

The Wakeup message shall be the first message sent whenever the device believes that the other side is asleep. The device shall then repeatedly send the Wakeup message until the Woken message is received. There shall be at least a one character gap between the sending of each Wakeup message to allow the UART to resynchronize. The Wakeup message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x05 0xFA. The Wakeup message shall be used after a device has sent a Sleep message. It is mandatory to respond to the Wakeup message.

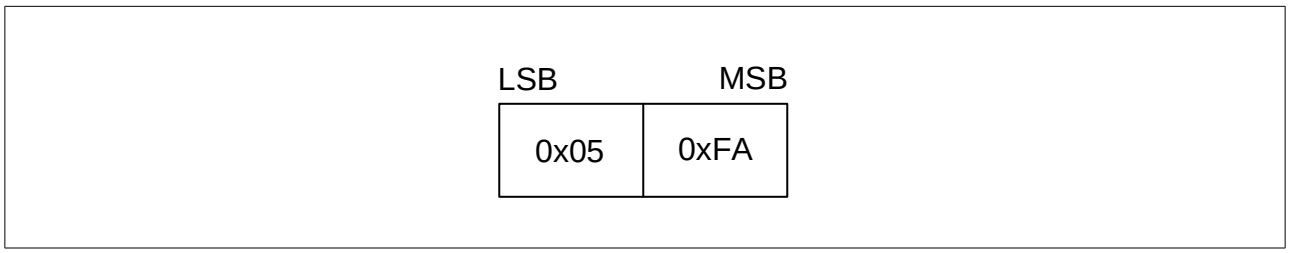



*Figure 9.1: Wakeup message payload format*

## 9.2 Woken message

The Woken message shall be sent whenever a Wakeup message is received even if the receiver is currently not asleep. Upon receiving a Woken message, a device can determine that the other device is not in a low power state and can send and receive data. The Woken message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x06 0xF9.

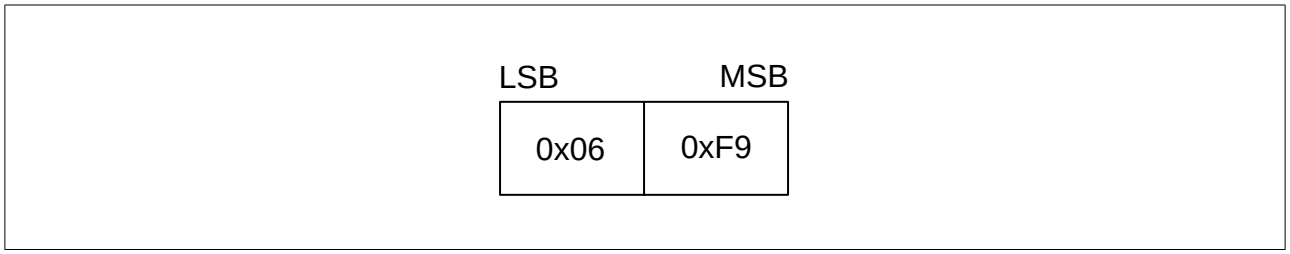



*Figure 9.2: Woken message payload format*

## 9.3 Sleep message

A Sleep message can be sent at any time after Link Establishment has finished. It notifies the other side that this device is going into a low power state, and that it may also go to sleep. If a device sends a Sleep message it shall use the Wakeup / Woken message sequence before sending any data. If a device receives a Sleep message, then it should use the Wakeup / Woken message sequence before sending any data. The Sleep message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x07 0x78.

The sending of this message is optional. The receiver of this message need not go to sleep, but cooperating devices may be able to schedule sleeping more effectively with this message.

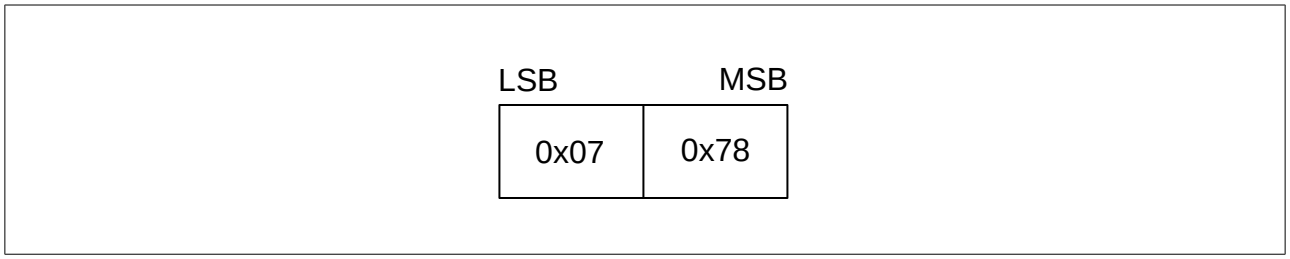



*Figure 9.3: Sleep message payload format*

# 10 OUT OF FRAME CONTROL

It is possible to embed information in the SLIP data stream after a SLIP ESCAPE character that can allow for Software Flow Control. This feature is optional and may be negotiated in the Link Establishment configuration messages.

## 10.1 Software Flow Control

If Software Flow Control is enabled, then the standard XON / XOFF (0x11 and 0x13) characters will control the flow of data over the transport. To allow the XON / XOFF characters to be sent in the payload, they shall be escaped as follows: 0x11 shall be changed to 0xDB 0xDE, 0x13 shall be changed to 0xDB DF. This means that the XON / XOFF characters in the data stream are used only by software flow control.

If Software Flow Control is disabled, then the SLIP escape sequences 0xDB 0xDE and 0xDB 0xDF are undefined. In this case, the original octets of 0x11 and 0x13 shall not be changed. Flow control should always be provided by the tunneled protocols, e.g. HCI Flow Control. Flow control is still available using the standard Sequence Number / Acknowledge Number. This can be done by not acknowledging packets until traffic can resume.

# 11 HARDWARE CONFIGURATION

The HCI Three-Wire UART Transport uses the following configurations.

## 11.1 Wires

There are three wires used by the HCI Three-Wire UART Transport. These are Transmit, Receive, and Ground.

### 11.1.1 Transmit & receive

The transmit line from one device shall be connected to the receive line of the other device.

### 11.1.2 Ground

A common ground reference shall be used.

## 11.2 Hardware flow

Hardware flow control may be used. The signaling shall be the same as a standard RS232 flow control lines. If used, the signals shall be connected in a null-modem fashion; for example, the local RTS shall be connected to the remote CTS and vice versa.

### 11.2.1 RTS & CTS

Request to Send indicates to the remote side that the local device is able to accept more data.

Clear to Send indicates if the remote side is able to receive data.

(See ITU.T recommendations V.24 [2] and V.28 [3])

# 12 RECOMMENDED PARAMETERS

## 12.1 Timing parameters

Because this transport protocol can be used with a wide variety of baud rates, it is not possible to specify a single timing value. However, it is possible to specify the time based on the baud rate in use. If $T_{max}$ is defined as the maximum time, in seconds, it will take to transmit the largest packet over this transport, $T_{max}$ can be expressed as:

$T_{max}$ = maximum size of a packet in bits / baud rate

The maximum size of a packet in bits is either the number of bits in a 4095 octet packet (32,760) or less if required in an embedded system or as determined by the Host or Controller[1]. Thus, at a baud rate of 921,600 and the maximum packet size of 4095 octets, $T_{max}$ is: (4095×10) ÷ 921,600 = 44.434 ms.

### 12.1.1 Acknowledgment of packets

It is not necessary to acknowledge every packet with a pure acknowledgment packet if there is a data packet that will be sent soon. The recommended maximum time before starting to send an acknowledgment is 2 × $T_{max}$.

### 12.1.2 Resending reliable packets

A reliable packet shall be resent until it is acknowledged. The recommended time between starting to send the same packet is 3 × $T_{max}$.

[1]This can be determined using the HCI_Read_Buffer_Size command.

# 13 REFERENCES

[1] IETF RFC 1055: A nonstandard for transmission of IP datagrams over serial lines: SLIP – http://www.ietf.org/rfc/rfc1055.txt

[2] ITU Recommendation V.24: List of definitions for interchange circuits between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) – http://www.itu.int/rec/recommendation.asp

[3] ITU Recommendation V.28: Electrical characteristics for unbalanced double-current interchange circuits – http://www.itu.int/rec/recommendation.asp

Host Controller Interface
Part E

# HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

*This Part describes the functional specification for the Host Controller interface (HCI). The HCI provides a uniform command interface to a Controller.*

# CONTENTS

**1 Introduction ..... 1817**
1.1 Lower Layers of the Bluetooth software stack ..... 1818
1.2 Cross-version issues ..... 1819

**2 Overview of Host Controller transport layer ..... 1821**
2.1 [This section is no longer used] ..... 1821

**3 Overview of commands and events ..... 1822**
3.1 LE Controller requirements ..... 1869
3.1.1 Legacy and extended advertising ..... 1869
3.2 Underlying Support ..... 1870
3.3 Feature Exchange ..... 1871

**4 HCI flow control ..... 1872**
4.1 Host to Controller data flow control ..... 1872
4.1.1 Packet-based data flow control ..... 1873
4.1.2 Data-block-based data flow control ..... 1874
4.2 Controller to Host data flow control ..... 1875
4.3 Disconnection behavior ..... 1876
4.4 Command flow control ..... 1876
4.5 Command error handling ..... 1878
4.5.1 Generic error handling ..... 1878
4.5.2 Error handling specific to a command ..... 1879
4.6 LMP transaction and LL procedure collisions ..... 1880
4.7 LE Host and Controller synchronization ..... 1880
4.8 Versioned events ..... 1881

**5 HCI data formats ..... 1882**
5.1 Correctness ..... 1882
5.2 Data and parameter formats ..... 1882
5.3 IDs and Handles ..... 1883
5.3.1 Controller handles ..... 1883
5.3.1.1 Broadcast Connection_Handles ..... 1884
5.3.2 [This section is no longer used] ..... 1885
5.4 Exchange of HCI-specific information ..... 1885
5.4.1 HCI Command packet ..... 1885
5.4.2 HCI ACL Data packets ..... 1887
5.4.3 HCI Synchronous Data packets ..... 1889
5.4.4 HCI Event packet ..... 1890
5.4.5 HCI ISO Data packets ..... 1892

5.5 Ignored parameters ..... 1894

**6 HCI configuration parameters ..... 1895**

6.1 Scan Enable ..... 1895
6.2 Inquiry Scan Interval ..... 1895
6.3 Inquiry Scan Window ..... 1896
6.4 Inquiry Scan Type ..... 1896
6.5 Inquiry mode ..... 1896
6.6 Page Timeout ..... 1897
6.7 Connection Accept Timeout ..... 1897
6.8 Page Scan Interval ..... 1898
6.9 Page Scan Window ..... 1898
6.10 [This section is no longer used] ..... 1898
6.11 Page Scan Type ..... 1898
6.12 Voice Setting ..... 1899
6.13 PIN Type ..... 1900
6.14 Link key ..... 1900
6.15 Failed Contact Counter ..... 1900
6.16 Authentication Enable ..... 1901
6.17 Hold Mode Activity ..... 1901
6.18 Link Policy Settings ..... 1902
6.19 Flush Timeout ..... 1902
6.20 Num Broadcast Retransmissions ..... 1903
6.21 Link Supervision Timeout ..... 1903
6.22 Synchronous Flow Control Enable ..... 1904
6.23 Local Name ..... 1904
6.24 Extended Inquiry response ..... 1904
6.25 Erroneous Data Reporting ..... 1905
6.26 Class of Device ..... 1905
6.27 Supported commands ..... 1905
6.28 [This section is no longer used] ..... 1918
6.29 [This section is no longer used] ..... 1918
6.30 [This section is no longer used] ..... 1918
6.31 [This section is no longer used] ..... 1918
6.32 [This section is no longer used] ..... 1918
6.33 Flow Control mode ..... 1918
6.34 LE Supported Host ..... 1918
6.35 [This section is no longer used] ..... 1919
6.36 Sync Train Interval ..... 1919
6.37 Sync Train Timeout ..... 1919
6.38 Service Data ..... 1919
6.39 Secure Connections Host Support ..... 1920
6.40 Authenticated Payload Timeout ..... 1920

6.41 Extended Page Timeout ..... 1920
6.42 Extended Inquiry Length ..... 1921

**7 HCI commands and events ..... 1922**
7.1 Link Control commands ..... 1922
7.1.1 Inquiry command ..... 1922
7.1.2 Inquiry Cancel command ..... 1925
7.1.3 Periodic Inquiry Mode command ..... 1926
7.1.4 Exit Periodic Inquiry Mode command ..... 1929
7.1.5 Create Connection command ..... 1930
7.1.6 Disconnect command ..... 1933
7.1.7 Create Connection Cancel command ..... 1935
7.1.8 Accept Connection Request command ..... 1937
7.1.9 Reject Connection Request command ..... 1939
7.1.10 Link Key Request Reply command ..... 1940
7.1.11 Link Key Request Negative Reply command ..... 1942
7.1.12 PIN Code Request Reply command ..... 1944
7.1.13 PIN Code Request Negative Reply command ..... 1946
7.1.14 Change Connection Packet Type command ..... 1948
7.1.15 Authentication Requested command ..... 1950
7.1.16 Set Connection Encryption command ..... 1952
7.1.17 Change Connection Link Key command ..... 1954
7.1.18 Link Key Selection command ..... 1955
7.1.19 Remote Name Request command ..... 1957
7.1.20 Remote Name Request Cancel command ..... 1959
7.1.21 Read Remote Supported Features command ..... 1961
7.1.22 Read Remote Extended Features command ..... 1962
7.1.23 Read Remote Version Information command ..... 1964
7.1.24 Read Clock Offset command ..... 1965
7.1.25 Read LMP Handle command ..... 1966
7.1.26 Setup Synchronous Connection command ..... 1968
7.1.27 Accept Synchronous Connection Request command 1972
7.1.28 Reject Synchronous Connection Request command 1975
7.1.29 IO Capability Request Reply command ..... 1976
7.1.30 User Confirmation Request Reply command ..... 1979
7.1.31 User Confirmation Request Negative Reply command ..... 1980
7.1.32 User Passkey Request Reply command ..... 1981
7.1.33 User Passkey Request Negative Reply command .... 1982
7.1.34 Remote OOB Data Request Reply command ..... 1983
7.1.35 Remote OOB Data Request Negative Reply command ..... 1985
7.1.36 IO Capability Request Negative Reply command ..... 1986

7.1.37 [This section is no longer used] ..... 1988
7.1.38 [This section is no longer used] ..... 1988
7.1.39 [This section is no longer used] ..... 1988
7.1.40 [This section is no longer used] ..... 1988
7.1.41 [This section is no longer used] ..... 1988
7.1.42 [This section is no longer used] ..... 1988
7.1.43 [This section is no longer used] ..... 1988
7.1.44 [This section is no longer used] ..... 1988
7.1.45 Enhanced Setup Synchronous Connection command ..... 1989
7.1.46 Enhanced Accept Synchronous Connection Request command ..... 1999
7.1.47 Truncated Page command ..... 2006
7.1.48 Truncated Page Cancel command ..... 2008
7.1.49 Set Connectionless Peripheral Broadcast command 2010
7.1.50 Set Connectionless Peripheral Broadcast Receive command ..... 2014
7.1.51 Start Synchronization Train command ..... 2018
7.1.52 Receive Synchronization Train command ..... 2019
7.1.53 Remote OOB Extended Data Request Reply command ..... 2021
7.2 Link Policy commands ..... 2023
7.2.1 Hold Mode command ..... 2023
7.2.2 Sniff Mode command ..... 2026
7.2.3 Exit Sniff Mode command ..... 2029
7.2.4 [This section is no longer used] ..... 2030
7.2.5 [This section is no longer used] ..... 2030
7.2.6 QoS Setup command ..... 2031
7.2.7 Role Discovery command ..... 2034
7.2.8 Switch Role command ..... 2036
7.2.9 Read Link Policy Settings command ..... 2038
7.2.10 Write Link Policy Settings command ..... 2040
7.2.11 Read Default Link Policy Settings command ..... 2042
7.2.12 Write Default Link Policy Settings command ..... 2043
7.2.13 Flow Specification command ..... 2044
7.2.14 Sniff Subrating command ..... 2047
7.3 Controller & Baseband commands ..... 2050
7.3.1 Set Event Mask command ..... 2050
7.3.2 Reset command ..... 2053
7.3.3 Set Event Filter command ..... 2054
7.3.4 Flush command ..... 2061
7.3.5 Read PIN Type command ..... 2063
7.3.6 Write PIN Type command ..... 2064

7.3.7 [This section is no longer used] ..... 2065
7.3.8 Read Stored Link Key command ..... 2066
7.3.9 Write Stored Link Key command ..... 2068
7.3.10 Delete Stored Link Key command ..... 2070
7.3.11 Write Local Name command ..... 2072
7.3.12 Read Local Name command ..... 2073
7.3.13 Read Connection Accept Timeout command ..... 2074
7.3.14 Write Connection Accept Timeout command ..... 2075
7.3.15 Read Page Timeout command ..... 2076
7.3.16 Write Page Timeout command ..... 2077
7.3.17 Read Scan Enable command ..... 2078
7.3.18 Write Scan Enable command ..... 2079
7.3.19 Read Page Scan Activity command ..... 2080
7.3.20 Write Page Scan Activity command ..... 2082
7.3.21 Read Inquiry Scan Activity command ..... 2083
7.3.22 Write Inquiry Scan Activity command ..... 2085
7.3.23 Read Authentication Enable command ..... 2086
7.3.24 Write Authentication Enable command ..... 2087
7.3.25 Read Class of Device command ..... 2088
7.3.26 Write Class of Device command ..... 2089
7.3.27 Read Voice Setting command ..... 2090
7.3.28 Write Voice Setting command ..... 2091
7.3.29 Read Automatic Flush Timeout command ..... 2092
7.3.30 Write Automatic Flush Timeout command ..... 2094
7.3.31 Read Num Broadcast Retransmissions command .... 2096
7.3.32 Write Num Broadcast Retransmissions command .... 2097
7.3.33 Read Hold Mode Activity command ..... 2098
7.3.34 Write Hold Mode Activity command ..... 2099
7.3.35 Read Transmit Power Level command ..... 2100
7.3.36 Read Synchronous Flow Control Enable command .. 2102
7.3.37 Write Synchronous Flow Control Enable command .. 2103
7.3.38 Set Controller To Host Flow Control command ..... 2104
7.3.39 Host Buffer Size command ..... 2106
7.3.40 Host Number Of Completed Packets command ..... 2109
7.3.41 Read Link Supervision Timeout command ..... 2111
7.3.42 Write Link Supervision Timeout command ..... 2113
7.3.43 Read Number Of Supported IAC command ..... 2115
7.3.44 Read Current IAC LAP command ..... 2116
7.3.45 Write Current IAC LAP command ..... 2118
7.3.46 Set AFH Host Channel Classification command ..... 2120
7.3.47 Read Inquiry Scan Type command ..... 2122
7.3.48 Write Inquiry Scan Type command ..... 2123
7.3.49 Read Inquiry Mode command ..... 2124

7.3.50 Write Inquiry Mode command ..... 2125
7.3.51 Read Page Scan Type command ..... 2126
7.3.52 Write Page Scan Type command ..... 2127
7.3.53 Read AFH Channel Assessment Mode command .... 2128
7.3.54 Write AFH Channel Assessment Mode command .... 2129
7.3.55 Read Extended Inquiry Response command ..... 2131
7.3.56 Write Extended Inquiry Response command ..... 2132
7.3.57 Refresh Encryption Key command ..... 2134
7.3.58 Read Simple Pairing Mode command ..... 2135
7.3.59 Write Simple Pairing Mode command ..... 2136
7.3.60 Read Local OOB Data command ..... 2137
7.3.61 Read Inquiry Response Transmit Power Level command ..... 2139
7.3.62 Write Inquiry Transmit Power Level command ..... 2140
7.3.63 Send Keypress Notification command ..... 2141
7.3.64 Read Default Erroneous Data Reporting command .. 2143
7.3.65 Write Default Erroneous Data Reporting command .. 2144
7.3.66 Enhanced Flush command ..... 2145
7.3.67 [This section is no longer used] ..... 2147
7.3.68 [This section is no longer used] ..... 2147
7.3.69 Set Event Mask Page 2 command ..... 2148
7.3.70 [This section is no longer used] ..... 2150
7.3.71 [This section is no longer used] ..... 2150
7.3.72 Read Flow Control Mode command ..... 2151
7.3.73 Write Flow Control Mode command ..... 2152
7.3.74 Read Enhanced Transmit Power Level command .... 2153
7.3.75 [This section is no longer used] ..... 2155
7.3.76 [This section is no longer used] ..... 2155
7.3.77 [This section is no longer used] ..... 2155
7.3.78 Read LE Host Support command ..... 2156
7.3.79 Write LE Host Support command ..... 2157
7.3.80 Set MWS Channel Parameters command ..... 2158
7.3.81 Set External Frame Configuration command ..... 2160
7.3.82 Set MWS Signaling command ..... 2163
7.3.83 Set MWS Transport Layer command ..... 2168
7.3.84 Set MWS Scan Frequency Table command ..... 2169
7.3.85 Set MWS_PATTERN Configuration command ..... 2171
7.3.86 Set Reserved LT_ADDR command ..... 2174
7.3.87 Delete Reserved LT_ADDR command ..... 2176
7.3.88 Set Connectionless Peripheral Broadcast Data command ..... 2178
7.3.89 Read Synchronization Train Parameters command .. 2181
7.3.90 Write Synchronization Train Parameters command .. 2183

7.3.91 Read Secure Connections Host Support command .. 2185
7.3.92 Write Secure Connections Host Support command .. 2186
7.3.93 Read Authenticated Payload Timeout command ..... 2188
7.3.94 Write Authenticated Payload Timeout command ..... 2190
7.3.95 Read Local OOB Extended Data command ..... 2192
7.3.96 Read Extended Page Timeout command ..... 2194
7.3.97 Write Extended Page Timeout command ..... 2195
7.3.98 Read Extended Inquiry Length command ..... 2196
7.3.99 Write Extended Inquiry Length command ..... 2197
7.3.100 Set Ecosystem Base Interval command ..... 2198
7.3.101 Configure Data Path command ..... 2200
7.3.102 Set Min Encryption Key Size command ..... 2202
7.4 Informational parameters ..... 2204
7.4.1 Read Local Version Information command ..... 2204
7.4.2 Read Local Supported Commands command ..... 2206
7.4.3 Read Local Supported Features command ..... 2208
7.4.4 Read Local Extended Features command ..... 2209
7.4.5 Read Buffer Size command ..... 2211
7.4.6 Read BD_ADDR command ..... 2214
7.4.7 Read Data Block Size command ..... 2215
7.4.8 Read Local Supported Codecs command ..... 2217
7.4.9 Read Local Simple Pairing Options command ..... 2220
7.4.10 Read Local Supported Codec Capabilities command ..... 2222
7.4.11 Read Local Supported Controller Delay command ... 2224
7.5 Status parameters ..... 2227
7.5.1 Read Failed Contact Counter command ..... 2227
7.5.2 Reset Failed Contact Counter command ..... 2229
7.5.3 Read Link Quality command ..... 2230
7.5.4 Read RSSI command ..... 2232
7.5.5 Read AFH Channel Map command ..... 2234
7.5.6 Read Clock command ..... 2236
7.5.7 Read Encryption Key Size command ..... 2238
7.5.8 [This section is no longer used] ..... 2240
7.5.9 [This section is no longer used] ..... 2240
7.5.10 [This section is no longer used] ..... 2240
7.5.11 Get MWS Transport Layer Configuration command .. 2241
7.5.12 Set Triggered Clock Capture command ..... 2244
7.6 Testing commands ..... 2247
7.6.1 Read Loopback Mode command ..... 2247
7.6.2 Write Loopback Mode command ..... 2249
7.6.3 Enable Implementation Under Test Mode command . 2252
7.6.4 Write Simple Pairing Debug Mode command ..... 2253

7.6.5 [This section is no longer used] ..... 2255
7.6.6 [This section is no longer used] ..... 2255
7.6.7 [This section is no longer used] ..... 2255
7.6.8 Write Secure Connections Test Mode command ..... 2256
7.7 Events ..... 2260
7.7.1 Inquiry Complete event ..... 2260
7.7.2 Inquiry Result event ..... 2261
7.7.3 Connection Complete event ..... 2263
7.7.4 Connection Request event ..... 2265
7.7.5 Disconnection Complete event ..... 2267
7.7.6 Authentication Complete event ..... 2269
7.7.7 Remote Name Request Complete event ..... 2270
7.7.8 Encryption Change event ..... 2271
7.7.9 Change Connection Link Key Complete event ..... 2273
7.7.10 Link Key Type Changed event ..... 2274
7.7.11 Read Remote Supported Features Complete event . 2276
7.7.12 Read Remote Version Information Complete event .. 2277
7.7.13 QoS Setup Complete event ..... 2279
7.7.14 Command Complete event ..... 2281
7.7.15 Command Status event ..... 2283
7.7.16 Hardware Error event ..... 2285
7.7.17 Flush Occurred event ..... 2286
7.7.18 Role Change event ..... 2287
7.7.19 Number Of Completed Packets event ..... 2288
7.7.20 Mode Change event ..... 2290
7.7.21 Return Link Keys event ..... 2292
7.7.22 PIN Code Request event ..... 2293
7.7.23 Link Key Request event ..... 2294
7.7.24 Link Key Notification event ..... 2295
7.7.25 Loopback Command event ..... 2297
7.7.26 Data Buffer Overflow event ..... 2298
7.7.27 Max Slots Change event ..... 2299
7.7.28 Read Clock Offset Complete event ..... 2300
7.7.29 Connection Packet Type Changed event ..... 2301
7.7.30 QoS Violation event ..... 2303
7.7.31 Page Scan Repetition Mode Change event ..... 2304
7.7.32 Flow Specification Complete event ..... 2305
7.7.33 Inquiry Result with RSSI event ..... 2308
7.7.34 Read Remote Extended Features Complete event ... 2310
7.7.35 Synchronous Connection Complete event ..... 2312
7.7.36 Synchronous Connection Changed event ..... 2315
7.7.37 Sniff Subrating event ..... 2317
7.7.38 Extended Inquiry Result event ..... 2319

7.7.39 Encryption Key Refresh Complete event ..... 2322
7.7.40 IO Capability Request event ..... 2323
7.7.41 IO Capability Response event ..... 2324
7.7.42 User Confirmation Request event ..... 2326
7.7.43 User Passkey Request event ..... 2327
7.7.44 Remote OOB Data Request event ..... 2328
7.7.45 Simple Pairing Complete event ..... 2329
7.7.46 Link Supervision Timeout Changed event ..... 2330
7.7.47 Enhanced Flush Complete event ..... 2331
7.7.48 User Passkey Notification event ..... 2332
7.7.49 Keypress Notification event ..... 2333
7.7.50 Remote Host Supported Features Notification event 2334
7.7.51 [This section is no longer used] ..... 2335
7.7.52 [This section is no longer used] ..... 2335
7.7.53 [This section is no longer used] ..... 2335
7.7.54 [This section is no longer used] ..... 2335
7.7.55 [This section is no longer used] ..... 2335
7.7.56 [This section is no longer used] ..... 2335
7.7.57 [This section is no longer used] ..... 2335
7.7.58 [This section is no longer used] ..... 2335
7.7.59 Number Of Completed Data Blocks event ..... 2336
7.7.60 [This section is no longer used] ..... 2338
7.7.61 [This section is no longer used] ..... 2338
7.7.62 [This section is no longer used] ..... 2338
7.7.63 [This section is no longer used] ..... 2338
7.7.64 [This section is no longer used] ..... 2338
7.7.65 LE Meta event ..... 2339
7.7.65.1 LE Connection Complete event ..... 2339
7.7.65.2 LE Advertising Report event ..... 2342
7.7.65.3 LE Connection Update Complete event ..... 2345
7.7.65.4 LE Read Remote Features Page 0 Complete event ..... 2347
7.7.65.5 LE Long Term Key Request event ..... 2349
7.7.65.6 LE Remote Connection Parameter Request event ..... 2350
7.7.65.7 LE Data Length Change event ..... 2352
7.7.65.8 LE Read Local P-256 Public Key Complete event ..... 2354
7.7.65.9 LE Generate DHKey Complete event ..... 2355
7.7.65.10 LE Enhanced Connection Complete event 2356
7.7.65.11 LE Directed Advertising Report event ..... 2361
7.7.65.12 LE PHY Update Complete event ..... 2363
7.7.65.13 LE Extended Advertising Report event ..... 2365

7.7.65.14 LE Periodic Advertising Sync Established event ..... 2371
7.7.65.15 LE Periodic Advertising Report event ..... 2376
7.7.65.16 LE Periodic Advertising Sync Lost event ... 2380
7.7.65.17 LE Scan Timeout event ..... 2381
7.7.65.18 LE Advertising Set Terminated event ..... 2382
7.7.65.19 LE Scan Request Received event ..... 2384
7.7.65.20 LE Channel Selection Algorithm event ..... 2386
7.7.65.21 LE Connectionless IQ Report event ..... 2387
7.7.65.22 LE Connection IQ Report event ..... 2391
7.7.65.23 LE CTE Request Failed event ..... 2395
7.7.65.24 LE Periodic Advertising Sync Transfer Received event ..... 2396
7.7.65.25 LE CIS Established event ..... 2401
7.7.65.26 LE CIS Request event ..... 2408
7.7.65.27 LE Create BIG Complete event ..... 2410
7.7.65.28 LE Terminate BIG Complete event ..... 2414
7.7.65.29 LE BIG Sync Established event ..... 2415
7.7.65.30 LE BIG Sync Lost event ..... 2418
7.7.65.31 LE Request Peer SCA Complete event ..... 2420
7.7.65.32 LE Path Loss Threshold event ..... 2422
7.7.65.33 LE Transmit Power Reporting event ..... 2424
7.7.65.34 LE BIGInfo Advertising Report event ..... 2427
7.7.65.35 LE Subrate Change event ..... 2431
7.7.65.36 LE Periodic Advertising Subevent Data Request event ..... 2433
7.7.65.37 LE Periodic Advertising Response Report event ..... 2435
7.7.65.38 LE Read All Remote Features Complete event ..... 2438
7.7.65.39 LE CS Read Remote Supported Capabilities Complete event ..... 2440
7.7.65.40 LE CS Read Remote FAE Table Complete event ..... 2451
7.7.65.41 LE CS Security Enable Complete event .... 2452
7.7.65.42 LE CS Config Complete event ..... 2453
7.7.65.43 LE CS Procedure Enable Complete event . 2460
7.7.65.44 LE CS Subevent Result event ..... 2464
7.7.65.45 LE CS Subevent Result Continue event .... 2477
7.7.65.46 LE CS Test End Complete event ..... 2481
7.7.65.47 LE Monitored Advertisers Report event ..... 2482
7.7.65.48 LE Frame Space Update Complete event . 2484
7.7.65.49 LE UTP Receive event ..... 2486

7.7.65.50 LE Connection Rate Change event ..... 2487
7.7.66 Triggered Clock Capture event ..... 2489
7.7.67 Synchronization Train Complete event ..... 2491
7.7.68 Synchronization Train Received event ..... 2492
7.7.69 Connectionless Peripheral Broadcast Receive event 2494
7.7.70 Connectionless Peripheral Broadcast Timeout event 2496
7.7.71 Truncated Page Complete event ..... 2497
7.7.72 Peripheral Page Response Timeout event ..... 2498
7.7.73 Connectionless Peripheral Broadcast Channel Map Change event ..... 2499
7.7.74 Inquiry Response Notification event ..... 2500
7.7.75 Authenticated Payload Timeout Expired event ..... 2501
7.7.76 SAM Status Change event ..... 2502
7.8 LE Controller commands ..... 2504
7.8.1 LE Set Event Mask command ..... 2504
7.8.2 LE Read Buffer Size command ..... 2508
7.8.3 LE Read Local Supported Features Page 0 command ..... 2511
7.8.4 LE Set Random Address command ..... 2512
7.8.5 LE Set Advertising Parameters command ..... 2514
7.8.6 LE Read Advertising Physical Channel Tx Power command ..... 2518
7.8.7 LE Set Advertising Data command ..... 2519
7.8.8 LE Set Scan Response Data command ..... 2521
7.8.9 LE Set Advertising Enable command ..... 2523
7.8.10 LE Set Scan Parameters command ..... 2525
7.8.11 LE Set Scan Enable command ..... 2528
7.8.12 LE Create Connection command ..... 2530
7.8.13 LE Create Connection Cancel command ..... 2536
7.8.14 LE Read Filter Accept List Size command ..... 2537
7.8.15 LE Clear Filter Accept List command ..... 2538
7.8.16 LE Add Device To Filter Accept List command ..... 2539
7.8.17 LE Remove Device From Filter Accept List command ..... 2541
7.8.18 LE Connection Update command ..... 2543
7.8.19 LE Set Host Channel Classification command ..... 2546
7.8.20 LE Read Channel Map command ..... 2547
7.8.21 LE Read Remote Features Page 0 command ..... 2549
7.8.22 LE Encrypt command ..... 2550
7.8.23 LE Rand command ..... 2552
7.8.24 LE Enable Encryption command ..... 2553
7.8.25 LE Long Term Key Request Reply command ..... 2555

7.8.26 LE Long Term Key Request Negative Reply command ..... 2557
7.8.27 LE Read Supported States command ..... 2558
7.8.28 LE Receiver Test command ..... 2562
7.8.29 LE Transmitter Test command ..... 2566
7.8.30 LE Test End command ..... 2571
7.8.31 LE Remote Connection Parameter Request Reply command ..... 2572
7.8.32 LE Remote Connection Parameter Request Negative Reply command ..... 2576
7.8.33 LE Set Data Length command ..... 2578
7.8.34 LE Read Suggested Default Data Length command . 2580
7.8.35 LE Write Suggested Default Data Length command . 2582
7.8.36 LE Read Local P-256 Public Key command ..... 2583
7.8.37 LE Generate DHKey command ..... 2584
7.8.38 LE Add Device To Resolving List command ..... 2586
7.8.39 LE Remove Device From Resolving List command .. 2589
7.8.40 LE Clear Resolving List command ..... 2591
7.8.41 LE Read Resolving List Size command ..... 2592
7.8.42 LE Read Peer Resolvable Address command ..... 2593
7.8.43 LE Read Local Resolvable Address command ..... 2595
7.8.44 LE Set Address Resolution Enable command ..... 2597
7.8.45 LE Set Resolvable Private Address Timeout command ..... 2599
7.8.46 LE Read Maximum Data Length command ..... 2601
7.8.47 LE Read PHY command ..... 2603
7.8.48 LE Set Default PHY command ..... 2605
7.8.49 LE Set PHY command ..... 2607
7.8.50 [This section is no longer used] ..... 2610
7.8.51 [This section is no longer used] ..... 2610
7.8.52 LE Set Advertising Set Random Address command . 2611
7.8.53 LE Set Extended Advertising Parameters command 2613
7.8.54 LE Set Extended Advertising Data command ..... 2622
7.8.55 LE Set Extended Scan Response Data command .... 2626
7.8.56 LE Set Extended Advertising Enable command ..... 2629
7.8.57 LE Read Maximum Advertising Data Length command ..... 2634
7.8.58 LE Read Number of Supported Advertising Sets command ..... 2635
7.8.59 LE Remove Advertising Set command ..... 2636
7.8.60 LE Clear Advertising Sets command ..... 2637
7.8.61 LE Set Periodic Advertising Parameters command ... 2638
7.8.62 LE Set Periodic Advertising Data command ..... 2643

7.8.63 LE Set Periodic Advertising Enable command ..... 2646
7.8.64 LE Set Extended Scan Parameters command ..... 2648
7.8.65 LE Set Extended Scan Enable command ..... 2652
7.8.66 LE Extended Create Connection command ..... 2656
7.8.67 LE Periodic Advertising Create Sync command ..... 2665
7.8.68 LE Periodic Advertising Create Sync Cancel command ..... 2670
7.8.69 LE Periodic Advertising Terminate Sync command ... 2671
7.8.70 LE Add Device To Periodic Advertiser List command 2672
7.8.71 LE Remove Device From Periodic Advertiser List command ..... 2674
7.8.72 LE Clear Periodic Advertiser List command ..... 2676
7.8.73 LE Read Periodic Advertiser List Size command ..... 2677
7.8.74 LE Read Transmit Power command ..... 2678
7.8.75 LE Read RF Path Compensation command ..... 2679
7.8.76 LE Write RF Path Compensation command ..... 2680
7.8.77 LE Set Privacy Mode command ..... 2682
7.8.78 [This section is no longer used] ..... 2684
7.8.79 [This section is no longer used] ..... 2684
7.8.80 LE Set Connectionless CTE Transmit Parameters command ..... 2685
7.8.81 LE Set Connectionless CTE Transmit Enable command ..... 2688
7.8.82 LE Set Connectionless IQ Sampling Enable command ..... 2690
7.8.83 LE Set Connection CTE Receive Parameters command ..... 2693
7.8.84 LE Set Connection CTE Transmit Parameters command ..... 2696
7.8.85 LE Connection CTE Request Enable command ..... 2699
7.8.86 LE Connection CTE Response Enable command ..... 2702
7.8.87 LE Read Antenna Information command ..... 2704
7.8.88 LE Set Periodic Advertising Receive Enable command ..... 2706
7.8.89 LE Periodic Advertising Sync Transfer command ..... 2708
7.8.90 LE Periodic Advertising Set Info Transfer command . 2710
7.8.91 LE Set Periodic Advertising Sync Transfer Parameters command ..... 2712
7.8.92 LE Set Default Periodic Advertising Sync Transfer Parameters command ..... 2716
7.8.93 [This section is no longer used] ..... 2719
7.8.94 LE Modify Sleep Clock Accuracy command ..... 2720
7.8.95 [This section is no longer used] ..... 2722

7.8.96 LE Read ISO TX Sync command ..... 2723
7.8.97 LE Set CIG Parameters command ..... 2725
7.8.98 LE Set CIG Parameters Test command ..... 2733
7.8.99 LE Create CIS command ..... 2742
7.8.100 LE Remove CIG command ..... 2745
7.8.101 LE Accept CIS Request command ..... 2747
7.8.102 LE Reject CIS Request command ..... 2749
7.8.103 LE Create BIG command ..... 2751
7.8.104 LE Create BIG Test command ..... 2756
7.8.105 LE Terminate BIG command ..... 2762
7.8.106 LE BIG Create Sync command ..... 2764
7.8.107 LE BIG Terminate Sync command ..... 2768
7.8.108 LE Request Peer SCA command ..... 2770
7.8.109 LE Setup ISO Data Path command ..... 2772
7.8.110 LE Remove ISO Data Path command ..... 2776
7.8.111 LE ISO Transmit Test command ..... 2778
7.8.112 LE ISO Receive Test command ..... 2780
7.8.113 LE ISO Read Test Counters command ..... 2782
7.8.114 LE ISO Test End command ..... 2784
7.8.115 LE Set Host Feature command ..... 2786
7.8.116 LE Read ISO Link Quality command ..... 2788
7.8.117 LE Enhanced Read Transmit Power Level command ..... 2791
7.8.118 LE Read Remote Transmit Power Level command ... 2794
7.8.119 LE Set Path Loss Reporting Parameters command .. 2796
7.8.120 LE Set Path Loss Reporting Enable command ..... 2799
7.8.121 LE Set Transmit Power Reporting Enable command 2801
7.8.122 LE Set Data Related Address Changes command ... 2803
7.8.123 LE Set Default Subrate command ..... 2805
7.8.124 LE Subrate Request command ..... 2808
7.8.125 LE Set Periodic Advertising Subevent Data command ..... 2812
7.8.126 LE Set Periodic Advertising Response Data command ..... 2816
7.8.127 LE Set Periodic Sync Subevent command ..... 2819
7.8.128 LE Read All Local Supported Features command .... 2821
7.8.129 LE Read All Remote Features command ..... 2822
7.8.130 LE CS Read Local Supported Capabilities command ..... 2824
7.8.131 LE CS Read Remote Supported Capabilities command ..... 2835
7.8.132 LE CS Write Cached Remote Supported Capabilities command ..... 2837

7.8.133 LE CS Security Enable command ..... 2849
7.8.134 LE CS Set Default Settings command ..... 2850
7.8.135 LE CS Read Remote FAE Table command ..... 2853
7.8.136 LE CS Write Cached Remote FAE Table command .. 2855
7.8.137 LE CS Create Config command ..... 2857
7.8.138 LE CS Remove Config command ..... 2864
7.8.139 LE CS Set Channel Classification command ..... 2866
7.8.140 LE CS Set Procedure Parameters command ..... 2868
7.8.141 LE CS Procedure Enable command ..... 2874
7.8.142 LE CS Test command ..... 2876
7.8.143 LE CS Test End command ..... 2892
7.8.144 LE Set Decision Data command ..... 2893
7.8.145 LE Set Decision Instructions command ..... 2895
7.8.146 LE Add Device To Monitored Advertisers List command ..... 2904
7.8.147 LE Remove Device From Monitored Advertisers List command ..... 2907
7.8.148 LE Clear Monitored Advertisers List command ..... 2909
7.8.149 LE Enable Monitoring Advertisers command ..... 2910
7.8.150 LE Read Monitored Advertisers List Size command . 2912
7.8.151 LE Frame Space Update command ..... 2913
7.8.152 LE Enable UTP OTA mode ..... 2916
7.8.153 LE UTP Send command ..... 2917
7.8.154 LE Connection Rate Request command ..... 2918
7.8.155 LE Set Default Rate Parameters command ..... 2923
7.8.156 LE Read Minimum Supported Connection Interval command ..... 2927
7.8.157 LE CS Set Security Requirements ..... 2930
7.8.158 CS Set Default Security Requirements ..... 2932

**Appendix A [This Appendix is no longer used] ..... 2934**

**Appendix B Removed commands and events ..... 2935**

# 1 INTRODUCTION

This Part of the specification describes the functional specifications for the Host Controller interface (HCI). The HCI provides a uniform command method for the Host to access Controller capabilities and to control connections to other Controllers. For the BR/EDR or LE Controller, these commands typically involve the Link Manager (LM) to exchange LMP commands or the Link Layer (LL) to exchange LL Control packets with remote Bluetooth devices. For details, see [Vol 2] Part C and [Vol 6] Part B.

The HCI layer spans the boundary between the Host and Controller. As a result, HCI has two roles: the Upper HCI, used by and residing within the Host, and the Lower HCI, used by and residing within the Controller.

The rest of this section provides a brief overview of the lower layers of the Bluetooth software stack and of the Bluetooth hardware. Section 2 provides an overview of the Host Controller Transport Layer. Section 3 lists the HCI commands and events and specifies the support requirements on different types of implementation. Section 4 describes the flow control used between the Host and the Controller. Section 5 describes the various data formats used by HCI. Section 6 further describes certain parameters that are common to several commands. Section 7 describes each of the HCI commands in detail, identifies parameters for each of the commands, and lists events associated with each command.

The specification is applicable to the following types of Controllers:

- BR/EDR Controller
- BR/EDR/LE Controller
- LE Controller

In the following sections, the term “BR/EDR Controller” is used to describe the BR/EDR functionality of a Controller which may be either a BR/EDR only Controller or a BR/EDR/LE Controller. Similarly, the term “LE Controller” is used to describe the LE functionality of a Controller which may be either an LE only Controller or a BR/EDR/LE Controller.

## 1.1 Lower Layers of the Bluetooth software stack

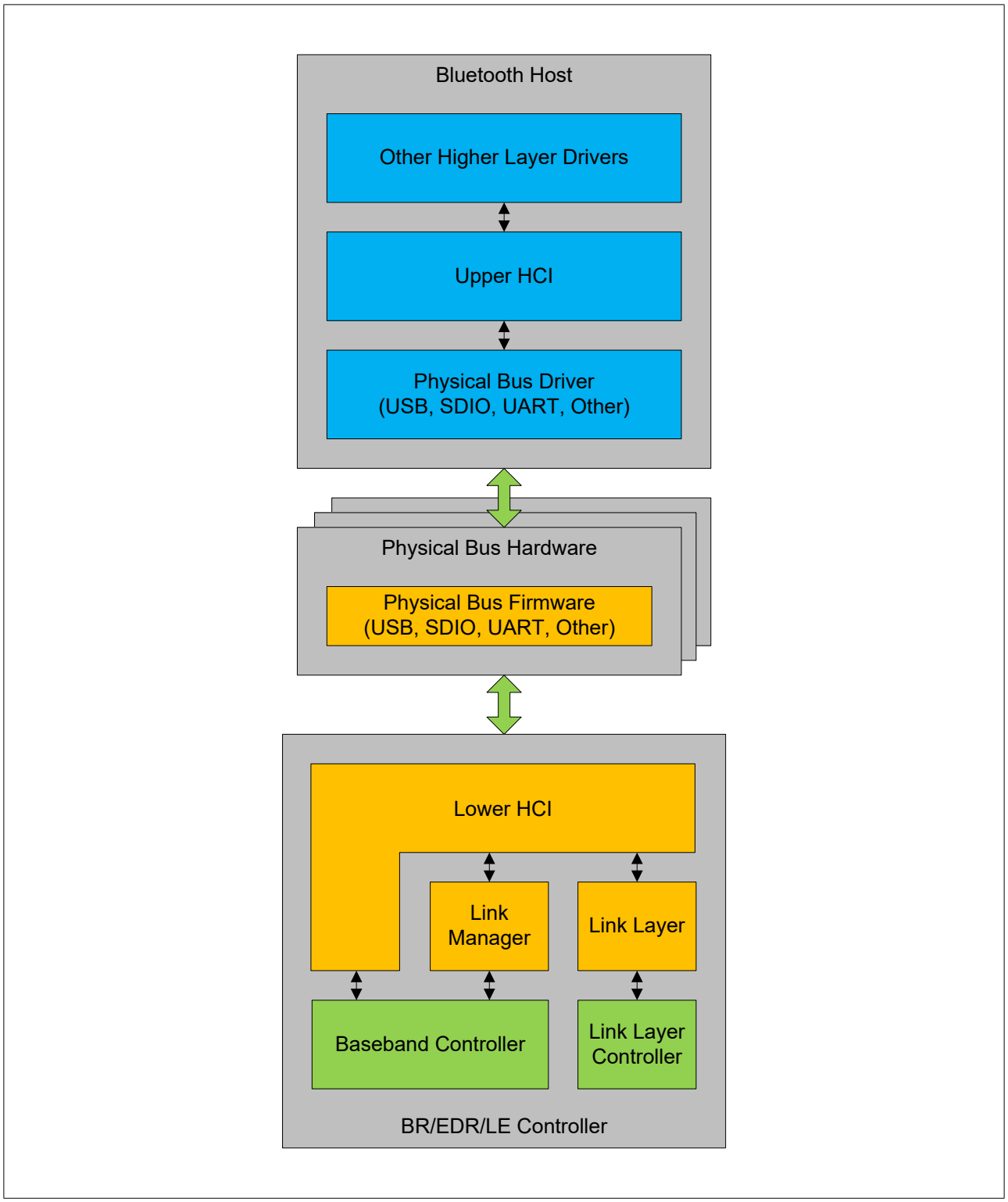



*Figure 1.1: Overview of the lower software layers*

Figure 1.1 provides an overview of the lower software layers.

Several layers may exist between the Upper HCI and the Lower HCI. These intermediate layers, collectively the "Host Controller Transport Layer", provide the ability to transfer data without intimate knowledge of the data.

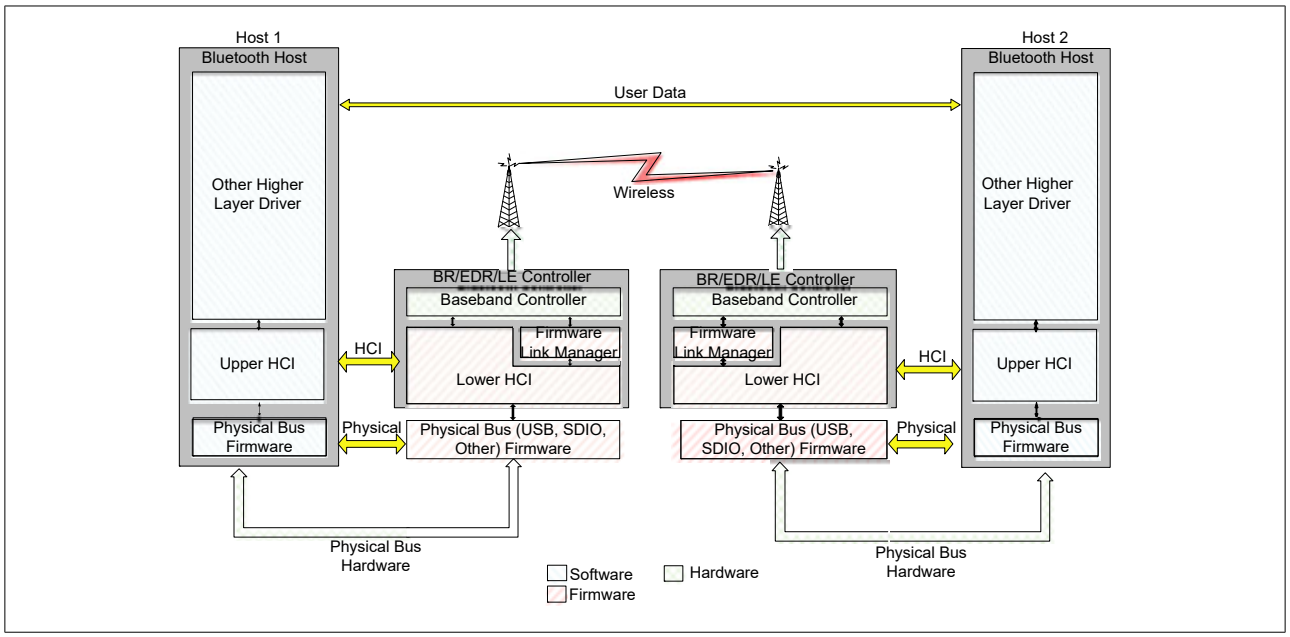



*Figure 1.2: End to end overview of lower software layers to transfer data*

Figure 1.2 illustrates the path of a data transfer from one device to another. The Upper HCI on the Host exchanges data and commands with the Lower HCI on the Bluetooth hardware. The Host Controller Transport Layer provides both HCI roles with the ability to exchange information with each other.

The Host will receive asynchronous notifications of HCI events independent of which Host Controller Transport Layer is used. HCI events are used for notifying the Host when something occurs. When the Host discovers that an event has occurred it will then parse the received event packet to determine which event occurred.

A BR/EDR/LE Controller uses one shared command buffer and flow control for BR/EDR and LE. Data buffers can be either shared between BR/EDR and LE or there may be separate data buffers for BR/EDR and LE. The configuration of a Controller is determined through HCI.

## 1.2 Cross-version issues

The Host and Controller communicating through HCI are not required to support the same version of this specification (see [Vol 0] Part D, Section 4).

If the Controller conforms to a newer version of the specification than the Host, events (including HCI_Command_Status and HCI_Command_Complete) may include values

that are reserved for future use in the version supported by the Host. In the case of the Status parameter of the HCI_Command_Status and HCI_Command_Complete events, a value that is reserved for future use must be treated as if it were the error code *Unspecified Error* (0x1F) (see [Vol 1] Part F, Section 1.3). In all other circumstances the Host must obey the requirements for devices receiving reserved values (see [Vol 1] Part E, Section 2.4).

Note: If the Host conforms to a newer version of the specification than the Controller and uses a command feature that is not part of the older specification version, then the Controller will report an error. Failure to handle such errors could result in unexpected behavior.

# 2 OVERVIEW OF HOST CONTROLLER TRANSPORT LAYER

The Host driver stack has a transport layer between the Host Controller Interface driver and the Host.

The main goal of this transport layer is transparency. The Host Controller Interface driver (which interfaces to the Controller) should be independent of the underlying transport technology. In addition, the transport should not require any understanding of the data that the Host Controller Interface driver passes to the Controller. This allows the logical interface (HCI) or the Controller to be upgraded without affecting the transport layer.

The specified Host Controller Transport Layers are described in the other parts of Volume 4.

## 2.1 [This section is no longer used]

# 3 OVERVIEW OF COMMANDS AND EVENTS

The commands and events are sent between the Host and the Controller.

Table 3.1 lists each HCI command and event together with specification version information, a summary description, and the support requirements.

A command or event may have more than one version. All versions of a command or event implement the same basic functionality but with detail differences, such as additional parameters relating to newer features. The different versions are indicated by "[v1]", etc. where necessary; if no version number is given, a reference to the command or event applies to all versions.

The specification version information gives the version number of the specification which first specified this command or event or, where more than one specification version number is shown, each version of this command or event.

Table 3.1 lists the requirements for a Controller to support each command or event or version thereof. Subject to Section 3.2, a Controller shall support the command or event if it is shown as mandatory for at least one of the transports (i.e., BR/EDR or LE) that the Controller supports, otherwise the Controller may support the command or event if it is shown as optional for at least one of the transports that the Controller supports, otherwise it shall not support the command or event. If the command or event has more than one version, then this determination is made for each version separately using the requirements listed in the table for that version. The LMP features mentioned in the conditions are defined in [Vol 2] Part C, Section 3.2 and the Link Layer features in [Vol 6] Part B, Section 4.6.

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Accept Connection Request command | 1.1 | This command is used to accept a new incoming BR/EDR connection request. | M | E |
| Accept Synchronous Connection Request command | 1.2 | This command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection. | C.134 | E |
| Authenticated Payload Timeout Expired event | 4.1 | This event is used to indicate that a packet containing a valid MIC on the Handle was not received within the *authenticatedPayloadTO*. | C.155 | C.155 |
| Authentication Complete event | 1.1 | This event occurs when authentication has been completed for the specified connection. | C.101 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Authentication Requested command | 1.1 | This command is used to establish authentication between the two devices associated with the specified Connection_Handle. | O | E |
| Change Connection Link Key command | 1.1 | This command is used to force both devices of a connection associated to the Connection_Handle, to generate a new link key. | O | E |
| Change Connection Link Key Complete event | 1.1 | This event is used to indicate that the change in the Link Key for the Connection_Handle specified by the Connection_Handle parameter had been completed. | C.102 | E |
| Change Connection Packet Type command | 1.1 | This command is used to change which packet types can be used for a connection that is currently established. | C.133 | E |
| Command Complete event | 1.1 | This event is used by the Controller to pass the return status of a command and the other parameters for each HCI command. | M | M |
| Command Status event | 1.1 | This event is used to indicate that the command described by the Command_Opcode parameter has been received and the Controller is currently performing the task for this command. | M | M |
| Configure Data Path command | 5.2 | This command is used by a Host to configure a data path to enable codec operation in the Controller. | C.156 | C.156 |
| Connection Complete event | 1.1 | This event indicates to both of the Hosts forming the connection that a new BR/EDR connection has been established. | M | E |
| Connection Packet Type Changed event | 1.1 | This event is used to indicate the completion of the process of the Link Manager changing the packet type mask used for the specified Connection_Handle. | C.133 | E |
| Connection Request event | 1.1 | This event is used to indicate that a new incoming BR/EDR connection is trying to be established. | M | E |
| Connectionless Peripheral Broadcast Channel Map Change event | CSA4 | This event indicates to the Host that the BR/EDR Controller has moved to a new AFH channel map for the PBD logical link. | C.201 | E |
| Connectionless Peripheral Broadcast Receive event | CSA4 | This event provides the Host with the data received from a Connectionless Peripheral Broadcast packet. | C.202 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Connectionless Peripheral Broadcast Timeout event | CSA4 | On the Connectionless Peripheral Broadcast Receiver, this event indicates to the Host that the BR/EDR Controller has lost synchronization with the Connectionless Peripheral Broadcast Transmitter.<br>On the Connectionless Peripheral Broadcast Transmitter, this event indicates to the Host that the BR/EDR Controller has been unable to transmit a Connectionless Peripheral Broadcast packet for the timeout interval specified in the HCI_Set_Connectionless_Peripheral_-Broadcast command. | C.202 | E |
| Create Connection Cancel command | 1.2 | This command is used to cancel an ongoing Create Connection. | M | E |
| Create Connection command | 1.1 | This command will cause the BR/EDR Link Manager to create an ACL connection to the BR/EDR Controller with the BD_ADDR specified by the parameters. | M | E |
| Data Buffer Overflow event | 1.1 | This event is used to indicate that the Controller's data buffers have overflowed, because the Host has sent more packets than allowed. | O | O |
| Delete Reserved LT_ADDR command | CSA4 | This command requests that the BR/EDR Controller cancel the reservation of a specific LT_ADDR reserved for the purposes of Connectionless Peripheral Broadcast. | C.201 | E |
| Delete Stored Link Key command | 1.1 | This command provides the ability to remove one or more of the link keys stored in the Controller. | C.121 | E |
| Disconnect command | 1.1 | This command is used to terminate an existing BR/EDR or LE connection. | M | C.3 |
| Disconnection Complete event | 1.1 | This event occurs when a connection has been terminated. | M | C.3 |
| Enable Implementation Under Test Mode command | 1.1 | This command will allow the local Controller to enter test mode via LMP test commands. The Host issues this command to request the local device to be the IUT for the Testing scenarios as described in the Bluetooth Test Mode document. | C.123 | E |
| Encryption Change event | 1.1<br>5.3 | This event is used to indicate that the change in encryption has been completed for the specified Connection_Handle. | [v1]<br>M<br>[v2]<br>C.158 | [v1]<br>C.4<br>[v2]<br>C.56 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Encryption Key Refresh Complete event | 2.1 + EDR | This event is used to indicate to the Host that the encryption key was refreshed on the given Connection_Handle any time encryption is paused and then resumed. | M | C.4 |
| Enhanced Accept Synchronous Connection Request command | CSA2 | This command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection. | C.135 | E |
| Enhanced Flush command | 2.1 + EDR | This command is used to discard specific packets currently pending for transmission in the Controller for the specified Handle. This command takes a parameter specifying the type of packets to be flushed. | M | E |
| Enhanced Flush Complete event | 2.1 + EDR | This event is used to indicate that an Enhanced Flush is complete. | M | E |
| Enhanced Setup Synchronous Connection command | CSA2 | This command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. | C.135 | E |
| Exit Periodic Inquiry Mode command | 1.1 | This command is used to end the Periodic Inquiry mode when the local device is in Periodic Inquiry Mode. | C.103 | E |
| Exit Sniff Mode command | 1.1 | This command is used to end Sniff mode for a Connection_Handle which is currently in Sniff mode. | C.214 | E |
| Extended Inquiry Result event | 2.1 + EDR | This event indicates that a BR/EDR Controller has responded with an extended inquiry response during the current Inquiry process. | C.147 | E |
| Flow Specification command | 1.2 | This command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the Connection_Handle. | M | E |
| Flow Specification Complete event | 1.2 | This event is used to inform the Host about the Quality of Service for the ACL connection the Controller is able to support. | M | E |
| Flush command | 1.1 | This command is used to discard all data that is currently pending for transmission in the Controller for the specified Connection_Handle. | M | E |
| Flush Occurred event | 1.1 | This event is used to indicate that, for the specified Handle, the data to be transmitted has been discarded. | M | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Get MWS Transport Layer Configuration command | CSA3 | This command reads the supported baud rates from the Controller. | C.109 | C.109 |
| Hardware Error event | 1.1 | This event is used to indicate some type of hardware failure for the Controller. | O | O |
| Hold Mode command | 1.1 | This command is used to initiate Hold mode. | C.213 | E |
| Host Buffer Size command | 1.1 | This command is used by the Host to notify the Controller about its buffer sizes for ACL and synchronous data. The Controller will fragment the data to be transmitted from the Controller to the Host, so that data contained in HCI Data packets will not exceed these sizes. | C.107 | C.107 |
| Host Number Of Completed Packets command | 1.1 | This command is used by the Host to indicate to the Controller when the Host is ready to receive more HCI packets for any Connection_Handle. | C.107 | C.107 |
| Inquiry Cancel command | 1.1 | This command will cause the BR/EDR Controller to stop the current Inquiry if the BR/EDR Controller is in Inquiry Mode. | C.127 | E |
| Inquiry command | 1.1 | This command will cause the BR/EDR Controller to enter Inquiry Mode. Inquiry Mode is used to discovery other nearby BR/EDR Controllers. | C.127 | E |
| Inquiry Complete event | 1.1 | This event indicates that the Inquiry is finished. | C.127 | E |
| Inquiry Response Notification event | CSA4 | This event indicates to the Host that the BR/EDR Controller responded to an inquiry message. | C.126 | E |
| Inquiry Result event | 1.1 | This event indicates that a BR/EDR Controller or multiple BR/EDR Controllers have responded so far during the current Inquiry process. | C.127 | E |
| Inquiry Result with RSSI event | 1.2 | This event indicates that a BR/EDR Controller or multiple BR/EDR Controllers have responded so far during the current Inquiry process. | C.128 | E |
| IO Capability Request event | 2.1 + EDR | This event is used to indicate that the IO capabilities of the Host are required for a Secure Simple Pairing process. | M | E |
| IO Capability Request Negative Reply command | 2.1 + EDR | This command is used to reject a pairing attempt after an HCI_IO_Capability_Request event has been received by the Host. | M | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| IO Capability Request Reply command | 2.1 + EDR | This command is used to reply to an HCI_IO_Capability_Request event from the Controller, and specifies the current IO capabilities of the Host. | M | E |
| IO Capability Response event | 2.1 + EDR | This event is used to indicate to the Host that IO capabilities from a remote device specified by BD_ADDR have been received during a Secure Simple Pairing process. | M | E |
| Keypress Notification event | 2.1 + EDR | This event is sent to the Host after a passkey notification has been received by the Link Manager on the given BD_ADDR. | M | E |
| LE Accept CIS Request command | 5.2 | This command is used by the Peripheral's Host to inform the Controller to accept the request for creating the CIS. | E | C.40 |
| LE Add Device To Filter Accept List command | 4.0 | This command will add a device to the Filter Accept List. | E | M |
| LE Add Device To Monitored Advertisers List command | 6.0 | This command will add a device to the Monitored Advertisers List. | E | C.78 |
| LE Add Device To Periodic Advertiser List command | 5.0 | This command will add a device to the Periodic Advertiser List. | E | C.21 |
| LE Add Device To Resolving List command | 4.2 | This command is used to add one device to the resolving list used to resolve Resolvable Private Addresses in the Controller. | E | C.9 |
| LE Advertising Report event | 4.0 | This event indicates that an advertising or scan response packet has been received. | E | C.98 |
| LE Advertising Set Terminated event | 5.0 | This event indicates that advertising in a given advertising set has stopped. | E | C.17 |
| LE BIG Create Sync command | 5.2 | This synchronizes and receives PDUs from one or more BISes. | E | C.42 |
| LE BIG Sync Established event | 5.2 | This event indicates that the Controller has completed an attempt to synchronize with the requested BISes. | E | C.42 |
| LE BIG Sync Lost event | 5.2 | This event indicates that the Controller stopped synchronizing with a BIG. | E | C.42 |
| LE BIG Terminate Sync command | 5.2 | This command stops or cancels synchronizing with a BIG. | E | C.42 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE BIGInfo Advertising Report event | 5.2 | This event indicates that the Controller has received an Advertising PDU that contained a BIGInfo field. | E | C.54 |
| LE Channel Selection Algorithm event | 5.0 | This event indicates the channel selection algorithm used on a connection. | E | C.23 |
| LE CIS Established event | 5.2<br>Erratum 18552 | This event indicates that the Controller established a CIS. | E | [v1] C.38<br>[v2] C.159 |
| LE CIS Request event | 5.2 | This event indicates that the Peripheral's Controller received a request from the Central to create a CIS. | E | C.40 |
| LE Clear Advertising Sets command | 5.0 | This command will remove all existing advertising sets from the Controller. | E | C.17 |
| LE Clear Filter Accept List command | 4.0 | This command will clear the Filter Accept List. | E | M |
| LE Clear Monitored Advertisers List command | 6.0 | This command will clear the Monitored Advertisers List. | E | C.78 |
| LE Clear Periodic Advertiser List command | 5.0 | This command will clear the Periodic Advertiser List. | E | C.21 |
| LE Clear Resolving List command | 4.2 | This command is used to remove all devices from the resolving list used to resolve Resolvable Private Addresses in the Controller. | E | C.9 |
| LE Connection Complete event | 4.0 | This event indicates to the Host that a new connection has been created. | E | C.3 |
| LE Connection CTE Request Enable command | 5.1 | This command will request the Controller to start or stop sending of LL_CTE_REQ PDUs on a connection. | E | C.25 |
| LE Connection CTE Response Enable command | 5.1 | This command will command the Controller to respond to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs. | E | C.26 |
| LE Connection IQ Report event | 5.1 | This event is used to report IQ samples from the Constant Tone Extension field of a received packet containing an LL_CTE_RSP PDU. | E | C.25 |
| LE Connection Rate Change event | 6.2 | This event is used to inform the Host about the result of the Connection Rate Update procedure. | E | C.85 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Connection Rate Request command | 6.2 | This command is used to initiate the Connection Rate Request procedure or the Connection Rate Update procedure. | E | C.85 |
| LE Connection Update command | 4.0 | This command will be used to change the connection parameters of an existing connection. | E | C.62 |
| LE Connection Update Complete event | 4.0 | This event indicates the completion of the process to change the connection parameters. | E | C.3 |
| LE Connectionless IQ Report event | 5.1 | This event reports IQ information from the Constant Tone Extension of a received advertising packet. | E | C.28 |
| LE Create BIG command | 5.2 | This command creates one or more BISes of a BIG. | E | C.41 |
| LE Create BIG Complete event | 5.2 | This event indicates that the Controller completed an attempt to create the BISes that were requested by the Host. | E | C.41 |
| LE Create BIG Test command | 5.2 | This command is used to create one or more BISes of a BIG for testing purposes. | E | C.41 |
| LE Create CIS command | 5.2 | This command is used by the Central's Host to create one or more CISes. | E | C.39 |
| LE Create Connection Cancel command | 4.0 | This command is used to cancel an ongoing HCI_LE_Create_Connection command. | E | C.94 |
| LE Create Connection command | 4.0 | This command is used to create a new connection. | E | C.59 |
| LE CS Config Complete event | 6.0 | This event is used to report the status of a CS configuration initiated by either a local or remote Controller. | E | C.75 |
| LE CS Create Config command | 6.0 | This command is used by a Host to create a CS configuration. | E | C.75 |
| LE CS Procedure Enable command | 6.0 | This command is used by a Host to enable the Controller to initiate the CS start procedure. | E | C.75 |
| LE CS Procedure Enable Complete event | 6.0 | This event is used to report the status after enabling a new CS procedure or disabling an ongoing CS procedure. | E | C.75 |
| LE CS Read Local Supported Capabilities command | 6.0<br>6.3 | This command is used by a Host to query CS capabilities. | E | [v1]<br>C.75<br>[v2]<br>C.48 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE CS Read Remote FAE Table command | 6.0 | This command is used by a Host to read the per-channel mode-0 Frequency Actuation Error table of the remote Controller. | E | C.75 |
| LE CS Read Remote FAE Table Complete event | 6.0 | This event is used to report the completion of a locally initiated Channel Sounding mode-0 FAE Table Request procedure. | E | C.75 |
| LE CS Read Remote Supported Capabilities command | 6.0 | This command is used by a Host to query the CS capabilities of a remote device. | E | C.75 |
| LE CS Read Remote Supported Capabilities Complete event | 6.0<br>6.3 | This event is used to report the CS capabilities of a remote device. | E | [v1]<br>C.75<br>[v2]<br>C.48 |
| LE CS Remove Config command | 6.0 | This command is used by a Host to remove an active CS configuration. | E | C.75 |
| LE CS Security Enable command | 6.0 | This command is used by a Host to start the CS Security procedure on the specified connection. | E | C.75 |
| LE CS Security Enable Complete event | 6.0 | This event is used to report the completion of a CS Security Start procedure initiated by either a local or remote Controller. | E | C.75 |
| LE CS Set Channel Classification command | 6.0 | This command is used by the Host to specify the channel classification to be used for CS procedures. | E | C.75 |
| LE CS Set Default Security Requirements command | 6.3 | This command is used by a Host to set the initial Channel Sounding security requirements in the Controller for future connections. | E | C.91 |
| LE CS Set Default Settings command | 6.0 | This command is used by a Host to set default CS settings in the local Controller. | E | C.75 |
| LE CS Set Procedure Parameters command | 6.0 | This command is used by the Host to specify the parameters to be used for scheduling CS procedures. | E | C.75 |
| LE CS Set Security Requirements command | 6.3 | This command is used by a Host to set the Channel Sounding security requirements in the Controller for a connection. | E | C.92 |
| LE CS Subevent Result Continue event | 6.0 | This event is used to report any remaining results of a CS subevent in the initiator or reflector. | E | C.75 |
| LE CS Subevent Result event | 6.0 | This event is used to report the results of a CS subevent in the initiator or reflector. | E | C.75 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE CS Test command | 6.0 | This command runs the CS test. | E | C.75 |
| LE CS Test End command | 6.0 | This command stops the CS test. | E | C.75 |
| LE CS Test End Complete event | 6.0 | This event is used to report the termination of a CS test. | E | C.75 |
| LE CS Write Cached Remote FAE Table command | 6.0 | This command is used by a Host to write the per-channel mode-0 Frequency Actuation Error table of the remote device in the local Controller. | E | C.76 |
| LE CS Write Cached Remote Supported Capabilities command | 6.0<br>6.3 | This command is used by a Host to write a cached copy of the remote CS capabilities in a local Controller. | E | [v1]<br>C.75<br>[v2]<br>C.48 |
| LE CTE Request Failed event | 5.1 | This event indicates a problem with a request generated by an HCI_LE_Connection_CTE_Request_Enable command for a peer device to send Constant Tone Extensions. | E | C.25 |
| LE Data Length Change event | 4.2 | This event is used to indicate a change in the maximum packet sizes by the Link Layer. | E | C.8 |
| LE Directed Advertising Report event | 4.2 | This event indicates that directed advertisements have been received where the advertiser is using a resolvable private address for the TargetA field in the ADV_DIRECT_IND PDU and the scanning filter policy is set to send this event to the Host. | E | C.63 |
| LE Enable Encryption command | 4.0 | This command is used to enable link level encryption. | E | C.60 |
| LE Enable Monitoring Advertisers command | 6.0 | This command will enable monitoring advertisers that are in the Monitored Advertisers List. | E | C.78 |
| LE Enable UTP OTA Mode command | 6.2 | This command is used by a device to enable the UTP OTA mode. | E | C.83 |
| LE Encrypt command | 4.0 | This command will encrypt a block of unencrypted data against a key and generate a block of encrypted data. | E | C.4 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Enhanced Connection Complete event | 4.2<br>5.4 | This event indicates to the Host that a new connection has been created. This event contains the additional parameters of the local and peer resolvable private addresses. | E | [v1]<br>C.24<br>[v2]<br>C.69 |
| LE Enhanced Read Transmit Power Level command | 5.2 | This command is used to read the current and maximum transmit power levels used by the local Controller on a specified PHY on an ACL connection. | E | C.51 |
| LE Extended Advertising Report event | 5.0 | This event indicates that an advertising packet has been received. | E | C.19 |
| LE Extended Create Connection command | 5.0<br>5.4 | This command is used to create a new connection supporting different initiating PHYs and to initiate a connection with a synchronized device. | E | [v1]<br>C.20<br>[v2]<br>C.67 |
| LE Frame Space Update command | 6.0 | This command is used to initiate the Frame Space Update procedure. | E | C.79 |
| LE Frame Space Update Complete event | 6.0 | This event is used to inform the Host of changes to the frame space values | E | C.79 |
| LE Generate DHKey command | 4.2<br>5.1 | This command is used to initiate generation of a Diffie-Hellman key in the Controller for use over the LE transport. | E | [v1]<br>C.99<br>[v2]<br>O |
| LE Generate DHKey Complete event | 4.2 | This event indicates that LE Diffie-Hellman key generation has been completed by the Controller. | E | C.81 |
| LE ISO Read Test Counters command | 5.2 | This command reads the test counters in the Controller which is configured in ISO Receive Test mode. | E | C.46 |
| LE ISO Receive Test command | 5.2 | This command configures a Link Layer to receive test payloads from an established CIS or a synchronized BIS. | E | C.46 |
| LE ISO Test End command | 5.2 | This command terminates the ISO Transmit and/or Receive Test mode. | E | C.47 |
| LE ISO Transmit Test command | 5.2 | This command configures an established CIS or BIS to transmit test payloads that are generated by the Controller. | E | C.45 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Long Term Key Request event | 4.0 | This event indicates that a Long Term Key is required for a connection. | E | C.61 |
| LE Long Term Key Request Negative Reply command | 4.0 | This command is used to reply to an HCI_LE_Long_Term_Key_Request event and indicates that the Host does not have a Long Term Key for that connection. | E | C.61 |
| LE Long Term Key Request Reply command | 4.0 | This command is used to reply to an HCI_LE_Long_Term_Key_Request event and includes the Long Term Key stored in the Host for that connection. | E | C.61 |
| LE Modify Sleep Clock Accuracy command | 5.1 | This command requests the Controller changes its sleep clock accuracy for testing purposes. | E | C.37 |
| LE Monitored Advertisers Report event | 6.0 | This event is used by a Controller to send reports on advertising devices being monitored. | E | C.78 |
| LE Path Loss Threshold event | 5.2 | This event is used to report a path loss threshold crossing on an ACL connection. | E | C.52 |
| LE Periodic Advertising Create Sync Cancel command | 5.0 | This command is used to cancel a pending HCI_LE_Periodic_Advertising_Create_Sync command. | E | C.16 |
| LE Periodic Advertising Create Sync command | 5.0 | This command is used to start receiving periodic advertising packets from an advertiser. | E | C.16 |
| LE Periodic Advertising Report event | 5.0<br>5.4 | This event indicates that a periodic advertising packet has been received. | E | [v1]<br>C.21<br>[v2]<br>C.87 |
| LE Periodic Advertising Set Info Transfer command | 5.1 | This command is used to send periodic advertising synchronization information, describing periodic advertising events that the Controller is transmitting, to a connected Controller. | E | C.34 |
| LE Periodic Advertising Response Report event | 5.4 | This event is used to report response data to a Host. | E | C.67 |
| LE Periodic Advertising Subevent Data Request event | 5.4 | This event is used to request subevent data from a Host. | E | C.67 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Periodic Advertising Sync Established event | 5.0<br>5.4 | This event indicates that the Controller has started receiving periodic advertising packets from an advertiser. | E | [v1]<br>C.16<br>[v2]<br>C.88 |
| LE Periodic Advertising Sync Lost event | 5.0 | This event indicates the Controller has ended receiving a periodic advertising train. | E | C.21 |
| LE Periodic Advertising Sync Transfer command | 5.1 | This command is used to send periodic advertising synchronization information to a connected Controller. | E | C.33 |
| LE Periodic Advertising Sync Transfer Received event | 5.1<br>5.4 | This event reports reception of periodic advertising synchronization information from a connected Controller. | E | [v1]<br>C.35<br>[v2]<br>C.89 |
| LE Periodic Advertising Terminate Sync command | 5.0 | This command is used to end receiving of a periodic advertising train. | E | C.21 |
| LE PHY Update Complete event | 5.0 | This event is used to inform the Host of the current PHY in each direction. | E | C.11 |
| LE Rand command | 4.0 | This command will generate a random number. | E | C.4 |
| LE Read Advertising Physical Channel Tx Power command | 4.0 | This command will read the transmit power level that will be used for advertising. | E | C.97 |
| LE Read All Local Supported Features command | 6.0 | This command will read all the features supported by the local LE Controller. | E | C.70 |
| LE Read All Remote Features command | 6.0 | This command is used to read all the features used on a connection and the features supported by a remote LE device. | E | C.71 |
| LE Read All Remote Features Complete event | 6.0 | This event indicates the completion of the process to read all the features used on a connection and the features supported by a remote LE device. | E | C.72 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Read Antenna Information command | 5.1 | This command allows the Host to read the switching rates, the sampling rates, the number of antennae, and the maximum length of the Constant Tone Extension supported by the Controller. | E | C.31 |
| LE Read Buffer Size command | 4.0<br>5.2 | This command returns the size of the HCI buffers. These buffers are used by the LE Controller to buffer data that is to be transmitted. | E | [v1]<br>C.3<br>[v2]<br>C.55 |
| LE Read Channel Map command | 4.0 | This command will read the current state of the channel map for a connection. | E | C.3 |
| LE Read Filter Accept List Size command | 4.0 | This command will read the number of Filter Accept List entries that this Controller can store at the present time. | E | M |
| LE Read ISO Link Quality command | 5.2 | This command returns the value of various counters related to link quality on an isochronous stream. | E | C.50 |
| LE Read ISO TX Sync command | 5.2 | This command is used to read the Time_Stamp and Time_Offset of a transmitted SDU. | E | C.45 |
| LE Read Local P-256 Public Key command | 4.2 | This command is used to return the local P-256 public key from the Controller. | E | C.82 |
| LE Read Local P-256 Public Key Complete event | 4.2 | This event is generated when local P-256 key generation is complete. | E | C.80 |
| LE Read Local Resolvable Address command | 4.2 | This command is used to get the current local Resolvable Private Address being used for the corresponding peer Identity Address. | E | C.10 |
| LE Read Local Supported Features Page 0 command | 4.0 | This command will read page 0 of the features supported by the local LE Controller. | E | M |
| LE Read Maximum Advertising Data Length command | 5.0 | This command will read the maximum length of advertising data that the advertising Controller supports in a given advertising set. | E | C.17 |
| LE Read Maximum Data Length command | 4.2 | This command allows the Host to read the Controller's supportedMaxTxOctets, supportedMaxTxTime, supportedMaxRxOctets, and supportedMaxRxTime parameters. | E | C.8 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Read Minimum Supported Connection Interval command | 6.2 | This command reads the Controller's minimum supported connection interval and resolution. | E | C.85 |
| LE Read Monitored Advertisers List Size command | 6.0 | This command will read the number of Monitored Advertisers List entries that the Controller supports. | E | C.78 |
| LE Read Number of Supported Advertising Sets command | 5.0 | This command will read the number of advertising sets that the advertising Controller can support at the present time. | E | C.17 |
| LE Read Peer Resolvable Address command | 4.2 | This command is used to get the current peer Resolvable Private Address being used for the corresponding peer Public and Random (static) Identity Address. | E | C.10 |
| LE Read Periodic Advertiser List Size command | 5.0 | This command will read the number of Periodic Advertiser List entries that the Controller can store at the present time. | E | C.21 |
| LE Read PHY command | 5.0 | This command will read the current PHY in each direction. | E | C.11 |
| LE Read Remote Features Page 0 command | 4.0 | This command is used to read page 0 of the features used on a connection and the features supported by a remote LE device. | E | C.3 |
| LE Read Remote Features Page 0 Complete event | 4.0 | This event indicates the completion of the process to read page 0 of the features used on a connection and the features supported by a remote LE device. | E | C.3 |
| LE Read Remote Transmit Power Level command | 5.2 | This command is used to read the transmit power level used by the remote Controller on a specified PHY on an ACL connection. | E | C.51 |
| LE Read Resolving List Size command | 4.2 | This command is used to read the total number of entries in the resolving list that can be stored in the Controller. | E | C.9 |
| LE Read RF Path Compensation command | 5.0 | This command is used to read the RF Path Compensation Value. | E | C.22 |
| LE Read Suggested Default Data Length command | 4.2 | This command allows the Host to read the initial MaxTxOctets and MaxTxTime values for new connections it suggested to the Controller. | E | C.8 |
| LE Read Supported States command | 4.0 | This command will read the current supported state and role combinations for the local LE Controllers. | E | M |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Read Transmit Power command | 5.0 | This command will read the minimum and maximum transmit powers supported by the Controller. | E | C.64 |
| LE Receiver Test command | 4.0<br>5.0<br>5.1 | This command will run the LE receiver test. | E | [v1]<br>C.2<br>[v2]<br>C.13<br>[v3]<br>C.30 |
| LE Reject CIS Request command | 5.2 | This command is used by the Peripheral's Host to inform the Controller to reject the request for creating the CIS. | E | C.40 |
| LE Remote Connection Parameter Request event | 4.1 | This event is used to indicate to the Host that the remote device is requesting a change in the connection parameters. | E | C.6 |
| LE Remote Connection Parameter Request Negative Reply command | 4.1 | This command is used to reject the remote device's request to change the connection parameters of the LE connection. | E | C.6 |
| LE Remote Connection Parameter Request Reply command | 4.1 | This command is used to accept the remote device's request to change the connection parameters of the LE connection. | E | C.6 |
| LE Remove Advertising Set command | 5.0 | This command will remove an advertising set from the Controller. | E | C.17 |
| LE Remove CIG command | 5.2 | This command is used by the Central's Host to remove a CIG from the Controller. | E | C.39 |
| LE Remove Device From Filter Accept List command | 4.0 | This command will remove a single device from the Filter Accept List. | E | M |
| LE Remove Device From Monitored Advertisers List command | 6.0 | This command will remove a device from the Monitored Advertisers list. | E | C.78 |
| LE Remove Device From Periodic Advertiser List command | 5.0 | This command will remove a single device from the Periodic Advertiser List. | E | C.21 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Remove Device From Resolving List command | 4.2 | This command is used to remove one device from the resolving list used to resolve Resolvable Private Addresses in the Controller. | E | C.9 |
| LE Remove ISO Data Path command | 5.2 | This command removes an isochronous data path between the Host and the Controller. | E | C.47 |
| LE Request Peer SCA command | 5.2 | This command requests the Sleep Clock Accuracy of the peer device. | E | C.44 |
| LE Request Peer SCA Complete event | 5.2 | This event indicates that the Controller completed the attempt to read the Sleep Clock Accuracy (SCA) of the peer device. | E | C.95 |
| LE Scan Request Received event | 5.0 | This event indicates that a scan request has been received. | E | C.17 |
| LE Scan Timeout event | 5.0 | This event indicates that scanning has finished. | E | C.19 |
| LE Set Address Resolution Enable command | 4.2 | This command is used to enable resolution of Resolvable Private Addresses in the Controller. | E | C.9 |
| LE Set Advertising Data command | 4.0 | This command will set the data transmitted when advertising. | E | C.97 |
| LE Set Advertising Enable command | 4.0 | This command will enable or disable advertising. | E | C.97 |
| LE Set Advertising Parameters command | 4.0 | This command will set the parameters used for advertising. | E | C.97 |
| LE Set Advertising Set Random Address command | 5.0 | This command will set the random address used in advertising. | E | C.17 |
| LE Set CIG Parameters command | 5.2 | This command is used by a Central's Host to set the parameters of one or more Connected Isochronous Streams (CISes) that are associated with a CIG in the Controller. | E | C.39 |
| LE Set CIG Parameters Test command | 5.2 | This command is used by a Central's Host to set the parameters of one or more CISes that are associated with a CIG in the Controller for testing purposes. | E | C.39 |
| LE Set Connection CTE Receive Parameters command | 5.1 | This command will set the antenna-switching pattern, switching and sampling slot durations for receiving the Constant Tone Extension on a connection. | E | C.25 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Set Connection CTE Transmit Parameters command | 5.1 | This command will set the antenna-switching pattern, switching and sampling slot durations for transmitting the Constant Tone Extension on a connection. | E | C.26 |
| LE Set Connectionless CTE Transmit Enable command | 5.1 | This command will request the Controller to enable or disable sending packets containing a Constant Tone Extension. | E | C.27 |
| LE Set Connectionless CTE Transmit Parameters command | 5.1 | This command will set the antenna-switching pattern and switching and sampling slot durations for the transmission of Constant Tone Extensions. | E | C.27 |
| LE Set Connectionless IQ Sampling Enable command | 5.1 | This command will request the Controller to enable or disable taking IQ samples from the Constant Tone Extension of advertising packets. | E | C.28 |
| LE Set Data Length command | 4.2 | This command is used to suggest maximum packet sizes to the Controller. | E | C.8 |
| LE Set Data Related Address Changes command | 5.3 | This command specifies circumstances when the Controller shall refresh any Resolvable Private Address used by an advertising set, whether or not the address timeout period has been reached. | E | C.10 |
| LE Set Decision Data command | 6.0 | This command will set the decision data transmitted when advertising. | E | C.73 |
| LE Set Decision Instructions command | 6.0 | This command will set the decision instructions used when scanning for advertisements that include decision data. | E | C.74 |
| LE Set Default Periodic Advertising Sync Transfer Parameters command | 5.1 | This command is used to specify the default behavior of the Controller when periodic advertising synchronization information is received from a connected Controller. | E | C.35 |
| LE Set Default PHY command | 5.0 | This command is used to configure preferred PHYs for new connections for the local device. | E | C.11 |
| LE Set Default Rate Parameters command | 6.2 | This command sets the range of the Connection Rate parameters on a Central that may be requested by a Peripheral. | E | C.85 |
| LE Set Default Subrate command | 5.3 | This command sets the range of the min and max subrates and other subrate parameters on a Central that may be requested by a Peripheral. | E | C.57 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Set Event Mask command | 4.0<br>6.3 | This command is used to control which events are generated by the HCI for the Host. | E | [v1]<br>M<br>[v2]<br>C.222 |
| LE Set Extended Advertising Data command | 5.0 | This command will set the advertising data transmitted when advertising. | E | C.17 |
| LE Set Extended Advertising Enable command | 5.0 | This command will enable or disable advertising. | E | C.17 |
| LE Set Extended Advertising Parameters command | 5.0<br>5.4 | This command will set the parameters used for advertising. | E | [v1]<br>C.65<br>[v2]<br>C.66 |
| LE Set Extended Scan Enable command | 5.0 | This command will enable or disable scanning on the primary advertising physical channels. | E | C.19 |
| LE Set Extended Scan Parameters command | 5.0 | This command will set the parameters used for scanning on the primary advertising physical channel. | E | C.19 |
| LE Set Extended Scan Response Data command | 5.0 | This command will set the data transmitted in a scan response. | E | C.17 |
| LE Set Host Channel Classification command | 4.0 | This command allows the Host to specify a channel classification based on its “local information”. | E | C.36 |
| LE Set Host Feature command | 5.2<br>6.0 | This command is used to set or clear a bit controlled by the Host in the Link Layer FeatureSet stored in the Controller. | E | [v1]<br>C.49<br>[v2]<br>C.77 |
| LE Set Path Loss Reporting Enable command | 5.2 | This command is used to enable or disable path loss reporting events for an ACL connection. | E | C.52 |
| LE Set Path Loss Reporting Parameters command | 5.2 | This command is used to set the path loss threshold and related parameters used to trigger reports for an ACL connection. | E | C.52 |
| LE Set Periodic Advertising Data command | 5.0 | This command will set the periodic advertising data transmitted when advertising. | E | C.18 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Set Periodic Advertising Enable command | 5.0 | This command will enable or disable periodic advertising. | E | C.18 |
| LE Set Periodic Advertising Parameters command | 5.0<br>5.4 | This command will set the parameters used for periodic advertising. | E | [v1]<br>C.18<br>[v2]<br>C.67 |
| LE Set Periodic Advertising Receive Enable command | 5.1 | This command will enable or disable periodic advertising reports once synchronized. | E | C.32 |
| LE Set Periodic Advertising Response Data command | 5.4 | This command is used to set the data for a response slot. | E | C.68 |
| LE Set Periodic Advertising Subevent Data command | 5.4 | This command is used to send subevent data for one or more subevents. | E | C.67 |
| LE Set Periodic Sync Subevent command | 5.4 | This command is used to configure the subset of subevents a device will synchronize with. | E | C.68 |
| LE Set Periodic Advertising Sync Transfer Parameters command | 5.1 | This command is used to allow the Host to specify the behavior of the Controller when periodic advertising synchronization information is received from a connected Controller. | E | C.35 |
| LE Set PHY command | 5.0 | This command is used to request a change of the PHY for a Connection_Handle. | E | C.11 |
| LE Set Privacy Mode command | 5.0 | This command is used to allow the Host to specify the privacy mode for an entry on the resolving list. | E | C.9 |
| LE Set Random Address command | 4.0 | This command will set the Random Device Address that may be used in a packet sent on the advertising physical channel. | E | C.43 |
| LE Set Resolvable Private Address Timeout command | 4.2<br>6.1 | This command sets the length of time the Controller uses a resolvable private address before a new resolvable private address is generated and starts being used. The [v2] version enables the timeout to be randomly varied within a specified range. | E | [v1]<br>C.9<br>[v2]<br>C.10 |
| LE Set Scan Enable command | 4.0 | This command will enable or disable scanning. | E | C.98 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE Set Scan Parameters command | 4.0 | This command will set the parameters used for scanning. | E | C.98 |
| LE Set Scan Response Data command | 4.0 | This command will set the data transmitted in a scan response. | E | C.15 |
| LE Set Transmit Power Reporting Enable command | 5.2 | This command is used to enable or disable reporting to the local Host of transmit power level changes on an ACL connection. | E | C.51 |
| LE Setup ISO Data Path command | 5.2 | This command identifies and creates the isochronous data path between the Host and the Controller and may configure the codec in the Controller. | E | C.47 |
| LE Subrate Change event | 5.3 | This event indicates that a new subrate factor has been applied to an existing ACL connection. | E | C.57 |
| LE Subrate Request command | 5.3 | This command modifies an existing ACL connection by applying a subrate factor. | E | C.57 |
| LE Terminate BIG command | 5.2 | This command terminates the transmission of all BISes of a BIG or cancels the process of creating a BIG. | E | C.41 |
| LE Terminate BIG Complete event | 5.2 | This event indicates that the transmission of all the BISes in the BIG have been terminated. | E | C.41 |
| LE Test End command | 4.0 | This command will end the current the receiver or transmitter test. | E | M |
| LE Transmit Power Reporting event | 5.2 | This event is used to report the transmit power level on the ACL connection. | E | C.51 |
| LE Transmitter Test command | 4.0<br>5.0<br>5.1<br>5.2 | This command will run the LE transmitter test. | E | [v1]<br>C.1<br>[v2]<br>C.12<br>[v3]<br>C.29<br>[v4]<br>C.53 |
| LE UTP Receive event | 6.2 | The HCI_LE_UTP_Receive event is used by the IUT to transmit UTP messages to the Upper Tester. | E | C.84 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| LE UTP Send command | 6.2 | The HCI_LE_UTP_Send command is used by the Upper Tester to transmit UTP messages to the IUT. | E | C.84 |
| LE Write RF Path Compensation command | 5.0 | This command is used to indicate the RF path gain or loss from the RF transceiver output to the antenna output contributed by intermediate components. | E | C.22 |
| LE Write Suggested Default Data Length command | 4.2 | This command allows the Host to suggest initial MaxTxOctets and MaxTxTime values for new connections. | E | C.8 |
| Link Key Notification event | 1.1 | This event is used to indicate to the Host that a new Link Key has been created for the connection with the BR/EDR Controller specified in BD_ADDR. | M | E |
| Link Key Request event | 1.1 | This event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR. | M | E |
| Link Key Request Negative Reply command | 1.1 | This command is used to reply to an HCI_Link_Key_Request event from the BR/EDR Controller if the Host does not have a stored Link Key for the connection with the other BR/EDR Controller specified by BD_ADDR. | M | E |
| Link Key Request Reply command | 1.1 | This command is used to reply to an HCI_Link_Key_Request event from the BR/EDR Controller, and specifies the Link Key stored on the Host to be used as the link key for the connection with the other BR/EDR Controller specified by BD_ADDR. | M | E |
| Link Key Selection command | 1.1 | This command is used to force both BR/EDR Controllers of a connection associated to the Connection_Handle to use the temporary link key of the Central or the regular link keys. | C.215 | E |
| Link Key Type Changed event | 1.1 | This event is used to indicate that the change in the temporary Link Key or in the semi-permanent link keys on the Bluetooth Central side has been completed. | C.215 | E |
| Link Supervision Timeout Changed event | 2.1 + EDR | This event indicates that the remote device changed the Link Supervision Timeout. | M | E |
| Loopback Command event | 1.1 | This event is used to loop back all commands that the Host sends to the BR/EDR Controller with some exceptions. | C.123 | E |
| Max Slots Change event | 1.1 | This event is used to indicate a change in the max slots by the LM. | C.132 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Mode Change event | 1.1 | This event is used to indicate that the current mode has changed. | C.144 | E |
| Number Of Completed Data Blocks event | 3.0 + HS | This event is used by the Controller to indicate to the Host how many HCI ACL Data packets have been completed and how many data block buffers have been freed for each Handle since the previous time this event was sent. | C.124 | E |
| Number Of Completed Packets event | 1.1 | This event is used by the Controller to indicate to the Host how many HCI Data packets have been completed for each Connection_Handle since the previous time this event was sent. | M | C.3 |
| Page Scan Repetition Mode Change event | 1.1 | This event indicates that the connected remote BR/EDR Controller with the specified Connection_Handle has successfully changed the Page Scan Repetition Mode (SR). | O | E |
| Periodic Inquiry Mode command | 1.1 | This command is used to configure the BR/EDR Controller to perform an automatic Inquiry based on a specified period range. | C.128 | E |
| Peripheral Page Response Timeout event | CSA4 | This event indicates to the Host that the *pageRespTO* has been exceeded on the BR/EDR Controller after the Controller responded to an ID packet. | O | E |
| PIN Code Request event | 1.1 | This event is used to indicate that a PIN code is required to create a new link key for a connection. | M | E |
| PIN Code Request Negative Reply command | 1.1 | This command is used to reply to an HCI_PIN_Code_Request event from the Controller when the Host cannot specify a PIN code to use for a connection. | M | E |
| PIN Code Request Reply command | 1.1 | This command is used to reply to an HCI_PIN_Code_Request event from the Controller and specifies the PIN code to use for a connection. | M | E |
| QoS Setup command | 1.1 | This command is used to specify Quality of Service parameters for a Connection_Handle. | M | E |
| QoS Setup Complete event | 1.1 | This event is used to indicate that QoS is set up. | M | E |
| QoS Violation event | 1.1 | This event is used to indicate the Controller's Link Manager is unable to provide the current QoS requirement for the Handle. | M | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read AFH Channel Assessment Mode command | 1.2 | This command will read the value for the AFH Channel Classification Mode parameter. This value is used to enable or disable the Controller's channel assessment scheme. | C.140 | C.58 |
| Read AFH Channel Map command | 1.2 | This command will read the current state of the channel map for a connection. | C.139 | E |
| Read Authenticated Payload Timeout command | 4.1 | This command is used to read the Authenticated Payload Timeout parameter, which is used to set the maximum time between packets being received from the remote device without a valid MIC. | C.155 | C.155 |
| Read Authentication Enable command | 1.1 | This command will read the value for the Authentication Enable parameter, which controls whether the Bluetooth device will require authentication for each connection with other Bluetooth devices. | C.111 | E |
| Read Automatic Flush Timeout command | 1.1 | This command will read the value for the Flush Timeout configuration parameter for the specified Connection_Handle. The Flush Timeout parameter is only used for ACL connections. | M | C.86 |
| Read BD_ADDR command | 1.1 | This command will read the value for the BD_ADDR parameter. | M | M |
| Read Buffer Size command | 1.1 | This command returns the size of the HCI buffers. These buffers are used by the Controller to buffer data that is to be transmitted. | M | E |
| Read Class of Device command | 1.1 | This command will read the value for the Class of Device configuration parameter, which is used to indicate its capabilities to other devices. | M | E |
| Read Clock command | 1.2 | This command will read an estimate of a piconet or the local Bluetooth Clock. | O | E |
| Read Clock Offset command | 1.1 | This command allows the Host to read the clock offset of remote BR/EDR Controllers. | O | E |
| Read Clock Offset Complete event | 1.1 | This event is used to indicate the completion of the process of the LM obtaining the Clock offset information. | C.104 | E |
| Read Connection Accept Timeout command | 1.1 | This command will read the value for the Connection Accept Timeout configuration parameter, which allows the Controller to automatically deny a connection request after a specified period has occurred, and to refuse a new connection. | M | C.40 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read Current IAC LAP command | 1.1 | This command will read the LAP(s) used to create the Inquiry Access Codes (IAC) that the local BR/EDR Controller is simultaneously scanning for during Inquiry Scans. | C.125 | E |
| Read Data Block Size command | 3.0 + HS | This command returns the maximum size of the HCI buffers. These buffers are used by the Controller to buffer data that is to be transmitted. | C.124 | E |
| Read Default Erroneous Data Reporting command | 2.1 + EDR | This command will read the value for the Erroneous Data Reporting configuration parameter, which controls whether the BR/EDR Controller will provide data for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data packets set according to HCI Synchronous Data packets. | C.112 | E |
| Read Default Link Policy Settings command | 1.2 | This command will read the Default Link Policy configuration parameter for all new connections. | C.141 | E |
| Read Encryption Key Size command | 3.0 + HS | This command is used to read the encryption key size on a given Connection_Handle. | M | E |
| Read Enhanced Transmit Power Level command | 3.0 + HS | This command will read the values for the GFSK, π/4-DQPSK and 8DPSK Transmit Power Level parameters for the specified Connection_Handle. | C.217 | E |
| Read Extended Inquiry Length command | 4.1 | This command is used to read the Extended Inquiry Length parameter from the Controller. | C.113 | E |
| Read Extended Inquiry Response command | 2.1 + EDR | This command will read the data that the BR/EDR Controller sends in the extended inquiry response packet during inquiry response. | C.205 | E |
| Read Extended Page Timeout command | 4.1 | This command is used to read the Extended Page Timeout parameter from the Controller. | C.114 | E |
| Read Failed Contact Counter command | 1.1 | This command will read the value for the Failed Contact Counter configuration parameter for a particular connection to another device. | M | E |
| Read Flow Control Mode command | 3.0 + HS | This command returns the value of the Flow_Control_Mode configuration parameter supported by this Controller. | C.124 | E |
| Read Hold Mode Activity command | 1.1 | This Activity command is used to read which activities should be suspended when the BR/EDR Controller is in Hold mode. | C.213 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read Inquiry Mode command | 1.2 | This command is used to read the Inquiry Mode configuration parameter of the local BR/EDR Controller. | C.115 | E |
| Read Inquiry Response Transmit Power Level command | 2.1 + EDR | This command will read the inquiry response Transmit Power level used to transmit the FHS and EIR data packets. This can be used directly in the Tx Power Level EIR data type. | C.125 | E |
| Read Inquiry Scan Activity command | 1.1 | This command will read the value for Inquiry Scan Interval and Inquiry Scan Window configuration parameters. Inquiry Scan Interval defines the amount of time between consecutive inquiry scans. Inquiry Scan Window defines the amount of time for the duration of the inquiry scan. | C.125 | E |
| Read Inquiry Scan Type command | 1.2 | This command is used to read the Inquiry Scan Type configuration parameter of the local BR/EDR Controller. The Inquiry Scan Type configuration parameter can set the inquiry scan to either normal or interlaced scan. | C.125 | E |
| Read LE Host Support command | 4.0 | This command reads the LE Supported Host setting from the BR/EDR Controller. | C.116 | E |
| Read Link Policy Settings command | 1.1 | This command will read the Link Policy configuration parameter for the specified Connection_Handle. The Link Policy settings allow the Host to specify which Link Modes the Link Manager can use for the specified Connection_Handle. | C.141 | E |
| Read Link Quality command | 1.1 | This command will read the value for the Link Quality for the specified Connection_Handle. | O | E |
| Read Link Supervision Timeout command | 1.1 | This command will read the value for the Link Supervision Timeout configuration parameter for the device. This parameter is used by the Controller to determine link loss. | C.117 | E |
| Read LMP Handle command | 1.2 | This command will read the current LMP Handle associated with the Connection_Handle. | C.134 | E |
| Read Local Extended Features command | 1.2 | This command requests a list of the supported extended features for the local device. | C.220 | E |
| Read Local Name command | 1.1 | This command provides the ability to read the stored user-friendly name for the BR/EDR Controller. | M | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read Local OOB Data command | 2.1 + EDR | This command is used to obtain a Secure Simple Pairing Hash C and Randomizer R which are intended to be transferred to a remote device using an OOB mechanism. | M | E |
| Read Local OOB Extended Data command | 4.1 | This command is used to obtain a Secure Simple Pairing Hash C and Randomizer R associated with both P-192 and P-256 public keys, which are intended to be transferred to a remote device using an OOB mechanism. | C.142 | E |
| Read Local Simple Pairing Options command | Erratum 10734 | This command is used to read the Secure Simple Pairing options and the maximum encryption key size supported. | O | E |
| Read Local Supported Codec Capabilities command | 5.2 | This command is used by a Host to query codec capabilities. | C.156 | C.156 |
| Read Local Supported Codecs command | CSA2<br>5.2 | This command is used by a Host to query a Controller's supported codecs. | [v1]<br>C.157<br>[v2]<br>O | [v1]<br>E<br>[v2]<br>O |
| Read Local Supported Commands command | 1.2<br>6.3 | This command requests a list of the supported HCI commands for the local device. | [v1]<br>M<br>[v2]<br>C.223 | [v1]<br>M<br>[v2]<br>C.223 |
| Read Local Supported Controller Delay command | 5.2 | This command is used by a Host to query a range of supported Controller delays for a given codec configuration. | C.156 | C.156 |
| Read Local Supported Features command | 1.1 | This command requests a list of the supported features for the local device. | M | M |
| Read Local Version Information command | 1.1 | This command will read the version information for the local Controller. | M | M |
| Read Loopback Mode command | 1.1 | This command will read the value for the setting of the BR/EDR Controller's Loopback Mode. The setting of the Loopback Mode will determine the path of information. | C.123 | E |
| Read Num Broadcast Retransmissions command | 1.1 | This command will read the parameter value for the Number of Broadcast Retransmissions for the BR/EDR Controller. | C.118 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read Number Of Supported IAC command | 1.1 | This command will read the value for the number of Inquiry Access Codes (IAC) that the local BR/EDR Controller can simultaneously listen for during an Inquiry Scan. | C.125 | E |
| Read Page Scan Activity command | 1.1 | This command will read the values for the Page Scan Interval and Page Scan Window configuration parameters. Page Scan Interval defines the amount of time between consecutive page scans. Page Scan Window defines the duration of the page scan. | M | E |
| Read Page Scan Type command | 1.2 | This command is used to read the page scan type of the local BR/EDR Controller. The Page Scan Type configuration parameter can set the page scan to either normal or interlaced scan. | C.119 | E |
| Read Page Timeout command | 1.1 | This command will read the value for the Page Reply Timeout configuration parameter, which determines the time the BR/EDR Controller will wait for the remote device to respond to a connection request before the local device returns a connection failure. | M | E |
| Read PIN Type command | 1.1 | This command is used for the Host to read the value that is specified to indicate whether the Host supports variable PINs or only fixed PINs. | C.120 | E |
| Read Remote Extended Features command | 1.2 | This command requests a list of the supported extended features of a remote device. | C.220 | E |
| Read Remote Extended Features Complete event | 1.2 | This event is used to indicate the completion of the process of the Link Manager obtaining the supported Extended features of the remote BR/EDR Controller specified by the Connection_Handle parameter. | C.220 | E |
| Read Remote Supported Features command | 1.1 | This command requests a list of the supported features of a remote device. | M | E |
| Read Remote Supported Features Complete event | 1.1 | This event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote BR/EDR Controller specified by the Connection_Handle parameter. | M | E |
| Read Remote Version Information command | 1.1 | This command will read the values for the version information for the remote device associated with the Connection_Handle. | O | C.3 |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Read Remote Version Information Complete event | 1.1 | This event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote device associated with the Connection_Handle parameter. | C.105 | C.3 |
| Read RSSI command | 1.1 | This command will read the value for the Received Signal Strength Indication (RSSI) for a Connection_Handle to another Controller. | O | C.3 |
| Read Scan Enable command | 1.1 | This command will read the value for the Scan Enable configuration parameter, which controls whether or not the BR/EDR Controller will periodically scan for page attempts and/or inquiry requests from other BR/EDR Controllers. | M | E |
| Read Secure Connections Host Support command | 4.1 | This command is used to read the Secure Connections Host Supports parameter from the Controller. | C.218 | E |
| Read Simple Pairing Mode command | 2.1 + EDR | This command reads the Secure Simple Pairing mode setting in the BR/EDR Controller. | M | E |
| Read Stored Link Key command | 1.1 | This command provides the ability to read whether one or more link keys are stored in the Controller. | C.121 | E |
| Read Synchronization Train Parameters command | CSA4 | This command returns the currently configured values for the Synchronization Train functionality in the BR/EDR Controller. | C.201 | E |
| Read Synchronous Flow Control Enable command | 1.1 | This command provides the ability to read the Synchronous Flow Control Enable setting. By using this setting, the Host can decide if the Controller will send HCI_Number_Of_Completed_Packets events for synchronous Connection_Handles. | C.122 | E |
| Read Transmit Power Level command | 1.1 | This command will read the values for the Transmit Power Level parameter for the specified Connection_Handle. | C.152 | C.3 |
| Read Voice Setting command | 1.1 | This command will read the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections. | C.134 | E |
| Receive Synchronization Train command | CSA4 | This command requests synchronization with the specified Connectionless Peripheral Broadcast transmitter. | C.202 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Refresh Encryption Key command | 2.1 + EDR | This command is used by the Host to cause the Controller to refresh the encryption key by pausing and resuming encryption | M | E |
| Reject Connection Request command | 1.1 | This command is used to decline a new incoming BR/EDR connection request. | M | E |
| Reject Synchronous Connection Request command | 1.2 | This command is used to decline an incoming request for a synchronous link. | C.134 | E |
| Remote Host Supported Features Notification event | 2.1 + EDR | This event is used to return the LMP extended features page containing the Host features. | C.106 | E |
| Remote Name Request Cancel command | 1.2 | This command is used to cancel an ongoing Remote Name Request. | C.106 | E |
| Remote Name Request command | 1.1 | This command is used to obtain the user-friendly name of another BR/EDR Controller. | O | E |
| Remote Name Request Complete event | 1.1 | This event is used to indicate a remote name request has been completed. | C.106 | E |
| Remote OOB Data Request event | 2.1 + EDR | This event is used to indicate that the Secure Simple Pairing Hash C and Randomizer R is required for the Secure Simple Pairing process involving the device identified by BD_ADDR. | M | E |
| Remote OOB Data Request Negative Reply command | 2.1 + EDR | This command is used to reply to an HCI_Remote_OOB_Data_Request event that the Host does not have the C and R | M | E |
| Remote OOB Data Request Reply command | 2.1 + EDR | This command is used to reply to an HCI_Remote_OOB_Data_Request event with the C and R values received via an OOB transfer from a remote BR/EDR Controller identified by BD_ADDR. | M | E |
| Remote OOB Extended Data Request Reply command | 4.1 | This command is used to reply to an HCI_Remote_OOB_Data_Request event with the C and R values received via an OOB transfer from a remote BR/EDR Controller identified by the BD_ADDR. | C.142 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Reset command | 1.1 | For a BR/EDR Controller, this command resets HCI, the Link Manager, and the Bluetooth radio. For an LE Controller, this command resets HCI, the Link Layer, and LE PHY. | M | M |
| Reset Failed Contact Counter command | 1.1 | This command will reset the value for the Failed Contact Counter configuration parameter for a particular connection to another device. | M | E |
| Return Link Keys event | 1.1 | This event is used to return stored link keys after an HCI_Read_Stored_Link_Key command is used. | C.121 | E |
| Role Change event | 1.1 | This event is used to indicate that the current BR/EDR Controller role related to the particular connection has been changed. | C.212 | E |
| Role Discovery command | 1.1 | This command is used for a BR/EDR Controller to determine which role the device is performing for a particular Connection_Handle. | O | E |
| SAM Status Change event | 5.0 | This event is used to indicate that either the local or remote SAM slot map on a particular connection has been changed. | C.219 | E |
| Send Keypress Notification command | 2.1 + EDR | This command is used during the Passkey Entry protocol by a device with KeyboardOnly IO capabilities. It is used by a Host to inform the remote device when keys have been entered or erased. | M | E |
| Set AFH Host Channel Classification command | 1.2 | This command allows the Host to specify a channel classification based on its “local information”. | C.140 | E |
| Set Connection Encryption command | 1.1 | This command is used to enable and disable the link level encryption. | M | E |
| Set Connectionless Peripheral Broadcast command | CSA4 | This command controls Connectionless Peripheral Broadcast functionality (for transmission) in the BR/EDR Controller including enabling and disabling the broadcast. | C.201 | E |
| Set Connectionless Peripheral Broadcast Data command | CSA4 | This command is used by the Host to set Connectionless Peripheral Broadcast data in the BR/EDR Controller. | C.201 | E |
| Set Connectionless Peripheral Broadcast Receive command | CSA4 | This command enables and disables Connectionless Peripheral Broadcast reception in the BR/EDR Controller. | C.202 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Set Controller To Host Flow Control command | 1.1 | This command is used by the Host to turn flow control on or off in the direction from the Controller to the Host. | O | C.96 |
| Set Ecosystem Base Interval command | 5.2 | This command indicates to the Controller the base interval of the ecosystem. | O | O |
| Set Event Filter command | 1.1 | This command is used by the Host to specify different event filters. The Host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters. | C.148 | E |
| Set Event Mask command | 1.1 | This command is used to control which events are generated by the HCI for the Host. | M | M |
| Set Event Mask Page 2 command | 3.0 + HS | This command is used to control which events are generated by the HCI for the Host. | C.145 | C.145 |
| Set External Frame Configuration command | CSA3 | This command enables an external device to describe a frame structure to the Controller. | C.108 | O |
| Set Min Encryption Key Size command | 5.3 | This command is used to modify the minimum encryption key size that may be negotiated by the Controller. | O | E |
| Set MWS Channel Parameters command | CSA3 | This command enables an MWS device to inform the Controller about the MWS channel configuration. | O | O |
| Set MWS Scan Frequency Table command | CSA3 | This command specifies the frequencies represented by the frequency index supplied by the MWS_SCAN_FREQUENCY signal. | O | O |
| Set MWS Signaling command | CSA3 | This command enables an MWS device to inform the Controller about the timing parameters for the MWS coexistence interface. | O | O |
| Set MWS Transport Layer command | CSA3 | This command selects the MWS coexistence signaling transport layer in the Controller. | C.109 | C.109 |
| Set MWS_PATTERN Configuration command | CSA3 | This command specifies the configuration of the pattern indicated over the MWS Coexistence Transport Layer. | C.136 | E |
| Set Reserved LT_ADDR command | CSA4 | This command requests that the BR/EDR Controller reserve a specific LT_ADDR for the purposes of Connectionless Peripheral Broadcast. | C.201 | E |
| Set Triggered Clock Capture command | CSA4 | This command is used to configure the Controller to return events containing an estimate of a piconet or the local Bluetooth clock. | O | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Setup Synchronous Connection command | 1.2 | This command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. | C.134 | E |
| Simple Pairing Complete event | 2.1 + EDR | This event is used to indicate that the Secure Simple Pairing process has completed. | M | E |
| Sniff Mode command | 1.1 | This command is used to alter the behavior of the LM and have the LM place the local or remote device into Sniff mode. | C.214 | E |
| Sniff Subrating command | 2.1 + EDR | This command is used to configure the sniff subrating parameters in the local device. | C.221 | E |
| Sniff Subrating event | 2.1 + EDR | This event is used to inform the Host of the local and remote transmit and receive latencies. | C.221 | E |
| Start Synchronization Train command | CSA4 | This command enables the Synchronization Train on the BR/EDR Controller using the currently configured Synchronization Train parameters. | C.201 | E |
| Switch Role command | 1.1 | This command is used to switch Central and Peripheral roles of the devices on either side of a connection. | C.212 | E |
| Synchronization Train Complete event | CSA4 | This event indicates that the Synchronization Train has completed. | C.201 | E |
| Synchronization Train Received event | CSA4 | This event provides the status of Synchronization Train packets received from the device with the given BD_ADDR. | C.202 | E |
| Synchronous Connection Changed event | 1.2 | This event indicates to the Host that an existing synchronous connection has been reconfigured. | C.134 | E |
| Synchronous Connection Complete event | 1.2 | This event indicates to both the Hosts that a new synchronous connection has been established. | C.134 | E |
| Triggered Clock Capture event | CSA4 | This event reports the Bluetooth clock when an external trigger occurred. | C.110 | E |
| Truncated Page Cancel command | CSA4 | This command is used to cancel an ongoing Truncated Page. | C.129 | E |
| Truncated Page command | CSA4 | This command will cause the BR/EDR Controller to page the BR/EDR Controller with the BD_ADDR specified by the parameters and abort the page sequence after receiving the ID response packet. | C.129 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Truncated Page Complete event | CSA4 | This event indicates to the Host that a Truncated Page has completed. | C.129 | E |
| User Confirmation Request event | 2.1 + EDR | This event is used to indicate that user confirmation of a numeric value is required. | M | E |
| User Confirmation Request Negative Reply command | 2.1 + EDR | This command is used to reply to an HCI_User_Confirmation_Request event and indicates that the user selected “no”. This command will terminate Secure Simple Pairing. | M | E |
| User Confirmation Request Reply command | 2.1 + EDR | This command is used to reply to an HCI_User_Confirmation_Request event and indicates that the user selected “yes”. It is also used when the Host has no input and no output capabilities. | M | E |
| User Passkey Notification event | 2.1 + EDR | This event is used to provide a passkey for the Host to display to the user as required as part of a Secure Simple Pairing process. | M | E |
| User Passkey Request event | 2.1 + EDR | This event is used to indicate that a passkey is required as part of a Secure Simple Pairing process. | M | E |
| User Passkey Request Negative Reply command | 2.1 + EDR | This command is used to reply to an HCI_User_Passkey_Request event and indicates the Host could not provide a passkey. This command will terminate Secure Simple Pairing. | M | E |
| User Passkey Request Reply command | 2.1 + EDR | This command is used to reply to an HCI_User_Passkey_Request event and specifies the Numeric_Value (passkey) entered by the user to be used in the Secure Simple Pairing process. | M | E |
| Write AFH Channel Assessment Mode command | 1.2 | This command will write the value for the Channel Classification Mode configuration parameter. This value is used to enable or disable the Controller’s channel assessment scheme. | C.140 | C.58 |
| Write Authenticated Payload Timeout command | 4.1 | This command is used to write the Authenticated Payload Timeout parameter, which is used to set the maximum time between packets being received from the remote device without a valid MIC. | C.151 | C.7 |
| Write Authentication Enable command | 1.1 | This command will write the value for the Authentication Enable parameter, which controls whether the Bluetooth device will require authentication for each connection with other Bluetooth devices. | O | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Write Automatic Flush Timeout command | 1.1 | This command will write the value for the Flush Timeout configuration parameter for the specified Connection_Handle. The Flush Timeout parameter is only used for ACL connections. | M | C.86 |
| Write Class of Device command | 1.1 | This command will write the value for the Class_of_Device configuration parameter, which is used to indicate its capabilities to other devices. | M | E |
| Write Connection Accept Timeout command | 1.1 | This command will write the value for the Connection Accept Timeout configuration parameter, which allows the Controller to automatically deny a connection request after a specified period has occurred, and to refuse a new connection. | M | C.40 |
| Write Current IAC LAP command | 1.1 | This command will write the LAP(s) used to create the Inquiry Access Codes (IAC) that the local BR/EDR Controller is simultaneously scanning for during Inquiry Scans. | C.125 | E |
| Write Default Erroneous Data Reporting command | 2.1 + EDR | This command will write the value for the Erroneous Data Reporting configuration parameter, which controls whether the Bluetooth Controller will provide data for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data packets set according to HCI Synchronous Data packets. | C.206 | E |
| Write Default Link Policy Settings command | 1.2 | This command will write the Default Link Policy configuration parameter for all new connections. | C.141 | E |
| Write Extended Inquiry Length command | 4.1 | This command is used to write the Extended Inquiry Length parameter to the Controller. | C.128 | E |
| Write Extended Inquiry Response command | 2.1 + EDR | This command will write the data that the BR/EDR Controller sends in the extended inquiry response packet during inquiry response. | C.205 | E |
| Write Extended Page Timeout command | 4.1 | This command is used to write the Extended Page Timeout parameter to the Controller. | O | E |
| Write Flow Control Mode command | 3.0 + HS | This command sets the value of the Flow_Control_Mode configuration parameter for this Controller. | C.124 | E |
| Write Hold Mode Activity command | 1.1 | This command is used to write which activities should be suspended when the BR/EDR Controller is in Hold mode. | C.213 | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Write Inquiry Mode command | 1.2 | This command is used to write the Inquiry Mode configuration parameter of the local BR/EDR Controller. | C.146 | E |
| Write Inquiry Scan Activity command | 1.1 | This command will write the value for Inquiry Scan Interval and Inquiry Scan Window configuration parameters. Inquiry Scan Interval defines the amount of time between consecutive inquiry scans. Inquiry Scan Window defines the amount of time for the duration of the inquiry scan. | C.125 | E |
| Write Inquiry Scan Type command | 1.2 | This command is used to write the Inquiry Scan Type configuration parameter of the local BR/EDR Controller. The Inquiry Scan Type configuration parameter can set the inquiry scan to either normal or interlaced scan. | C.125 | E |
| Write Inquiry Transmit Power Level command | 2.1 + EDR | This command is used to write the transmit power level used to transmit the inquiry (ID) data packets. | C.127 | E |
| Write LE Host Support command | 4.0 | This command writes the LE Supported Host setting to the BR/EDR Controller. | C.153 | E |
| Write Link Policy Settings command | 1.1 | This command will write the Link Policy configuration parameter for the specified Connection_Handle. The Link Policy settings allow the Host to specify which Link Modes the Link Manager can use for the specified Connection_Handle. | C.141 | E |
| Write Link Supervision Timeout command | 1.1 | This command will write the value for the Link Supervision Timeout configuration parameter for the device. This parameter is used by the Controller to determine link loss. | O | E |
| Write Local Name command | 1.1 | This command provides the ability to modify the user-friendly name for the BR/EDR Controller. | M | E |
| Write Loopback Mode command | 1.1 | This command will write the value for the setting of the BR/EDR Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information. | C.123 | E |
| Write Num Broadcast Retransmissions command | 1.1 | This command will write the parameter value for the Number of Broadcast Retransmissions for the BR/EDR Controller. | O | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Write Page Scan Activity command | 1.1 | This command will write the value for Page Scan Interval and Page Scan Window configuration parameters. Page Scan Interval defines the amount of time between consecutive page scans. Page Scan Window defines the duration of the page scan. | M | E |
| Write Page Scan Type command | 1.2 | This command is used to write the page scan type of the local BR/EDR Controller. The Page Scan Type configuration parameter can set the page scan to either normal or interlaced scan. | C.154 | E |
| Write Page Timeout command | 1.1 | This command will write the value for the Page Reply Timeout configuration parameter, which allows the BR/EDR Controller to define the amount of time a connection request will wait for the remote device to respond before the local device returns a connection failure. | M | E |
| Write PIN Type command | 1.1 | This command is used for the Host to specify whether the Host supports variable PIN or only fixed PINs. | O | E |
| Write Scan Enable command | 1.1 | This command will write the value for the Scan Enable configuration parameter, which controls whether or not the BR/EDR Controller will periodically scan for page attempts and/or inquiry requests from other BR/EDR Controllers. | M | E |
| Write Secure Connections Host Support command | 4.1 | This command is used to write the Secure Connections Host Supports parameter to the Controller. | C.218 | E |
| Write Secure Connections Test Mode command | 4.1 | This command is used to put the Controller in a test mode where DM1 packets are not allowed to be used for ACL-U traffic and/or the contents of eSCO payloads can be looped back. | C.138 | E |
| Write Simple Pairing Debug Mode command | 2.1 + EDR | This command configures the BR/EDR Controller to use a predefined Diffie Hellman private key for Secure Simple Pairing to enable debug equipment to monitor the encrypted connection. | M | E |
| Write Simple Pairing Mode command | 2.1 + EDR | This command writes the Secure Simple Pairing mode setting in the BR/EDR Controller. | M | E |
| Write Stored Link Key command | 1.1 | This command provides the ability to write one or more link keys to be stored in the Controller. | O | E |

| Name | Vers. | Summary Description | BR/EDR | LE |
|---|---|---|---|---|
| Write Synchronization Train Parameters command | CSA4 | This command configures the Synchronization Train functionality in the BR/EDR Controller. | C.201 | E |
| Write Synchronous Flow Control Enable command | 1.1 | This command provides the ability to write the Synchronous Flow Control Enable setting. By using this setting, the Host can decide if the Controller will send HCI_Number_Of_Completed_Packets events for synchronous Connection_Handles. | C.135 | E |
| Write Voice Setting command | 1.1 | This command will write the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections. | C.134 | E |

*Table 3.1: Alphabetical list of commands and events*

C.1: Mandatory if the LE Controller supports transmitting packets, otherwise excluded.

C.2: Mandatory if the LE Controller supports receiving packets, otherwise excluded.

C.3: Mandatory if the LE Controller supports Connection State, otherwise excluded.

C.4: Mandatory if LE Feature (LE Encryption) is supported, otherwise excluded.

C.5: *Previously used*

C.6: Mandatory if LE Feature (Connection Parameters Request procedure) is supported, otherwise excluded.

C.7: Mandatory if LE Feature (LE Encryption) and LE Feature (LE Ping) are supported, otherwise excluded.

C.8: Mandatory if LE Feature (LE Data Packet Length Extension) is supported, otherwise optional.

C.9: Mandatory if LE Feature (LL Privacy) is supported, otherwise excluded.

C.10: Optional if LE Feature (LL Privacy) is supported, otherwise excluded.

C.11: Mandatory if LE Feature (LE 2M PHY) or LE Feature (LE Coded PHY) is supported, otherwise optional.

C.12: Mandatory if LE Feature (LE 2M PHY) or LE Feature (LE Coded PHY) or LE Feature (Stable Modulation Index - Transmitter) is supported, otherwise optional if the LE Controller supports transmitting packets, otherwise excluded.

C.13: Mandatory if LE Feature (LE 2M PHY) or LE Feature (LE Coded PHY) or LE Feature (Stable Modulation Index - Receiver) is supported, otherwise optional if the LE Controller supports receiving packets, otherwise excluded.

C.14: *Previously used*

C.15: Mandatory if LE Controller supports transmitting scannable advertisements, otherwise excluded.

C.16: Mandatory if LE Feature (Periodic Advertising) is supported and the LE Controller supports both Scanning State and Synchronization State, otherwise excluded.

C.17: Mandatory if LE Feature (Extended Advertising) is supported and the LE Controller supports Advertising State, otherwise excluded.

C.18: Mandatory if LE Feature (Periodic Advertising) is supported and the LE Controller supports Advertising State, otherwise excluded.

C.19: Mandatory if LE Feature (Extended Advertising) is supported and the LE Controller supports Scanning State, otherwise excluded.

C.20: Mandatory if LE Feature (Extended Advertising) is supported and the LE Controller supports Initiating State, otherwise excluded.

C.21: Mandatory if LE Feature (Periodic Advertising) is supported and the LE Controller supports Synchronization State, otherwise excluded.

C.22: Mandatory if the LE Controller supports sending Transmit Power in advertisements or if LE Feature (LE Power Control Request) is supported, otherwise optional.

C.23: Mandatory if LE Feature (LE Channel Selection Algorithm #2) is supported, otherwise excluded.

C.24: Mandatory if the LE Controller supports Connection State and either LE Feature (LL Privacy) or LE Feature (Extended Advertising) is supported, otherwise optional if the LE Controller supports Connection State, otherwise excluded.

C.25: Mandatory if LE Feature (Connection CTE Request) is supported, otherwise excluded.

C.26: Mandatory if LE Feature (Connection CTE Response) is supported, otherwise excluded.

C.27: Mandatory if LE Feature (Connectionless CTE Transmitter) is supported, otherwise excluded.

C.28: Mandatory if LE Feature (Connectionless CTE Receiver) is supported, otherwise excluded.

C.29: Mandatory if LE Feature (Connection CTE Response) or LE Feature (Connectionless CTE Transmitter) is supported, otherwise optional if the LE Controller supports transmitting packets, otherwise excluded.

C.30: Mandatory if LE Feature (Connection CTE Request) or LE Feature (Connectionless CTE Receiver) is supported, otherwise optional if the LE Controller supports receiving packets, otherwise excluded.

C.31: Mandatory if LE Feature (Connection CTE Request) or LE Feature (Connection CTE Response) or LE Feature (Connectionless CTE Transmitter) or LE Feature (Connectionless CTE Receiver) is supported, otherwise excluded.

C.32: Mandatory if LE Feature (Periodic Advertising Sync Transfer – Recipient) is supported, otherwise optional if LE Feature (Periodic Advertising) is supported and the LE Controller supports Synchronization State, otherwise excluded.

C.33: Mandatory if LE Feature (Periodic Advertising Sync Transfer – Sender) is supported and the LE Controller supports Scanning State, otherwise excluded.

C.34: Mandatory if LE Feature (Periodic Advertising Sync Transfer – Sender) is supported and the LE Controller supports Advertising State, otherwise excluded.

C.35: Mandatory if LE Feature (Periodic Advertising Sync Transfer – Recipient) is supported, otherwise excluded.

C.36: Mandatory if the LE Controller supports Central role or supports both Peripheral role and LE Feature (Channel Classification), otherwise optional if LE Feature (Extended Advertising) is supported and the LE Controller supports Advertising State or if LE Feature (Isochronous Broadcaster) is supported, otherwise excluded.

C.37: Mandatory if the LE Controller can change its sleep clock accuracy, otherwise excluded.

C.38: Mandatory if LE Feature (Connected Isochronous Stream - Central) or LE Feature (Connected Isochronous Stream - Peripheral) is supported, otherwise excluded.

C.39: Mandatory if LE Feature (Connected Isochronous Stream - Central) is supported, otherwise excluded.

C.40: Mandatory if LE Feature (Connected Isochronous Stream - Peripheral) is supported, otherwise excluded.

C.41: Mandatory if LE Feature (Isochronous Broadcaster) is supported, otherwise excluded.

C.42: Mandatory if LE Feature (Synchronized Receiver role) is supported, otherwise excluded.

C.43: Mandatory if the LE Controller supports Random Device Addresses, otherwise optional if the LE Controller supports transmitting packets, otherwise excluded.

C.44: Mandatory if LE Feature (Sleep Clock Accuracy Updates) and either LE Feature (Connected Isochronous Stream - Central) or LE Feature (Connected Isochronous Stream - Peripheral) are supported, otherwise optional if LE Feature (Sleep Clock Accuracy Updates) is supported, otherwise excluded.

C.45: Mandatory if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Isochronous Broadcaster) is supported, otherwise excluded.

C.46: Mandatory if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Synchronized Receiver role) is supported, otherwise excluded.

C.47: Mandatory if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Isochronous Broadcaster), or LE Feature (Synchronized Receiver role) is supported, otherwise excluded.

C.48: Mandatory if LE Feature (Channel Sounding Enhancement #1) is supported, otherwise optional if LE Feature (Channel Sounding) is supported, otherwise excluded.

C.49: Mandatory if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Connection Subrating), or LE Feature (Advertising Coding Selection), or LE Feature (Channel Sounding) is supported, otherwise optional.

C.50: Optional if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Synchronized Receiver role) is supported, otherwise excluded.

C.51: Mandatory if LE Feature (LE Power Control Request) is supported, otherwise excluded.

C.52: Mandatory if LE Feature (LE Path Loss Monitoring) is supported, otherwise excluded.

C.53: Mandatory if LE Feature (LE Power Control Request) is supported, otherwise optional if the LE Controller supports transmitting packets, otherwise excluded.

C.54: Mandatory if LE Feature (Synchronized Receiver) is supported, otherwise optional.

C.55: Mandatory if LE Feature (Connected Isochronous Stream - Central), or LE Feature (Connected Isochronous Stream - Peripheral), or LE Feature (Isochronous Broadcaster) is supported, otherwise optional if the LE Controller supports Connection State, otherwise excluded.

C.56: Optional if LE Feature (LE Encryption) is supported, otherwise excluded.

C.57: Mandatory if LE Feature (Connection Subrating) is supported, otherwise excluded.

C.58: Mandatory if LE Feature (Channel Classification) is supported, otherwise excluded.

C.59: Mandatory if the LE Controller supports Central role, otherwise excluded.

C.60: Mandatory if the LE Controller supports Central role and LE Feature (LE Encryption), otherwise excluded.

C.61: Mandatory if the LE Controller supports Peripheral role and LE Feature (LE Encryption), otherwise excluded.

C.62: Mandatory if the LE Controller supports Central role or supports both Peripheral role and LE Feature (Connection Parameters Request Procedure), otherwise excluded.

C.63: Mandatory if the LE Controller supports Scanning state and LE Feature (LL Privacy), otherwise excluded.

C.64: Optional if the Controller supports transmitting packets, otherwise excluded.

C.65: Mandatory if LE Set Extended Advertising Parameters command [v2] is supported, otherwise mandatory if LE Feature (Extended Advertising) is supported and the LE Controller supports Advertising State, otherwise excluded.

C.66: Mandatory if LE Feature (Advertising Coding Selection) is supported, otherwise optional if LE Feature (Extended Advertising) is supported and the LE Controller supports Advertising state, otherwise excluded.

C.67: Mandatory if LE Feature (Periodic Advertising with Responses - Advertiser) is supported, otherwise excluded.

C.68: Mandatory if LE Feature (Periodic Advertising with Responses - Scanner) is supported, otherwise excluded.

C.69: Mandatory if LE Feature (Periodic Advertising with Responses - Advertiser) or LE Feature (Periodic Advertising with Responses - Scanner) is supported, otherwise excluded.

C.70: Mandatory if the LE Controller supports LE Feature (LL Extended Feature Set), otherwise optional.

C.71: Mandatory if the LE Controller supports Connection State and LE Feature (LL Extended Feature Set), otherwise optional if the LE Controller supports Connection State, otherwise excluded.

C.72: Mandatory if the LE Controller supports the LE Read All Remote Features command, otherwise excluded.

C.73: Mandatory if LE Feature (Decision-Based Advertising Filtering) is supported and the LE Controller supports Advertising State, otherwise excluded.

C.74: Mandatory if LE Feature (Decision-Based Advertising Filtering) is supported and the LE Controller supports Scanning State, otherwise excluded.

C.75: Mandatory if LE Feature (Channel Sounding) is supported, otherwise excluded.

C.76: Mandatory if LE Feature (Channel Sounding) and initiator role are supported, otherwise excluded.

C.77: Mandatory if LE Feature (Shorter Connection Intervals) is supported, otherwise optional if the LE Set Host Feature command [v1] is supported, otherwise excluded.

C.78: Mandatory if LE Feature (Monitoring Advertisers) is supported, otherwise excluded.

C.79: Mandatory if LE Feature (Frame Space Update) is supported, otherwise excluded.

C.80: Mandatory if the LE Read Local P-256 Public Key command is supported, otherwise excluded.

C.81: Mandatory if the LE Generate DHKey command is supported, otherwise excluded.

C.82: Mandatory if the LE Generate DHKey command is supported, otherwise optional.

C.83: Mandatory if LE Feature (UTP OTA mode) is supported, otherwise excluded.

C.84: Mandatory if LE Feature (UTP HCI mode) is supported, otherwise excluded.

C.85: Mandatory if LE Feature (Shorter Connection Intervals) is supported, otherwise excluded.

C.86: Mandatory if LE Feature (LE Flushable ACL Data packets) is supported, otherwise excluded.

C.87: Mandatory if LE Feature (Periodic Advertising with Responses - Scanner) is supported, otherwise optional if the LE Periodic Advertising Report event [v1] is supported, otherwise excluded.

C.88: Mandatory if LE Feature (Periodic Advertising with Responses - Scanner) is supported, otherwise optional if the LE Periodic Advertising Sync Established event [v1] is supported, otherwise excluded.

C.89: Mandatory if LE Feature (Periodic Advertising with Responses - Scanner) is supported, otherwise optional if the LE Periodic Advertising Sync Transfer Received event [v1] is supported, otherwise excluded.

C.91 Optional if LE Feature (Channel Sounding) is supported, otherwise excluded.

C.92 Mandatory if the LE CS Set Default Security Requirements command is supported, otherwise excluded.

C.94: Mandatory if the LE Create Connection or LE Extended Create Connection command is supported, otherwise excluded.

C.95: Mandatory if the LE Request Peer SCA command is supported, otherwise excluded.

C.96: Optional if the LE Controller supports Connection State, otherwise excluded.

C.97: Mandatory if Advertising State is supported, otherwise excluded.

C.98: Mandatory if Scanning State is supported, otherwise excluded.

C.99: Mandatory if LE Generate DHKey command [v2] is supported, otherwise optional.

C.101: Mandatory if the Authentication Requested command is supported, otherwise excluded.

C.102: Mandatory if the Change Connection Link Key command is supported, otherwise excluded.

C.103: Mandatory if the Periodic Inquiry Mode command is supported, otherwise excluded.

C.104: Mandatory if the Read Clock Offset command is supported, otherwise excluded.

C.105: Mandatory if the Read Remote Version Information command is supported, otherwise excluded.

C.106: Mandatory if the Remote Name Request command is supported, otherwise excluded.

C.107: Mandatory if the Set Controller To Host Flow Control command is supported, otherwise excluded.

C.108: Mandatory if the Set MWS_PATTERN Configuration command is supported, otherwise optional.

C.109: Mandatory if the Set MWS Signaling command is supported, otherwise excluded.

C.110: Mandatory if the Set Triggered Clock Capture command is supported, otherwise excluded.

C.111: Mandatory if the Write Authentication Enable command is supported, otherwise excluded.

C.112: Mandatory if the Write Default Erroneous Data Reporting command is supported, otherwise excluded.

C.113: Mandatory if the Write Extended Inquiry Length command is supported, otherwise excluded.

C.114: Mandatory if the Write Extended Page Timeout command is supported, otherwise excluded.

C.115: Mandatory if the Write Inquiry Mode command is supported, otherwise excluded.

C.116: Mandatory if the Write LE Host Support command is supported, otherwise excluded.

C.117: Mandatory if the Write Link Supervision Timeout command is supported, otherwise excluded.

C.118: Mandatory if the Write Num Broadcast Retransmissions command is supported, otherwise excluded.

C.119: Mandatory if the Write Page Scan Type command is supported, otherwise excluded.

C.120: Mandatory if the Write PIN Type command is supported, otherwise excluded.

C.121: Mandatory if the Write Stored Link Key command is supported, otherwise excluded.

C.122: Mandatory if the Write Synchronous Flow Control Enable command is supported, otherwise excluded.

C.123: Mandatory if BR/EDR test mode is supported, otherwise excluded.

C.124: Mandatory if Data block based flow control is supported, otherwise excluded.

C.125: Mandatory if Inquiry Scan is supported, otherwise excluded.

C.126: Optional if Inquiry Scan is supported, otherwise excluded.

C.127: Mandatory if Inquiry is supported, otherwise excluded.

C.128: Optional if Inquiry is supported, otherwise excluded.

C.129: Mandatory if Truncated page state is supported, otherwise excluded.

C.130: *Previously used*

C.131: *Previously used*

C.132: Mandatory if multi-slot ACL packets are is supported, otherwise excluded.

C.133: Mandatory if HV2, HV3, or multi-slot or EDR ACL packets are is supported, otherwise excluded.

C.134: Mandatory if SCO or eSCO is supported, otherwise excluded.

C.135: Optional if SCO or eSCO is supported, otherwise excluded.

C.136: Optional if Slot Availability Mask is supported, otherwise excluded.

C.138: Mandatory if Secure Connections (Controller) is supported, otherwise optional if eSCO is supported, otherwise excluded.

C.139: Mandatory if the Controller is AFH capable in either role, otherwise excluded.

C.140: Mandatory if the Controller supports AFH classification in either role or is an AFH capable Central, otherwise optional if Connectionless Peripheral Broadcast - Transmitter is supported, otherwise excluded.

C.141: Mandatory if Role Switch, Hold mode, or Sniff mode is supported, otherwise excluded.

C.142: Mandatory if Secure Connections (Controller) or Secure Simple Pairing (Controller) is supported, otherwise excluded.

C.143: *Previously used*

C.144: Mandatory if Hold Mode or Sniff Mode is supported, otherwise excluded.

C.145: Mandatory if any event in event mask page 2 is supported, otherwise optional.

C.146: Mandatory if the Extended Inquiry Result event or the IO Capability Request event is supported, otherwise optional if Inquiry is supported, otherwise excluded.

C.147: Optional if the Inquiry Result with RSSI event is supported, otherwise excluded.

C.148: Optional if any of the Connection Complete, Connection Request, Extended Inquiry Result, Inquiry Result with RSSI, IO Capability Request, or Synchronous Connection Complete events is supported, otherwise excluded.

C.149: *Previously used*

C.150: *Previously used*

C.151: Mandatory if Secure Connections (Controller) and Ping are supported, otherwise excluded.

C.152: Mandatory if Power Control is supported, otherwise optional.

C.153: Mandatory if LE supported in the Controller, otherwise optional.

C.154: Mandatory if Interlaced Page Scan is supported, otherwise optional.

C.155: Mandatory if the Write Authenticated Payload Timeout command is supported, otherwise excluded.

C.156: Mandatory if the Read Local Supported Codecs command [v2] is supported, otherwise excluded.

C.157: Mandatory if the Read Local Supported Codecs command [v2] is supported, otherwise optional.

C.158: Mandatory if the Set Min Encryption Key Size command is supported, otherwise optional.

C.159: Optional if the LE CIS Established event [v1] is supported, otherwise excluded.

C.201: Mandatory if Connectionless Peripheral Broadcast - Transmitter is supported, otherwise excluded.

C.202: Mandatory if Connectionless Peripheral Broadcast - Receiver is supported, otherwise excluded.

C.203: *Previously used*

C.204: *Previously used*

C.205: Mandatory if Extended Inquiry Response is supported, otherwise excluded.

C.206: Mandatory if Erroneous Synchronous Data Reporting is supported, otherwise excluded.

C.212: Mandatory if Role Switch is supported, otherwise excluded.

C.213: Mandatory if Hold mode is supported, otherwise excluded.

C.214: Mandatory if Sniff mode is supported, otherwise excluded.

C.215: Mandatory if Broadcast Encryption is supported, otherwise excluded.

C.217: Mandatory if BR/EDR Enhanced Power Control is supported, otherwise excluded.

C.218: Mandatory if Secure Connections (Controller) is supported, otherwise excluded.

C.219: Mandatory if Slot Availability Mask is supported, otherwise excluded.

C.220: Mandatory if LMP Extended Features mask is supported, otherwise excluded.

C.221: Mandatory if Sniff subrating is supported, otherwise excluded.

C.222: Mandatory if any event assigned a bit number of 64 or above is supported, otherwise optional.

C.223: Mandatory if any command listed in octet 64 or above is supported, otherwise optional.

## 3.1 LE Controller requirements

### 3.1.1 Legacy and extended advertising

Table 3.2 lists the legacy and extended advertising commands and events.

If a Controller supports any legacy advertising command or event listed in the table and also supports the LE Feature (Extended Advertising), it shall support the corresponding extended advertising command or event in the same row of the table.

If, since the last power-on or reset, the Host has ever issued a legacy advertising command and then issues an extended advertising command, or has ever issued an extended advertising command and then issues a legacy advertising command, the Controller shall return the error code *Command Disallowed* (0x0C).

A Host should not issue legacy commands to a Controller that supports the LE Feature (Extended Advertising).

| Legacy advertising command or event | Extended advertising command or event |
|---|---|
| HCI_LE_Advertising_Report event | HCI_LE_Extended_Advertising_Report event |
| HCI_LE_Directed_Advertising_Report event | HCI_LE_Extended_Advertising_Report event |
| HCI_LE_Set_Advertising_Parameters command | HCI_LE_Set_Extended_Advertising_Parameters command |
| HCI_LE_Read_Advertising_Physical_Channel_Tx_Power command | *none* |
| HCI_LE_Set_Advertising_Data command | HCI_LE_Set_Extended_Advertising_Data command |
| HCI_LE_Set_Scan_Response_Data command | HCI_LE_Set_Extended_Scan_Response_Data command |
| HCI_LE_Set_Advertising_Enable command | HCI_LE_Set_Extended_Advertising_Enable command |
| *none* | HCI_LE_Read_Maximum_Advertising_Data_Length command |
| *none* | HCI_LE_Read_Number_of_Supported_Advertising_Sets command |
| *none* | HCI_LE_Remove_Advertising_Set command |
| *none* | HCI_LE_Clear_Advertising_Sets command |
| *none* | HCI_LE_Set_Periodic_Advertising_Parameters command |
| *none* | HCI_LE_Set_Periodic_Advertising_Data command |

| Legacy advertising command or event | Extended advertising command or event |
|---|---|
| *none* | HCI_LE_Set_Periodic_Advertising_Enable command |
| HCI_LE_Set_Scan_Parameters command | HCI_LE_Set_Extended_Scan_Parameters command |
| HCI_LE_Set_Scan_Enable command | HCI_LE_Set_Extended_Scan_Enable command |
| HCI_LE_Create_Connection command | HCI_LE_Extended_Create_Connection command |
| *none* | HCI_LE_Periodic_Advertising_Create_Sync command |
| *none* | HCI_LE_Periodic_Advertising_Create_Sync_Cancel command |
| *none* | HCI_LE_Periodic_Advertising_Terminate_Sync command |
| *none* | HCI_LE_Add_Device_To_Periodic_Advertiser_List command |
| *none* | HCI_LE_Remove_Device_From_Periodic_Advertiser_List command |
| *none* | HCI_LE_Clear_Periodic_Advertiser_List command |
| *none* | HCI_LE_Read_Periodic_Advertiser_List_Size command |
| *none* | HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command |
| *none* | HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters command |
| *none* | HCI_LE_Periodic_Advertising_Sync_Transfer_Received event |

*Table 3.2: Legacy and extended advertising commands and events*

## 3.2 Underlying Support

Except as stated in this section, if a command or event is supported by a Controller then the feature underlying the command or event shall also be fully supported. If the feature applies to more than one transport, it shall be supported on all supported transports.

If the Controller supports ACL connections on BR/EDR but does not support Connection State on LE then, for each command or event that it supports that has a handle parameter, it shall only support the underlying functionality on BR/EDR (such Controllers will not support the commands or events that create LE connections and therefore all valid handles will represent BR/EDR connections).

For each of the commands and events in Table 3.3, the requirements in Table 3.1 shall be evaluated separately for each supported transport. If the requirement for a given supported transport evaluates to Mandatory, or evaluates to Optional and the Controller supports the command or event on that transport, then the underlying feature shall be fully supported on that transport. Otherwise the Controller shall not support the underlying feature on that transport and:

- For a command, if the Host issues the command with the Handle or Connection_Handle parameter referring to a connection on that transport, the Controller shall return the error code *Unsupported Feature or Parameter Value* (0x11).
- For an event, the Controller shall not generate that event with the Connection_Handle parameter referring to a connection on that transport.

| |
|---|
| HCI_Authenticated_Payload_Timeout_Expired event |
| HCI_Encryption_Change event |
| HCI_Encryption_Key_Refresh_Complete event |
| HCI_Read_Authenticated_Payload_Timeout command |
| HCI_Read_Remote_Version_Information command |
| HCI_Read_Remote_Version_Information_Complete event |
| HCI_Read_RSSI command |
| HCI_Read_Transmit_Power_Level command |
| HCI_Write_Authenticated_Payload_Timeout command |

*Table 3.3: Commands and events that have separate requirements for each transport*

## 3.3 Feature Exchange

If a command has a Connection_Handle parameter, returns an HCI_Command_Status event followed by a completion event (see Section 4.4), and the specification of that command does not refer to feature exchange, then the command may perform feature exchange on the specified connection before any other procedure in order to determine whether the peer device supports a necessary feature.

# 4 HCI FLOW CONTROL

Flow control for data shall be used in the direction from the Host to the Controller to avoid overflowing the Controller data buffers with data destined for a remote device (using a Connection_Handle) that is not responding. The Host manages the data buffers of the Controller. Packet based flow control is the default for BR/EDR traffic and the only option for LE traffic. Flow control for data moving from the Controller to the Host may be in accordance with Section 4.2.

Command flow control is covered in Section 4.4 and Section 4.5.

## 4.1 Host to Controller data flow control

Two methods of data flow control are defined: 'packet-based' flow control and 'data-block-based' flow control, known as buffer management. Selection of the data flow control mechanism is performed with the HCI_Write_Flow_Control_Mode command (see Section 7.3.73).

If a BR/EDR/LE Controller implements separate buffers for ACL Data:

1. The Host shall use the HCI_LE_Read_Buffer_Size command to determine the buffers that are used for ACL Data on an LE-U logical link.
2. The Host shall use separate packet based flow control for each set of buffers.
3. The Connection_Handle contained in the ACL Data packet shall be used by the Controller to determine which set of buffers to use and the logical link (ACL-U, APB-U, or LE-U) over which the data is to be sent.

If a BR/EDR/LE Controller does not implement separate buffers, then all ACL Data shall use the BR/EDR buffer management as described below, and only the logical link (ACL-U, APB-U, or LE-U) shall be determined by the Connection_Handle.

A packet is said to have completed when the Controller no longer needs the buffer space to store the data from the packet and has freed the corresponding buffer for re-use. This nominally happens when the data is transmitted or flushed, but can be delayed for implementation reasons or can happen early if the implementation transfers the data to other storage.

For each individual Connection_Handle, the data shall be sent to the Controller in HCI Data packets in the order in which it was provided by the Host and shall arrive at the Controller in that order.

### 4.1.1 Packet-based data flow control

When the packet based flow control mechanism is enabled, on initialization, a Host that supports LE shall issue the HCI_LE_Read_Buffer_Size command (see Section 7.8.2). Two of the return parameters of this command determine the maximum size of HCI ACL (excluding header) Data packets that can be used to transmit ACL data for an LE transport sent from the Host to the Controller. There is an additional return parameter that specifies the total number of HCI ACL Data packets that the Controller may have waiting for transmission in those buffers. A Controller that supports BR/EDR and LE may return zero for the total number of HCI ACL packets used to transmit ACL data for an LE transport. In this case the Host shall then send all BR/EDR and LE data using the HCI ACL Data packets into the buffers identified using the HCI_Read_Buffer_Size command. A Controller that does not support BR/EDR shall not return zero for the total number of HCI ACL packets used to transmit ACL data for an LE transport.

When the packet-based flow control mechanism is enabled, on initialization, a Host that supports isochronous data over HCI in either the Connected Isochronous Stream Central role, Connected Isochronous Stream Peripheral role, or Isochronous Broadcaster role shall issue the HCI_LE_Read_Buffer_Size command.

An HCI ISO Data packet is used to transfer isochronous data between the Host and Controller for transmission of isochronous data on an isochronous transport. The ISO_Data_Packet_Length parameter of this command specifies the maximum buffer size for each HCI ISO Data packet (excluding the header but including optional fields such as ISO_SDU_Length). The return parameter Total_Num_ISO_Data_Packets of this command specifies the maximum number of HCI ISO Data packets that the Controller can have waiting for transmission in its buffers.

Note: The ISO_Data_Packet_Length and Total_Num_ISO_Data_Packets return parameters are only available when using v2 or above of the HCI_LE_Read_Buffer_Size command (see Section 7.8.2).

In a BR/EDR Controller, when the packet based flow control mechanism is enabled, on initialization, the Host shall issue the HCI_Read_Buffer_Size command. Two of the return parameters of this command determine the maximum size of HCI ACL and Synchronous Data packets (excluding header) sent from the Host to the Controller. There are also two additional return parameters that specify the total number of HCI ACL and Synchronous Data packets that the Controller may have waiting for transmission in its buffers. A Controller that supports BR/EDR shall not return zero for the total number of HCI ACL Data packets or their maximum size. A Controller that supports SCO or eSCO over HCI shall not return zero for the total number of HCI Synchronous Data packets or their maximum size.

When there is at least one connection to another device, the Controller is broadcasting a BIG, or when in local loopback mode on a BR/EDR Controller, the Controller shall

use the HCI_Number_Of_Completed_Packets event to control the flow of data from the Host. This event contains a list of Connection_Handles and a corresponding number of HCI Data packets and/or HCI ISO Data packets that have been completed since the previous time the event was returned (or since the connection was established, if the event has not been returned before for a particular Connection_Handle).

The Host chooses the Connection_Handles for the following HCI Data packets and/or HCI ISO Data packets based on the information returned in this event, and/or the HCI_LE_Read_Buffer_Size commands.

Every time it has sent an HCI Data packet, the Host shall reduce its count of the free buffer space for the corresponding link type (ACL, SCO, or eSCO) in the Controller by one HCI Data packet.

Every time the Host sends an HCI ISO Data packet, the Host shall reduce its count of the free buffer space for the corresponding logical link type (LE-S or LE-F) in the Controller by one HCI ISO Data packet.

Each HCI_Number_Of_Completed_Packets event received by the Host provides information about how many HCI Data packets and/or HCI ISO Data packets have been completed for each Connection_Handle since the previous HCI_Number_Of_Completed_Packets event was sent to the Host. It can then calculate the actual current buffer usage.

The Host shall not send an HCI Data packet or HCI ISO Data packet to the Controller when its count of the free buffer space for the corresponding link type is zero.

When the Controller has completed one or more HCI Data packet(s) and/or HCI ISO Data packet(s) it shall send an HCI_Number_Of_Completed_Packets event to the Host, until it finally reports that all the pending HCI Data packets have been completed. The frequency at which this event is sent is manufacturer specific.

Note: The HCI_Number_Of_Completed_Packets events will not report on synchronous Connection_Handles if Synchronous Flow Control is disabled. (See Section 7.3.36 and Section 7.3.37.)

### 4.1.2 Data-block-based data flow control

When the data-block-based flow control mechanism is enabled, on initialization the Host shall issue the Read Data Block Size command. Two of the return parameters of this command determine the maximum size of HCI ACL Data packets (excluding header) sent from the Host to the Controller. A further return parameter specifies the total number of HCI ACL Data packets that the Controller may have waiting for transmission in its buffers.

The Controller shall use the HCI_Number_Of_Completed_Data_Blocks event to control the flow of data from the Host. This event contains a list of Handles and a corresponding number of HCI Data packets that have been completed since the previous time the event was returned (or since the link was established, if the event has not been returned before for a particular Handle).

Based on the information returned in this event, and the return parameters of the HCI_Read_Data_Block_Size command that specify the total number of HCI ACL Data packets that can be stored in the Controller, the Host decides for which Handles the following HCI Data packets should be sent.

Every time it has sent an HCI Data packet, the Host shall reduce its count of the free buffer space for the corresponding ACL link type in the Controller by one HCI Data packet.

Each HCI_Number_Of_Completed_Data_Blocks event received by the Host provides information about how many HCI Data packets have been completed for each Handle since the previous HCI_Number_Of_Completed_Data_Blocks event was sent to the Host. It can then calculate the actual current buffer usage.

The Host shall not send an HCI Data packet to the Controller when its count of the free buffer space for the corresponding link type is zero.

When the Controller has completed one or more HCI Data packet(s) it shall send an HCI_Number_Of_Completed_Data_Blocks event to the Host until it finally reports that all the pending HCI Data packets have been completed. The frequency at which this event is sent is manufacturer specific.

## 4.2 Controller to Host data flow control

In some implementations, flow control may also be necessary in the direction from the Controller to the Host. The HCI_Set_Controller_To_Host_Flow_Control command can be used to turn flow control on or off in that direction.

On initialization, the Host uses the HCI_Host_Buffer_Size command to notify the Controller about the maximum size of HCI ACL and Synchronous Data packets sent from the Controller to the Host. The command also contains two additional command parameters to notify the Controller about the total number of ACL and Synchronous Data packets that can be stored in the data buffers of the Host.

The Host uses the HCI_Host_Number_Of_Completed_Packets command in exactly the same way as the Controller uses the HCI_Number_Of_Completed_Packets event as was previously described in this section, but a packet is completed when the Host is ready to free the corresponding buffer.

The Controller shall not send an HCI Data packet to the Host when its count of the free buffer space for the corresponding link type is zero.

The HCI_Host_Number_Of_Completed_Packets command is a special command for which no command flow control is used, and which can be sent anytime there is a connection or when in local loopback mode. The command also has no event after the command has completed. This makes it possible for the flow control to work in exactly the same way in both directions, and the flow of normal commands will not be disturbed.

For each individual Connection_Handle, the data shall be sent to the Host in HCI Data packets in the order in which it was provided by the Controller and shall arrive at the Host in that order.

## 4.3 Disconnection behavior

When the Host receives an HCI_Disconnection_Complete event, the Host shall assume that all unacknowledged HCI Data packets that have been sent to the Controller for the returned Handle have been flushed, and that the corresponding data buffers have been freed. A Controller does not have to notify the Host about this in an HCI_Number_Of_Completed_Packets or an HCI_Number_Of_Completed_Data_Blocks event before the disconnection event.

If flow control is also enabled in the direction from the Controller to the Host, the Controller may, after it has sent an HCI_Disconnection_Complete event, assume that the Host will flush its data buffers for the sent Handle when it receives the HCI_Disconnection_Complete event. The Host does not have to notify the Controller about this in an HCI_Host_Number_Of_Completed_Packets command.

## 4.4 Command flow control

On initial power-on, and after a reset, the Host shall send a maximum of one outstanding HCI Command packet until an HCI_Command_Complete or HCI_Command_Status event has been received.

The HCI_Command_Complete and HCI_Command_Status events contain a parameter called Num_HCI_Command_Packets, which indicates the number of HCI Command packets the Host is currently allowed to send to the Controller. The Controller may buffer one or more HCI Command packets, but the Controller shall start performing the commands in the order in which they are received. The Controller can start performing a command before it completes previous commands. Therefore, the commands do not always complete in the order they are started.

To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller may generate an HCI_Command_Complete or HCI_Command_Status event with the Command Opcode set to 0x0000 and the Num_HCI_Command_Packets

parameter set to 1 or more. Command Opcode 0x0000 is a special value indicating that this event is not associated with a command sent by the Host. The Controller can send an HCI_Command_Complete or HCI_Command_Status event with Command Opcode 0x0000 at any time to change the number of outstanding HCI Command packets that the Host can send before waiting. If the Controller generates an HCI_Command_Complete or HCI_Command_Status event with Num_HCI_Command_Packets set to zero, then the Host shall stop sending commands.

HCI commands may take different amounts of time to be completed. Therefore, the results of commands will be reported back to the Host in the form of an event. For example, for most HCI commands the Controller will generate the HCI_Command_Complete event when a command is completed. This event contains the return parameters for the completed HCI command. For enabling the Host to detect errors on the HCI-Transport Layer, there needs to be a timeout between the transmission of the Host's command and the reception of the Controller's response (e.g. an HCI_Command_Complete or HCI_Command_Status event). Since the maximum response timeout is strongly dependent on the HCI-Transport Layer used, it is recommended to use a default value of one second for this timer. This amount of time is also dependent on the number of commands unprocessed in the command queue.

There are two separate patterns of command execution. For the first type (used by those commands which are expected to complete quickly and are carried out entirely in the local Controller), the Controller shall send the Host an HCI_Command_Complete event when it has completed the command. For the second type (used by those commands that are expected to take a significant length of time, usually because they involve interaction with a peer device), the Controller shall send the Host an HCI_Command_Status event when it has received the command and checked the parameters. When the actions associated with the command have finished, a separate event that is associated with the command (the "completion event") shall be sent by the Controller to the Host. Between the times that these two events are generated (even if they have not yet been received by the Host), this type of command is described as "pending".

If a command of the second type does not begin to execute (for example, if there was a parameter error or the command is currently not allowed), the HCI_Command_Status event shall be returned with the appropriate error code in the Status parameter and no completion event is generated.

When a Connection_Handle is deleted and there are pending commands relating to that Connection_Handle, the Controller may return the completion event for each command to the Host. Each such event shall have a non-zero status and shall precede the event indicating the deletion of the Connection_Handle. No events for that Connection_Handle shall be sent after the event indicating the deletion of the Connection_Handle.

## 4.5 Command error handling

### 4.5.1 Generic error handling

Unless explicitly stated otherwise in the description of a command, any error in any parameter means that the command will not begin to execute; it will only return an error code. If more than one error code is applicable, the implementation shall choose one of them.

If an error occurs for a command for which an HCI_Command_Complete event is returned, the error shall be reported in the Status parameter.

If an error occurs for a command for which an HCI_Command_Status event and a completion event are returned, there are two possibilities; which happens will depend on the specific error. If the error is one which means the command will not begin to execute (including any errors in parameters), the error shall be returned as a non-zero Status parameter in the HCI_Command_Status event and no completion event will be returned. If the error is not detected until after the HCI_Command_Status event has been generated, the HCI_Command_Status event shall have the Status parameter set to zero and the error shall be returned in the Status parameter of the completion event.

If the Controller does not support an issued command, it shall return the error code *Unknown HCI Command* (0x01) in the Status parameter of either an HCI_Command_Complete event or an HCI_Command_Status event; which event is used is vendor-specific.

If an error occurs for a command for which an HCI_Command_Complete event is returned, the Return Parameters field may only contain some of the return parameters specified for the command. The Status parameter, which explains the error reason and which is the first return parameter, shall always be returned. If there is a Handle parameter or a BD_ADDR parameter right after the Status parameter, this parameter shall also be returned so that the Host can identify to which instance of a command the HCI_Command_Complete event belongs. In this case, the Handle or BD_ADDR parameter shall have exactly the same value as that in the corresponding command parameter. It is implementation specific whether more parameters will be returned in case of an error; if they are not, the event will be shorter than if they were.

The above also applies to commands that have associated command specific completion events with a Status parameter in their completion event, with the exceptions shown in Table 4.1, which indicates the only parameters (other than Status) that are valid. The validity of other parameters is likewise implementation specific for failed commands in this group, but they shall be sent in any case.

| Event | Valid parameters |
|---|---|
| Connection_Complete | BD_ADDR |
| Synchronous_Connection_Complete | BD_ADDR |
| LE_Connection_Complete | *none* |
| LE_Enhanced_Connection_Complete | *none* |
| LE_Periodic_Advertising_Sync_Established | *none* |
| LE_CIS_Established | Connection_Handle |
| LE_CIS_Request | *none* |
| LE_Create_BIG_Complete | BIG_Handle |
| LE_Terminate_BIG_Complete | BIG_Handle |
| LE_BIG_Sync_Established | BIG_Handle |
| LE_Request_Peer_SCA_Complete | Connection_Handle |

*Table 4.1: Valid parameters for command completion events reporting an error*

For the purposes of this section, the Subevent_Code parameter of the HCI_LE_Meta event (see Section 7.7.65) is not treated as a parameter and is always valid.

Note: The BD_ADDR return parameter of the command HCI_Read_BD_ADDR is not used to identify to which instance of the HCI_Read_BD_ADDR command the HCI_Command_Complete event belongs. It is optional for the Controller to return this parameter in case of an error.

Note: The LE_Generate_DHKey_Complete event (see Section 7.7.65.9) has special rules in the case of an error.

### 4.5.2 Error handling specific to a command

Some controller command descriptions in Section 7 include a table describing error conditions and error handling specific to that command in addition to the command error handling described in Section 4.5.1. Table 4.2 describes the types of command error handling used in the command-specific error conditions and error handling tables.

| Type | Error Condition | Command Rejected with Error Code | Return Specified Error Code |
|---|---|---|---|
| MC | The Controller shall return the specified error code and not execute the command. | Mandatory | Mandatory |
| M | The Controller shall return an error code and not execute the command. The error code returned should be the one specified. | Mandatory | Recommended |

| Type | Error Condition | Command Rejected with Error Code | Return Specified Error Code |
|---|---|---|---|
| RC | The Controller should return an error code and not execute the command. If it does return an error code, it shall be the one specified. | Recommended | Mandatory |
| R | The Controller should return an error code and not execute the command. If it does return an error code, it should be the one specified. | Recommended | Recommended |
| OC | The Controller shall return either success or the specified error code. | Optional | Mandatory |
| O | The Controller shall return either success or an error code. If it returns an error code, it should be the one specified. | Optional | Recommended |

*Table 4.2: Types of error handling*

An error situation occurs if any of the conditions in the table apply at the moment the Controller starts to process the command parameters (if any) and before the Controller starts to execute the command. For example, an error condition because a conflicting mode was enabled would not apply if that mode is disabled between the Host sending the command to the HCI transport and the Controller starting to check the command parameters.

## 4.6 LMP transaction and LL procedure collisions

If the Host issues a command that returns a successful HCI_Command_Status event but then triggers an LMP transaction or Link Layer procedure that terminates with the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23) because the peer has also initiated the same transaction or procedure, then the Controller shall take one of the following actions:

- The Controller shall wait until the peer-initiated transaction or procedure has completed and use the result to generate the command-specific completion event. In this case, the collision is not reported to the Host.
- The Controller shall generate the command-specific completion event with the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23).

## 4.7 LE Host and Controller synchronization

To synchronize the timing of isochronous data received in the Host with the timing of the isochronous data received in the Controller from the isochronous physical channel, the Controller can include a time stamp (see [Vol 6] Part G, Section 1) in each HCI ISO Data packet. Similarly, the Host can include a time stamp in each SDU (see [Vol 6] Part G, Section 1) sent to the Controller.

A Host can use the HCI_LE_Read_ISO_TX_Sync command at any time to read the time stamp and packet sequence number of the last SDU scheduled for transmission.

## 4.8 Versioned events

If an event has more than one version and the event is generated, the Controller shall use the latest version that is both supported and enabled (“unmasked”) in the relevant event mask (see Section 7.3.1, Section 7.3.69, and Section 7.8.1).

# 5 HCI DATA FORMATS

## 5.1 Correctness

The Controller shall set the value of return and event parameters so as to correctly represent the data or circumstances being reported. For example, in the HCI_Connection_Complete event (see Section 7.7.3), the value of the Link_Type parameter must correctly indicate the type of connection being reported.

## 5.2 Data and parameter formats

- All values are in binary and hexadecimal little-endian formats unless otherwise noted.
- Unless noted otherwise, the order of parameters in an HCI Command packet or HCI Event packet is the order the parameters are listed in the command or event.
- Arrayed parameters are specified using the following notation: ParameterA[i]. If more than one set of arrayed parameters are specified (e.g. ParameterA[i], ParameterB[i]), then, unless noted otherwise, the order of the parameters are as follows: ParameterA[0], ParameterB[0], ParameterA[1], ParameterB[1], ParameterA[2], ParameterB[2], ... ParameterA[n], ParameterB[n]. The description of an arrayed parameter will actually describe a single element of the array.
- Unless noted otherwise, all parameter values are sent and received in little-endian format (i.e. for multi-octet parameters the rightmost (Least Significant Octet) is transmitted first).
- Most command and event parameters that are not-arrayed and all elements in an arrayed parameter have fixed sizes (an integer number of octets). Where a parameter, or an element of an arrayed parameter, has a variable length, this will be noted in the specific command; the length will then be specified in another parameter.The parameters and the size of each not-arrayed parameter (or of each element in an arrayed parameter) contained in a command or an event is specified for each command or event. The number of elements in an arrayed parameter is not fixed.
  - Where a command or event has one or more arrayed parameters or parameters with variable length, the maximum value specified for the parameter that determines the size can result in an array or parameter that is too big to fit in the HCI command or event packet. When this happens, the effective maximum size will be less than the specified maximum and can depend on the size or value of other parameters.
- Where bit strings are specified, the low order bit is the right hand bit, e.g. 0 is the low order bit in 0b10.

- Where a parameter value is described as "0xXX" or "N = 0xXX" (with an appropriate number of "X"s), then the description applies to all possible values other than any for which a separate description is given. Where the description specifies a range, values outside that range (if any) are reserved for future use.
- Parameter values or opcodes that an implementation does not know how to interpret shall be ignored and the operation that is being attempted shall be completed with the correct signaling. The Host or Controller shall not stop functioning because of receiving a reserved value.
- If not all possible values for a parameter are described, then all omitted values or bit numbers are reserved for future use.
- Unless noted otherwise, such as in a "Mandatory Range" statement or because a value relates to an optional feature that is not supported, the Controller shall support all valid values of all parameters (values reserved for future use are not valid).

## 5.3 IDs and Handles

Two types of identifiers are used in HCI commands and events.

IDs, assigned by the Host, are identifiers used between two peer devices and sent over the air in PDUs. Each different type of ID has a separate number space. The CIS_ID has a separate number space for each CIG_ID.

Handles, assigned by either the Host in a command or by the Controller in an event, are identifiers used between the Host and the Controller but not sent over the air. Some handles share the same number spaces and others have separate number spaces.

When a device allocates an ID or Handle for a new object (e.g. when the Controller allocates a Connection_Handle for a new ACL connection), it shall not use a handle that is currently allocated to another object in the same number space.

After a device deletes an ID or Handle (e.g. after the Controller deletes a Connection_Handle for a disconnected ACL connection), it may reuse the ID or Handle but should use an unused ID or Handle in the same number space or, if all values have been used, then the least recently used ID or Handle in the same number space. The device should not reuse an ID or Handle while there is a possibility that an entity associated with that ID or Handle still exists.

### 5.3.1 Controller handles

Connection_Handles, Sync_Handles, Advertising_Handles, and BIG_Handles are Controller Handles used to identify logical channels between the Host and the Controller.

Connection_Handles are assigned by the Controller when a new logical transport is created or reserved and reported to the

Host in one of the following events: HCI_Connection_Complete, HCI_Synchronous_Connection_Complete, HCI_LE_Connection_Complete, HCI_LE_Enhanced_Connection_Complete, HCI_LE_CIS_Request, HCI_LE_Create_BIG_Complete, HCI_LE_BIG_Sync_Established, or HCI_Command_Complete events following the HCI_LE_Set_CIG_Parameters command. Broadcast Connection_Handles that use the BR/EDR transport are handled differently, and are described in Section 5.3.1.1.

Sync_Handles are assigned by the Controller when a new logical transport is created and reported to the Host in the HCI_LE_Periodic_Advertising_Sync_Established event.

Advertising_Handles are assigned by the Host when a new advertising set is created by using the HCI_LE_Set_Extended_Advertising_Parameters command.

BIG_Handles are assigned by the Host when a new BIG is created using one of the following commands: HCI_LE_Create_BIG, HCI_LE_Create_BIG_Test, and LE_BIG_Create_Sync.

All connection handles that are assigned by the Controller shall be derived from the same number space. Sync handles shall be assigned by the Controller from a separate number space.

Advertising handles and BIG handles are assigned by the Host from separate number spaces.

#### 5.3.1.1 Broadcast Connection_Handles

The first time the Host sends an HCI ACL Data packet with Broadcast_Flag set to 0b01 (active Peripheral broadcast) after a power-on or a reset, the value of the Connection_Handle parameter shall be a value which is not currently assigned by the Controller.

The BR/EDR Controller shall then continue to use the same Connection_Handle for broadcast until a reset is made. The Controller shall not re-allocate a Connection_Handle that it knows is used for broadcast.

In some situations, it may happen that the Controller sends a Connection Complete event before having interpreted a Broadcast packet received from the Host, and that the Connection_Handles of both Connection Complete event and HCI ACL Data packet are the same. This conflict is avoided as follows:

If a Connection Complete event is received containing the Connection_Handle used for broadcast, the Host shall wait before sending any packets for the new connection until it receives a Number Of Completed Packets event indicating that there are no pending broadcast packets belonging to the Connection_Handle. The Host shall also change the

Connection_Handle used for broadcast to a Connection_Handle which is currently not assigned by the Controller. This Connection_Handle shall then be used for all following broadcasts of until a reset is performed or the same conflict situation happens again. However, this will occur very rarely.

By following this procedure, the Controller can distinguish between the Broadcast message sent by the Host and the new connection made (this could be even a new synchronous link) even though the Connection_Handles are the same.

For an HCI ACL Data packet sent from the Controller to the Host where the Broadcast_Flag is 01, the Connection_Handle parameter should contain the Connection_Handle for the ACL connection to the Central that sent the broadcast.

For Connectionless Peripheral Broadcast, no Connection_Handle is assigned.

### 5.3.2 [This section is no longer used]

## 5.4 Exchange of HCI-specific information

The Host Controller Transport Layer provides transparent exchange of HCI specific information. These transporting mechanisms provide the ability for the Host to send HCI commands, receive HCI events, and send and receive data to the Controller. Since the Host Controller Transport Layer provides transparent exchange of HCI-specific information, the HCI specification specifies the format of the commands, events, and data exchange between the Host and the Controller(s). The next sections specify the HCI packet formats.

### 5.4.1 HCI Command packet

The HCI Command packet is used to send commands to the Controller from the Host. The format of the HCI Command packet is shown in Figure 5.1, and the definition of each field is explained below.

Controllers shall be able to accept HCI Command packets with up to 255 bytes of data excluding the HCI Command packet header. The HCI Command packet header is the first 3 octets of the packet.

Each command is assigned a 2 byte Opcode used to uniquely identify different types of commands. The Opcode parameter is divided into two fields, called the Opcode Group Field (OGF) and Opcode Command Field (OCF). The OGF occupies the upper 6 bits of the Opcode, while the OCF occupies the remaining 10 bits. Any opcode not mentioned in this Part is reserved for future use.

The OGF value 0x3E is reserved for specification development purposes.

The OGF of 0x3F is reserved for vendor-specific debug commands.

The organization of the opcodes allows additional information to be inferred without fully decoding the entire Opcode.

Note: The OGF composed of all ‘ones’ has been reserved for vendor-specific debug commands. These commands are vendor-specific and are used during manufacturing, for a possible method for updating firmware, and for debugging.

On receipt of a Vendor Specific Debug command the Controller should respond with either:

1. An HCI_Command_Status event. If the status indicates success (Section 7.7.15) then this event shall be followed by an HCI event with Event Code field of 0xFF (Section 5.4.4).
2. An HCI_Command_Complete event specifying the corresponding Vendor Specific Debug command opcode.

The Host shall assume that sending of an HCI_Vendor_Specific_Debug command will consume an HCI command credit.

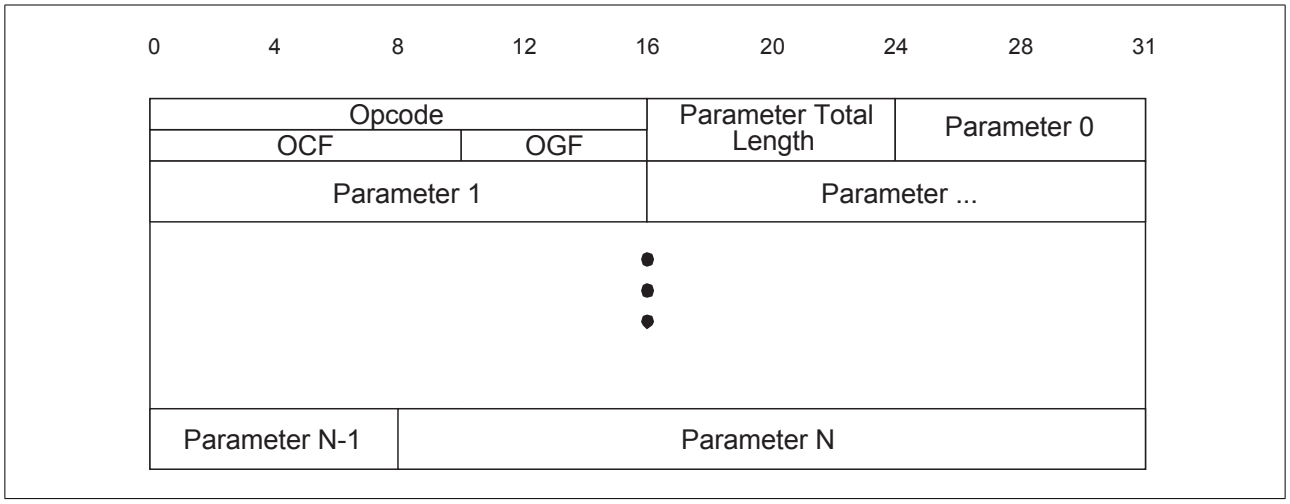



*Figure 5.1: HCI Command packet*

*Opcode:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | OGF Range (6 bits): 0x00 to 0x3F (0x3F reserved for vendor-specific debug commands)<br>OCF Range (10 bits): 0x0000 to 0x03FF |

*Parameter_Total_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Lengths of all of the parameters contained in this packet measured in octets. (N.B.: total length of parameters, not number of parameters) |

*Parameter 0 - N:* *Size: Parameter_Total_Length*

| Parameter Description |
|---|
| Each command has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each command. Each parameter is an integer number of octets in size. |

### 5.4.2 HCI ACL Data packets

HCI ACL Data packets are used to exchange data between the Host and Controller. There are two types of HCI ACL Data packets:

- Automatically-Flushable
- Non-Automatically-Flushable

Automatically-Flushable HCI ACL Data packets are flushed based on the setting of an automatic flush timer (see Section 7.3.29). Non-Automatically-Flushable HCI ACL Data packets are not controlled by the automatic flush timeout and shall not be automatically flushed. The format of the HCI ACL Data packet is shown in Figure 5.2. The definition for each of the fields in the data packets is explained below.

Hosts and Controllers shall be able to accept HCI ACL Data packets with up to 27 bytes of data excluding the HCI ACL Data packet header on Connection_Handles associated with an LE-U logical link.The HCI ACL Data packet header is the first 4 octets of the packet.

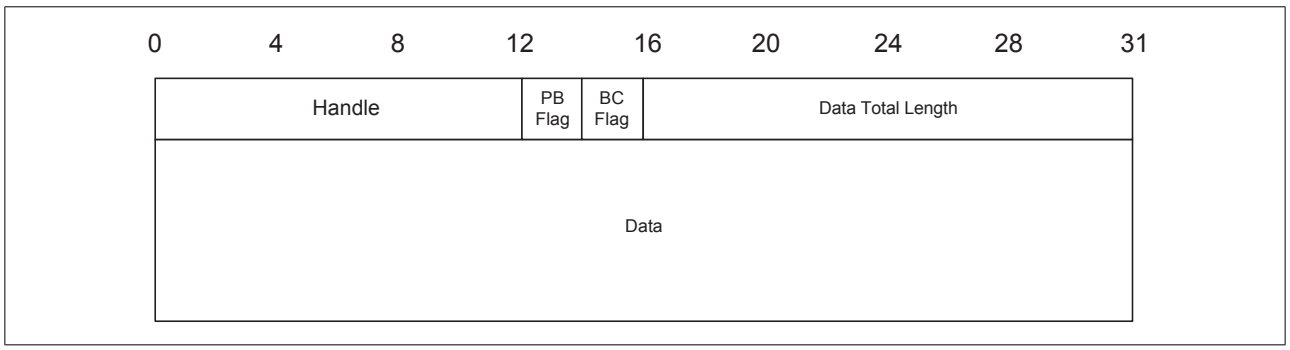



*Figure 5.2: HCI ACL Data packet*

*Handle:* *Size: 12 Bits*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection_Handle to be used for transmitting a data packet over a Controller.<br>Range: 0x000 to 0xEFF (all other values reserved for future use) |

The Flag Bits consist of the Packet_Boundary_Flag and Broadcast_Flag. The Packet_Boundary_Flag is located in bit 4 and bit 5, and the Broadcast_Flag is located in bit 6 and bit 7 in the second octet of the HCI ACL Data packet.

*Packet_Boundary_Flag:* *Size: 2 Bits*

| Value | Parameter Description | | APB-U | ACL-U | LE-U |
|---|---|---|---|---|---|
| 0b00 | First non-automatically-flushable packet of a higher layer message (start of a non-automatically-flushable L2CAP PDU) from Host to Controller. | Host to Controller | Not allowed | Allowed | Allowed |
| | | Controller to Host | Not allowed | Not allowed (except during loopback) | Not allowed |
| 0b01 | Continuing fragment of a higher layer message | Host to Controller | Allowed | Allowed | Allowed |
| | | Controller to Host | Allowed | Allowed | Allowed |
| 0b10 | First automatically flushable packet of a higher layer message (start of an automatically-flushable L2CAP PDU). | Host to Controller | Allowed | Allowed | C.1 |
| | | Controller to Host | Allowed | Allowed | Allowed |
| 0b11 | Previously used | | | | |
| C.1: Allowed if LE Feature (LE Flushable ACL Data packets) is supported, otherwise not allowed. | | | | | |

The start of a non-flushable packet of a higher layer message (start of a non-automatically-flushable L2CAP PDU) with the PBF of 0b00 shall be transmitted with an LLID of 0b10. All continuing fragment packets of a higher layer message shall be transmitted with an LLID of 0b01.

*Broadcast_Flag:* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 0b00 | Point-to-point (ACL-U or LE-U) |
| 0b01 | BR/EDR broadcast (APB-U) |

| Value | Parameter Description |
|---|---|
| 0b10 | Reserved for future use. |
| 0b11 | Reserved for future use. |

The Broadcast_Flag value 0b01 shall only be used in packets from Host to Controller on the Central of a piconet and from Controller to Host on the Peripheral of a piconet.

Note: Peripherals in Sniff mode will only receive a broadcast packet if it happens to be sent in a sniff slot when the Peripheral is listening.

*Data_Total_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length of data measured in octets. |

### 5.4.3 HCI Synchronous Data packets

HCI Synchronous Data packets are used to exchange synchronous data (SCO and eSCO) between the Host and Controller. The Controller may support HCI Synchronous Data packets if the Baseband supports synchronous data, but not otherwise.

The format of the HCI Synchronous Data packet is shown in Figure 5.3. The definition for each of the fields in the data packets is explained below. The HCI Synchronous Data packet header is the first 3 octets of the packet.

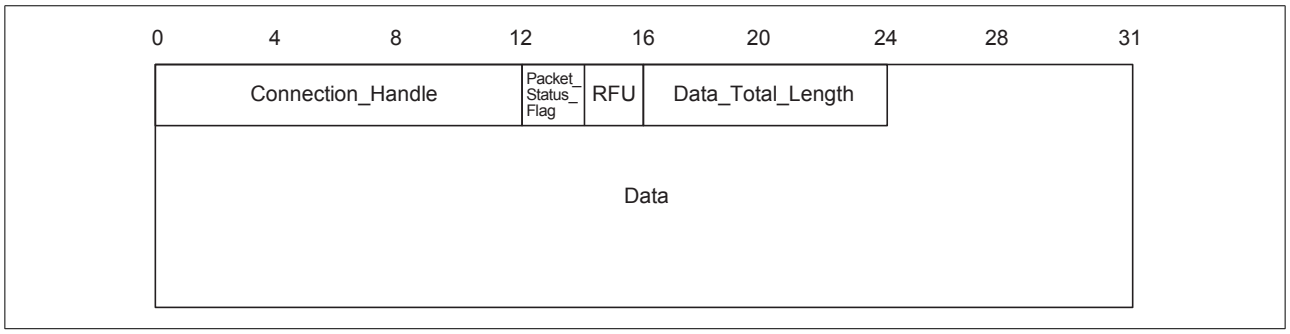



*Figure 5.3: HCI Synchronous Data packet*

*Connection_Handle:* *Size: 12 Bits*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection_Handle to be used to for transmitting a synchronous data packet.<br>Range: 0x0000 to 0x0EFF |

The Packet_Status_Flag bits consist of two bits, which are located from bit 4 to 5 in the second octet of the HCI Synchronous Data packet.

The Host shall set the Packet_Status_Flag bits to 0b00.

If the Erroneous_Data_Reporting parameter was set to disabled when the synchronous connection was created, the Controller shall set the Packet_Status_Flag bits to 0b00 and whether or not data is provided for cases when a valid (e)SCO packet was not received is unspecified.

If the Erroneous_Data_Reporting parameter was set to enabled when the synchronous connection was created, the Controller shall set the Packet_Status_Flag according to the following table.

*Packet_Status_Flag (in packets sent by the Controller):* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 0b00 | Correctly received data. The payload data belongs to received eSCO or SCO packets that the Baseband marked as “good data”. |
| 0b01 | Possibly invalid data. At least one eSCO packet has been marked by the Baseband as “data with possible errors” and all others have been marked as “good data” in the eSCO interval(s) corresponding to the HCI Synchronous Data packet. |
| 0b10 | No data received. All data from the Baseband received during the (e)SCO interval(s) corresponding to the HCI Synchronous Data packet have been marked as "lost data" by the Baseband. The Payload data octets shall be set to 0. |
| 0b11 | Data partially lost. Not all, but at least one (e)SCO packet has been marked as “lost data” by the Baseband in the (e)SCO intervals corresponding to the HCI Synchronous Data packet. The payload data octets corresponding to the missing (e)SCO packets shall be set to 0. |

Note: Some HCI transports and/or Controller implementations will align the HCI Synchronous Data packets with the (e)SCO Baseband packets such that data integrity can be explicitly marked in the Packet_Status_Flag. For HCI transports or Controller implementations that do not preserve this alignment, information in the Packet_Status_Flag might be ambiguous.

*Data_Total_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of synchronous data measured in octets |

### 5.4.4 HCI Event packet

The HCI Event packet is used by the Controller to notify the Host when events occur. If the Controller sends an HCI Event Packet containing an Event Code or an LE subevent code that the Host has not masked out and does not support, the Host shall ignore that packet. Any event code or LE subevent code not mentioned in this Part is reserved for future use. The Host shall be able to accept HCI Event packets with up to 255 octets

of data excluding the HCI Event packet header. The format of the HCI Event packet is shown in Figure 5.4, and the definition of each field is explained below. The HCI Event packet header is the first 2 octets of the packet.

The event code 0xFE is reserved for specification development purposes. The event code 0xFF is reserved for vendor-specific debugging events.

Note: An LE Controller uses a single Event Code (see Section 7.7.65) to transmit all LE specific events from the Controller to the Host. The first Event Parameter is always a subevent code identifying the specific event.

If Event Codes 0xFE and 0xFF are being used for more than one event, then these can also use subevent codes.

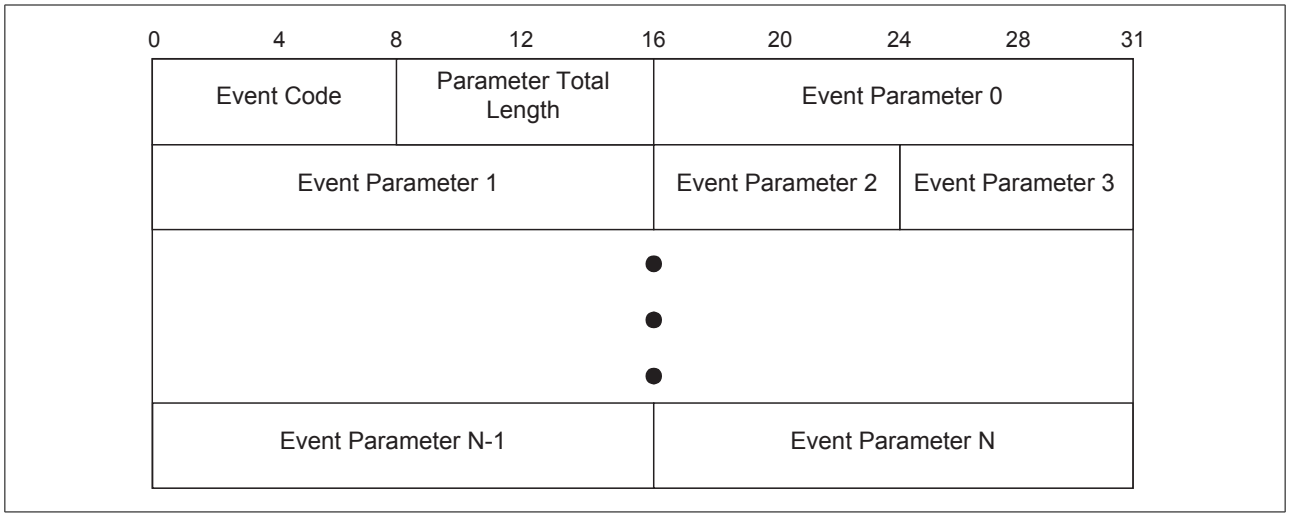



*Figure 5.4: HCI Event packet*

*Event_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Each event is assigned a 1-Octet event code used to uniquely identify different types of events. |

*Parameter_Total_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of all of the parameters contained in this packet, measured in octets |

*Event_Parameter 0 - N:* *Size: Parameter_Total_Length*

| Value | Parameter Description |
|---|---|
| 0xXX | Each event has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each event. Each parameter is an integer number of octets in size. |

### 5.4.5 HCI ISO Data packets

HCI ISO Data packets are used to exchange isochronous data between the Host and Controller. The Controller may support HCI ISO Data packets if the Link Layer supports any of the LE features Connected Isochronous Stream - Central, Connected Isochronous Stream - Peripheral, Isochronous Broadcaster, or Synchronized Receiver, but not otherwise.

An HCI ISO Data packet holds either an SDU or part of an SDU. In the Host to Controller direction, it cannot contain more data than the size of the buffer supported by the Controller. If the length of an SDU is greater than the Controller's buffer size, the Host may need to fragment that SDU. The Controller shall not start sending an SDU or fragments of an SDU to the Host until all the PDUs containing data from that SDU have either been received or can no longer be received because the last opportunity for them to be transmitted has passed. SDU fragments generated over HCI are unrelated to the SDU fragments generated by ISOAL.

The format of the HCI ISO Data packet is shown in Figure 5.5. The definition of each field in the packet is given below. The HCI ISO Data packet header is the first 4 octets of the packet.

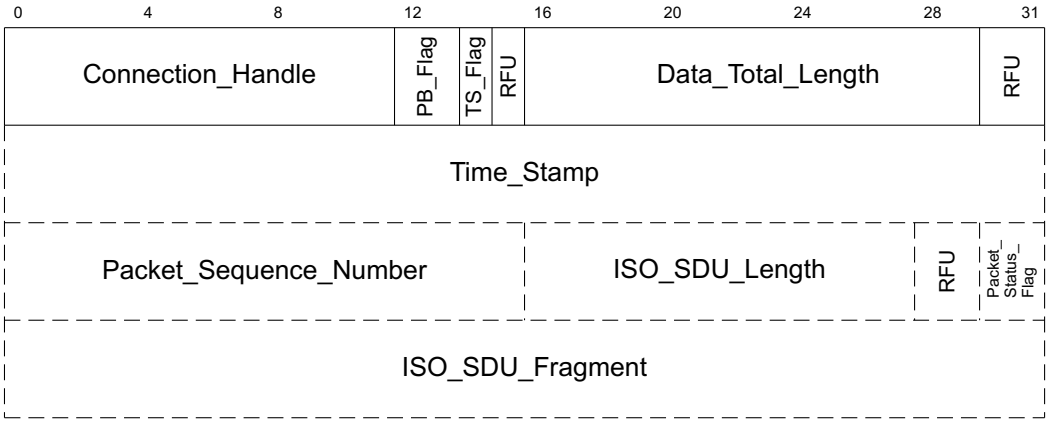



*Figure 5.5: Format of an HCI ISO Data packet*

If PB_Flag equals 0b00 or 0b10, then the Packet_Sequence_Number, ISO_SDU_Length, and Packet_Status_Flag fields (plus the intermediate RFU field)

shall all be present in the packet and the Time_Stamp field may be present. If PB_Flag equals 0b01 or 0b11, then none of these fields shall be included in the packet.

*Connection_Handle:* *Size: 12 bits*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection_Handle to be used for transmitting an ISO SDU or fragment.<br>Range: 0x000 to 0xEFF |

*PB_Flag:* *Size: 2 bits*

| Value | Parameter Description |
|---|---|
| 0b00 | The ISO_SDU_Fragment field contains the first fragment of a fragmented SDU. |
| 0b01 | The ISO_SDU_Fragment field contains an intermediate fragment of an SDU. |
| 0b10 | The ISO_SDU_Fragment field contains a complete SDU. |
| 0b11 | The ISO_SDU_Fragment field contains the last fragment of an SDU. |

*TS_Flag:* *Size: 1 bit*

| Value | Parameter Description |
|---|---|
| 0 | The Time_Stamp field is not present in the packet. |
| 1 | The Time_Stamp field is present in the packet. |

*Data_Total_Length:* *Size: 14 bits*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length of the packet, excluding the packet header, in octets. |

In the Host to Controller direction, Data_Total_Length shall be less than or equal to the size of the buffer supported by the Controller (which is returned using the ISO_Data_Packet_Length return parameter of the LE Read Buffer Size command).

If PB_Flag is 0b01 or 0b11, then Data_Total_Length may be zero. Otherwise Data_Total_Length is at least 4 if TS_Flag is 0 and at least 8 if TS_Flag is 1.

*Time_Stamp:* *Size: 32 bits*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | A time, in microseconds (see [Vol 6] Part G, Section 3). |

*Packet_Sequence_Number:* *Size: 16 bits*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The sequence number of the SDU (see [Vol 6] Part G, Section 2). |

*ISO_SDU_Length:* *Size: 12 bits*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The total length of the SDU (and not of any individual fragments), in octets. |

The Packet_Status_Flag field indicates the status of the packet that the Controller receives over the isochronous physical channel. The Packet_Status_Flag field is only valid in HCI ISO Data packets sent by the Controller and is reserved for future use in packets sent by the Host.

*Packet_Status_Flag (in packets sent by the Controller)* *Size: 2 bits*

| Value | Parameter Description |
|---|---|
| 0b00 | Valid data. The complete SDU was received correctly. |
| 0b01 | Possibly invalid data. The contents of the ISO_SDU_Fragment may contain errors or part of the SDU may be missing. This is reported as "data with possible errors". |
| 0b10 | Part(s) of the SDU were not received correctly. This is reported as "lost data". |
| All other values | Reserved for future use |

The ISO_SDU_Fragment field shall contain the isochronous data (the SDU or fragment of the SDU). This field may be empty. When Packet_Status_Flag is set to 0b10 in packets from the Controller to the Host, there is no data, PB_Flag shall be set to 0b10, and ISO_SDU_Length shall be set to zero.

## 5.5 Ignored parameters

When a parameter of a command or event is described as “ignored” (this is usually only in a specific circumstance), then the value of this parameter shall not affect the behavior of the recipient (whether Controller or Host ) in any way. In particular, the Controller shall not report an error because the value is out of range, reserved for future use, or represents an inappropriate entity (e.g. a Connection_Handle may be unassigned or for the wrong logical channel), or because two related values have the wrong relationship (e.g. minimum > maximum).

# 6 HCI CONFIGURATION PARAMETERS

## 6.1 Scan Enable

The Scan_Enable parameter controls whether or not the BR/EDR Controller will periodically scan for page attempts and/or inquiry requests from other BR/EDR Controllers. If Page Scan is enabled, then the device will enter page scan mode based on the value of the Page_Scan_Interval and Page_Scan_Window parameters. If Inquiry Scan is enabled, then the BR/EDR Controller will enter Inquiry Scan mode based on the value of the Inquiry_Scan_Interval and Inquiry_Scan_Window parameters.

*Scan_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled. |
| 0x01 | Inquiry Scan enabled.<br>Page Scan always disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| All other values | Reserved for future use |

## 6.2 Inquiry Scan Interval

The Inquiry_Scan_Interval configuration parameter defines the amount of time between consecutive inquiry scans. This is defined as the time interval from when the BR/EDR Controller started its last inquiry scan until it begins the next inquiry scan.

*Inquiry_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0012 to 0x1000; only even values are valid<br>Default: 0x1000<br>Time = N × 0.625 ms<br>Time Range: 11.25 to 2560 ms<br>Time Default: 2.56 s |

## 6.3 Inquiry Scan Window

The Inquiry_Scan_Window configuration parameter defines the amount of time for the duration of the inquiry scan. The Inquiry_Scan_Window can only be less than or equal to the Inquiry_Scan_Interval.

*Inquiry_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0011 to 0x1000<br>Default: 0x0012<br>Mandatory Range: 0x0011 to Inquiry Scan Interval<br>Time = N × 0.625 ms<br>Time Range: 10.625 ms to 2560 ms<br>Time Default: 11.25 ms |

## 6.4 Inquiry Scan Type

The Inquiry_Scan_Type configuration parameter indicates whether inquiry scanning will be done using non-interlaced scan or interlaced scan. Currently, one mandatory inquiry scan type and one optional inquiry scan type are defined. For details, see [Vol 2] Part B, Section 8.4.1.

*Inquiry_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| All other values | Reserved for future use |

## 6.5 Inquiry mode

The Inquiry_Mode configuration parameter indicates whether inquiry returns Inquiry Result events in the standard format, with RSSI, or with RSSI and extended inquiry response information.

*Inquiry_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format |
| 0x01 | Inquiry Result format with RSSI |

| Value | Parameter Description |
|---|---|
| 0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| All other values | Reserved for future use |

## 6.6 Page Timeout

The Page_Timeout configuration parameter together with Extended_Page_Timeout defines the maximum time the local Link Manager will wait for a Baseband page response from the remote device at a locally initiated connection attempt. If this time expires and the remote BR/EDR Controller has not responded to the page at Baseband level, the connection attempt will be considered to have failed.

*Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0001 to 0xFFFF<br>Default: 0x2000<br>Mandatory Range: 0x0016 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 0.625 ms to 40.9 s<br>Time Default: 5.12 s |

## 6.7 Connection Accept Timeout

The Connection_Accept_Timeout configuration parameter allows the BR/EDR or LE Controller to automatically deny a connection request after a specified time period has occurred and the new connection is not accepted. The parameter defines the time duration from when the BR/EDR Controller sends an HCI_Connection_Request event or the LE Controller sends an HCI_LE_CIS_Request event until the Controller will automatically reject an incoming connection.

*Connection_Accept_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0001 to 0xB540<br>Default: 0x1F40<br>Mandatory Range: 0x00A0 to 0xB540<br>Time = N × 0.625 ms<br>Time Range: 0.625 ms to 29 s<br>Time Default: 5 s |

## 6.8 Page Scan Interval

The Page_Scan_Interval configuration parameter defines the amount of time between consecutive page scans. This time interval is defined from when the Controller started its last page scan until it begins the next page scan.

*Page_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0012 to 0x1000; only even values are valid<br>Default: 0x0800<br>Time = N × 0.625 ms<br>Time Range: 11.25 ms to 2560 ms<br>Time Default: 1.28 s |

## 6.9 Page Scan Window

The Page_Scan_Window configuration parameter defines the amount of time for the duration of the page scan. The Page_Scan_Window can only be less than or equal to the Page_Scan_Interval.

*Page_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0011 to 0x1000<br>Default: 0x0012<br>Mandatory Range: 0x0011 to Page Scan Interval<br>Time = N × 0.625 ms<br>Time Range: 10.625 ms to Page Scan Interval<br>Time Default: 11.25 ms |

## 6.10 [This section is no longer used]

## 6.11 Page Scan Type

The Page_Scan_Type parameter indicates whether page scanning will be done using non-interlaced scan or interlaced scan. For details, see [Vol 2] Part B, Section 8.3.1.

*Page_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| All other values | Reserved for future use |

## 6.12 Voice Setting

The Voice_Setting parameter controls all the various settings for voice connections. The Voice_Setting parameter controls the configuration for voice connections: Input Coding, Air coding format, input data format, Input sample size, and linear PCM parameter. The air coding format bits in the Voice_Setting command parameter specify which air coding format the local device requests. The air coding format bits do not specify which air coding format(s) the local device accepts when a remote device requests an air coding format. This is determined by the hardware capabilities of the local BR/EDR Controller.

*Voice_Setting:* *Size: 1 octet*

| Bit Number | Parameter description | |
|---|---|---|
| 0 – 1 | Air coding format | 0: CVSD<br>1: μ-law<br>2: A-law<br>3. transparent data |
| 2 – 4 | Linear PCM bit position - number of bit positions that the MSB of the sample is from the MSB of the value (only for linear PCM) | |
| 5 | Input sample size (only for linear PCM) | 0: 8 bits<br>1: 16 bits |
| 6 – 7 | Input data format | 0: one’s complement<br>1: two’s complement<br>2: sign and magnitude<br>3: unsigned |
| 8 – 9 | Input coding format | 0: linear<br>1: μ-law<br>2: A-law<br>3: Reserved for future use |
| 10 – 15 | Reserved for future use | |

## 6.13 PIN Type

The PIN_Type configuration parameter determines whether the Link Manager assumes that the Host supports variable PIN codes or a fixed PIN code. The Controller uses the PIN_Type information during pairing.

*PIN_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

## 6.14 Link key

The Controller can store a limited number of link keys for other BR/EDR Controllers. Link keys are shared between two BR/EDR Controllers, and are used for all security transactions between the two BR/EDR Controllers. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the BR/EDR Controller when needed. A Link_Key is associated with a BD_ADDR.

*Link_Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Link Key for an associated BD_ADDR. |

## 6.15 Failed Contact Counter

The Failed_Contact_Counter records the number of consecutive incidents in which either the local or remote device did not respond after the flush timeout had expired, and the L2CAP PDU that was currently being transmitted was automatically ‘flushed’. When this occurs, the Failed_Contact_Counter is incremented by 1.

The Failed_Contact_Counter is maintained for each Connection_Handle.

The Failed_Contact_Counter for a connection is reset to zero on the following conditions:

1. When a new connection is established
2. When the Failed_Contact_Counter is not zero and an L2CAP PDU is acknowledged for that connection
3. When the HCI_Reset_Failed_Contact_Counter command has been issued

*Failed_Contact_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of consecutive failed contacts for a connection corresponding to the Connection_-Handle |

## 6.16 Authentication Enable

The Authentication_Enable parameter controls if the local device requires to authenticate the remote device at connection setup (between the HCI_Create_Connection command or acceptance of an incoming ACL connection and the corresponding Connection Complete event). At connection setup, only the device(s) with the Authentication_Enable parameter enabled will try to authenticate the other device.

Note: Changing this parameter does not affect existing connections.

*Authentication_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required. |
| 0x01 | Authentication required for all connections. |
| All other values | Reserved for future use |

## 6.17 Hold Mode Activity

The Hold_Mode_Activity value is used to determine what activities should be suspended when the BR/EDR Controller is in Hold mode. After the hold period has expired, the device will return to the previous mode of operation. If no activities are suspended, then all of the current Periodic Inquiry, Inquiry Scan, and Page Scan settings remain valid during the Hold mode. If the Hold_Mode_Activity parameter is set to Suspend Page Scan, Suspend Inquiry Scan, and Suspend Periodic Inquiries, then the device can enter a low-power state during the Hold mode period and all activities are suspended. The Hold Mode Activity is only valid if all connections are in Hold mode.

*Hold_Mode_Activity:* *Size: 1 octet*

| Bit | Parameter Description |
|---|---|
| 0 | Suspend Page Scan. |
| 1 | Suspend Inquiry Scan. |
| 2 | Suspend Periodic Inquiries. |
| All other bits | Reserved for future use. |

## 6.18 Link Policy Settings

The Link_Policy_Settings parameter determines the behavior of the local Link Manager when it receives a request from a remote Link Manager or it determines itself to change role or to enter Hold or Sniff mode. The local Link Manager will automatically accept or reject such a request from the remote device, and may even autonomously request itself, depending on the value of the Link_Policy_Settings parameter for the corresponding Connection_Handle. When the value of the Link_Policy_Settings parameter is changed for a certain Connection_Handle, the new value will only be used for requests from a remote device or from the local Link Manager itself made after this command has been completed. By enabling each mode individually, the Host can choose any combination needed to support various modes of operation. Multiple LM policies may be specified for the Link_Policy_Settings parameter by performing a bitwise OR operation of the different activity types.

Once a connection to a device has been put into Hold mode through an LMP request because the Link_Policy_Settings allowed it, the same connection shall be forced into Hold mode by any further LMP requests irrespective of the value of Link_Policy_Settings. The Link_Policy_Settings value shall apply to a subsequent connection between the same two devices.

*Link_Policy_Settings:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable Role switch. |
| 1 | Enable Hold mode. |
| 2 | Enable Sniff mode. |
| All other bits | Reserved for future use. |

## 6.19 Flush Timeout

The Flush_Timeout configuration parameter is used for ACL connections on a BR/EDR or an LE Controller. The Flush Timeout for a BR/EDR Controller is defined in [Vol 2] Part B, Section 7.6.3. The Flush Timeout for an LE Controller is defined in [Vol 6] Part B, Section 4.5.17.1.This parameter allows automatically-flushable ACL packets to be automatically flushed without the Host device issuing an HCI_Flush command. This provides support for ACL-U isochronous data, such as audio or sensor states. Non-automatically-flushable ACL packets shall not be affected by this parameter (see Section 5.4.2) providing support for both asynchronous and isochronous data on the same ACL connection. When an L2CAP packet on a BR/EDR ACL-U logical link that is currently being transmitted is automatically ‘flushed’, the Failed_Contact_Counter is incremented by one.

*Flush_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Timeout = ∞; No Automatic Flush |
| N = 0xXXXX | Range: 0x0001 to 0x07FF<br>Mandatory Range: 0x0002 to 0x07FF<br>Time = N × 0.625 ms<br>Time Range: 0.625 ms to 1279.375 ms |

## 6.20 Num Broadcast Retransmissions

Broadcast packets are not acknowledged and are unreliable. The Num_Broadcast_Retransmissions parameter, N, is used to increase the reliability of a broadcast message by retransmitting the broadcast message multiple times. This sets the value $N_{BC}$ in the Baseband to one greater than the Num_Broadcast_Retransmissions value. (See [Vol 2] Part B, Section 7.6.5) This parameter should be adjusted as the link quality measurement changes.

*Num_Broadcast_Retransmissions:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | $N_{BC}$ = N + 1<br>Range: 0x00 to 0xFE |

## 6.21 Link Supervision Timeout

The Link_Supervision_Timeout parameter is used by the BR/EDR Controller to monitor link loss. If, for any reason, no packets are received from that Connection_Handle for a duration longer than the Link_Supervision_Timeout, the connection shall be disconnected. The same timeout value is used for both synchronous and ACL connections for the device specified by the Connection_Handle.

Note: Setting the Link_Supervision_Timeout to No link supervision timeout (0x0000) will disable the Link_Supervision_Timeout check for the specified Connection_Handle.

*Link_Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No link supervision timeout. |

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0001 to 0xFFFF<br>Default: 0x7D00<br>Mandatory Range: 0x0190 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 0.625 ms to 40.9 s<br>Time Default: 20 s |

## 6.22 Synchronous Flow Control Enable

The Synchronous_Flow_Control_Enable configuration parameter allows the Host to decide if the BR/EDR Controller will send HCI_Number_Of_Completed_Packets events for synchronous Connection_Handles. This setting allows the Host to enable and disable synchronous flow control.

*Synchronous_Flow_Control_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronous Flow Control is disabled. No HCI_Number_Of_Completed_Packets events will be sent from the BR/EDR Controller for synchronous Connection_Handles. |
| 0x01 | Synchronous Flow Control is enabled. HCI_Number_Of_Completed_Packets events will be sent from the BR/EDR Controller for synchronous Connection_Handles. |

## 6.23 Local Name

The user-friendly Local_Name provides the user the ability to distinguish one BR/EDR Controller from another. The Local_Name configuration parameter is a UTF-8 encoded string with the type utf8s{248} defined in [Vol 1] Part E, Section 2.9.3.

Note: The Local_Name configuration parameter is a string parameter. Endianness does therefore not apply to the Local_Name configuration parameter. The first octet of the name is received first.

*Local_Name:* *Size: 248 octets*

| Value | Parameter Description |
|---|---|
| | A UTF-8 encoded User Friendly Descriptive Name for the device with type utf8s{248}. |

## 6.24 Extended Inquiry response

The Extended_Inquiry_Response provides information about the local device in response to inquiry from remote devices. The configuration parameter has two parts, a significant part followed by a non-significant part. The non-significant part contains

only all-zero octets. The data format of the significant part is defined in [Vol 3] Part C, Section 8.

*Extended_Inquiry_Response:* *Size: 240 octets*

| Value | Parameter Description |
|---|---|
| | Information about the local device that will be sent in an extended inquiry response packet to remote devices during inquiry response. |

## 6.25 Erroneous Data Reporting

The Erroneous_Data_Reporting configuration parameter shall be used for SCO and eSCO connections only. This parameter determines if the BR/EDR Controller is required to provide data to the Host for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data packets set according to Section 5.4.3.

*Erroneous_Data_Reporting:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous data reporting disabled |
| 0x01 | Erroneous data reporting enabled. |
| All other values | Reserved for future use |

## 6.26 Class of Device

The Class_Of_Device parameter is used to indicate the capabilities of the local device to other devices.

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

## 6.27 Supported commands

The Supported_Commands configuration parameter lists which HCI commands the local Controller supports.

The Supported_Commands is a 251 octet bit field. If a bit is set to 1, then this command is supported.

*Supported_Commands:* *Size: 251 octets*

| Octet | Bit | Command Supported |
|---|---|---|
| **0** | 0 | HCI_Inquiry |
| | 1 | HCI_Inquiry_Cancel |
| | 2 | HCI_Periodic_Inquiry_Mode |
| | 3 | HCI_Exit_Periodic_Inquiry_Mode |
| | 4 | HCI_Create_Connection |
| | 5 | HCI_Disconnect |
| | 6 | Previously used |
| | 7 | HCI_Create_Connection_Cancel |
| **1** | 0 | HCI_Accept_Connection_Request |
| | 1 | HCI_Reject_Connection_Request |
| | 2 | HCI_Link_Key_Request_Reply |
| | 3 | HCI_Link_Key_Request_Negative_Reply |
| | 4 | HCI_PIN_Code_Request_Reply |
| | 5 | HCI_PIN_Code_Request_Negative_Reply |
| | 6 | HCI_Change_Connection_Packet_Type |
| | 7 | HCI_Authentication_Requested |
| **2** | 0 | HCI_Set_Connection_Encryption |
| | 1 | HCI_Change_Connection_Link_Key |
| | 2 | HCI_Link_Key_Selection |
| | 3 | HCI_Remote_Name_Request |
| | 4 | HCI_Remote_Name_Request_Cancel |
| | 5 | HCI_Read_Remote_Supported_Features |
| | 6 | HCI_Read_Remote_Extended_Features |
| | 7 | HCI_Read_Remote_Version_Information |
| **3** | 0 | HCI_Read_Clock_Offset |
| | 1 | HCI_Read_LMP_Handle |
| | 2 | Reserved for future use |
| | 3 | Reserved for future use |
| | 4 | Reserved for future use |
| | 5 | Reserved for future use |
| | 6 | Reserved for future use |
| | 7 | Reserved for future use |

| Octet | Bit | Command Supported |
|---|---|---|
| **4** | 0 | Reserved for future use |
| | 1 | HCI_Hold_Mode |
| | 2 | HCI_Sniff_Mode |
| | 3 | HCI_Exit_Sniff_Mode |
| | 4 | Previously used |
| | 5 | Previously used |
| | 6 | HCI_QoS_Setup |
| | 7 | HCI_Role_Discovery |
| **5** | 0 | HCI_Switch_Role |
| | 1 | HCI_Read_Link_Policy_Settings |
| | 2 | HCI_Write_Link_Policy_Settings |
| | 3 | HCI_Read_Default_Link_Policy_Settings |
| | 4 | HCI_Write_Default_Link_Policy_Settings |
| | 5 | HCI_Flow_Specification |
| | 6 | HCI_Set_Event_Mask |
| | 7 | HCI_Reset |
| **6** | 0 | HCI_Set_Event_Filter |
| | 1 | HCI_Flush |
| | 2 | HCI_Read_PIN_Type |
| | 3 | HCI_Write_PIN_Type |
| | 4 | Previously used |
| | 5 | HCI_Read_Stored_Link_Key |
| | 6 | HCI_Write_Stored_Link_Key |
| | 7 | HCI_Delete_Stored_Link_Key |
| **7** | 0 | HCI_Write_Local_Name |
| | 1 | HCI_Read_Local_Name |
| | 2 | HCI_Read_Connection_Accept_Timeout |
| | 3 | HCI_Write_Connection_Accept_Timeout |
| | 4 | HCI_Read_Page_Timeout |
| | 5 | HCI_Write_Page_Timeout |
| | 6 | HCI_Read_Scan_Enable |
| | 7 | HCI_Write_Scan_Enable |

| Octet | Bit | Command Supported |
|---|---|---|
| **8** | 0 | HCI_Read_Page_Scan_Activity |
| | 1 | HCI_Write_Page_Scan_Activity |
| | 2 | HCI_Read_Inquiry_Scan_Activity |
| | 3 | HCI_Write_Inquiry_Scan_Activity |
| | 4 | HCI_Read_Authentication_Enable |
| | 5 | HCI_Write_Authentication_Enable |
| | 6 | Previously used |
| | 7 | Previously used |
| **9** | 0 | HCI_Read_Class_Of_Device |
| | 1 | HCI_Write_Class_Of_Device |
| | 2 | HCI_Read_Voice_Setting |
| | 3 | HCI_Write_Voice_Setting |
| | 4 | HCI_Read_Automatic_Flush_Timeout |
| | 5 | HCI_Write_Automatic_Flush_Timeout |
| | 6 | HCI_Read_Num_Broadcast_Retransmissions |
| | 7 | HCI_Write_Num_Broadcast_Retransmissions |
| **10** | 0 | HCI_Read_Hold_Mode_Activity |
| | 1 | HCI_Write_Hold_Mode_Activity |
| | 2 | HCI_Read_Transmit_Power_Level |
| | 3 | HCI_Read_Synchronous_Flow_Control_Enable |
| | 4 | HCI_Write_Synchronous_Flow_Control_Enable |
| | 5 | HCI_Set_Controller_To_Host_Flow_Control |
| | 6 | HCI_Host_Buffer_Size |
| | 7 | HCI_Host_Number_Of_Completed_Packets |
| **11** | 0 | HCI_Read_Link_Supervision_Timeout |
| | 1 | HCI_Write_Link_Supervision_Timeout |
| | 2 | HCI_Read_Number_Of_Supported_IAC |
| | 3 | HCI_Read_Current_IAC_LAP |
| | 4 | HCI_Write_Current_IAC_LAP |
| | 5 | Previously used |
| | 6 | Previously used |
| | 7 | Previously used |
| **12** | 0 | Previously used |

| Octet | Bit | Command Supported |
|---|---|---|
| | 1 | HCI_Set_AFH_Host_Channel_Classification |
| | 2 | HCI_LE_CS_Read_Remote_FAE_Table |
| | 3 | HCI_LE_CS_Write_Cached_Remote_FAE_Table |
| | 4 | HCI_Read_Inquiry_Scan_Type |
| | 5 | HCI_Write_Inquiry_Scan_Type |
| | 6 | HCI_Read_Inquiry_Mode |
| | 7 | HCI_Write_Inquiry_Mode |
| **13** | 0 | HCI_Read_Page_Scan_Type |
| | 1 | HCI_Write_Page_Scan_Type |
| | 2 | HCI_Read_AFH_Channel_Assessment_Mode |
| | 3 | HCI_Write_AFH_Channel_Assessment_Mode |
| | 4 | Reserved for future use |
| | 5 | Reserved for future use |
| | 6 | Reserved for future use |
| | 7 | Reserved for future use |
| **14** | 0 | Reserved for future use |
| | 1 | Reserved for future use |
| | 2 | Reserved for future use |
| | 3 | HCI_Read_Local_Version_Information |
| | 4 | Reserved for future use |
| | 5 | HCI_Read_Local_Supported_Features |
| | 6 | HCI_Read_Local_Extended_Features |
| | 7 | HCI_Read_Buffer_Size |
| **15** | 0 | Previously used |
| | 1 | HCI_Read_BD_ADDR |
| | 2 | HCI_Read_Failed_Contact_Counter |
| | 3 | HCI_Reset_Failed_Contact_Counter |
| | 4 | HCI_Read_Link_Quality |
| | 5 | HCI_Read_RSSI |
| | 6 | HCI_Read_AFH_Channel_Map |
| | 7 | HCI_Read_Clock |
| **16** | 0 | HCI_Read_Loopback_Mode |
| | 1 | HCI_Write_Loopback_Mode |

| Octet | Bit | Command Supported |
|---|---|---|
| | 2 | HCI_Enable_Implementation_Under_Test_Mode |
| | 3 | HCI_Setup_Synchronous_Connection |
| | 4 | HCI_Accept_Synchronous_Connection_Request |
| | 5 | HCI_Reject_Synchronous_Connection_Request |
| | 6 | HCI_LE_CS_Create_Config |
| | 7 | HCI_LE_CS_Remove_Config |
| **17** | 0 | HCI_Read_Extended_Inquiry_Response |
| | 1 | HCI_Write_Extended_Inquiry_Response |
| | 2 | HCI_Refresh_Encryption_Key |
| | 3 | Reserved for future use |
| | 4 | HCI_Sniff_Subrating |
| | 5 | HCI_Read_Simple_Pairing_Mode |
| | 6 | HCI_Write_Simple_Pairing_Mode |
| | 7 | HCI_Read_Local_OOB_Data |
| **18** | 0 | HCI_Read_Inquiry_Response_Transmit_Power_Level |
| | 1 | HCI_Write_Inquiry_Transmit_Power_Level |
| | 2 | HCI_Read_Default_Erroneous_Data_Reporting |
| | 3 | HCI_Write_Default_Erroneous_Data_Reporting |
| | 4 | Reserved for future use |
| | 5 | Reserved for future use |
| | 6 | Reserved for future use |
| | 7 | HCI_IO_Capability_Request_Reply |
| **19** | 0 | HCI_User_Confirmation_Request_Reply |
| | 1 | HCI_User_Confirmation_Request_Negative_Reply |
| | 2 | HCI_User_Passkey_Request_Reply |
| | 3 | HCI_User_Passkey_Request_Negative_Reply |
| | 4 | HCI_Remote_OOB_Data_Request_Reply |
| | 5 | HCI_Write_Simple_Pairing_Debug_Mode |
| | 6 | HCI_Enhanced_Flush |
| | 7 | HCI_Remote_OOB_Data_Request_Negative_Reply |
| **20** | 0 | Reserved for future use |
| | 1 | Reserved for future use |
| | 2 | HCI_Send_Keypress_Notification |

| Octet | Bit | Command Supported |
|---|---|---|
| | 3 | HCI_IO_Capability_Request_Negative_Reply |
| | 4 | HCI_Read_Encryption_Key_Size |
| | 5 | HCI_LE_CS_Read_Local_Supported_Capabilities [v1] |
| | 6 | HCI_LE_CS_Read_Remote_Supported_Capabilities |
| | 7 | HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities [v1] |
| **21** | 0 | Previously used |
| | 1 | Previously used |
| | 2 | Previously used |
| | 3 | Previously used |
| | 4 | Previously used |
| | 5 | Previously used |
| | 6 | Previously used |
| | 7 | Previously used |
| **22** | 0 | Previously used |
| | 1 | Previously used |
| | 2 | HCI_Set_Event_Mask_Page_2 |
| | 3 | Previously used |
| | 4 | Previously used |
| | 5 | Previously used |
| | 6 | Previously used |
| | 7 | Previously used |
| **23** | 0 | HCI_Read_Flow_Control_Mode |
| | 1 | HCI_Write_Flow_Control_Mode |
| | 2 | HCI_Read_Data_Block_Size |
| | 3 | HCI_LE_CS_Test |
| | 4 | HCI_LE_CS_Test_End |
| | 5 | Previously used |
| | 6 | Previously used |
| | 7 | Previously used |
| **24** | 0 | HCI_Read_Enhanced_Transmit_Power_Level |
| | 1 | HCI_LE_CS_Security_Enable |
| | 2 | Previously used |
| | 3 | Previously used |

| Octet | Bit | Command Supported |
|---|---|---|
| | 4 | Previously used |
| | 5 | HCI_Read_LE_Host_Support |
| | 6 | HCI_Write_LE_Host_Support |
| | 7 | HCI_LE_CS_Set_Default_Settings |
| **25** | 0 | HCI_LE_Set_Event_Mask [v1] |
| | 1 | HCI_LE_Read_Buffer_Size [v1] |
| | 2 | HCI_LE_Read_Local_Supported_Features_Page_0 |
| | 3 | Reserved for future use |
| | 4 | HCI_LE_Set_Random_Address |
| | 5 | HCI_LE_Set_Advertising_Parameters |
| | 6 | HCI_LE_Read_Advertising_Physical_Channel_Tx_Power |
| | 7 | HCI_LE_Set_Advertising_Data |
| **26** | 0 | HCI_LE_Set_Scan_Response_Data |
| | 1 | HCI_LE_Set_Advertising_Enable |
| | 2 | HCI_LE_Set_Scan_Parameters |
| | 3 | HCI_LE_Set_Scan_Enable |
| | 4 | HCI_LE_Create_Connection |
| | 5 | HCI_LE_Create_Connection_Cancel |
| | 6 | HCI_LE_Read_Filter_Accept_List_Size |
| | 7 | HCI_LE_Clear_Filter_Accept_List |
| **27** | 0 | HCI_LE_Add_Device_To_Filter_Accept_List |
| | 1 | HCI_LE_Remove_Device_From_Filter_Accept_List |
| | 2 | HCI_LE_Connection_Update |
| | 3 | HCI_LE_Set_Host_Channel_Classification |
| | 4 | HCI_LE_Read_Channel_Map |
| | 5 | HCI_LE_Read_Remote_Features_Page_0 |
| | 6 | HCI_LE_Encrypt |
| | 7 | HCI_LE_Rand |
| **28** | 0 | HCI_LE_Enable_Encryption |
| | 1 | HCI_LE_Long_Term_Key_Request_Reply |
| | 2 | HCI_LE_Long_Term_Key_Request_Negative_Reply |
| | 3 | HCI_LE_Read_Supported_States |
| | 4 | HCI_LE_Receiver_Test [v1] |

| Octet | Bit | Command Supported |
|---|---|---|
| | 5 | HCI_LE_Transmitter_Test [v1] |
| | 6 | HCI_LE_Test_End |
| | 7 | HCI_LE_Enable_Monitoring_Advertisers |
| **29** | 0 | HCI_LE_CS_Set_Channel_Classification |
| | 1 | HCI_LE_CS_Set_Procedure_Parameters |
| | 2 | HCI_LE_CS_Procedure_Enable |
| | 3 | HCI_Enhanced_Setup_Synchronous_Connection |
| | 4 | HCI_Enhanced_Accept_Synchronous_Connection_Request |
| | 5 | HCI_Read_Local_Supported_Codecs [v1] |
| | 6 | HCI_Set_MWS_Channel_Parameters |
| | 7 | HCI_Set_External_Frame_Configuration |
| **30** | 0 | HCI_Set_MWS_Signaling |
| | 1 | HCI_Set_MWS_Transport_Layer |
| | 2 | HCI_Set_MWS_Scan_Frequency_Table |
| | 3 | HCI_Get_MWS_Transport_Layer_Configuration |
| | 4 | HCI_Set_MWS_PATTERN_Configuration |
| | 5 | HCI_Set_Triggered_Clock_Capture |
| | 6 | HCI_Truncated_Page |
| | 7 | HCI_Truncated_Page_Cancel |
| **31** | 0 | HCI_Set_Connectionless_Peripheral_Broadcast |
| | 1 | HCI_Set_Connectionless_Peripheral_Broadcast_Receive |
| | 2 | HCI_Start_Synchronization_Train |
| | 3 | HCI_Receive_Synchronization_Train |
| | 4 | HCI_Set_Reserved_LT_ADDR |
| | 5 | HCI_Delete_Reserved_LT_ADDR |
| | 6 | HCI_Set_Connectionless_Peripheral_Broadcast_Data |
| | 7 | HCI_Read_Synchronization_Train_Parameters |
| **32** | 0 | HCI_Write_Synchronization_Train_Parameters |
| | 1 | HCI_Remote_OOB_Extended_Data_Request_Reply |
| | 2 | HCI_Read_Secure_Connections_Host_Support |
| | 3 | HCI_Write_Secure_Connections_Host_Support |
| | 4 | HCI_Read_Authenticated_Payload_Timeout |
| | 5 | HCI_Write_Authenticated_Payload_Timeout |

| Octet | Bit | Command Supported |
|---|---|---|
| | 6 | HCI_Read_Local_OOB_Extended_Data |
| | 7 | HCI_Write_Secure_Connections_Test_Mode |
| **33** | 0 | HCI_Read_Extended_Page_Timeout |
| | 1 | HCI_Write_Extended_Page_Timeout |
| | 2 | HCI_Read_Extended_Inquiry_Length |
| | 3 | HCI_Write_Extended_Inquiry_Length |
| | 4 | HCI_LE_Remote_Connection_Parameter_Request_Reply |
| | 5 | HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply |
| | 6 | HCI_LE_Set_Data_Length |
| | 7 | HCI_LE_Read_Suggested_Default_Data_Length |
| **34** | 0 | HCI_LE_Write_Suggested_Default_Data_Length |
| | 1 | HCI_LE_Read_Local_P-256_Public_Key |
| | 2 | HCI_LE_Generate_DHKey [v1] |
| | 3 | HCI_LE_Add_Device_To_Resolving_List |
| | 4 | HCI_LE_Remove_Device_From_Resolving_List |
| | 5 | HCI_LE_Clear_Resolving_List |
| | 6 | HCI_LE_Read_Resolving_List_Size |
| | 7 | HCI_LE_Read_Peer_Resolvable_Address |
| **35** | 0 | HCI_LE_Read_Local_Resolvable_Address |
| | 1 | HCI_LE_Set_Address_Resolution_Enable |
| | 2 | HCI_LE_Set_Resolvable_Private_Address_Timeout [v1] |
| | 3 | HCI_LE_Read_Maximum_Data_Length |
| | 4 | HCI_LE_Read_PHY |
| | 5 | HCI_LE_Set_Default_PHY |
| | 6 | HCI_LE_Set_PHY |
| | 7 | HCI_LE_Receiver_Test [v2] |
| **36** | 0 | HCI_LE_Transmitter_Test [v2] |
| | 1 | HCI_LE_Set_Advertising_Set_Random_Address |
| | 2 | HCI_LE_Set_Extended_Advertising_Parameters [v1] |
| | 3 | HCI_LE_Set_Extended_Advertising_Data |
| | 4 | HCI_LE_Set_Extended_Scan_Response_Data |
| | 5 | HCI_LE_Set_Extended_Advertising_Enable |
| | 6 | HCI_LE_Read_Maximum_Advertising_Data_Length |

| Octet | Bit | Command Supported |
|---|---|---|
| | 7 | HCI_LE_Read_Number_of_Supported_Advertising_Sets |
| **37** | 0 | HCI_LE_Remove_Advertising_Set |
| | 1 | HCI_LE_Clear_Advertising_Sets |
| | 2 | HCI_LE_Set_Periodic_Advertising_Parameters [v1] |
| | 3 | HCI_LE_Set_Periodic_Advertising_Data |
| | 4 | HCI_LE_Set_Periodic_Advertising_Enable |
| | 5 | HCI_LE_Set_Extended_Scan_Parameters |
| | 6 | HCI_LE_Set_Extended_Scan_Enable |
| | 7 | HCI_LE_Extended_Create_Connection [v1] |
| **38** | 0 | HCI_LE_Periodic_Advertising_Create_Sync |
| | 1 | HCI_LE_Periodic_Advertising_Create_Sync_Cancel |
| | 2 | HCI_LE_Periodic_Advertising_Terminate_Sync |
| | 3 | HCI_LE_Add_Device_To_Periodic_Advertiser_List |
| | 4 | HCI_LE_Remove_Device_From_Periodic_Advertiser_List |
| | 5 | HCI_LE_Clear_Periodic_Advertiser_List |
| | 6 | HCI_LE_Read_Periodic_Advertiser_List_Size |
| | 7 | HCI_LE_Read_Transmit_Power |
| **39** | 0 | HCI_LE_Read_RF_Path_Compensation |
| | 1 | HCI_LE_Write_RF_Path_Compensation |
| | 2 | HCI_LE_Set_Privacy_Mode |
| | 3 | HCI_LE_Receiver_Test [v3] |
| | 4 | HCI_LE_Transmitter_Test [v3] |
| | 5 | HCI_LE_Set_Connectionless_CTE_Transmit_Parameters |
| | 6 | HCI_LE_Set_Connectionless_CTE_Transmit_Enable |
| | 7 | HCI_LE_Set_Connectionless_IQ_Sampling_Enable |
| **40** | 0 | HCI_LE_Set_Connection_CTE_Receive_Parameters |
| | 1 | HCI_LE_Set_Connection_CTE_Transmit_Parameters |
| | 2 | HCI_LE_Connection_CTE_Request_Enable |
| | 3 | HCI_LE_Connection_CTE_Response_Enable |
| | 4 | HCI_LE_Read_Antenna_Information |
| | 5 | HCI_LE_Set_Periodic_Advertising_Receive_Enable |
| | 6 | HCI_LE_Periodic_Advertising_Sync_Transfer |
| | 7 | HCI_LE_Periodic_Advertising_Set_Info_Transfer |

| Octet | Bit | Command Supported |
|---|---|---|
| **41** | 0 | HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters |
| | 1 | HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters |
| | 2 | HCI_LE_Generate_DHKey [v2] |
| | 3 | HCI_Read_Local_Simple_Pairing_Options |
| | 4 | HCI_LE_Modify_Sleep_Clock_Accuracy |
| | 5 | HCI_LE_Read_Buffer_Size [v2] |
| | 6 | HCI_LE_Read_ISO_TX_Sync |
| | 7 | HCI_LE_Set_CIG_Parameters |
| **42** | 0 | HCI_LE_Set_CIG_Parameters_Test |
| | 1 | HCI_LE_Create_CIS |
| | 2 | HCI_LE_Remove_CIG |
| | 3 | HCI_LE_Accept_CIS_Request |
| | 4 | HCI_LE_Reject_CIS_Request |
| | 5 | HCI_LE_Create_BIG |
| | 6 | HCI_LE_Create_BIG_Test |
| | 7 | HCI_LE_Terminate_BIG |
| **43** | 0 | HCI_LE_BIG_Create_Sync |
| | 1 | HCI_LE_BIG_Terminate_Sync |
| | 2 | HCI_LE_Request_Peer_SCA |
| | 3 | HCI_LE_Setup_ISO_Data_Path |
| | 4 | HCI_LE_Remove_ISO_Data_Path |
| | 5 | HCI_LE_ISO_Transmit_Test |
| | 6 | HCI_LE_ISO_Receive_Test |
| | 7 | HCI_LE_ISO_Read_Test_Counters |
| **44** | 0 | HCI_LE_ISO_Test_End |
| | 1 | HCI_LE_Set_Host_Feature [v1] |
| | 2 | HCI_LE_Read_ISO_Link_Quality |
| | 3 | HCI_LE_Enhanced_Read_Transmit_Power_Level |
| | 4 | HCI_LE_Read_Remote_Transmit_Power_Level |
| | 5 | HCI_LE_Set_Path_Loss_Reporting_Parameters |
| | 6 | HCI_LE_Set_Path_Loss_Reporting_Enable |
| | 7 | HCI_LE_Set_Transmit_Power_Reporting_Enable |
| **45** | 0 | HCI_LE_Transmitter_Test [v4] |

| Octet | Bit | Command Supported |
|---|---|---|
| | 1 | HCI_Set_Ecosystem_Base_Interval |
| | 2 | HCI_Read_Local_Supported_Codecs [v2] |
| | 3 | HCI_Read_Local_Supported_Codec_Capabilities |
| | 4 | HCI_Read_Local_Supported_Controller_Delay |
| | 5 | HCI_Configure_Data_Path |
| | 6 | HCI_LE_Set_Data_Related_Address_Changes |
| | 7 | HCI_Set_Min_Encryption_Key_Size |
| **46** | 0 | HCI_LE_Set_Default_Subrate |
| | 1 | HCI_LE_Subrate_Request |
| | 2 | HCI_LE_Set_Extended_Advertising_Parameters [v2] |
| | 3 | HCI_LE_Set_Decision_Data |
| | 4 | HCI_LE_Set_Decision_Instructions |
| | 5 | HCI_LE_Set_Periodic_Advertising_Subevent_Data |
| | 6 | HCI_LE_Set_Periodic_Advertising_Response_Data |
| | 7 | HCI_LE_Set_Periodic_Sync_Subevent |
| **47** | 0 | HCI_LE_Extended_Create_Connection [v2] |
| | 1 | HCI_LE_Set_Periodic_Advertising_Parameters [v2] |
| | 2 | HCI_LE_Read_All_Local_Supported_Features |
| | 3 | HCI_LE_Read_All_Remote_Features |
| | 4 | HCI_LE_Set_Host_Feature [v2] |
| | 5 | HCI_LE_Add_Device_To_Monitored_Advertisers_List |
| | 6 | HCI_LE_Remove_Device_From_Monitored_Advertisers_List |
| | 7 | HCI_LE_Clear_Monitored_Advertisers_List |
| **48** | 0 | HCI_LE_Read_Monitored_Advertisers_List_Size |
| | 1 | HCI_LE_Frame_Space_Update |
| | 2 | HCI_LE_Set_Resolvable_Private_Address_Timeout [v2] |
| | 3 | HCI_LE_Enable_OTA_UTP_Mode |
| | 4 | HCI_LE_UTP_Send |
| | 5 | HCI_LE_Connection_Rate_Request |
| | 6 | HCI_LE_Set_Default_Rate_Parameters |
| | 7 | HCI_LE_Read_Minimum_Supported_Connection_Interval |
| **49** | 0 | HCI_Read_Local_Supported_Commands [v2] |
| | 1 | HCI_LE_Set_Event_Mask [v2] |

| Octet | Bit | Command Supported |
|---|---|---|
| | 2 | HCI_LE_CS_Read_Local_Supported_Capabilities [v2] |
| | 3 | HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities [v2] |
| | 4 | HCI_LE_CS_Set_Security_Requirements |
| | 5 | HCI_LE_CS_Set_Default_Security_Requirements |
| **All other octets** | | Reserved for future use |

## 6.28 [This section is no longer used]

## 6.29 [This section is no longer used]

## 6.30 [This section is no longer used]

## 6.31 [This section is no longer used]

## 6.32 [This section is no longer used]

## 6.33 Flow Control mode

The Flow_Control_Mode configuration parameter allows the Host to select the HCI Data flow control mode used by the Controller for ACL Data traffic other than LE traffic, which shall always use packet-based data flow control mode.

*Flow_Control_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Packet based data flow control mode (default) |
| 0x01 | Data block based data flow control mode |
| All other values | Reserved for future use |

## 6.34 LE Supported Host

The LE_Supported_Host parameter allows the Host to read and set the Link Manager Protocol feature bit LE Supported (Host). See [Vol 2] Part C, Section 3.2.

*LE_Supported_Host:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE Supported (Host) disabled (**default**) |
| 0x01 | LE Supported (Host) enabled |
| All other values | Reserved for future use |

## 6.35 [This section is no longer used]

## 6.36 Sync Train Interval

The Sync_Train_Interval configuration parameter defines the time between Synchronization Train transmit events on a single transmit RF channel.

*Sync_Train_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0020 to 0xFFFE; only even values are valid<br>Default: 0x0080<br>Mandatory Range: 0x0020 to 0x1000<br>Time = N × 0.625 ms<br>Time Range: 20 ms to 40.9 s<br>Time Default: 80 ms |

## 6.37 Sync Train Timeout

The Sync_Train_Timeout configuration parameter is used by the Controller to terminate the Synchronization Train after it has been started via the HCI_Start_Synchronization_Train command.

*Sync_Train_Timeout:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXXXXXX | Range: 0x00000002 to 0x07FFFFE; only even values are valid<br>Default: 0x0002EE00<br>Time = N × 0.625 ms<br>Time Range: 1.25 ms to 23.3 hours<br>Time Default: 120 s |

## 6.38 Service Data

The Service_Data configuration parameter defines the value of the service data field in the Synchronization Train.

*Service_Data:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: 0x00 to 0xFF<br>Default: 0x00 |

## 6.39 Secure Connections Host Support

The Secure_Connections_Host_Support configuration parameter allows the Host to indicate whether it supports Secure Connections or not. When Secure Connections Host Support is set to 'enabled' the Controller shall use the enhanced reporting mechanisms for the Encryption_Enabled parameter in the HCI_Encryption_Change event (see Section 7.7.8) and the Key_Type parameter in the HCI_Link_Key_Notification event (see Section 7.7.24).

*Secure_Connections_Host_Support:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure_Conections_Host_Support is ‘disabled’. Host does not support Secure Connections (default) |
| 0x01 | Secure_Connections_Host_Support is ‘enabled’. Host supports Secure Connections |
| All other values | Reserved for future use |

## 6.40 Authenticated Payload Timeout

The Authenticated_Payload_Timeout configuration parameter allows the Host to configure the maximum interval between packets containing a MIC received from the remote device when AES-CCM encryption is enabled.

*Authenticated_Payload_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum amount of time specified between packets authenticated by a MIC.<br>Range: 0x0001 to 0xFFFF<br>Time = N × 10 ms<br>Time Range: 10 ms to 655,350 ms |

## 6.41 Extended Page Timeout

The Extended_Page_Timeout configuration parameter together with Page_Timeout defines the maximum time the local Link Manager will wait for a Baseband page

response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at Baseband level, the connection attempt will be considered to have failed.

*Extended_Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0000 to 0xFFFF<br>Default: 0x0000<br>Time = N × 0.625 ms<br>Time Range: 0 to 40.9 s<br>Time Default: 0 s |

## 6.42 Extended Inquiry Length

The Extended_Inquiry_Length configuration parameter together with Inquiry_Length defines the maximum time the local Link Manager will wait for a Baseband inquiry response messages from the remote device at a locally initiated inquiry or periodic inquiry.

*Extended_Inquiry_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0000 to 0xFFFF<br>Default: 0x0000<br>Time = N × 0.625 ms<br>Time Range: 0 to 40.9 s<br>Time Default: 0 s |

# 7 HCI COMMANDS AND EVENTS

## 7.1 Link Control commands

The Link Control commands allow a Controller to control connections to other BR/EDR Controllers. Some Link Control commands are used only with a BR/EDR Controller whereas other Link Control commands are also used with an LE Controller.

In the BR/EDR Controller, when the Link Control commands are used, the Link Manager (LM) controls how the Bluetooth piconets and scatternets are established and maintained. These commands instruct the LM to create and modify Link Layer connections with Bluetooth remote devices, perform Inquiries of other BR/EDR Controllers in range, and other LMP commands.

In the LE Controller, Link Control commands are used to disconnect physical links.

For the Link Control commands, the OGF is defined as 0x01.

### 7.1.1 Inquiry command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Inquiry | 0x0001 | LAP,<br>Inquiry_Length,<br>Num_Responses | *none* |

**Description:**

This command causes the BR/EDR Controller to enter Inquiry Mode. Inquiry Mode is used to discover other nearby BR/EDR Controllers. The LAP parameter contains the LAP from which the inquiry access code shall be derived when the inquiry procedure is made. The Inquiry_Length parameter, added to Extended_Inquiry_Length (see Section 6.42), specifies the total duration of the Inquiry Mode and, when this time expires, Inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the Inquiry is halted. HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result events will be sent to report the details of nearby BR/EDR Controllers that have responded to this inquiry. The HCI_Inquiry_Complete event is sent to report that Inquiry Mode has ended.

A device which responds during an inquiry or inquiry period should always be reported to the Host in an HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using

the command HCI_Set_Event_Filter. If the device has been reported earlier during the current inquiry or inquiry period, whether it is reported again will depend on the implementation (e.g. whether earlier results have been saved in the BR/EDR Controller and in that case how many responses have been saved). It is recommended that the BR/EDR Controller tries to report a particular device only once during an inquiry or inquiry period.

**Command parameters:**

*LAP:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | The LAP from which the inquiry access code should be derived when the inquiry procedure is made; see Assigned Numbers.<br>Range: 0x9E8B00 to 0x9E8B3F |

*Inquiry_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Maximum amount of time (added to Extended_Inquiry_Length) specified before the Inquiry is halted.<br>Range: 0x01 to 0x30<br>Time = N × 1.28 s<br>Range: 1.28 to 61.44 s |

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unlimited number of responses. |
| 0xXX | Maximum number of responses from the Inquiry before the Inquiry is halted.<br>Range: 0x01 to 0xFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

An HCI_Command_Status event shall be sent from the BR/EDR Controller to the Host when the BR/EDR Controller has started the Inquiry process. Unless filtered, an HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result event shall be created for each BR/EDR Controller which responds to the Inquiry message. In addition, multiple BR/EDR Controllers which respond to the Inquire

message may be combined into the same event. An HCI_Inquiry_Complete event shall be generated when the Inquiry process has completed.

### 7.1.2 Inquiry Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Inquiry_Cancel | 0x0002 | *none* | Status |

**Description:**

This command shall cause the BR/EDR Controller to stop the current Inquiry if the BR/EDR Controller is in Inquiry Mode. This command allows the Host to interrupt the BR/EDR Controller and request the BR/EDR Controller to perform a different task. The command should only be issued after the HCI_Inquiry command has been issued, an HCI_Command_Status event has been received for the Inquiry command, and before the HCI_Inquiry_Complete event occurs.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Inquiry_Cancel command succeeded. |
| 0x01 to 0xFF | HCI_Inquiry_Cancel command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Inquiry_Cancel command has completed, an HCI_Command_Complete event shall be generated. No HCI_Inquiry_Complete event will be generated for the cancelled Inquiry process.

### 7.1.3 Periodic Inquiry Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Periodic_Inquiry_Mode | 0x0003 | Max_Period_Length,<br>Min_Period_Length,<br>LAP,<br>Inquiry_Length,<br>Num_Responses | Status |

**Description:**

This command is used to configure the BR/EDR Controller to enter the Periodic Inquiry Mode that performs an automatic Inquiry. Max_Period_Length and Min_Period_Length define the time range between two consecutive inquiries, from the beginning of an inquiry until the start of the next inquiry. The BR/EDR Controller shall use this range to determine a new random time between two consecutive inquiries for each Inquiry. The LAP parameter contains the LAP from which the inquiry access code shall be derived when the inquiry procedure is made. The Inquiry_Length parameter, added to Extended_Inquiry_Length (see Section 6.42), specifies the total duration of the Inquiry Mode and, when time expires, Inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the Inquiry is halted. This command is completed when the Inquiry process has been started by the BR/EDR Controller, and an HCI_Command_Complete event is sent from the Controller to the Host. When each of the periodic Inquiry processes are completed, the Controller will send an HCI_Inquiry_Complete event to the Host indicating that the latest periodic Inquiry process has finished. When a BR/EDR Controller responds to the Inquiry message, an HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result event will occur to notify the Host of the discovery.

Max_Period_Length shall be greater than Min_Period_Length. Min_Period_Length shall be greater than (Inquiry_Length + Extended Inquiry_Length).

A device which responds during an inquiry or inquiry period should always be reported to the Host in an HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using the command HCI_Set_Event_Filter. If the device has been reported earlier during the current inquiry or inquiry period, whether it is reported again will depend on the implementation (e.g. whether earlier results have been saved in the BR/EDR Controller and in that case how many responses have been saved). It is recommended that the BR/EDR Controller tries to report a particular device only once during an inquiry or inquiry period.

**Command parameters:**

*Max_Period_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum amount of time specified between consecutive inquiries.<br>Range: 0x0003 to 0xFFFF<br>Time = N × 1.28 s<br>Range: 3.84 to 83884.8 s<br>0.0 to 23.3 hours |

*Min_Period_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum amount of time specified between consecutive inquiries.<br>Range: 0x0002 to 0xFFFE<br>Time = N × 1.28 s<br>Range: 2.56 to 83883.52 s<br>0.0 to 23.3 hours |

*LAP:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | The LAP from which the inquiry access code should be derived when the inquiry procedure is made;<br>see Assigned Numbers.<br>Range: 0x9E8B00 to 0x9E8B3F |

*Inquiry_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Maximum amount of time (added to Extended_Inquiry_Length) specified before the Inquiry is halted.<br>Range: 0x01 to 0x30<br>Time = N × 1.28 s<br>Range: 1.28 to 61.44 s |

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unlimited number of responses. |
| 0xXX | Maximum number of responses from the Inquiry before the Inquiry is halted.<br>Range: 0x01 to 0xFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Periodic_Inquiry_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Periodic_Inquiry_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

The HCI Periodic Inquiry Mode begins when the BR/EDR Controller sends the HCI_Command_Complete event for this command to the Host. Unless filtered, an HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result event shall be created for each remote device that has responded to the Inquiry message. In addition, responses to the Inquiry message from multiple BR/EDR Controllers may be combined into the same event. An HCI_Inquiry_Complete event shall be generated when each of the periodic Inquiry processes has completed. No HCI_Inquiry_Complete event will be generated for the cancelled Inquiry process.

### 7.1.4 Exit Periodic Inquiry Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Exit_Periodic_Inquiry_Mode | 0x0004 | *none* | Status |

**Description:**

This command is used to end the Periodic Inquiry mode when the local device is in Periodic Inquiry Mode. If the BR/EDR Controller is currently in an Inquiry process, the Inquiry process shall be stopped directly and the BR/EDR Controller shall no longer perform periodic inquiries until the HCI_Periodic_Inquiry_Mode command is reissued.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Exit_Periodic_Inquiry_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Exit_Periodic_Inquiry_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

An HCI_Command_Complete event for this command shall occur when the local device is no longer in Periodic Inquiry Mode. No HCI_Inquiry_Complete event will be generated for the cancelled Inquiry process.

### 7.1.5 Create Connection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Create_Connection | 0x0005 | BD_ADDR,<br>Packet_Type,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Clock_Offset,<br>Allow_Role_Switch | *none* |

**Description:**

This command causes the Link Manager to create a connection to the remote device with the BD_ADDR specified by the command parameters. This command causes the local BR/EDR Controller to begin the Page process to create a link level connection. The Link Manager will determine how the new ACL connection is established. This ACL connection is determined by the current state of the device, its piconet, and the state of the device to be connected. The Packet_Type parameter specifies which packet types the Link Manager shall use for the ACL connection; the Host shall not specify packet types that the local Controller does not support. When sending HCI ACL Data packets the Link Manager shall only use the packet type(s) specified by Packet_Type or the always-allowed DM1 packet type. Multiple packet types may be specified for the Packet Type parameter by performing a bit-wise OR operation of the different packet types. The Link Manager may choose which packet type to be used from the list of acceptable packet types. The Page_Scan_Repetition_Mode parameter specifies the Page Scan Repetition mode supported by the remote device with the BD_ADDR. This is the most recent version of the information that was acquired either during the inquiry process or from an HCI_Page_Scan_Repetition_Mode_Change event (see Section 7.7.31). The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 to 16 of the difference are used, and they are mapped to this parameter as bits 0 to 14 respectively. A Clock_Offset_Valid_Flag, located in bit 15 of the Clock_Offset parameter, is used to indicate if the Clock Offset is valid or not. A Connection_Handle for this connection is returned in the HCI_Connection_Complete event (see below). The Allow_Role_Switch parameter specifies if the local device accepts or rejects the request from the remote device to switch roles at connection setup (in the Role parameter of the HCI_Accept_Connection_Request command) (before the local Controller returns an HCI_Connection_Complete event). For a definition of the different packet types see [Vol 2] Part B, Section 6.5.

Note: Enabling as many packet types as possible allows the Link Manager to perform more efficiently.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to be connected. |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 1 | 2-DH1 shall not be used. |
| 2 | 3-DH1 shall not be used. |
| 3 | Ignored; DM1 may be used whether or not this bit is set. |
| 4 | DH1 may be used. |
| 8 | 2-DH3 shall not be used. |
| 9 | 3-DH3 shall not be used. |
| 10 | DM3 may be used. |
| 11 | DH3 may be used. |
| 12 | 2-DH5 shall not be used. |
| 13 | 3-DH5 shall not be used. |
| 14 | DM5 may be used. |
| 15 | DH5 may be used. |
| All other bits | Reserved for future use. |

*Page_Scan_Repetition_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use. |

*Reserved:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved, shall be set to 0x00. |

*Clock_Offset:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0-14 | Bits 16-2 of CLKNPeripheral - CLK |
| 15 | Clock_Offset_Valid_Flag<br>Invalid Clock Offset = 0<br>Valid Clock Offset = 1 |

*Allow_Role_Switch:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The local device will be a Central, and will not accept a role switch requested by the remote device at the connection setup. |
| 0x01 | The local device may be a Central, or may become a Peripheral after accepting a role switch requested by the remote device at the connection setup. |
| All other values | Reserved for future use. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Create_Connection command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. In addition, when the Link Manager determines the connection is established, the BR/EDR Controller, on both BR/EDR Controllers that form the connection, shall send an HCI_Connection_Complete event to each Host. The HCI_Connection_Complete event contains the Connection_Handle if this command is successful.

### 7.1.6 Disconnect command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Disconnect | 0x0006 | Connection_Handle,<br>Reason | *none* |

**Description:**

This command is used to terminate an existing connection. The Connection_Handle parameter indicates which connection is to be disconnected. The Reason parameter indicates the reason for ending the connection and is copied into the error code field of the LMP_DETACH PDU on a BR/EDR connection or the error code field of the LL_TERMINATE_IND or LL_CIS_TERMINATE_IND PDU on an LE connection. All SCO, eSCO, and CIS connections on a physical link should be disconnected before the ACL connection on the same physical connection is disconnected. If it does not, they will be implicitly disconnected as part of the ACL disconnection.

If the Host issues this command when there is a pending HCI_LE_Create_CIS command for the same CIS but before the CIS is created, then this command shall be successful and the CIS shall not be created.

Note: The CIS is created when the Central sends an LL_CIS_IND PDU to the Peripheral for that CIS (see [Vol 6] Part B, Section 5.1.15).

Note: If the Controller follows the requirements of a version of this specification lower than version 6.0, then it can return an error if this command is issued before the CIS is created.

Note: As specified in Section 7.7.5, on the Central, the handle for a CIS remains valid even after disconnection and, therefore, the Host can recreate a disconnected CIS at a later point in time using the same connection handle.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | On the Central, the Controller has not received the HCI_LE_Create_CIS command for the same CIS (a previous CIS with the same CIS_ID in the same CIG that has been terminated or considered lost is not the same CIS for this purpose). | *Command Disallowed* (0x0C) |
| MC | On the Peripheral, the Controller has not generated the HCI_LE_CIS_Established event for that CIS. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x05,<br>0x13 to 0x15,<br>0x1A,<br>0x29,<br>0x3B | *Authentication Failure* error code (0x05), *Other End Terminated Connection* error codes (0x13 to 0x15), *Unsupported Remote Feature* error code (0x1A), *Pairing with Unit Key Not Supported* error code (0x29) and *Unacceptable Connection Parameters* error code (0x3B), see [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Disconnect command, it shall send the HCI_Command_Status event to the Host. The HCI_Disconnection_Complete event will occur at each Host when the termination of the connection has completed, and on the local Host also indicates that this command has been completed. The Reason parameter in the event on the local Host shall be set to the value *Connection Terminated by Local Host* (0x16), while that on the remote Host shall be set to the value of Reason. However, if the termination procedure completes because a timer expires and, therefore, the local Controller cannot determine whether or not Reason was received by the remote Controller, the Reason parameter on the local Host should instead be set to the value *LMP Response Timeout / LL Response Timeout* (0x22).

If this command is issued for a CIS on the Central and the CIS is successfully terminated before being created, or after being created but before being established, then an HCI_LE_CIS_Established event shall also be sent for this CIS with the Status *Operation Cancelled by Host* (0x44).

### 7.1.7 Create Connection Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Create_Connection_Cancel | 0x0008 | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to request cancellation of the ongoing connection creation process, which was started by an HCI_Create_Connection command of the local BR/EDR Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The connection is already established and the Controller has sent the HCI_Connection_Complete event. | *Connection Already Exists* (0x0B) |
| MC | The Controller has not received an HCI_Create_Connection command for the same BD_ADDR. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the HCI_Create_Connection command that was issued before and is the subject of this cancellation request. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Create_Connection_Cancel command succeeded |
| 0x01 to 0xFF | HCI_Create_Connection_Cancel command failed. See [Vol 1] Part F, Controller Error Codes for list of error codes |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the HCI_Create_Connection command that was issued before and is the subject of this cancellation request. |

**Event(s) generated (unless masked away):**

When the HCI_Create_Connection_Cancel command has completed, an HCI_Command_Complete event shall be generated.

If the connection is already established by the Baseband, but the BR/EDR Controller has not yet sent the HCI_Connection_Complete event, then the local device shall detach the link and return an HCI_Command_Complete event with the status “Success”.

The HCI_Connection_Complete event for the corresponding HCI_Create_Connection command shall always be sent. The HCI_Connection_Complete event shall be sent after the HCI_Command_Complete event for the HCI_Create_Connection_Cancel command. If the cancellation was successful, the HCI_Connection_Complete event will be generated with the error code *Unknown Connection Identifier* (0x02).

### 7.1.8 Accept Connection Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Accept_Connection_Request | 0x0009 | BD_ADDR,<br>Role | *none* |

**Description:**

This command is used to accept a new incoming connection request.
The HCI_Accept_Connection_Request command shall only be issued after an HCI_Connection_Request event has occurred. The HCI_Connection_Request event will return the BD_ADDR of the device which is requesting the connection. This command will cause the Link Manager to create a connection to the BR/EDR Controller, with the BD_ADDR specified by the command parameters. The Link Manager will determine how the new connection will be established. This will be determined by the current state of the device, its piconet, and the state of the device to be connected. The Role parameter allows the Host to specify if the Link Manager shall request a role switch and become the Central for this connection. This is a preference and not a requirement. If the Role Switch fails then the connection will still be accepted, and the HCI_Role_Discovery command will reflect the current role.

The Link Manager may terminate the connection if it would be low on resources if the role switch fails. The decision to accept a connection should be completed before the connection accept timeout expires on the local Bluetooth Module.

Note: When accepting a synchronous connection request, the Role parameter is not used and will be ignored by the BR/EDR Controller.

Note: See Section 7.3.3 for the behavior when the HCI_Connection_Request event is masked or the connection is auto accepted.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to be connected |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Become the Central for this connection. The LM will perform the role switch. |
| 0x01 | Remain the Peripheral for this connection. The LM will **NOT** perform the role switch. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

The HCI_Accept_Connection_Request command shall cause the HCI_Command_Status event to be sent from the BR/EDR Controller when the BR/EDR Controller begins setting up the connection. In addition, when the Link Manager determines the connection is established, the local BR/EDR Controller shall send an HCI_Connection_Complete event to its Host, and the remote Controller will send an HCI_Connection_Complete event or an HCI_Synchronous_Connection_Complete event to the Host. The HCI_Connection_Complete event contains the Connection_Handle if this command is successful.

### 7.1.9 Reject Connection Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reject_Connection_Request | 0x000A | BD_ADDR,<br>Reason | *none* |

**Description:**

This command is used to decline a new incoming connection request.
The HCI_Reject_Connection_Request command shall only be called after an HCI_Connection_Request event has occurred. The HCI_Connection_Request event will return the BD_ADDR of the device that is requesting the connection. The Reason parameter will be returned to the connecting device in the Status parameter of the HCI_Connection_Complete event returned to the Host of the connection device, to indicate why the connection was declined.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to reject the connection from. |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0D to 0x0F | Host Reject error code. See [Vol 1] Part F, Controller Error Codes for list of error codes and descriptions. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Reject_Connection_Request command, the Controller shall send the HCI_Command_Status event to the Host. Then, the local BR/EDR Controller will send an HCI_Connection_Complete event to its Host, and the remote device shall send an HCI_Connection_Complete event or an HCI_Synchronous_Connection_Complete event to the Host. The Status parameter of the HCI_Connection_Complete event, which is sent to the Host of the device attempting to make the connection, will contain the Reason parameter from this command.

### 7.1.10 Link Key Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Link_Key_Request_Reply | 0x000B | BD_ADDR,<br>Link_Key | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_Link_Key_Request event from the Controller, and specifies the Link Key stored on the Host to be used as the link key for the connection with the other BR/EDR Controller specified by BD_ADDR. The HCI_Link_Key_Request event will be generated when the BR/EDR Controller needs a Link Key for a connection.

When the BR/EDR Controller generates an HCI_Link_Key_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_Link_Key_Request_-Reply or HCI_Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C.)

When the BR/EDR Controller supports the Secure Connections (Controller Support) feature, it shall discard the Link Key once the connection has been disconnected.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of which the Link Key is for. |

*Link_Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Link Key for the associated BD_ADDR. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Link_Key_Request_Reply command succeeded. |
| 0x01 to 0xFF | HCI_Link_Key_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of which the Link Key request reply has completed. |

### Event(s) generated (unless masked away):

When the HCI_Link_Key_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.11 Link Key Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Link_Key_Request_Negative_Reply | 0x000C | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_Link_Key_Request event from the BR/EDR Controller if the Host does not have a stored Link Key for the connection with the other BR/EDR Controller specified by BD_ADDR. The HCI_Link_Key_Request event will be generated when the BR/EDR Controller needs a Link Key for a connection.

When the Controller generates an HCI_Link_Key_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_Link_Key_Request_Reply or HCI_Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the Link Key is for. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Link_Key_Request_Negative_Reply command succeeded. |
| 0x01 to 0xFF | HCI_Link_Key_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which the HCI_Link_Key_Request_Negative_Reply command has completed. |

**Event(s) generated (unless masked away):**

When the HCI_Link_Key_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.12 PIN Code Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_PIN_Code_Request_Reply | 0x000D | BD_ADDR,<br>PIN_Code_Length,<br>PIN_Code | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_PIN_Code_Request event from the BR/EDR Controller, and specifies the PIN code to use for a connection. The HCI_PIN_Code_Request event will be generated when a connection with remote initiating device has requested pairing.

When the BR/EDR Controller generates an HCI_PIN_Code_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_PIN_Code_Request_Reply or HCI_PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the PIN code is for. |

*PIN_Code_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The PIN code length specifics the length, in octets, of the PIN code to be used.<br>Range: 0x01 to 0x10 |

*PIN_Code:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | PIN code for the device that is to be connected.<br>Note: The PIN_Code parameter is a string parameter. Endianness does therefore not apply to the PIN_Code parameter. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_PIN_Code_Request_Reply command succeeded. |
| 0x01 to 0xFF | HCI_PIN_Code_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which the HCI_PIN_Code_Request_Reply command has completed. |

### Event(s) generated (unless masked away):

When the HCI_PIN_Code_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.13 PIN Code Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_PIN_Code_Request_Negative_Reply | 0x000E | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to a PIN Code request event from the BR/EDR Controller when the Host cannot specify a PIN code to use for a connection. This command will cause the pair request with remote device to fail.

When the BR/EDR Controller generates an HCI_PIN_Code_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_PIN_Code_Request_Reply or HCI_PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which this command is responding to. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_PIN_Code_Request_Negative_Reply command succeeded. |
| 0x01 to 0xFF | HCI_PIN_Code_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which the HCI_PIN_Code_Request_Negative_Reply command has completed. |

**Event(s) generated (unless masked away):**

When the HCI_PIN_Code_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.14 Change Connection Packet Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Change_Connection_Packet_Type | 0x000F | Connection_Handle,<br>Packet_Type | *none* |

**Description:**

This command is used to change which packet types can be used for a connection that is currently established. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type parameter specifies which packet types the Link Manager can use for the connection; the Host shall not specify packet types that the local Controller does not support. When sending HCI ACL Data packets the Link Manager shall only use the packet type(s) specified by Packet_Type or the always-allowed DM1 packet type. The interpretation of the value of Packet_Type will depend on the Link_Type parameter returned in the HCI_Connection_Complete event at the connection setup. Multiple packet types may be specified for Packet_Type by bitwise OR operation of the different packet types. For a definition of the different packet types see [Vol 2] Part B, Section 6.5.

Note: Enabling as many packet types as possible allows the Link Manager to perform more efficiently.

Note: Use the HCI_Setup_Synchronous_Connection command to change an eSCO connection.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Packet_Type:* *Size: 2 octets*

*For ACL Link_Type*

| Bit Number | Parameter Description |
|---|---|
| 1 | 2-DH1 shall not be used. |
| 2 | 3-DH1 shall not be used. |
| 3 | Ignored; DM1 may be used whether or not this bit is set. |
| 4 | DH1 may be used. |

| Bit Number | Parameter Description |
|---|---|
| 8 | 2-DH3 shall not be used. |
| 9 | 3-DH3 shall not be used. |
| 10 | DM3 may be used. |
| 11 | DH3 may be used. |
| 12 | 2-DH5 shall not be used. |
| 13 | 3-DH5 shall not be used. |
| 14 | DM5 may be used. |
| 15 | DH5 may be used. |
| All other bits | Reserved for future use. |

*For SCO Link_Type*

| Bit Number | Parameter Description |
|---|---|
| 5 | HV1 may be used. |
| 6 | HV2 may be used. |
| 7 | HV3 may be used. |
| All other bits | Reserved for future use. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Change_Connection_Packet_Type command, the Controller shall send the HCI_Command_Status event to the Host. In addition, when the Link Manager determines the packet type has been changed for the connection, the Controller on the local device will send an HCI_Connection_Packet_Type_Changed event to the Host. This will be done at the local side only.

### 7.1.15 Authentication Requested command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Authentication_Requested | 0x0011 | Connection_Handle | *none* |

**Description:**

This command is used to try to authenticate the remote device associated with the specified Connection_Handle, which shall be for an ACL connection. On an authentication failure, the BR/EDR Controller or Link Manager shall not automatically detach the link. The Host is responsible for issuing an HCI_Disconnect command to terminate the link if the action is appropriate.

The Controller shall always perform the authentication with the remote device even if the link has already been authenticated or the Controller already has a stored link key.

Note: The Connection_Handle parameter is used to identify the other BR/EDR Controller which forms the connection. The authentication will apply to all Connection_Handles with the same remote BR/EDR Controller.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Authentication_Requested command, it shall send the HCI_Command_Status event to the Host.

If Secure Simple Pairing Mode is enabled, the HCI_Link_Key_Request event shall be generated, and Secure Simple Pairing shall be started only if the Host replies to it with the HCI_Link_Key_Request_Negative_Reply command; if the Host replies to it with the HCI_Link_Key_Request_Reply command, only the authentication procedure (see [Vol 2] Part C, Section 4.2.1) shall be performed and no Secure Simple Pairing shall be started.

If Secure Simple Pairing Mode is not enabled, then the BR/EDR Controller may, but should not, use an existing stored link key. If authentication fails, the HCI_PIN_Code_Request event may be generated.

Using an existing stored link key when Secure Simple Pairing mode is disabled is discouraged because it does not offer the Host a method for enhancing the security of an existing link (e.g., in the case where a profile mandating a minimum passkey length is started over a link that is already authenticated with shorter passkey than the new service requires).

The HCI_Authentication_Complete event is generated when the authentication has been completed for the connection and is the indication that this command has been completed.

### 7.1.16 Set Connection Encryption command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Connection_Encryption | 0x0013 | Connection_Handle,<br>Encryption_Enable | *none* |

**Description:**

This command is used to enable and disable the link level encryption. The Connection_Handle parameter is used to identify the other BR/EDR Controller which forms the connection. The Connection_Handle should be a Connection_Handle for an ACL connection. The encryption setting will apply to all Connection_Handles with the same remote BR/EDR Controller. While the encryption is being changed, the Link Manager will suspend all ACL-U traffic on the connection.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Both devices support both the Secure Connections (Controller Support) and Secure Connections (Host Support) features, Encryption_Enable is set to “Turn Link Level Encryption OFF”, and encryption is enabled on the specified Connection_Handle. | *Encryption Mode Not Acceptable* (0x25) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Encryption_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Turn Link Level Encryption OFF. |
| 0x01 | Turn Link Level Encryption ON. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Set_Connection_Encryption command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the Link Manager has completed enabling/disabling encryption for the connection, the local BR/EDR Controller shall send an HCI_Encryption_Change event to the Host, and the BR/EDR Controller on the remote device will also generate an HCI_Encryption_Change event.

### 7.1.17 Change Connection Link Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Change_Connection_Link_Key | 0x0015 | Connection_Handle | *none* |

**Description:**

This command is used to force both devices of a connection associated with the Connection_Handle to generate a new link key. The link key is used for authentication and encryption of connections. The Connection_Handle shall be a Connection_Handle for an ACL connection.

Note: The Connection_Handle parameter is used to identify the other BR/EDR Controller forming the connection.

Note: The resulting link key, generated as a result of HCI_Change_Connection_Link_Key command, will be of equal link key strength to the previously used link key.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Change_Connection_Link_Key command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the Link Manager has changed the Link Key for the connection, the local BR/EDR Controller shall send an HCI_Link_Key_Notification event and an HCI_Change_Connection_Link_Key_Complete event to the Host, and the remote BR/EDR Controller will also generate an HCI_Link_Key_Notification event. The HCI_Link_Key_Notification event indicates that a new connection link key is valid for the connection.

### 7.1.18 Link Key Selection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Link_Key_Selection | 0x0017 | Key_Flag | *none* |

**Description:**

This command is used to force the device that is Central of the piconet to use the temporary link key or the semi-permanent link keys. The temporary link key is used for encryption of broadcast messages within a piconet, and the semi-permanent link keys are used for private encrypted point-to-point communication. The Key_Flag parameter is used to indicate which Link Key (temporary link key or the semi-permanent link keys) shall be used.

Note: When at least one Peripheral in the piconet cannot support AES-CCM encryption, encrypted broadcast packets will not be received by Peripherals where both the Controller and Host support Secure Connections. When all Peripherals in the piconet support AES-CCM encryption, broadcast packets will not be encrypted and can be received by Peripherals that have AES-CCM encryption enabled.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | All Peripherals in the piconet support AES-CCM encryption and Key_Flag is set to “Use Temporary Link Key”. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Key_Flag:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use semi-permanent Link Keys. |
| 0x01 | Use Temporary Link Key. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Link_Key_Selection command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When

the Link Manager has changed link key, the BR/EDR Controller on both the local and the remote device shall send an HCI_Link_Key_Type_Changed event to the Host. If no change is required or the command fails, only the Controller on the local device shall send the event. The Connection_Handle on the Central side shall be a Connection_Handle for one of the existing connections to a Peripheral. On the Peripheral side, the Connection_Handle shall be a Connection_Handle to the initiating Central.

The HCI_Link_Key_Type_Changed event contains the status of this command.

### 7.1.19 Remote Name Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_Name_Request | 0x0019 | BD_ADDR,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Clock_Offset | *none* |

**Description:**

This command is used to obtain the user-friendly name of another BR/EDR Controller. The user-friendly name is used to enable the user to distinguish one BR/EDR Controller from another. The BD_ADDR parameter is used to identify the device for which the user-friendly name is to be obtained. The Page_Scan_Repetition_Mode parameter specifies the Page Scan Repetition mode supported by the remote device with the BD_ADDR. This is the most recent version of the information that was acquired either during the inquiry process or from an HCI_Page_Scan_Repetition_Mode_Change event (see Section 7.7.31). The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 to 16 of the difference are used and they are mapped to this parameter as bits 0 to 14 respectively. A Clock_Offset_Valid_Flag, located in bit 15 of Clock_Offset, is used to indicate if Clock Offset is valid or not.

When the HCI_Remote_Supported_Host_Features_Notification event is unmasked and when the HCI_Remote_Name_Request command initiates a connection, the Link Manager shall read the remote LMP features mask pages 0 and 1.

Note: If no connection exists between the local device and the device corresponding to the BD_ADDR, a temporary Link Layer connection will be established to obtain the LMP features and name of the remote device.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device whose name is requested. |

*Page_Scan_Repetition_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use. |

*Reserved:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved, shall be set to 0x00. |

*Clock_Offset:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 to 14 | Bits 2 to 16 of CLKNPeripheral - CLK |
| 15 | Clock_Offset_Valid_Flag<br>Invalid Clock Offset = 0<br>Valid Clock Offset = 1 |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Remote_Name_Request command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. If a temporary Link Layer connection was established, then when the Link Manager has completed the LMP sequence to obtain the remote Host supported features, if present, the BR/EDR Controller on the local device shall send an HCI_Remote_Host_Supported_Features_Notification event. When the Link Manager has completed the LMP messages to obtain the remote name, the local BR/EDR Controller shall send an HCI_Remote_Name_Request_Complete event to the Host. If the remote Host supported features page is present, the HCI_Remote_Host_Supported_Features_Notification event shall be sent before the HCI_Remote_Name_Request_Complete event. If not, only the HCI_Remote_Name_Request_Complete event shall be sent.

### 7.1.20 Remote Name Request Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_Name_Request_Cancel | 0x001A | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to request cancellation of the ongoing remote name request process, which was started by the HCI_Remote_Name_Request command.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the HCI_Remote_Name_Request command that was issued before and that is subject of this cancellation request |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Remote_Name_Request_Cancel command succeeded |
| 0x01 to 0xFF | HCI_Remote_Name_Request_Cancel command failed. See [Vol 1] Part F, Controller Error Codes for list of error codes |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the HCI_Remote_Name_Request_Cancel command that was issued before and that was subject of this cancellation request |

**Event(s) generated (unless masked away):**

When the HCI_Remote_Name_Request_Cancel command has completed, an HCI_Command_Complete event shall be generated.

If the HCI_Remote_Name_Request_Cancel command is sent to the BR/EDR Controller without a preceding HCI_Remote_Name_Request command to the same device, the Controller shall return an HCI_Command_Complete event with the error code *Invalid HCI Command Parameters* (0x12).

The HCI_Remote_Name_Request_Complete event for the corresponding HCI_Remote_Name_Request command shall always be sent. The HCI_Remote_-Name_Request_Complete event shall be sent after the HCI_Command_Complete event for the HCI_Remote_Name_Request_Cancel command. If the cancellation was successful, the HCI_Remote_Name_Request_Complete event shall be generated with the error code *Unknown Connection Identifier* (0x02).

### 7.1.21 Read Remote Supported Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Supported_Features | 0x001B | Connection_Handle | *none* |

**Description:**

This command requests a list of the supported features for the remote device identified by the Connection_Handle parameter. The Connection_Handle shall be a Connection_Handle for an ACL-U logical link. The HCI_Read_Remote_Supported_Features_Complete event will return a list of the LMP features. For details see [Vol 2] Part C, Section 3.3.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Read_Remote_Supported_Features command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the Link Manager has completed the LMP sequence to determine the remote features or has determined that it will be using a cached copy, the BR/EDR Controller on the local device shall send an HCI_Read_Remote_Supported_Features_Complete event to the Host. The HCI_Read_Remote_Supported_Features_Complete event contains the status of this command, and parameters describing the supported features of the remote device.

### 7.1.22 Read Remote Extended Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Extended_Features | 0x001C | Connection_Handle,<br>Page_Number | *none* |

**Description:**

This command returns the requested page of the extended LMP features for the remote device identified by the specified Connection_Handle. The Connection_Handle shall be the Connection_Handle for an ACL-U logical link. This command is only available if the extended features feature is implemented by the remote device. The HCI_Read_Remote_Extended_Features_Complete event will return the requested information. For details see [Vol 2] Part C, Section 3.3.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Requests the normal LMP features as returned by the HCI_Read_Remote_Supported_Features command |
| 0x01 to 0xFF | Return the corresponding page of features |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Read_Remote_Extended_Features command the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the Link Manager has completed the LMP sequence to determine the remote extended features or has determined that it will be using a cached copy, the Controller on the local device shall generate an HCI_Read_Remote_Extended_Features_Complete event to the Host. The

HCI_Read_Remote_Extended_Features_Complete event contains the page number and the remote features returned by the remote device.

### 7.1.23 Read Remote Version Information command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Version_Information | 0x001D | Connection_Handle | *none* |

**Description:**

This command will obtain the values for the version information for the remote device identified by the Connection_Handle parameter. The Connection_Handle shall be a Connection_Handle for an ACL-U or LE-U logical link.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Read_Remote_Version_Information command, the Controller shall send the HCI_Command_Status event to the Host.
When the Link Manager or Link Layer has completed the sequence to determine the remote version information, the local Controller shall send an HCI_Read_Remote_Version_Information_Complete event to the Host. The HCI_Read_Remote_Version_Information_Complete event contains the status of this command, and parameters describing the version and subversion of the LMP or Link Layer used by the remote device.

### 7.1.24 Read Clock Offset command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Clock_Offset | 0x001F | Connection_Handle | *none* |

**Description:**

Both the System Clock and the clock offset to a remote device are used to determine what hopping frequency is used by a remote device for page scan. This command allows the Host to read the clock offset of remote devices. The clock offset can be used to speed up the paging procedure when the local device tries to establish a connection to a remote device, for example, when the local Host has issued an HCI_Create_Connection or HCI_Remote_Name_Request command. The Connection_Handle shall be a Connection_Handle for an ACL-U logical link.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Read_Clock_Offset command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. If this command is requested at the Central then, when the Link Manager has completed the LMP messages to obtain the Clock Offset information, the BR/EDR Controller on the local BR/EDR Controller shall send an HCI_Read_Clock_Offset_Complete event to the Host. If this command is requested at the Peripheral, the LM shall immediately send an HCI_Read_Clock_Offset_Complete event to the Host, without an exchange of LMP PDUs.

### 7.1.25 Read LMP Handle command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_LMP_Handle | 0x0020 | Connection_Handle | Status,<br>Connection_Handle,<br>LMP_Handle,<br>Reserved |

**Description:**

This command reads the current LMP Handle associated with the Connection_Handle. The Connection_Handle shall identify a SCO or eSCO connection. If the Connection_Handle is a SCO Connection_Handle, then this command shall read the LMP SCO Handle for this connection. If the Connection_Handle is an eSCO Connection_Handle, then this command shall read the LMP eSCO Handle for this connection.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_LMP_Handle command succeeded. |
| 0x01 to 0xFF | HCI_Read_LMP_Handle command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*LMP_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The LMP Handle is the LMP Handle that is associated with this Connection_Handle.<br>For a synchronous handle, this would be the LMP Synchronous Handle used when negotiating the synchronous connection in the link manager. |

*Reserved:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 | Reserved, shall be set to 0x00000000. |

**Event(s) generated (unless masked away):**

When the HCI_Read_LMP_Handle command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.26 Setup Synchronous Connection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Setup_Synchronous_Connection | 0x0028 | Connection_Handle,<br>Transmit_Bandwidth,<br>Receive_Bandwidth,<br>Max_Latency,<br>Voice_Setting,<br>Retransmission_Effort,<br>Packet_Type | *none* |

**Description:**

This command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. If the Connection_Handle refers to an ACL link a new synchronous logical transport will be added. If the Connection_Handle refers to an already existing synchronous logical transport (eSCO only) this link will be modified. The parameters are specified per connection. This synchronous connection can be used to transfer synchronous voice at 64 kb/s or transparent synchronous data.

When used to setup a new synchronous logical transport, the Connection_Handle parameter shall specify an ACL connection with which the new synchronous connection will be associated. The other parameters relate to the negotiation of the link, and may be reconfigured during the lifetime of the link. The transmit and receive bandwidth specify how much bandwidth shall be available for transmitting and for receiving data. While in many cases the receive and transmit bandwidth parameters may be equal, they may be different. The latency specifies an upper limit to the time between the eSCO (or SCO) instants, plus the size of the retransmission window, plus the length of the reserved synchronous slots for this logical transport. The content format specifies the settings for voice or transparent data on this connection. The retransmission effort specifies the extra resources that are allocated to this connection if a packet may need to be retransmitted. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to don't care.

When used to modify an existing synchronous logical transport, the Transmit_Bandwidth, Receive_Bandwidth and Voice_Setting shall be set to the same values as were used during the initial setup. The Packet_Type, Retransmission_Effort and Max_Latency parameters may be modified.

The Packet_Type field is a bitmap specifying which packet types the LM shall accept in the negotiation of the link parameters. Multiple packet types are specified by bitwise OR of the packet type codes in the table. At least one packet type shall be specified for

each negotiation. It is recommended to enable as many packet types as possible. The Host may enable packet types that are not supported by the local Controller.

A Connection_Handle for the new synchronous connection will be returned in an HCI_Synchronous_Connection_Complete event.

The link manager may choose any combination of packet types, timing, and retransmission window sizes that satisfy the parameters given. This may be achieved by using more frequent transmissions of smaller packets. The link manager may choose to set up either a SCO or an eSCO connection, if the parameters allow, using the corresponding LMP sequences.

Note: To modify a SCO connection, use the HCI_Change_Connection_Packet_Type command.

If the lower layers cannot achieve the exact transmit and receive bandwidth requested subject to the other parameters, then the link shall be rejected.

A synchronous connection may only be created when an ACL connection already exists.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The ACL link has encryption enabled using AES-CCM and the Controller cannot establish an eSCO transport (e.g., Packet_Type only allows SCO packet types). | *Rejected due to Security Reasons* (0x0E) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Transmit_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Transmit bandwidth in octets per second. |

*Receive_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Receive bandwidth in octets per second. |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0003 | Reserved for future use |
| 0x0004 to 0xFFFE | This is a value, in milliseconds, representing the upper limit of the sum of the synchronous interval and the size of the eSCO window, where the eSCO window is the reserved slots plus the retransmission window. (See [Vol 2] Part B, Figure 8.9) |
| 0xFFFF | Don't care. |

*Voice_Setting:* *Size: 2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12. | |

*Retransmission_Effort:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No retransmissions (SCO or eSCO connection allowed) |
| 0x01 | At least one retransmission, optimize for power consumption (eSCO connection required). |
| 0x02 | At least one retransmission, optimize for link quality (eSCO connection required) |
| 0xFF | Don’t care (SCO or eSCO connection allowed) |
| All other values | Reserved for future use |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | HV1 may be used. |
| 1 | HV2 may be used. |
| 2 | HV3 may be used. |
| 3 | EV3 may be used. |
| 4 | EV4 may be used. |
| 5 | EV5 may be used. |

| Bit Number | Parameter Description |
|---|---|
| 6 | 2-EV3 shall not be used. |
| 7 | 3-EV3 shall not be used. |
| 8 | 2-EV5 shall not be used. |
| 9 | 3-EV5 shall not be used. |
| All other bits | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Setup_Synchronous_Connection command, it shall send the HCI_Command_Status event to the Host. In addition, when the LM determines the connection is established, the local BR/EDR Controller shall send an HCI_Synchronous_Connection_Complete event to the local Host, and the remote Controller will send an HCI_Synchronous_Connection_Complete event or an HCI_Connection_Complete event to the remote Host. The HCI_Synchronous_Connection_Complete event contains the Connection_Handle if this command is successful.

If this command is used to change the parameters of an existing eSCO link, the HCI_Synchronous_Connection_Changed event is sent to both Hosts. In this case no HCI_Synchronous_Connection_Complete event, HCI_Connection_Request event, or HCI_Connection_Complete event will be sent to either Host. This command cannot be used to change the parameters of a SCO link.

### 7.1.27 Accept Synchronous Connection Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Accept_-Synchronous_Connection_Request | 0x0029 | BD_ADDR,<br>Transmit_Bandwidth,<br>Receive_Bandwidth,<br>Max_Latency,<br>Voice_Setting,<br>Retransmission_Effort,<br>Packet_Type | *none* |

**Description:**

This command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection.

The command shall only be issued after an HCI_Connection_Request event with link type SCO or eSCO has occurred. The HCI_Connection_Request event contains the BD_ADDR of the device requesting the connection. The decision to accept a connection should be taken before the timer Connection_Accept_Timeout expires on the local device.

The Host shall include in the Packet_Type parameter at least one packet type for the transport (SCO or eSCO) specified in the incoming request. The Controller shall ignore any packet types in the Packet_Type parameter for the other transport.

If the ACL link has encryption enabled using AES-CCM then the Host shall not accept a request where the link type is SCO.

The parameter set of the HCI_Accept_Synchronous_Connection_Request command is the same as for the HCI_Setup_Synchronous_Connection command. The Transmit_Bandwidth and Receive_Bandwidth values are required values for the new link and shall be met. The Max_Latency is an upper bound to the acceptable latency for the Link, as defined in the HCI_Setup_Synchronous_Connection command (see Section 7.1.26) and shall not be exceeded. Voice_Setting specifies the encoding in the same way as in the HCI_Setup_Synchronous_Connection command and shall be met. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to don't care. The Packet_Type parameter is a bit mask specifying the synchronous packet types that are allowed on the link.

If the Link Type of the incoming request is SCO, then the Controller shall ignore the Transmit_Bandwidth, Receive_Bandwidth, and Retransmission_Effort parameters.

Note: See Section 7.3.3 for the behavior when the HCI_Connection_Request event is masked or the connection is auto accepted.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device requesting the connection |

*Transmit_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Transmit bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care (default) |

*Receive_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Receive bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care (default) |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0003 | Reserved for future use |
| 0x0004 to 0xFFFE | This is a value in milliseconds representing the upper limit of the sum of the synchronous interval and the size of the eSCO window, where the eSCO window is the reserved slots plus the retransmission window. (See [Vol 2] Part B, Figure 8.9) |
| 0xFFFF | Don't care (default) |

*Voice_Setting:* *Size: 2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12. When links are auto-accepted, the default is to use the values written by the HCI_Write_Voice_Setting command. | |

*Retransmission_Effort:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No retransmissions |
| 0x01 | At least one retransmission, optimize for power consumption |
| 0x02 | At least one retransmission, optimize for link quality |
| 0xFF | Don't care (default) |
| All other values | Reserved for future use |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | HV1 may be used. |
| 1 | HV2 may be used. |
| 2 | HV3 may be used. |
| 3 | EV3 may be used. |
| 4 | EV4 may be used. |
| 5 | EV5 may be used. |
| 6 | 2-EV3 shall not be used. |
| 7 | 3-EV3 shall not be used |
| 8 | 2-EV5 shall not be used. |
| 9 | 3-EV5 shall not be used. |
| All other bits | Reserved for future use |

Default: 0x003F - means all defined packet types may be used.

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

The HCI_Accept_Synchronous_Request command shall cause the HCI_Command_-Status event to be sent from the BR/EDR Controller when the BR/EDR Controller starts setting up the connection. When the link setup is complete, the local BR/EDR Controller shall send an HCI_Synchronous_Connection_Complete event to its Host, and the remote BR/EDR Controller will send an HCI_Synchronous_Connection_-Complete event to the Host. The HCI_Synchronous_Connection_Complete will contain the Connection_Handle and the link parameters if the setup is successful.

### 7.1.28 Reject Synchronous Connection Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reject_-Synchronous_Connection_Request | 0x002A | BD_ADDR,<br>Reason | *none* |

**Description:**

This command is used to decline an incoming request for a synchronous link. It shall only be issued after an HCI_Connection_Request event with Link Type equal to SCO or eSCO has occurred. The HCI_Connection_Request event contains the BD_ADDR of the device requesting the connection. The Reason parameter will be returned to the initiating Host in the Status parameter of the HCI_Synchronous_Connection_Complete event on the remote side.

If the ACL link has encryption enabled using AES-CCM and the requested link type was SCO, the Host shall reject the request using this command with the Reason parameter set to *Rejected due to Security Reasons* (0x0E).

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device requesting the connection |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0D to 0x0F | Host Reject error code. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Reject_Synchronous_Connection_Request, it shall send an HCI_Command_Status event to the Host. When the setup is terminated, the local Controller shall send an HCI_Synchronous_Connection_Complete event to its Host, and the remote Controller will send an HCI_Synchronous_Connection_Complete event to the Host with the Reason code from this command.

### 7.1.29 IO Capability Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_IO_Capability_Request_Reply | 0x002B | BD_ADDR,<br>IO_Capability,<br>OOB_Data_Present,<br>Authentication_Requirements | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_IO_Capability_Request event from the Controller, and specifies the current IO capabilities of the Host. This includes the Host input, output and out-of-band (OOB) capabilities.

If an authenticated link key is not required by the Host, the Authentication Requirements parameter may be set to one of the following:

- MITM Protection Not Required – No Bonding
- MITM Protection Not Required – Dedicated Bonding
- MITM Protection Not Required – General Bonding

If both Hosts set the Authentication_Requirements parameter to one of the above values, the Link Managers shall use the numeric comparison authentication procedure and the Hosts shall use the Just Works association model.

If an authenticated link key is required by the Host, the Authentication Requirements parameter shall be set to one of the following:

- MITM Protection Required – No Bonding
- MITM Protection Required – Dedicated Bonding
- MITM Protection Required – General Bonding

In addition, the following requirements apply:

- If one or both Hosts set the Authentication Requirements parameter to one of the above values, the Link Managers shall use the IO_Capability parameter to determine the authentication procedure.
- A Host that sets the Authentication_Requirements parameter to one of the above values shall verify that the resulting Link Key type meets the security requirements requested.

If the Host has received OOB authentication data from a device with the same BD_ADDR sent in the HCI_IO_Capabilty_Request event, then the OOB_Data_Present parameter shall be set to:

- "P-192 OOB authentication data from remote device present" when the Host has received only P-192 OOB data,
- "P-256 OOB authentication data from remote device present" when the Host has received only P-256 OOB data, or
- "P-192 and P-256 OOB authentication data from remote device present" when the Host has received both P-192 and P-256 OOB data.

Otherwise OOB_Data_Present shall be set to "OOB authentication data not present".

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

*IO_Capability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | DisplayOnly |
| 0x01 | DisplayYesNo |
| 0x02 | KeyboardOnly |
| 0x03 | NoInputNoOutput |
| All other values | Reserved for future use |

*OOB_Data_Present:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | OOB authentication data not present |
| 0x01 | P-192 OOB authentication data from remote device present |
| 0x02 | P-256 OOB authentication data from remote device present |
| 0x03 | P-192 and P-256 OOB authentication data from remote device present |
| All other values | Reserved for future use |

*Authentication_Requirements:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | MITM Protection Not Required – No Bonding. Numeric comparison with automatic accept allowed. |
| 0x01 | MITM Protection Required – No Bonding. Use IO Capabilities to determine authentication procedure |
| 0x02 | MITM Protection Not Required – Dedicated Bonding. Numeric comparison with automatic accept allowed. |
| 0x03 | MITM Protection Required – Dedicated Bonding. Use IO Capabilities to determine authentication procedure |
| 0x04 | MITM Protection Not Required – General Bonding. Numeric Comparison with automatic accept allowed. |
| 0x05 | MITM Protection Required – General Bonding. Use IO capabilities to determine authentication procedure. |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_IO_Capability_Request_Reply command succeeded |
| 0x01 to 0xFF | HCI_IO_Capability_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

### Event(s) generated (unless masked away):

When the HCI_IO_Capability_Request_Reply command has completed, an HCI_Command_Complete event shall be generated. When the device is the initiator of Secure Simple Pairing, an HCI_IO_Capability_Response event shall be generated. Additionally, when the OOB_Data_Present parameter indicates that OOB authentication data from the remote device is present, the HCI_Remote_OOB_Data_Request event shall be generated.

### 7.1.30 User Confirmation Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Confirmation_Request_Reply | 0x002C | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_User_Confirmation_Request event and indicates that the user selected "yes". It is also used when the Host has no input and no output capabilities.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_User_Confirmation_Request_Reply command succeeded |
| 0x01 to 0xFF | HCI_User_Confirmation_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_User_Confirmation_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.31 User Confirmation Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Confirmation_Request_-Negative_Reply | 0x002D | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_User_Confirmation_Request event and indicates that the user selected "no". This command shall cause the initiating Link Manager to transmit an LMP_NUMERIC_COMPARISON_FAILED PDU and terminate Secure Simple Pairing.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_User_Confirmation_Request_Negative_Reply command succeeded |
| 0x01 to 0xFF | HCI_User_Confirmation_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_User_Confirmation_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.32 User Passkey Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Passkey_Request_Reply | 0x002E | BD_ADDR,<br>Numeric_Value | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_User_Passkey_Request event and specifies the Numeric_Value (passkey) entered by the user to be used in the Secure Simple Pairing process.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

*Numeric_Value:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0x000F423F | Numeric value (passkey) entered by user. Valid values are decimal 000000 to 999999. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_User_Passkey_Request_Reply command succeeded |
| 0x01 to 0xFF | HCI_User_Passkey_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_User_Passkey_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.33 User Passkey Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Passkey_Request_Negative_-Reply | 0x002F | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_User_Passkey_Request event and indicates the Host could not provide a passkey. This command shall cause the initiating Link Manager to transmit an LMP_PASSKEY_ENTRY_FAILED PDU and terminate Secure Simple Pairing.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_User_Passkey_Request_Negative_Reply command succeeded |
| 0x01 to 0xFF | HCI_User_Passkey_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_User_Passkey_Negative_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.34 Remote OOB Data Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_OOB_Data_Request_Reply | 0x0030 | BD_ADDR,<br>C,<br>R | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_Remote_OOB_Data_Request event with the C and R values received via an OOB transfer from a remote device identified by BD_ADDR. If the R value is not present in the received OOB data from the remote device, the Host shall set R to zeros.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device from which the C and R values were received |

*C:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Hash C |

*R:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Randomizer R |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Remote_OOB_Data_Request_Reply command succeeded |
| 0x01to 0xFF | HCI_Remote_OOB_Data_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_Remote_OOB_Data_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.35 Remote OOB Data Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_-OOB_Data_Request_Negative_Reply | 0x0033 | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_Remote_OOB_Data_Request event that the Host does not have the C and R values associated with the remote device identified by BD_ADDR.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Remote_OOB_Data_Request_Negative_Reply command succeeded |
| 0x01 to 0xFF | HCI_Remote_OOB_Data_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_Remote_OOB_Data_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.36 IO Capability Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_IO_Capability_Request_Negative_-Reply | 0x0034 | BD_ADDR,<br>Reason | Status,<br>BD_ADDR |

**Description:**

This command shall be used to reject a pairing attempt after an HCI_IO_Capability_Request event has been received by the Host. The reason for the rejection is given in the Reason parameter. The error code *Secure Simple Pairing Not Supported by Host* (0x37) shall not be used in the Reason parameter.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason that Secure Simple Pairing rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_IO_Capability_Request_Negative_Reply command succeeded. |
| 0x01 to 0xFF | HCI_IO_Capability_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_IO_Capability_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.1.37 [This section is no longer used]

### 7.1.38 [This section is no longer used]

### 7.1.39 [This section is no longer used]

### 7.1.40 [This section is no longer used]

### 7.1.41 [This section is no longer used]

### 7.1.42 [This section is no longer used]

### 7.1.43 [This section is no longer used]

### 7.1.44 [This section is no longer used]

### 7.1.45 Enhanced Setup Synchronous Connection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enhanced_-Setup_Synchronous_-Connection | 0x003D | Connection_Handle,<br>Transmit_Bandwidth,<br>Receive_Bandwidth,<br>Transmit_Coding_Format,<br>Receive_Coding_Format,<br>Transmit_Codec_Frame_Size,<br>Receive_Codec_Frame_Size,<br>Input_Bandwidth,<br>Output_Bandwidth,<br>Input_Coding_Format,<br>Output_Coding_Format,<br>Input_Coded_Data_Size,<br>Output_Coded_Data_Size,<br>Input_PCM_Data_Format,<br>Output_PCM_Data_Format,<br>Input_PCM_Sample_Payload_MSB_Position,<br>Output_PCM_Sample_Payload_MSB_Position,<br>Input_Data_Path,<br>Output_Data_Path,<br>Input_Transport_Unit_Size,<br>Output_Transport_Unit_Size,<br>Max_Latency,<br>Packet_Type,<br>Retransmission_Effort | *none* |

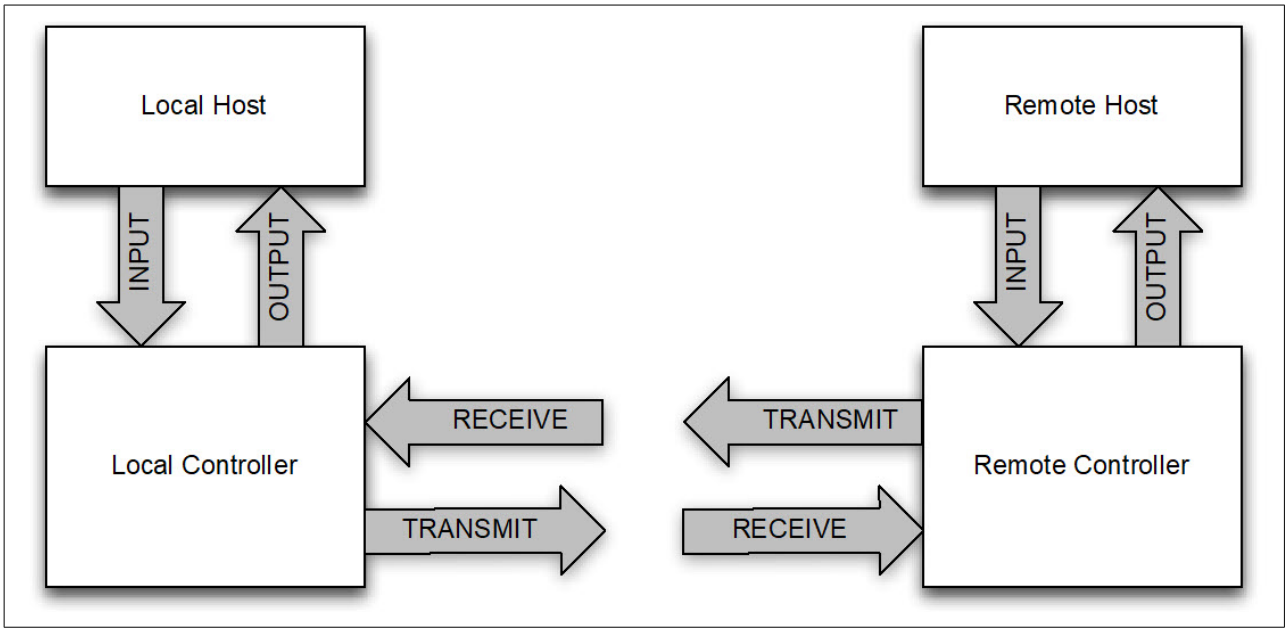
**Description:**

This command adds a new, or modifies an existing, synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. If the Connection_Handle refers to an ACL link, then a new synchronous logical transport shall be added. If the Connection_Handle refers to an existing synchronous logical transport (eSCO only), then this link shall be modified. The parameters are specified per connection. This synchronous connection can be used to transfer synchronous voice data or transparent synchronous data. If the ACL link has encryption enabled using AES-CCM and the Controller cannot establish an eSCO transport (e.g. the Host parameters restricting the packet types to SCO packet types),

the Controller shall return the error code *Rejected due to Security Reasons* (0x0E) and a SCO transport will not be established.

When used to setup a new synchronous logical transport, the Connection_Handle parameter shall specify an ACL connection with which the new synchronous connection shall be associated. The other parameters relate to the negotiation of the link, and may be reconfigured during the lifetime of the link.

The following terms are used to describe the four different audio paths: Transmit, Receive, Input and Output. The Transmit and Receive paths are from the perspective of the local Controller's radio. The Input and Output paths are from the perspective of the Controller.




The following parameters are used to describe the transmit and receive paths over the air:

- The Transmit_Bandwidth and Receive_Bandwidth parameters specify how much bandwidth shall be available for transmitting and for receiving data. The Host shall set the Transmit_Bandwidth and Receive_Bandwidth parameters to be equal or shall set one of them to be zero and the other non-zero.
- The Transmit_Coding_Format and Receive_Coding_Format parameters specify the coding format used for transmitted or received data. The Host shall set the Transmit_Coding_Format and Receive_Coding_Formats to be equal. When the Transmit_Coding_Format and Receive_Coding_Format parameters are not equal to CVSD, A-law or μ-law, the Link Manager shall map these to Transparent air mode.
- The Transmit_Codec_Frame_Size and Receive_Codec_Frame_Size parameters specify the frame size produced by the codecs in the context of over-the-air coding.

The over-the-air packet size should have the following relationship with the codec frame size:

Packet_Size = Frame_Size × N, or

Packet_Size = Frame_Size ÷ N

where N is an integer.

The following parameters are used to describe the coding format used prior to encapsulating over the audio data transport path:

- The Input_Bandwidth and Output_Bandwidth specify the nominal rate at which the Host or Controller transfers data (for HCI transports this omits the HCI header). The Host shall either set the Input_Bandwidth and Output_Bandwidth to be equal, or shall set one of them to be zero and the other non-zero.
- The Input_Coding_Format and Output_Coding_Format parameters specify the coding format used over the transport. The Host shall set the Input_Coding_Format and Output_Coding_Format to be equal.
- The Input_Coded_Data_Size and Output_Coded_Data_Size specify the number of bits in each sample or frame of data. For CVSD, a frame of data shall be 8 bits.
- The Input_PCM_Data_Format and Output_PCM_Data_Format parameters specify the data format over the transport for linear samples. They shall be ignored when the data is encoded in any other way.
- The Input_PCM_Sample_Payload_MSB_Position and Output_PCM_Sample_Payload_MSB_Position parameters indicate, for linear samples, how many bit positions that the MSB of the sample is away from starting at the MSB of the data. They shall be ignored when the data is encoded in any other way. For example, if Input_Coded_Data_Size =16 and Input_PCM_Sample_Payload_MSB_Position = 3, then each sample is actually only 13 bits, the MSB (which is the sign bit for signed formats) is bit 12 (counting from the LSB at bit 0), and the contents of bits 13, 14, and 15 of each sample shall be ignored.

The following parameters describe the audio data transport path characteristics:

- The Input_Data_Path and Output_Data_Path parameters specify the audio data transport path. When set to 0x00, the audio data path shall be over the HCI transport. When set to 0xFF, audio test mode (see Section 7.6.2) is selected (this is only applicable during test mode). When set to 0x01 to 0xFE, the audio data path shall use non-HCI transport data paths (e.g. PCM interface) with logical transport channel numbers. The meanings of these logical transport channel numbers are vendor specific.
- The Input_Transport_Unit_Size and Output_Transport_Unit_Size indicate how many bits are in each unit of data delivered by the audio data transport. Except for HCI,

the meaning of “unit” depends on the Host transport used and, therefore, is vendor specific (for example, on a PCM transport this should indicate the number of bits transported per sync pulse, and would normally be 8 or 16). The Host shall set the Input_Transport_Unit_Size and Output_Transport_Unit_Size to be equal. For HCI Host transport the Host shall set them to 0.

The following parameters are used by the Link Manager to negotiate the synchronous transport:

- The Max_Latency parameter specifies an upper limit to the time between the eSCO (or SCO) instants, plus the size of the retransmission window, plus the length of the reserved synchronous slots for this logical transport.
- The Packet_Type parameter is a bitmap specifying which synchronous packet types may be used by the Link Manager in the negotiation of the link parameters. Multiple packet types are specified by bitwise OR of the packet type codes in the table. At least one packet type shall be specified for each negotiation. It is recommended to enable as many packet types as possible. The Host may enable packet types that are not supported by the local Controller.
- The Retransmission_Effort parameter specifies the extra resources that are allocated to this connection if a packet may need to be retransmitted. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to “Don’t care”.

The following restrictions shall apply:

- Either both the Transmit_Coding_Format and Input_Coding_Format shall be “transparent” or neither shall be. If both are “transparent”, the Transmit_Bandwidth and the Input_Bandwidth shall be the same and the Controller shall not modify the data sent to the remote device.
- Either both the Receive_Coding_Format and Output_Coding_Format shall be “transparent” or neither shall be. If both are “transparent”, the Receive_Bandwidth and the Output_Bandwidth shall be the same and the Controller shall not modify the data sent to the Host.

A Connection_Handle for the new synchronous connection will be returned in the HCI_Synchronous_Connection_Complete event if the command is used to set up a new synchronous connection.

When used to modify an existing synchronous logical transport, only the Packet_Type, Retransmission_Effort and Max_Latency parameters may be modified.

The Link Manager may choose any combination of packet types, timing, and retransmission window sizes that satisfy the parameters given. This may be achieved

by using more frequent transmissions of smaller packets. The link manager may choose to set up either a SCO or an eSCO connection, if the parameters allow, using the corresponding LMP sequences.

Note: To modify a SCO connection, use the HCI_Change_Connection_Packet_Type command.

If the lower layers cannot achieve the exact transmit and receive bandwidth requested subject to the other parameters, or cannot achieve the transcoding or resampling implied by the parameters, then the link creation or link modification shall be rejected. A synchronous connection may only be created when an ACL connection already exists.

The data at the audio data transport interface shall be treated as a stream of bits. The bits in each unit of data delivered by the transport shall be taken LSB first, and the units shall be taken in the order of delivery. The samples, encoded samples, frames, or other entity to be transcoded for transmission, or that has been transcoded after reception, shall be taken in the order of transmission or reception, with each entity taken LSB first.

For example, if the audio data transport uses 16 bit units and the Input or Output coding format is A-law, each unit represents two samples with the first in the 8 least significant bits and the second in the 8 most significant bits. Similarly, if the audio data transport uses 8 bit units and the Input or Output coding format is linear PCM with a size of 16 bits, the 8 least significant bits of each sample are transmitted first.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Transmit_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Transmit bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care |

*Receive_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Receive bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care |

*Transmit_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Transmit_Coding_Format is not 0xFF. |

*Receive_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Receive_Coding_Format is not 0xFF. |

*Transmit_Codec_Frame_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: 0x0001 to 0xFFFF, the actual size of the over-the-air encoded frame in octets. |

*Receive_Codec_Frame_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: 0x0001 to 0xFFFF, the actual size of the over-the-air encoded frame in octets. |

*Input_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Host to Controller nominal data rate in octets per second. |

*Output_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Controller to Host nominal data rate in octets per second. |

*Input_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Input_Coding_Format is not 0xFF. |

*Output_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Output_Coding_Format is not 0xFF. |

*Input_Coded_Data_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Size, in bits, of the sample or framed data |

*Output_Coded_Data_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Size, in bits, of the sample or framed data |

*Input_PCM_Data_Format:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers for PCM_Data_Format |

*Output_PCM_Data_Format:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers for PCM_Data_Format |

*Input_PCM_Sample_Payload_MSB_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of bit positions within an audio sample that the MSB of the sample is away from starting at the MSB of the data. |

*Output_PCM_Sample_Payload_MSB_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of bit positions within an audio sample that the MSB of the sample is away from starting at the MSB of the data. |

*Input_Data_Path:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI |
| 0x01 to 0xFE | Logical_Channel_Number. The meaning of the logical channels will be vendor specific. |
| 0xFF | Audio test mode |

*Output_Data_Path:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI |
| 0x01 to 0xFE | Logical_Channel_Number. The meaning of the logical channels will be vendor specific. |
| 0xFF | Audio test mode |

*Input_Transport_Unit_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 1 to 255 | The number of bits in each unit of data received from the Host over the audio data transport. |
| 0 | Not applicable (implied by the choice of audio data transport) |

*Output_Transport_Unit_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 1 to 255 | The number of bits in each unit of data sent to the Host over the audio data transport. |
| 0 | Not applicable (implied by the choice of audio data transport) |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0003 | Reserved for future use |
| 0x0004 to 0xFFFE | The value, in milliseconds, representing the upper limit of the sum of the synchronous interval and the size of the eSCO window, where the eSCO window is the reserved slots plus the retransmission window. (See [Vol 2] Part B, Figure 8.9) |
| 0xFFFF | Don't care. |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | HV1 may be used |
| 1 | HV2 may be used |
| 2 | HV3 may be used |
| 3 | EV3 may be used |
| 4 | EV4 may be used |
| 5 | EV5 may be used |
| 6 | 2-EV3 shall not be used |
| 7 | 3-EV3 shall not be used |
| 8 | 2-EV5 shall not be used |
| 9 | 3-EV5 shall not be used |
| All other bits | Reserved for future use |

Note: 0x003F means all packet types can be used.

*Retransmission_Effort:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No retransmission (SCO or eSCO connection allowed) |
| 0x01 | At least one retransmission, optimize for power consumption (eSCO connection required) |
| 0x02 | At least one retransmission, optimize for link quality (eSCO connection required) |
| 0xFF | Don't care (SCO or eSCO connection allowed) |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Enhanced_Setup_Synchronous_Connection command, it shall send the HCI_Command_Status event to the Host. In addition, when the LM determines that the connection is established, the local BR/EDR Controller shall send an HCI_Synchronous_Connection_Complete event to the local Host, and the remote Controller will send an HCI_Synchronous_Connection_Complete event or an HCI_Connection_Complete event to the remote Host. The HCI_Synchronous_Connection_Complete event contains the Connection_Handle if this command is successful.

This command cannot be used to change the parameters of a SCO link.

### 7.1.46 Enhanced Accept Synchronous Connection Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enhanced_-Accept_Synchronous_-Connection_Request | 0x003E | BD_ADDR,<br>Transmit_Bandwidth,<br>Receive_Bandwidth,<br>Transmit_Coding_Format,<br>Receive_Coding_Format,<br>Transmit_Codec_Frame_Size,<br>Receive_Codec_Frame_Size,<br>Input_Bandwidth,<br>Output_Bandwidth,<br>Input_Coding_Format,<br>Output_Coding_Format,<br>Input_Coded_Data_Size,<br>Output_Coded_Data_Size,<br>Input_PCM_Data_Format,<br>Output_PCM_Data_Format,<br>Input_PCM_Sample_Payload_MSB_Position,<br>Output_PCM_Sample_Payload_MSB_Position,<br>Input_Data_Path,<br>Output_Data_Path,<br>Input_Transport_Unit_Size,<br>Output_Transport_Unit_Size,<br>Max_Latency,<br>Packet_Type,<br>Retransmission_Effort | *none* |

**Description:**

This command is used to accept an incoming request for a synchronous connection and to present the local Link Manager with the acceptable parameter values for the synchronous connection. This command shall only be issued after an HCI_Connection_Request event, with link type SCO or eSCO, has occurred. An HCI_Connection_Request event contains the BD_ADDR of the device requesting the connection. The command to accept a connection must be received by the Controller before the timer Connection_Accept_Timeout expires on the local device.

The parameter set of the HCI_Enhanced_Accept_Synchronous_Connection_Request command is the same as for the HCI_Enhanced_Setup_Synchronous_Connection command except for the Connection_Handle in the HCI_Enhanced_Setup_-Synchronous_Connection command, which is replaced by the BD_ADDR in the HCI_Enhanced_Accept_Synchronous_Connection_Request command. See Section 7.1.45 for the descriptions of these parameters.

The Host shall include in the Packet_Type parameter at least one packet type for the transport (SCO or eSCO) specified in the incoming request. The Controller shall ignore any packet types in the Packet_Type parameter for the other transport.

If the Link Type of the incoming request is SCO, then the Controller shall ignore the Transmit_Bandwidth, Receive_Bandwidth, and Retransmission_Effort parameters.

Note: See Section 7.3.3 for the behavior when the HCI_Connection_Request event is masked or the connection is auto accepted.

If the ACL link has encryption enabled using AES-CCM then the Host shall not accept a request where the link type is SCO.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device requesting the connection |

*Transmit_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Transmit bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care |

*Receive_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFE | Receive bandwidth in octets per second. |
| 0xFFFFFFFF | Don’t care |

*Transmit_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Transmit_Coding_Format is not 0xFF. |

*Receive_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Receive_Coding_Format is not 0xFF. |

*Transmit_Codec_Frame_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: 0x0001 to 0xFFFF, the actual size of the over-the-air encoded frame in octets. |

*Receive_Codec_Frame_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: 0x0001 to 0xFFFF, the actual size of the over-the-air encoded frame in octets. |

*Input_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Host to Controller nominal data rate in octets per second. |

*Output_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Controller to Host nominal data rate in octets per second. |

*Input_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Input_Coding_Format is not 0xFF. |

*Output_Coding_Format:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding_Format |
| Octets 1 to 4 | Octet 1 to 2: Company ID, see Assigned Numbers for Company Identifier.<br>Octet 3 to 4: Vendor specific codec ID.<br>Shall be ignored if octet 0 of Output_Coding_Format is not 0xFF. |

*Input_Coded_Data_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Size, in bits, of the sample or framed data |

*Output_Coded_Data_Size:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Size, in bits, of the sample or framed data |

*Input_PCM_Data_Format:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers for PCM_Data_Format |

*Output_PCM_Data_Format:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers for PCM_Data_Format |

*Input_PCM_Sample_Payload_MSB_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of bit positions within an audio sample that the MSB of the sample is away from starting at the MSB of the data. |

*Output_PCM_Sample_Payload_MSB_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of bit positions within an audio sample that the MSB of the sample is away from starting at the MSB of the data. |

*Input_Data_Path:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI |
| 0x01 to 0xFE | Logical_Channel_Number. The meaning of the logical channels will be vendor specific. |
| 0xFF | Audio test mode |

*Output_Data_Path:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI |
| 0x01 to 0xFE | Logical_Channel_Number. The meaning of the logical channels will be vendor specific. |
| 0xFF | Audio test mode |

*Input_Transport_Unit_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 1 to 255 | The number of bits in each unit of data received from the Host over the audio data transport. |
| 0 | Not applicable (implied by the choice of audio data transport) |

*Output_Transport_Unit_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 1 to 255 | The number of bits in each unit of data sent to the Host over the audio data transport. |
| 0 | Not applicable (implied by the choice of audio data transport) |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0003 | Reserved for future use |
| 0x0004 to 0xFFFE | The value in milliseconds representing the upper limit of the sum of the synchronous interval, and the size of the eSCO window, where the eSCO window is the reserved slots plus the retransmission window. (See [Vol 2] Part B, Figure 8.9) |
| 0xFFFF | Don't care. |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | HV1 may be used |
| 1 | HV2 may be used |
| 2 | HV3 may be used |
| 3 | EV3 may be used |
| 4 | EV4 may be used |
| 5 | EV5 may be used |
| 6 | 2-EV3 shall not be used |
| 7 | 3-EV3 shall not be used |
| 8 | 2-EV5 shall not be used |
| 9 | 3-EV5 shall not be used |
| All other bits | Reserved for future use |

Note: 0x003F means all packet types can be used.

*Retransmission_Effort:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No retransmission |
| 0x01 | At least one retransmission, optimize for power consumption |
| 0x02 | At least one retransmission, optimize for link quality |
| 0xFF | Don't care |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

The HCI_Enhanced_Accept_Synchronous_Request command requests the local BR/EDR Controller to start setting up the connection. When this action commences, the HCI_Command_Status event shall be sent by the BR/EDR Controller. When the link setup is complete, the BR/EDR Controller shall send an HCI_Synchronous_Connection_Complete event to its Host, and the remote BR/EDR Controller will send an HCI_Synchronous_Connection_Complete event to its Host. The HCI_Synchronous_Connection_Complete will contain the Connection_Handle and the link parameters if the setup is successful.

### 7.1.47 Truncated Page command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Truncated_Page | 0x003F | BD_ADDR,<br>Page_Scan_Repetition_Mode,<br>Clock_Offset | *none* |

**Description:**

This command is used to page the BR/EDR Controller with the specified BD_ADDR and then abort the paging sequence after an ID response has been received. See [Vol 2] Part B, Section 8.3.3 for additional information.

The Page_Scan_Repetition_Mode parameter specifies the Page Scan Repetition mode supported by the remote BR/EDR Controller with the BD_ADDR. This is the most recent version of the information that was acquired either during the inquiry process or from an HCI_Page_Scan_Repetition_Mode_Change event (see Section 7.7.31).

The Clock_Offset parameter is the difference between the local BR/EDR Controller's own clock and the clock of the remote BR/EDR Controller with BD_ADDR. Only bits 2 to 16 of the difference are used, and they are mapped to this parameter as bits 0 to 14 respectively. A Clock_Offset_Valid_Flag, located in bit 15 of the Clock_Offset parameter, indicates if the Clock Offset is valid or not.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to page |

*Page_Scan_Repetition_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use. |

*Clock_Offset:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 to 14 | Bits 2 to 16 of CLKNPeripheral - CLK |
| 15 | Clock Offset is valid |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Truncated_Page command the BR/EDR Controller shall send the HCI_Command_Status event to the Host. In addition, when the Truncated Page procedure has completed, the BR/EDR Controller shall send an HCI_Truncated_Page_Complete event to the Host.

### 7.1.48 Truncated Page Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Truncated_Page_Cancel | 0x0040 | BD_ADDR | Status,<br>BD_ADDR |

**Description:**

This command is used to request cancellation of an ongoing Truncated_Page process previously started by an HCI_Truncated_Page command.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device to which the HCI_Truncated_Page command was previously issued and that is the subject of the cancellation request. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Truncated_Page_Cancel command succeeded. |
| 0x01 to 0xFF | HCI_Truncated_Page_Cancel command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device to which the HCI_Truncated_Page command was previously issued before and that is the subject of the cancellation request. |

**Event(s) generated (unless masked away):**

When the HCI_Truncated_Page_Cancel command has completed, an HCI_Command_Complete event shall be generated.

If the truncated page procedure has already completed, but the BR/EDR Controller has not yet sent the HCI_Truncated_Page_Complete event, then the local device shall return an HCI_Command_Complete event with status “Success”.

If the HCI_Truncated_Page_Cancel command is sent to the BR/EDR Controller without a pending HCI_Truncated_Page command to the same device, the BR/EDR

Controller shall return an HCI_Command_Complete event with the error code *Unknown Connection Identifier* (0x02).

Note: From the BR/EDR Controller perspective this is identical to the situation where the HCI_Truncated_Page command has already completed and the HCI_Truncated_Page_Complete event already sent.

### 7.1.49 Set Connectionless Peripheral Broadcast command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Connectionless_Peripheral_-Broadcast | 0x0041 | Enable,<br>LT_ADDR,<br>LPO_Allowed,<br>Packet_Type,<br>Interval_Min,<br>Interval_Max,<br>Supervision_Timeout | Status,<br>LT_ADDR,<br>Interval |

**Description:**

This command controls the Connectionless Peripheral Broadcast functionality in the BR/EDR Controller. Connectionless Peripheral Broadcast mode may be enabled or disabled by the Enable parameter. If Enable is set to 0x00 and the Synchronization Train substate is active, then the Controller shall also exit the Synchronization Train substate. If Enable is set to 0x00, the remaining parameters shall be ignored.

The LT_ADDR indicated in the Set_Connectionless_Peripheral_Broadcast shall be pre-allocated using the HCI_Set_Reserved_LT_ADDR command.

The LPO_Allowed parameter informs the BR/EDR Controller whether it is allowed to sleep.

The Packet_Type parameter specifies which packet types are allowed. The Host shall either enable BR packet types only, or shall enable EDR and DM1 packet types only.

The Interval_Min and Interval_Max parameters specify the range from which the BR/EDR Controller shall select the Connectionless Peripheral Broadcast Interval. The selected Interval is returned.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The LT_ADDR has not been reserved. | *Unknown Connection Identifier* (0x02) |
| MC | The Controller is unable to reserve sufficient band-width for the requested activity. | *Rejected due to Limited Resources* (0x0D) |

**Command parameters:**

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disabled |
| 0x01 | Enabled |
| All other values | Reserved for future use |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR used for Connectionless Peripheral Broadcast |
| All other values | Reserved for future use |

*LPO_Allowed:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | BR/EDR Controller shall not sleep (that is, clock accuracy shall be equal to or better than ± 20 ppm) |
| 0x01 | BR/EDR Controller may sleep (that is, clock accuracy shall be equal to or better than ± 250 ppm) |
| All other values | Reserved for future use |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 1 | 2-DH1 shall not be used |
| 2 | 3-DH1 shall not be used |
| 3 | DM1 may be used |
| 4 | DH1 may be used |
| 8 | 2-DH3 shall not be used |
| 9 | 3-DH3 shall not be used |
| 10 | DM3 may be used |
| 11 | DH3 may be used |
| 12 | 2-DH5 shall not be used |
| 13 | 3-DH5 shall not be used |
| 14 | DM5 may be used |

| Bit Number | Parameter Description |
|---|---|
| 15 | DH5 may be used |
| All other bits | Reserved for future use. |

*Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum interval between Connectionless Peripheral Broadcast packets in slots.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

*Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum interval between Connectionless Peripheral Broadcast packets in slots.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration in slots after which the BR/EDR Controller reports an HCI_Connectionless_-Peripheral_Broadcast_Timeout event if it is unable to transmit a Connectionless Peripheral Broadcast packet.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Connectionless_Peripheral_Broadcast command succeeded. |
| 0x01 to 0xFF | HCI_Set_Connectionless_Peripheral_Broadcast command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR used for Connectionless Peripheral Broadcast |
| All other values | Reserved for future use |

*Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Actual interval between Connectionless Peripheral Broadcast packets in slots.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

**Event(s) generated (unless masked away):**

When the HCI_Set_Connectionless_Peripheral_Broadcast command has completed, an HCI_Command_Complete event shall be generated.

If the BR/EDR Controller is unable to transmit a Connectionless Peripheral Broadcast packet for *Supervision_Timeout slots*, it shall generate an HCI_Connectionless_Peripheral_Broadcast_Timeout event.

### 7.1.50 Set Connectionless Peripheral Broadcast Receive command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_-Connectionless_-Peripheral_-Broadcast_Receive | 0x0042 | Enable,<br>BD_ADDR,<br>LT_ADDR,<br>Interval,<br>Clock_Offset,<br>Next_Connectionless_Peripheral_Broadcast_Clock,<br>Supervision_Timeout,<br>Remote_Timing_Accuracy,<br>Skip,<br>Packet_Type,<br>AFH_Channel_Map | Status,<br>BD_ADDR,<br>LT_ADDR |

**Description:**

This command controls the reception of Connectionless Peripheral Broadcast packets in the BR/EDR Controller of a Connectionless Peripheral Broadcast Receiver. If the Enable parameter is set to Disabled, the BR/EDR Controller does not attempt to receive Connectionless Peripheral Broadcast packets and the remaining parameters shall be ignored. If the Enable parameter is set to Enabled, the BR/EDR Controller starts attempting to receive Connectionless Peripheral Broadcast packets on the specified LT_ADDR.

The Interval parameter specifies the interval of the Connectionless Peripheral Broadcast to be used by the BR/EDR Controller.

The Skip parameter specifies the number of consecutive Connectionless Peripheral Broadcast instants which the receiver may skip after successfully receiving a Connectionless Peripheral Broadcast packet.

The Packet_Type parameter specifies which packet types are allowed. The Host shall either enable BR packet types only or shall enable EDR and DM1 packet types only.

The AFH_Channel_Map parameter is the AFH channel map used by the Transmitter for the PBD logical link, and is obtained by the Receiver's Host from the HCI_Synchronization_Train_Received event.

**Command parameters:**

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disabled |
| 0x01 | Enabled |
| All other values | Reserved for future use |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Connectionless Peripheral Broadcast transmitter |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR to use for receiving Connectionless Peripheral Broadcast messages |
| All other values | Reserved for future use |

*Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Interval between Connectionless Peripheral Broadcast packets instants in slots.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

*Clock_Offset:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | (CLKNreceiver - CLKNtransmitter) *mod* $2^{28}$ |

*Next_Connectionless_Peripheral_Broadcast_Clock:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | CLK for next Connectionless Peripheral Broadcast instant |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration in slots to continue listening for Connectionless Peripheral Broadcast packets after the last successfully received Connectionless Peripheral Broadcast packet.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

*Remote_Timing_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Timing accuracy of the Transmitter in ppm. Typical values are 20 ppm and 250 ppm. |

*Skip:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Connectionless Peripheral Broadcast instants to skip after successfully receiving a Broadcast packet. |

*Packet_Type:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 1 | 2-DH1 shall not be used |
| 2 | 3-DH1 shall not be used |
| 3 | DM1 may be used |
| 4 | DH1 may be used |
| 8 | 2-DH3 shall not be used |
| 9 | 3-DH3 shall not be used |
| 10 | DM3 may be used |
| 11 | DH3 may be used |
| 12 | 2-DH5 shall not be used |
| 13 | 3-DH5 shall not be used |
| 14 | DM5 may be used |
| 15 | DH5 may be used |
| All other bits | Reserved for future use. |

*AFH_Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX<br>XXXXXXXXXX<br>XX | This parameter contains 80 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>0: channel *n* is unused<br>1: channel *n* is used<br>The most significant bit (bit 79) is reserved for future use<br>At least $N_{min}$ channels shall be marked as used (see [Vol 2] Part B, Section 2.3.1) |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Connectionless_Peripheral_Broadcast_Receive command succeeded. |
| 0x01 to 0xFF | HCI_Set_Connectionless_Peripheral_Broadcast_Receive command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Connectionless Peripheral Broadcast transmitter |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR used for receiving Connectionless Peripheral Broadcast messages |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Set_Connectionless_Peripheral_Broadcast_Receive command has been received, an HCI_Command_Complete event shall be generated. Completion of the HCI_Set_Connectionless_Peripheral_Broadcast_Receive command does not require reception of a Connectionless Peripheral Broadcast packet.
If the BR/EDR Controller does not receive a Connectionless Peripheral Broadcast packet for *CPB_supervisionTO* slots, it shall generate an HCI_Connectionless_Peripheral_Broadcast_Timeout event.

### 7.1.51 Start Synchronization Train command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Start_Synchronization_Train | 0x0043 | *none* | *none* |

**Description:**

This command controls the Synchronization Train functionality in the BR/EDR Controller. Connectionless Peripheral Broadcast mode shall be enabled on the BR/EDR Controller before this command may be used. After receiving this command and returning an HCI_Command_Status event, the Baseband starts attempting to send synchronization train packets containing information related to the enabled Connectionless Peripheral Broadcast packet timing.

Note: The AFH_Channel_Map used in the synchronization train packets is configured by the HCI_Set_AFH_Host_Channel_Classification command and the local channel classification in the BR/EDR Controller.

The synchronization train packets will be sent using the parameters specified by the latest HCI_Write_Synchronization_Train_Parameters command. The Synchronization Train will continue until *synchronization_trainTO* slots (as specified in the last HCI_Write_Synchronization_Train_Parameters command) have passed or until the Host disables the Connectionless Peripheral Broadcast logical transport.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connectionless Peripheral Broadcast mode is not enabled. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

None

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Start_Synchronization_Train command, it shall send an HCI_Command_Status event to the Host.

### 7.1.52 Receive Synchronization Train command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Receive_Synchronization_Train | 0x0044 | BD_ADDR,<br>Sync_Scan_Timeout,<br>Sync_Scan_Window,<br>Sync_Scan_Interval | *none* |

**Description:**

This command requests synchronization with the specified Connectionless Peripheral Broadcast Transmitter. The Sync_Scan_Window parameter specifies the duration of each scan and the Sync_Scan_Interval parameter specifies the interval between the start of consecutive scan windows. An HCI_Synchronization_Train_Received event shall be sent if a synchronization train packet is received with a non-zero Connectionless Peripheral Broadcast LT_ADDR or if the BR/EDR Controller fails to receive a synchronization train packet for *synchronization_scanTO* slots.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Connectionless Peripheral Broadcast transmitter |

*Sync_Scan_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration in slots to search for the synchronization train<br>Shall be greater than or equal to Sync_Scan_Window; only even values are valid |

*Sync_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration in slots to listen for a synchronization train packet on a single frequency<br>Range: 0x0022 to 0xFFFC; only even values are valid |

*Sync_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration in slots between the start of consecutive scan windows<br>Shall be greater than or equal to Sync_Scan_Window+0x0002; only even values are valid |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Receive_Synchronization_Train command, it shall send an HCI_Command_Status event to the Host. In addition, when the BR/EDR Controller receives, or fails to receive within the duration specified by *Sync_Scan_Timeout*, the synchronization train from the Connectionless Peripheral Broadcast transmitter, it shall send an HCI_Synchronization_Train_Received event to the Host.

### 7.1.53 Remote OOB Extended Data Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_-<br>OOB_Extended_Data_Request_Reply | 0x0045 | BD_ADDR,<br>C_192,<br>R_192,<br>C_256,<br>R_256 | Status,<br>BD_ADDR |

**Description:**

This command is used to reply to an HCI_Remote_OOB_Data_Request event with the C and R values derived with the P-192 public key and the C and R values associated with the P-256 public key received via an OOB transfer from a remote device identified by BD_ADDR. If the C_192 and R_192 values are not present in the received OOB data from the remote device, the Host shall set C_192 and R_192 to zeros. If the C_256 and R_256 values are not present in the received OOB data from the remote device, the Host shall set C_256 and R_256 to zeros.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Neither Secure Connections (Host Support) nor Secure Simple Pairing (Host Support) has been enabled. | *Command Disallowed* (0x0C) |

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device from which the C and R values were received |

*C_192:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX<br>XXXXXXXXXXX<br>XXXXXXXXX | Secure Simple Pairing Hash C derived from the P-192 public key. |

*R_192:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Randomizer associated with the P-192 public key. |

*C_256:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Hash C derived from the P-256 public key. |

*R_256:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Randomizer associated with the P-256 public key. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Remote_OOB_Extended_Data_Request_Reply command succeeded. |
| 0x01 to 0xFF | HCI_Remote_OOB_Extended_Data_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device from which the C and R values were received |

**Event(s) generated (unless masked away):**

When the HCI_Remote_OOB_Extended_Data_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

## 7.2 Link Policy commands

The Link Policy commands provide methods for the Host to affect how the Link Manager manages the piconet. When Link Policy commands are used, the LM still controls how Bluetooth piconets and scatternets are established and maintained, depending on adjustable policy parameters. These policy commands modify the Link Manager behavior that can result in changes to the Link Layer connections with Bluetooth remote devices.

Note: Only one ACL connection can exist between two BR/EDR Controllers, and therefore there can only be one ACL HCI Connection_Handle for each physical Link Layer Connection. The BR/EDR Controller provides policy adjustment mechanisms to provide support for a number of different policies. This capability allows one Bluetooth module to be used to support many different usage models, and the same Bluetooth module can be incorporated in many different types of BR/EDR Controllers.

For the Link Policy commands, the OGF is defined as 0x02.

### 7.2.1 Hold Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Hold_Mode | 0x0001 | Connection_Handle,<br>Hold_Mode_Max_Interval,<br>Hold_Mode_Min_Interval | *none* |

**Description:**

This command is used to alter the behavior of the Link Manager, and have it place the ACL Baseband connection associated by the specified Connection_Handle into the Hold mode. The Connection_Handle shall identify an ACL connection and not a SCO or eSCO connection. The Hold_Mode_Max_Interval and Hold_Mode_Min_Interval command parameters specify the length of time the Host is requesting that the connection is in the Hold mode. The local and remote devices will negotiate the length in the Hold mode. The Hold_Mode_Max_Interval parameter is used to specify the maximum length of the Hold interval for which the Host may actually enter into the Hold mode after negotiation with the remote device. The Hold interval defines the amount of time between when the Hold mode begins and when the Hold mode is completed. The Hold_Mode_Min_Interval parameter is used to specify the minimum length of the Hold interval for which the Host may actually enter into the Hold mode after the negotiation with the remote device. Therefore the Hold_Mode_Min_Interval shall not be greater than the Hold_Mode_Max_Interval. The BR/EDR Controller will return the actual Hold interval in the Interval parameter of the HCI_Mode_Change event, if the command is successful. This command enables the Host to support a low-power policy for itself or

several other BR/EDR Controllers, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.

Note: If the Host sends data to the BR/EDR Controller with a Connection_Handle corresponding to a connection in Hold mode, the BR/EDR Controller will keep the data in its buffers until either the data can be transmitted (the Hold mode has ended) or a flush, a flush timeout or a disconnection occurs. This is valid even if the Host has not yet been notified of the Hold mode through an HCI_Mode_Change event when it sends the data.

Note: The above is not valid for an HCI ACL Data packet sent from the Host to the BR/EDR Controller on the Central side where the Connection_Handle is a Connection_Handle used for broadcast and the Broadcast_Flag is set to Active Broadcast. The broadcast data will then never be received by Peripherals in Hold mode.

The Hold_Mode_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Hold_Mode_Max_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum acceptable number of Baseband slots to wait in Hold mode.<br>Time Length of the Hold = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0002 to 0xFFFE; only even values are valid.<br>Time Range: 1.25 ms to 40.9 s<br>Mandatory Range: 0x0014 to 0x8000 |

*Hold_Mode_Min_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum acceptable number of Baseband slots to wait in Hold mode.<br>Time Length of the Hold = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0002 to 0xFF00; only even values are valid<br>Time Range: 1.25 ms to 40.9 s<br>Mandatory Range: 0x0014 to 0x8000 |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Hold_Mode command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. The HCI_Mode_Change event shall occur when the Hold mode has started and the HCI_Mode_Change event shall occur again when the Hold mode has completed for the specified Connection_Handle. The HCI_Mode_Change event signaling the end of the Hold mode is an estimation of the Hold mode ending if the event is for a remote BR/EDR Controller.

### 7.2.2 Sniff Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Sniff_Mode | 0x0003 | Connection_Handle,<br>Sniff_Max_Interval,<br>Sniff_Min_Interval,<br>Sniff_Attempt,<br>Sniff_Timeout | *none* |

**Description:**

This command is used to alter the behavior of the Link Manager and have it place the ACL Baseband connection associated with the specified Connection_Handle into Sniff mode. The Connection_Handle parameter is used to identify which ACL link connection is to be placed in Sniff mode and shall identify an ACL connection, not a SCO or eSCO, connection. The Sniff_Max_Interval and Sniff_Min_Interval command parameters are used to specify the requested acceptable maximum and minimum periods in Sniff mode. The Sniff_Min_Interval shall not be greater than the Sniff_Max_Interval. The sniff interval defines the amount of time between each consecutive sniff period. The BR/EDR Controller will return the actual sniff interval in the Interval parameter of the HCI_Mode_Change event, if the command is successful. For a description of the meaning of the Sniff_Attempt and Sniff_Timeout parameters, see [Vol 2] Part B, Section 8.7. Sniff_Attempt is there called $N_{sniff\ attempt}$ and Sniff_Timeout is called $N_{sniff\ timeout}$. This command enables the Host to support a low-power policy for itself or several other BR/EDR Controllers, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.

Note: If the Host sends data to the BR/EDR Controller with a Connection_Handle corresponding to a connection in Sniff mode, the BR/EDR Controller will keep the data in its buffers until either the data can be transmitted or a flush, a flush timeout or a disconnection occurs. This is valid even if the Host has not yet been notified of Sniff mode through an HCI_Mode_Change event when it sends the data.

Note: It is possible for the Central to transmit data to a Peripheral without exiting Sniff mode (see description in [Vol 2] Part B, Section 8.7).

Note: The above is not valid for an HCI ACL Data packet sent from the Host to the BR/EDR Controller on the Central side where the Connection_Handle is a Connection_Handle used for broadcast and the Broadcast_Flag is set to Active Broadcast. In that case, the broadcast data will only be received by a Peripheral in Sniff mode if that Peripheral happens to listen to the Central when the broadcast is made.

The Sniff_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Sniff_Max_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0002 to 0xFFFE; only even values are valid<br>Mandatory Range: 0x0006 to 0x0540<br>Time = N × 0.625 ms<br>Time Range: 1.25 ms to 40.9 s |

*Sniff_Min_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0002 to 0xFFFE; only even values are valid<br>Mandatory Range: 0x0006 to 0x0540<br>Time = N × 0.625 ms<br>Time Range: 1.25 ms to 40.9 s |

*Sniff_Attempt:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Number of Baseband receive slots for sniff attempt.<br>Length = N × 1.25 ms<br>Range: 0x0001 to 0x7FFF<br>Time Range: 1.25 ms to 40.9 s<br>Mandatory Range for Controller: 1 to $T_{sniff}$÷2 |

*Sniff_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Number of Baseband receive slots for sniff timeout.<br>Length = N × 1.25 ms<br>Range: 0x0000 to 0x7FFF<br>Time Range: 0 ms to 40.9 s<br>Mandatory Range for Controller: 0 to 0x0028 |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Sniff_Mode command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. The HCI_Mode_Change event shall occur when Sniff mode has started for the specified Connection_Handle.

### 7.2.3 Exit Sniff Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Exit_Sniff_Mode | 0x0004 | Connection_Handle | *none* |

**Description:**

This command is used to end Sniff mode for a Connection_Handle, which is currently in Sniff mode. The Connection_Handle shall identify an ACL connection and not a SCO or eSCO connection. The Link Manager will determine and issue the appropriate LMP commands to remove Sniff mode for the associated Connection_Handle.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When BR/EDR Controller receives the HCI_Exit_Sniff_Mode command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. The HCI_Mode_Change event shall occur when Sniff mode has ended for the specified Connection_Handle.

### 7.2.4 [This section is no longer used]

### 7.2.5 [This section is no longer used]

### 7.2.6 QoS Setup command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_QoS_Setup | 0x0007 | Connection_Handle,<br>Unused,<br>Service_Type,<br>Token_Rate,<br>Peak_Bandwidth,<br>Latency,<br>Delay_Variation | *none* |

**Description:**

This command is used to specify Quality of Service parameters for a Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection. These QoS parameter are the same parameters as L2CAP QoS. For more detail see [Vol 3] Part A, Logical Link Control and Adaptation Protocol Specification. This allows the Link Manager to have all of the information about what the Host is requesting for each connection. The LM will determine if the QoS parameters can be met. BR/EDR Controllers that are both Peripherals and Centrals can use this command. When a device is a Peripheral, this command will trigger an LMP request to the Central to provide the Peripheral with the specified QoS as determined by the LM. When a device is a Central, this command is used to request a Peripheral to accept the specified QoS as determined by the LM of the Central. Connection_Handle is used to identify for which connection the QoS request is requested.

The Unused parameter is reserved for future use.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | This value shall be used by the Host. |
| All other values | Reserved for future use. |

*Service_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic. |
| 0x01 | Best Effort. |
| 0x02 | Guaranteed. |
| All other values | Reserved for future use. |

*Token_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second. |

*Peak_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second. |

*Latency:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Latency in microseconds. |

*Delay_Variation:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Delay Variation in microseconds. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_QoS_Setup command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the Link Manager has completed the LMP messages to establish the requested QoS parameters, the BR/EDR Controller shall send an HCI_QoS_Setup_Complete event to the Host, and the event may also be generated on the remote side if there was LMP negotiation. The values of the parameters of the HCI_QoS_Setup_Complete event may, however, be different on the initiating and the remote side. The HCI_QoS_Setup_Complete event returned by the BR/EDR Controller on the local side

contains the status of this command, and returned QoS parameters describing the supported QoS for the connection.

Note: If the Link Manager performs an LMP transaction that involves the flow parameter values on the remote side, the remote Controller can send an HCI_Flow_Specification_Complete event or HCI_QoS_Setup_Complete event to the remote Host.

### 7.2.7 Role Discovery command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Role_Discovery | 0x0009 | Connection_Handle | Status,<br>Connection_Handle,<br>Current_Role |

**Description:**

This command is used for a Host to determine which role the device is performing for a particular Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Role_Discovery command succeeded |
| 0x01 to 0xFF | HCI_Role_Discovery command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Current_Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Current Role is Central for this Connection_Handle. |
| 0x01 | Current Role is Peripheral for this Connection_Handle. |

**Event(s) generated (unless masked away):**

When the HCI_Role_Discovery command has completed, an HCI_Command_Complete event shall be generated.

### 7.2.8 Switch Role command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Switch_Role | 0x000B | BD_ADDR,<br>Role | *none* |

**Description:**

This command is used to switch the current BR/EDR role the device is performing for a particular connection with another specified BR/EDR Controller. The BD_ADDR parameter indicates for which connection the role switch is to be performed and shall specify a BR/EDR Controller for which a connection already exists. The Role parameter indicates the requested new role that the local device performs.

If there is an (e)SCO connection between the local device and the device identified by the BD_ADDR parameter, an attempt to perform a role switch shall be rejected by the local device.

If the connection between the local device and the device identified by the BD_ADDR parameter is placed in Sniff mode, an attempt to perform a role switch shall be rejected by the local device.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the connected device with which a role switch is to be performed. |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Change own Role to Central for this BD_ADDR. |
| 0x01 | Change own Role to Peripheral for this BD_ADDR. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Switch_Role command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host. When the role

switch is performed, an HCI_Role_Change event shall occur to indicate that the roles have been changed, and will be communicated to both Hosts. If no change is required, only the Controller on the local device shall send the event. If a Baseband role switch is attempted but fails, the local Controller shall send the event and the remote Controller may send it.

### 7.2.9 Read Link Policy Settings command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Policy_Settings | 0x000C | Connection_Handle | Status,<br>Connection_Handle,<br>Link_Policy_Settings |

**Description:**

This command will read the Link Policy setting for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection. Section 6.18.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Link_Policy_Settings command succeeded. |
| 0x01 to 0xFF | HCI_Read_Link_Policy_Settings command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Link_Policy_Settings:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable Role Switch. |
| 1 | Enable Hold mode. |

| Bit Number | Parameter Description |
|---|---|
| 2 | Enable Sniff mode. |
| All other bits | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Link_Policy_Settings command has completed, an HCI_Command_Complete event shall be generated.

### 7.2.10 Write Link Policy Settings command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Link_Policy_Settings | 0x000D | Connection_Handle,<br>Link_Policy_Settings | Status,<br>Connection_Handle |

**Description:**

This command writes the Link Policy setting for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection. See Section 6.18.

The default value is the value set by the HCI_Write_Default_Link_Policy_Settings command.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Link_Policy_Settings:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable Role Switch. |
| 1 | Enable Hold mode. |
| 2 | Enable Sniff mode. |
| All other bits | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Link_Policy_Settings command succeeded. |
| 0x01 to 0xFF | HCI_Write_Link_Policy_Settings command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_Write_Link_Policy_Settings command has completed, an HCI_Command_Complete event shall be generated.

### 7.2.11 Read Default Link Policy Settings command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Default_Link_Policy_-Settings | 0x000E | *none* | Status,<br>Default_Link_Policy_Settings |

**Description:**

This command reads the Default Link Policy setting for all new BR/EDR connections.

Note: See the Link Policy Settings configuration parameter (Section 6.18) for more information.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Link_Policy_Settings command succeeded |
| 0x01 to 0xFF | HCI_Read_Link_Policy_Settings command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Default_Link_Policy_Settings:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable Role Switch |
| 1 | Enable Hold mode |
| 2 | Enable Sniff mode |
| All other bits | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Default_Link_Policy_Settings command has completed, an HCI_Command_Complete event shall be generated.

### 7.2.12 Write Default Link Policy Settings command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Default_Link_Policy_-Settings | 0x000F | Default_Link_Policy_Settings | Status |

**Description:**

This command writes the Default Link Policy configuration value. The Default_Link_Policy_Settings parameter determines the initial value of the Link_Policy_Settings for all new BR/EDR connections.

Note: See the Link Policy Settings configuration parameter (Section 6.18) for more information.

**Command parameters:**

*Default_Link_Policy_Settings:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable Role Switch |
| 1 | Enable Hold mode |
| 2 | Enable Sniff mode |
| All other bits | Reserved for future use |

The default value is 0x0000.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Link_Policy_Settings command succeeded |
| 0x01 to 0xFF | HCI_Write_Link_Policy_Settings command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Default_Link_Policy_Settings command has completed, an HCI_Command_Complete event will be generated.

### 7.2.13 Flow Specification command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Flow_Specification | 0x0010 | Connection_Handle,<br>Unused,<br>Flow_Direction,<br>Service_Type,<br>Token_Rate,<br>Token_Bucket_Size,<br>Peak_Bandwidth,<br>Access_Latency | *none* |

**Description:**

This command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the Connection_Handle. The Connection_Handle parameter shall be a Connection_Handle for an ACL connection and is used to identify for which connection the Flow Specification is requested. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_Direction field. The HCI_Flow_Specification command allows the Link Manager to have the parameters of the outgoing as well as the incoming flow for the ACL connection. The flow parameters are defined in the L2CAP specification [Vol 3] Part A, Section 5.3, Quality of Service (QoS) option. The Link Manager will determine if the flow parameters can be supported. BR/EDR Controllers that are both Central and Peripheral can use this command.

The Unused parameter is reserved for future use.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | This value shall be used by the Host. |
| All other values | Reserved for future use. |

*Flow_Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Outgoing Flow i.e., traffic sent over the ACL connection |
| 0x01 | Incoming Flow i.e., traffic received over the ACL connection |
| All other values | Reserved for future use. |

*Service_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic |
| 0x01 | Best Effort |
| 0x02 | Guaranteed |
| All other values | Reserved for future use |

*Token_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second |

*Token_Bucket_Size:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Bucket Size in octets |

*Peak_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second |

*Access_Latency:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Latency in microseconds |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Flow_Specification command, the BR/EDR Controller shall send the HCI_Command_Status event to the Host.

When the Link Manager has determined if the Flow specification can be supported, the BR/EDR Controller on the local BR/EDR Controller shall send an HCI_Flow_Specification_Complete event to the Host. The HCI_Flow_Specification_Complete event returned by the Controller on the local side contains the status of this command, and returned Flow parameters describing the supported QoS for the ACL connection.

Note: If the Link Manager performs an LMP transaction that involves the flow parameter values on the remote side, the remote Controller can send an HCI_Flow_Specification_Complete event or HCI_QoS_Setup_Complete event to the remote Host.

### 7.2.14 Sniff Subrating command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Sniff_Subrating | 0x0011 | Connection_Handle,<br>Max_Latency,<br>Min_Remote_Timeout,<br>Min_Local_Timeout | Status,<br>Connection_Handle |

**Description:**

This command specifies the parameters for sniff subrating for a given link. The interval shall be determined from the sniff interval and the maximum subrate latency parameters from the command. The link may have smaller subrates and therefore lower latencies and longer timeouts than those specified. When the sniff subrate has been exchanged an HCI_Sniff_Subrating event shall be generated. If this command is used on a link in Sniff mode this shall cause sniff subrating to be negotiated at the Link Manager, otherwise sniff subrating shall be negotiated only after the device has entered Sniff mode.

The Connection_Handle shall be the primary Connection_Handle between the two devices.

The Maximum Latency parameter shall define the maximum allowed sniff subrate of the remote device.

If the Host does not write the sniff subrating parameters prior to sniff subrating being initiated by the Link Manager the default values shall be used.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The Maximum Latency parameter shall be used to calculate the maximum_sniff subrate that the remote device may use.<br>Default: $T_{sniff}$<br>Latency = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0002 to 0xFFFE<br>Time Range: 1.25 ms to 40.9 s |

*Min_Remote_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum sniff mode timeout ($T_{sniff_mode_timeout}$) that the remote device may use<br>Default: 0x0000<br>Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

*Min_Local_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum sniff mode timeout ($T_{sniff_mode_timeout}$) that the local device may use.<br>Default: 0x0000<br>Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_Sniff_Subrating command succeeded. |
| 0x01 to 0xFF | HCI_Sniff_Subrating command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_Sniff_Subrating command has been received by the BR/EDR Controller, an HCI_Command_Complete event shall be generated.

An HCI_Sniff_Subrating event shall occur when the sniff subrating has been negotiated for the specified Connection_Handle.

## 7.3 Controller & Baseband commands

The Controller & Baseband commands provide access and control to various capabilities of the Bluetooth hardware. These parameters provide control of BR/EDR Controllers and of the capabilities of the Link Manager and Baseband in the BR/EDR Controller and the Link Layer in an LE Controller. The Host can use these commands to modify the behavior of the local Controller.

For the HCI Control and Baseband commands, the OGF is defined as 0x03.

### 7.3.1 Set Event Mask command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Event_Mask | 0x0001 | Event_Mask | Status |

**Description:**

This command is used to control which events are generated by the HCI for the Host. If the bit in the Event_Mask is set to a one, then the event associated with that bit will be enabled. For an LE Controller, the “LE Meta event” bit in the event_Mask shall enable or disable all LE events in the LE Meta event (see Section 7.7.65). The event mask allows the Host to control how much it is interrupted.

The Controller shall ignore those bits which are reserved for future use or represent events which it does not support. If the Host sets any of these bits to 1, the Controller shall act as if they were set to 0.

**Command parameters:**

*Event_Mask:* *Size: 8 octets*

| Bit | Parameter Description |
|---|---|
| 0 | HCI_Inquiry_Complete event |
| 1 | HCI_Inquiry_Result event |
| 2 | HCI_Connection_Complete event |
| 3 | HCI_Connection_Request event |
| 4 | HCI_Disconnection_Complete event |
| 5 | HCI_Authentication_Complete event |
| 6 | HCI_Remote_Name_Request_Complete event |
| 7 | HCI_Encryption_Change event [v1] |
| 8 | HCI_Change_Connection_Link_Key_Complete event |

| Bit | Parameter Description |
|---|---|
| 9 | HCI_Link_Key_Type_Changed event |
| 10 | HCI_Read_Remote_Supported_Features_Complete event |
| 11 | HCI_Read_Remote_Version_Information_Complete event |
| 12 | HCI_QoS_Setup_Complete event |
| 15 | HCI_Hardware_Error event |
| 16 | HCI_Flush_Occurred event |
| 17 | HCI_Role_Change event |
| 19 | HCI_Mode_Change event |
| 20 | HCI_Return_Link_Keys event |
| 21 | HCI_PIN_Code_Request event |
| 22 | HCI_Link_Key_Request event |
| 23 | HCI_Link_Key_Notification event |
| 24 | HCI_Loopback_Command event |
| 25 | HCI_Data_Buffer_Overflow event |
| 26 | HCI_Max_Slots_Change event |
| 27 | HCI_Read_Clock_Offset_Complete event |
| 28 | HCI_Connection_Packet_Type_Changed event |
| 29 | HCI_QoS_Violation event |
| 30 | Previously used |
| 31 | HCI_Page_Scan_Repetition_Mode_Change event |
| 32 | HCI_Flow_Specification_Complete event |
| 33 | HCI_Inquiry_Result_with_RSSI event |
| 34 | HCI_Read_Remote_Extended_Features_Complete event |
| 43 | HCI_Synchronous_Connection_Complete event |
| 44 | HCI_Synchronous_Connection_Changed event |
| 45 | HCI_Sniff_Subrating event |
| 46 | HCI_Extended_Inquiry_Result event |
| 47 | HCI_Encryption_Key_Refresh_Complete event |
| 48 | HCI_IO_Capability_Request event |
| 49 | HCI_IO_Capability_Response event |
| 50 | HCI_User_Confirmation_Request event |
| 51 | HCI_User_Passkey_Request event |
| 52 | HCI_Remote_OOB_Data_Request event |

| Bit | Parameter Description |
|---|---|
| 53 | HCI_Simple_Pairing_Complete event |
| 55 | HCI_Link_Supervision_Timeout_Changed event |
| 56 | HCI_Enhanced_Flush_Complete event |
| 58 | HCI_User_Passkey_Notification event |
| 59 | HCI_Keypress_Notification event |
| 60 | HCI_Remote_Host_Supported_Features_Notification event |
| 61 | HCI_LE_Meta event |
| All other bits | Reserved for future use |

The value with all bits set to 0 indicates that no events are specified. The default is for bits 0 to 44 (the value 0x0000 1FFF FFFF FFFF) to be set.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Event_Mask command succeeded. |
| 0x01 to 0xFF | HCI_Set_Event_Mask command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Event_Mask command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.2 Reset command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reset | 0x0003 | *none* | Status |

**Description:**

This command will reset the Controller and the Link Manager on the BR/EDR Controller or the Link Layer on an LE Controller. If the Controller supports both BR/EDR and LE then the HCI_Reset command shall reset the Link Manager, Baseband and Link Layer. The HCI_Reset command shall not affect the used HCI transport layer since the HCI transport layers may have reset mechanisms of their own. After the reset is completed, the current operational state will be lost, the Controller will enter standby mode and the Controller will automatically revert to the default values for the parameters for which default values are defined in the specification.

Note: The HCI_Reset command will not necessarily perform a hardware reset. This is implementation defined.

The Host shall not send additional HCI commands before the HCI_Command_Complete event related to the HCI_Reset command has been received.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Reset command succeeded, was received and will be executed. |
| 0x01 to 0xFF | HCI_Reset command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the reset has been performed, an HCI_Command_Complete event shall be generated.

### 7.3.3 Set Event Filter command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Event_Filter | 0x0005 | Filter_Type,<br>Filter_Condition_Type,<br>Condition | Status |

**Description:**

This command is used by the Host to specify different event filters. The Host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters. The event filters are used by the Host to specify items of interest, which allow the BR/EDR Controller to send only events which interest the Host. Only some of the events have event filters. By default (before this command has been issued after power-on or Reset) no filters are set, and the Auto_Accept_Flag is off (incoming connections are not automatically accepted). An event filter is added each time this command is sent from the Host and the Filter_Condition_Type is not equal to 0x00. (The old event filters will not be overwritten). To clear all event filters, the Filter_Type = 0x00 is used. The Auto_Accept_Flag will then be set to off. To clear event filters for only a certain Filter_Type, the Filter_Condition_Type = 0x00 is used.

The Inquiry Result filter allows the BR/EDR Controller to filter out HCI_Inquiry_Result, HCI_Inquiry_Result_with_RSSI, or HCI_Extended_Inquiry_Result events. The Inquiry Result filter allows the Host to specify that the BR/EDR Controller only sends Inquiry Results to the Host if the report meets one of the specified conditions set by the Host. For the Inquiry Result filter, the Host can specify one or more of the following Filter Condition Types:

1. Return responses from all devices during the Inquiry process
2. A device with a specific Class of Device responded to the Inquiry process
3. A device with a specific BD_ADDR responded to the Inquiry process

The Inquiry Result filter is used in conjunction with the HCI_Inquiry and HCI_Periodic_Inquiry_Mode commands.

The Connection Setup filter allows the Host to specify that the Controller only sends an HCI_Connection_Complete, HCI_Synchronous_Connection_Complete, or HCI_Connection_Request event to the Host if the event meets one of the specified

conditions set by the Host. For the Connection Setup filter, the Host can specify one or more of the following Filter Condition Types:

1. Allow Connections from all devices
2. Allow Connections from a device with a specific Class of Device
3. Allow Connections from a device with a specific BD_ADDR

For each of these conditions, an Auto_Accept_Flag parameter allows the Host to specify what action should be done when the condition is met. The Auto_Accept_Flag allows the Host to specify if the incoming connection should be auto accepted (in which case the BR/EDR Controller will send a HCI_Connection_Complete event to the Host when an ACL or SCO connection is completed or an HCI_Synchronous_Connection_Complete event when an eSCO connection is completed) or if the Host should make the decision (in which case the BR/EDR Controller will send the HCI_Connection_Request event to the Host, to elicit a decision on the connection). If the Auto_Accept_Flag is off and the Host has masked the HCI_Connection_Request event, the Controller shall reject the connection attempt.

Note: Auto-accept does not override any requirement to reject a connection in this specification, such as the requirement in [Vol 2] Part C, Section 4.5.1 to reject a SCO connection when AES-CCM encryption is in use.

If a synchronous connection is auto-accepted, then the default parameter settings of the Accept_Synchronous_Connection_Request command (see Section 7.1.27) should be used by the local Link Manager when negotiating the SCO or eSCO link parameters.

The Connection Setup filter is used in conjunction with the HCI_Read/Write_Scan_-Enable commands. If the local device is in the process of a page scan, and is paged by another device which meets one on the conditions set by the Host, and the Auto_Accept_Flag is off for this device, then an HCI_Connection_Request event will be sent to the Host by the BR/EDR Controller. An HCI_Connection_Complete event will be sent later on after the Host has responded to the incoming connection attempt. In this same example, if the Auto_Accept_Flag is on, then an HCI_Connection_Complete event will be sent to the Host by the Controller. (No HCI_Connection_Request event will be sent in that case.)

The BR/EDR Controller will store these filters in volatile memory until the Host clears the event filters using the HCI_Set_Event_Filter command or until the HCI_Reset command is issued. The number of event filters the BR/EDR Controller can store is implementation dependent.

Note: The Clear All Filters has no Filter Condition Types or Conditions.

Note: In the condition that a connection is auto accepted, an HCI_Link_Key_Request event and possibly also an HCI_PIN_Code_Request event and an HCI_Link_-Key_Notification event could be sent to the Host by the Controller before the HCI_Connection_Complete event is sent.

If there is a contradiction between event filters, the latest set event filter will override older ones. An example is an incoming connection attempt where more than one Connection Setup filter matches the incoming connection attempt, but the Auto-Accept_Flag has different values in the different filters.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Host tries to set more filters than the BR/EDR Controller can store. | *Memory Capacity Exceeded* (0x07) |

**Command parameters:**

*Filter_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Clear All Filters<br>In this case, the Filter_Condition_Type and Condition parameters should not be given; they should have a length of zero octets. Filter_Type should be the only parameter. |
| 0x01 | Inquiry Result. |
| 0x02 | Connection Setup. |
| All other values | Reserved for future use. |

*Filter Condition Types: For each Filter Type one or more Filter Condition types exists.*

*Inquiry_Result_Filter_Condition_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Return responses from all devices during the Inquiry process.<br>Note: A device can be reported to the Host in an HCI_Inquiry_Result, HCI_Inquiry_-Result_with_RSSI, or HCI_Extended_Inquiry_Result event more than once during an inquiry or inquiry period depending on the implementation; see description in Section 7.1.1 and Section 7.1.3. |
| 0x01 | A device with a specific Class of Device responded to the Inquiry process. |

| Value | Parameter Description |
|---|---|
| 0x02 | A device with a specific BD_ADDR responded to the Inquiry process. |
| All other values | Reserved for future use |

*Connection_Setup_Filter_Condition_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Allow Connections from all devices. |
| 0x01 | Allow Connections from a device with a specific Class of Device. |
| 0x02 | Allow Connections from a device with a specific BD_ADDR. |
| All other values | Reserved for future use. |

*Condition: For each Filter Condition Type defined for the Inquiry Result Filter and the Connection Setup Filter, zero or more Condition parameters are required – depending on the filter condition type and filter type.*

*Condition for Inquiry_Result_Filter_Condition_Type = 0x00*

*Condition:* *Size: 0 octets*

| Value | Parameter Description |
|---|---|
| | The Condition parameter is not used. |

*Condition for Inquiry_Result_Filter_Condition_Type = 0x01*

*Condition:* *Size: 6 octets*

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000000 | Return All Devices (default). |
| 0xXXXXXX | *Class of Device* of Interest. |

*Class_Of_Device_Mask:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Bit Mask used to determine which bits of the Class of Device parameter are ‘don’t care’. Zero-value bits in the mask indicate the ‘don’t care’ bits of the Class of Device. |

*Condition for Inquiry_Result_Filter_Condition_Type = 0x02*

*Condition:* *Size: 6 octets*

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of Interest |

*Condition for Connection_Setup_Filter_Condition_Type = 0x00*

*Condition:* *Size: 1 octet*

*Auto_Accept_Flag:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on).<br>Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02. |
| All other values | Reserved for future use. |

*Condition for Connection_Setup_Filter_Condition_Type = 0x01*

*Condition:* *Size: 7 octets*

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000000 | Return All Devices (default). |
| 0xXXXXXX | *Class of Device* of Interest. |

*Class_Of_Device_Mask:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Bit Mask used to determine which bits of the Class of Device parameter are ‘don’t care’. Zero-value bits in the mask indicate the ‘don’t care’ bits of the Class of Device.<br>Note: For an incoming SCO connection, if the Class of Device is unknown then the connection will be accepted. |

*Auto_Accept_Flag:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on).<br>Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02. |
| All other values | Reserved for future use. |

*Condition for Connection_Setup_Filter_Condition_Type = 0x02*

*Condition:* *Size: 7 octets*

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of Interest. |

*Auto_Accept_Flag:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on).<br>Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Event_Filter command succeeded. |
| 0x01 to 0xFF | HCI_Set_Event_Filter command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

An HCI_Command_Complete event for this command shall occur when the Controller has enabled the filtering of events. When one of the conditions are met, a specific event shall occur.

### 7.3.4 Flush command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Flush | 0x0008 | Connection_Handle | Status,<br>Connection_Handle |

**Description:**

This command is used to discard all data that is currently pending for transmission in the Controller for the specified Connection_Handle, even if there currently are chunks of data that belong to more than one L2CAP packet in the Controller. Both automatically-flushable and non-automatically-flushable packets shall be discarded (see Section 5.4.2). After this, all data that is sent to the Controller for the same Connection_Handle will be discarded by the Controller until an HCI ACL Data packet with one of the start Packet_Boundary_Flag values (0x00 or 0x02) is received. When this happens, a new transmission attempt can be made.

This command, when used on a BR/EDR Controller, will allow higher-level software to control how long the Baseband should try to retransmit a Baseband packet for a Connection_Handle before all data that is currently pending for transmission in the Controller should be flushed.

Note: The HCI_Flush command is used for ACL connections only. In addition to the HCI_Flush command, the automatic flush timers (see Section 7.3.29) can be used to automatically flush an automatically-flushable L2CAP packet that is currently being transmitted after the specified flush timer has expired.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Flush command succeeded. |
| 0x01 to 0xFF | HCI_Flush command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

The HCI_Flush_Occurred event shall occur once the flush is completed. An HCI_Flush_Occurred event could be from an automatic Flush or could be caused by the Host issuing the HCI_Flush command. When the HCI_Flush command has completed, an HCI_Command_Complete event shall be generated, to indicate that the Host caused the Flush.

### 7.3.5 Read PIN Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_PIN_Type | 0x0009 | *none* | Status,<br>PIN_Type |

**Description:**

This command is used to read the PIN_Type configuration parameter. See Section 6.13.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_PIN_Type command succeeded. |
| 0x01 to 0xFF | HCI_Read_PIN_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*PIN_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

**Event(s) generated (unless masked away):**

When the HCI_Read_PIN_Type command has completed, an HCI_Command_Complete event will be generated.

### 7.3.6 Write PIN Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_PIN_Type | 0x000A | PIN_Type | Status |

**Description:**

This command is used to write the PIN Type configuration parameter. See Section 6.13.

**Command parameters:**

*PIN_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_PIN_Type command succeeded. |
| 0x01 to 0xFF | HCI_Write_PIN_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_PIN_Type command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.7 [This section is no longer used]

### 7.3.8 Read Stored Link Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Stored_Link_Key | 0x000D | BD_ADDR,<br>Read_All | Status,<br>Max_Num_Keys,<br>Num_Keys_Read |

**Description:**

This command provides the ability to read whether one or more link keys are stored in the BR/EDR Controller. The BR/EDR Controller can store a limited number of link keys for other BR/EDR Controllers. Link keys are shared between two BR/EDR Controllers, and are used for all security transactions between the two devices. The HCI_Read_Stored_Link_Key command shall not return the link key's value. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the BR/EDR Controller when needed. The Read_All parameter is used to indicate if all of the stored Link Keys should be returned. If Read_All indicates that all Link Keys are to be returned, then the BD_ADDR parameter shall be ignored. Otherwise, BD_ADDR is used to identify which link key to read. The stored Link Keys are returned by one or more HCI_Return_Link_Keys events. See Section 6.14.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the stored link key to be read. |

*Read_All:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Return Link Key for specified BD_ADDR. |
| 0x01 | Return all stored Link Keys. |
| All other values | Reserved for future use. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Stored_Link_Key command succeeded. |
| 0x01 to 0xFF | HCI_Read_Stored_Link_Key command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Max_Num_Keys:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total Number of Link Keys that the Controller can store.<br>Range: 0x0000 to 0xFFFF |

*Num_Keys_Read:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of Link Keys Read.<br>Range: 0x0000 to 0xFFFF |

### Event(s) generated (unless masked away):

Zero or more instances of the HCI_Return_Link_Keys event shall occur after the command is issued. When there are no link keys stored, no HCI_Return_Link_Keys events will be returned. When there are link keys stored, the number of link keys returned in each HCI_Return_Link_Keys event is implementation specific. When the HCI_Read_Stored_Link_Key command has completed an HCI_Command_Complete event shall be generated.

### 7.3.9 Write Stored Link Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Stored_Link_Key | 0x0011 | Num_Keys_To_Write,<br>BD_ADDR[i],<br>Link_Key[i] | Status,<br>Num_Keys_Written |

**Description:**

This command provides the ability to write one or more link keys to be stored in the BR/EDR Controller. The BR/EDR Controller can store a limited number of link keys for other BR/EDR Controllers. If no additional space is available in the BR/EDR Controller then no additional link keys will be stored. If space is limited and if all the link keys to be stored will not fit in the limited space, then the order of the list of link keys without any error will determine which link keys are stored. Link keys at the beginning of the list will be stored first. The Num_Keys_Written parameter will return the number of link keys that were successfully stored. If no additional space is available, then the Host must delete one or more stored link keys before any additional link keys are stored. The link key replacement algorithm is implemented by the Host and not the BR/EDR Controller. Link keys are shared between two BR/EDR Controllers and are used for all security transactions between the two devices. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the BR/EDR Controller when needed. See Section 6.14.

Note: Link Keys are only stored by issuing this command.

A Host in Secure Connections Only Mode should not store link keys in the Controller.

**Command parameters:**

*Num_Keys_To_Write:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys to Write.<br>Range: 0x01 to 0x0B |

*BD_ADDR[i]:* *Size: Num_Keys_To_Write × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the associated Link Key. |

*Link_Key[i]:* *Size: Num_Keys_To_Write × 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Link Key for an associated BD_ADDR. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Stored_Link_Key command succeeded. |
| 0x01 to 0xFF | HCI_Write_Stored_Link_Key command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Keys_Written:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys successfully written.<br>Range: 0x00 to 0x0B |

**Event(s) generated (unless masked away):**

When the HCI_Write_Stored_Link_Key command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.10 Delete Stored Link Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Delete_Stored_Link_Key | 0x0012 | BD_ADDR,<br>Delete_All | Status,<br>Num_Keys_Deleted |

**Description:**

This command provides the ability to remove one or more of the link keys stored in the BR/EDR Controller. The BR/EDR Controller can store a limited number of link keys for other BR/EDR devices.

The Delete_All parameter indicates whether one or all of the stored Link Keys are to be deleted. If Delete_All indicates that all Link Keys are to be deleted, then the BD_ADDR parameter shall be ignored. Otherwise, BD_ADDR identifies which link key to delete.

If a link key to be deleted is currently in use for a connection, then the link key shall be deleted immediately from the store but only completely deleted when all of the connections using it are disconnected. Otherwise the link key shall be deleted immediately. In either case, the command is completed when the key has (or all the keys have) been deleted from the store and, therefore, the key(s) cannot subsequently be read using the HCI_Read_Stored_Link_Key command. See Section 6.14.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the link key to be deleted. |

*Delete_All:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Delete only the Link Key for specified BD_ADDR. |
| 0x01 | Delete all stored Link Keys. |
| All other values | Reserved for future use. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Delete_Stored_Link_Key command succeeded. |
| 0x01 to 0xFF | HCI_Delete_Stored_Link_Key command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Num_Keys_Deleted:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of Link Keys deleted from the store |

### Event(s) generated (unless masked away):

When the HCI_Delete_Stored_Link_Key command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.11 Write Local Name command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Local_Name | 0x0013 | Local_Name | Status |

**Description:**

This command provides the ability to modify the user-friendly name for the BR/EDR Controller. See Section 6.23.

**Command parameters:**

*Local_Name:* *Size: 248 octets*

| Value | Parameter Description |
|---|---|
| | A UTF-8 encoded User-Friendly Descriptive Name for the device with type utf8s{248} (see [Vol 1] Part E, Section 2.9.3). |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Local_Name command succeeded. |
| 0x01 to 0xFF | HCI_Write_Local_Name command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Local_Name command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.12 Read Local Name command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Name | 0x0014 | *none* | Status,<br>Local_Name |

**Description:**

This command provides the ability to read the stored user-friendly name for the BR/EDR Controller. See Section 6.23.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Name command succeeded |
| 0x01 to 0xFF | HCI_Read_Local_Name command failed (see [Vol 1] Part F, Controller Error Codes for list of error codes). |

*Local_Name:* *Size: 248 octets*

| Parameter Description |
|---|
| A UTF-8 encoded User Friendly Descriptive Name for the device with type utf8s{248} (see [Vol 1] Part E, Section 2.9.3) |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Name command has completed an HCI_Command_Complete event shall be generated.

### 7.3.13 Read Connection Accept Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Connection_Accept_-Timeout | 0x0015 | *none* | Status,<br>Connection_Accept_Timeout |

**Description:**

This command reads the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Connection_Accept_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Read_Connection_Accept_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Accept_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection Accept Timeout measured in number of BR/EDR Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xB540<br>Time Range: 0.625 ms to 29 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Connection_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.14 Write Connection Accept Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Connection_Accept_-Timeout | 0x0016 | Connection_Accept_Timeout | Status |

**Description:**

This command writes the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7.

**Command parameters:**

*Connection_Accept_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection Accept Timeout measured in number of BR/EDR Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xB540<br>Time Range: 0.625 ms to 29 s<br>Default: 0x1FA0 (5.06 s)<br>Mandatory Range for Controller: 0x00A0 to 0xB540 |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Connection_Accept_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Write_Connection_Accept_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Connection_Accept_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.15 Read Page Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Timeout | 0x0017 | *none* | Status,<br>Page_Timeout |

**Description:**

This command reads the value for the Page_Timeout configuration parameter. See Section 6.6.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Page_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Read_Page_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Page Timeout measured in number of Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xFFFF<br>Time Range: 0.625 ms to 40.9 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Page_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.16 Write Page Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Timeout | 0x0018 | Page_Timeout | Status |

**Description:**

This command writes the value for the Page_Timeout configuration parameter. See Section 6.6.

**Command parameters:**

*Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Page Timeout measured in number of Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xFFFF<br>Time Range: 0.625 ms to 40.9 s<br>Default: 0x2000 (5.12 s)<br>Mandatory Range for Controller: 0x0016 to 0xFFFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Page_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Write_Page_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Page_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.17 Read Scan Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Scan_Enable | 0x0019 | *none* | Status,<br>Scan_Enable |

**Description:**

This command reads the value for the Scan_Enable parameter configuration parameter. See Section 6.1.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Scan_Enable command succeeded. |
| 0x01 to 0xFF | HCI_Read_Scan_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Scan_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled. |
| 0x01 | Inquiry Scan enabled.<br>Page Scan disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Scan_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.18 Write Scan Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Scan_Enable | 0x001A | Scan_Enable | Status |

**Description:**

This command writes the value for the Scan_Enable configuration parameter. See Section 6.1.

**Command parameters:**

*Scan_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled (default) |
| 0x01 | Inquiry Scan enabled.<br>Page Scan disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Scan_Enable command succeeded. |
| 0x01 to 0xFF | HCI_Write_Scan_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Scan_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.19 Read Page Scan Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Scan_Activity | 0x001B | *none* | Status,<br>Page_Scan_Interval,<br>Page_Scan_Window |

**Description:**

This command reads the value for Page_Scan_Interval and Page_Scan_Window configuration parameters. See Section 6.8 and Section 6.9.

Note: Page Scan is only performed when Page_Scan is enabled (see Section 6.1, Section 7.3.17, and Section 7.3.18). A changed Page_Scan_Interval could change the local Page Scan Repetition Mode (see [Vol 2] Part B, Section 8.3.1).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Page_Scan_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Read_Page_Scan_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Page_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0012 to 0x1000<br>Time = N × 0.625 ms<br>Range: 11.25 ms to 2560 ms; only even values are valid |

*Page_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0011 to 0x1000<br>Time = N × 0.625 ms<br>Range: 10.625 ms to 2560 ms |

**Event(s) generated (unless masked away):**

When the HCI_Read_Page_Scan_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.20 Write Page Scan Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Scan_Activity | 0x001C | Page_Scan_Interval,<br>Page_Scan_Window | Status |

**Description:**

This command writes the values for the Page_Scan_Interval and Page_Scan_Window configuration parameters. The Page_Scan_Window shall be less than or equal to the Page_Scan_Interval. See Section 6.8 and Section 6.9.

Note: Page Scan is only performed when Page_Scan is enabled (see Section 6.1, Section 7.3.17, and Section 7.3.18). A changed Page_Scan_Interval could change the local Page Scan Repetition Mode (see [Vol 2] Part B, Section 8.3.1).

**Command parameters:**

*Page_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| See Section 6.8 | |

*Page_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| See Section 6.9 | |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Page_Scan_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Write_Page_Scan_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Page_Scan_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.21 Read Inquiry Scan Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Scan_Activity | 0x001D | *none* | Status,<br>Inquiry_Scan_Interval,<br>Inquiry_Scan_Window |

**Description:**

This command reads the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameter. See Section 6.2 and Section 6.3.

Note: Inquiry Scan is only performed when Inquiry_Scan is enabled see Section 6.1, Section 7.3.17, and Section 7.3.18).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Inquiry_Scan_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Read_Inquiry_Scan_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Inquiry_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0012 to 0x1000<br>Time = N × 0.625 ms<br>Range: 11.25 to 2560 ms; only even values are valid |

*Inquiry_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0011 to 0x1000<br>Time = N × 0.625 ms<br>Range: 10.625 ms to 2560 ms |

**Event(s) generated (unless masked away):**

When the HCI_Read_Inquiry_Scan_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.22 Write Inquiry Scan Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Scan_Activity | 0x001E | Inquiry_Scan_Interval,<br>Inquiry_Scan_Window | Status |

**Description:**

This command writes the values for the Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. The Inquiry_Scan_Window shall be less than or equal to the Inquiry_Scan_Interval. See Section 6.2 and Section 6.3.

Note: Inquiry Scan is only performed when Inquiry_Scan is enabled (see Section 6.1, Section 7.3.17, and Section 7.3.18).

**Command parameters:**

*Inquiry_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| See Section 6.2. | |

*Inquiry_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| See Section 6.3. | |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Inquiry_Scan_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Write_Inquiry_Scan_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Inquiry_Scan_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.23 Read Authentication Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Authentication_Enable | 0x001F | *none* | Status,<br>Authentication_Enable |

**Description:**

This command reads the value for the Authentication_Enable configuration parameter. See Section 6.16.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Authentication_Enable command succeeded. |
| 0x01 to 0xFF | HCI_Read_Authentication_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Authentication_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required. |
| 0x01 | Authentication required for all connections. |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Authentication_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.24 Write Authentication Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Authentication_Enable | 0x0020 | Authentication_Enable | Status |

**Description:**

This command writes the value for the Authentication_Enable configuration parameter. See Section 6.16.

The Authentication_Enable configuration parameter shall only apply to connections (e.g. send an LMP_IN_RAND or LMP_AU_RAND) when the remote device's Host or BR/EDR Controller does not support Secure Simple Pairing or when the local Host does not support Secure Simple Pairing.

**Command parameters:**

*Authentication_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required (default) |
| 0x01 | Authentication required for all connections. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Authentication_Enable command succeeded. |
| 0x01 to 0xFF | HCI_Write_Authentication_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Authentication_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.25 Read Class of Device command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Class_Of_Device | 0x0023 | *none* | Status,<br>Class_Of_Device |

**Description:**

This command reads the value for the Class_Of_Device parameter. See Section 6.26.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Class_Of_Device command succeeded. |
| 0x01 to 0xFF | HCI_Read_Class_Of_Device command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Class_Of_Device command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.26 Write Class of Device command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Class_Of_Device | 0x0024 | Class_Of_Device | Status |

**Description:**

This command writes the value for the Class_Of_Device parameter.

See Section 6.26.

**Command parameters:**

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Class_Of_Device command succeeded. |
| 0x01 to 0xFF | HCI_Write_Class_Of_Device command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Class_Of_Device command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.27 Read Voice Setting command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Voice_Setting | 0x0025 | *none* | Status,<br>Voice_Setting |

**Description:**

This command reads the values for the Voice_Setting configuration parameter. See Section 6.12.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Voice_Setting command succeeded. |
| 0x01 to 0xFF | HCI_Read_Voice_Setting command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Voice_Setting:* *Size: 2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12. | |

**Event(s) generated (unless masked away):**

When the HCI_Read_Voice_Setting command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.28 Write Voice Setting command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Voice_Setting | 0x0026 | Voice_Setting | Status |

**Description:**

This command writes the values for the Voice_Setting configuration parameter. See Section 6.12.

**Command parameters:**

*Voice_Setting:* *Size: 2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12. | |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Voice_Setting command succeeded. |
| 0x01 to 0xFF | HCI_Write_Voice_Setting command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Voice_Setting command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.29 Read Automatic Flush Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Automatic_Flush_Timeout | 0x0027 | Connection_Handle | Status,<br>Connection_Handle,<br>Flush_Timeout |

**Description:**

This command reads the value for the Flush_Timeout parameter for the specified Connection_Handle. See Section 6.19.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Automatic_Flush_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Read_Automatic_Flush_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Flush_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Timeout = ∞; No Automatic Flush |
| N = 0xXXXX | Flush Timeout = N × 0.625 ms<br>Range: 0x0001 to 0x07FF |

**Event(s) generated (unless masked away):**

When the HCI_Read_Automatic_Flush_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.30 Write Automatic Flush Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Automatic_Flush_Timeout | 0x0028 | Connection_Handle,<br>Flush_Timeout | Status,<br>Connection_Handle |

**Description:**

This command writes the value for the Flush_Timeout parameter for the specified Connection_Handle. See Section 6.19.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Flush_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Timeout = ∞; No Automatic Flush (default) |
| N = 0xXXXX | Flush Timeout = N × 0.625 ms<br>Range: 0x0001 to 0x07FF<br>Mandatory Range for Controller: 0x0002 to 0x07FF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Automatic_Flush_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Write_Automatic_Flush_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_Write_Automatic_Flush_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.31 Read Num Broadcast Retransmissions command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Num_Broadcast_-Retransmissions | 0x0029 | *none* | Status,<br>Num_Broadcast_Retransmissions |

**Description:**

This command reads the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.20

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Num_Broadcast_Retransmissions command succeeded. |
| 0x01 to 0xFF | HCI_Read_Num_Broadcast_Retransmissions command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Broadcast_Retransmissions:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| See Section 6.20. | |

**Event(s) generated (unless masked away):**

When the HCI_Read_Num_Broadcast_Retransmission command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.32 Write Num Broadcast Retransmissions command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Num_Broadcast_-Retransmissions | 0x002A | Num_Broadcast_Retransmissions | Status |

**Description:**

This command writes the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.20.

**Command parameters:**

*Num_Broadcast_Retransmissions:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| See Section 6.20. | |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Num_Broadcast_Retransmissions command succeeded. |
| 0x01 to 0xFF | HCI_Write_Num_Broadcast_Retransmissions command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Num_Broadcast_Retransmissions command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.33 Read Hold Mode Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Hold_Mode_Activity | 0x002B | *none* | Status,<br>Hold_Mode_Activity |

**Description:**

This command reads the value for the Hold_Mode_Activity parameter.

See Section 6.16.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Hold_Mode_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Read_Hold_Mode_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Hold_Mode_Activity:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Maintain current Power State. |
| 0x01 | Suspend Page Scan. |
| 0x02 | Suspend Inquiry Scan. |
| 0x04 | Suspend Periodic Inquiries. |
| All other values | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Hold_Mode_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.34 Write Hold Mode Activity command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Hold_Mode_Activity | 0x002C | Hold_Mode_Activity | Status |

**Description:**

This command writes the value for the Hold_Mode_Activity parameter.

See Section 6.16.

**Command parameters:**

*Hold_Mode_Activity:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Maintain current Power State (default) |
| 0x01 | Suspend Page Scan. |
| 0x02 | Suspend Inquiry Scan. |
| 0x04 | Suspend Periodic Inquiries. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Hold_Mode_Activity command succeeded. |
| 0x01 to 0xFF | HCI_Write_Hold_Mode_Activity command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Hold_Mode_Activity command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.35 Read Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Transmit_Power_Level | 0x002D | Connection_Handle,<br>Type | Status,<br>Connection_Handle,<br>TX_Power_Level |

**Description:**

This command reads the values for the TX_Power_Level parameter for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read Current Transmit Power Level. |
| 0x01 | Read Maximum Transmit Power Level. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Transmit_Power_Level command succeeded. |
| 0x01 to 0xFF | HCI_Read_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -30 to 20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the HCI_Read_Transmit_Power_Level command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.36 Read Synchronous Flow Control Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Synchronous_-Flow_Control_Enable | 0x002E | *none* | Status,<br>Synchronous_Flow_Control_Enable |

**Description:**

This command provides the ability to read the Synchronous_Flow_Control_Enable parameter. See Section 6.22.

The Synchronous_Flow_Control_Enable parameter shall only be changed if no connection exists.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Synchronous_Flow_Control_Enable command succeeded |
| 0x01 to 0xFF | HCI_Read_Synchronous_Flow_Control_Enable command failed (see [Vol 1] Part F, Controller Error Codes for list of error codes). |

*Synchronous_Flow_Control_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronous Flow Control is disabled. No HCI_Number_Of_Completed_Packets events will be sent from the Controller for synchronous Connection_Handles. |
| 0x01 | Synchronous Flow Control is enabled. HCI_Number_Of_Completed_Packets events will be sent from the Controller for synchronous Connection_Handles. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Synchronous_Flow_Control_Enable command has completed an HCI_Command_Complete event shall be generated.

### 7.3.37 Write Synchronous Flow Control Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Synchronous_-Flow_Control_Enable | 0x002F | Synchronous_Flow_-Control_Enable | Status |

**Description:**

This command provides the ability to write the Synchronous_Flow_Control_Enable parameter. See Section 6.22.

The Synchronous_Flow_Control_Enable parameter can only be changed if no connections exist.

**Command parameters:**

*Synchronous_Flow_Control_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronous Flow Control is disabled. No HCI_Number_Of_Completed_Packets events shall be sent from the Controller for synchronous Connection_Handles.<br>**Default** |
| 0x01 | Synchronous Flow Control is enabled. HCI_Number_Of_Completed_Packets events shall be sent from the Controller for synchronous Connection_Handles. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Synchronous_Flow_Control_Enable command succeeded |
| 0x01 to 0xFF | HCI_Write_Synchronous_Flow_Control_Enable command failed (see [Vol 1] Part F, Controller Error Codes for list of error codes.) |

**Event(s) generated (unless masked away):**

When the HCI_Write_Synchronous_Flow_Control_Enable command has completed an HCI_Command_Complete event shall be generated.

### 7.3.38 Set Controller To Host Flow Control command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Controller_To_Host_Flow_-Control | 0x0031 | Flow_Control_Enable | Status |

**Description:**

This command is used by the Host to turn flow control on or off for data and/or voice sent in the direction from the Controller to the Host. If flow control is turned off, the Host should not send the HCI_Host_Number_Of_Completed_Packets command. That command will be ignored by the Controller if it is sent by the Host and flow control is off. If flow control is turned on for HCI ACL Data packets and off for HCI Synchronous Data packets, HCI_Host_Number_Of_Completed_Packets commands sent by the Host should only contain Connection_Handles for ACL connections. If flow control is turned off for HCI ACL Data packets and on for HCI Synchronous Data packets, HCI_Host_Number_Of_Completed_Packets commands sent by the Host should only contain Connection_Handles for synchronous connections. If flow control is turned on for HCI ACL Data packets and HCI Synchronous Data packets, the Host will send HCI_Host_Number_Of_Completed_Packets commands both for ACL connections and synchronous connections.

The Flow_Control_Enable parameter shall only be changed if no connections exist.

**Command parameters:**

*Flow_Control_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Flow control **off** in direction from Controller to Host (default) |
| 0x01 | Flow control **on** for HCI ACL Data packets and **off** for HCI Synchronous Data packets in direction from Controller to Host. |
| 0x02 | Flow control **off** for HCI ACL Data packets and **on** for HCI Synchronous Data packets in direction from Controller to Host. |
| 0x03 | Flow control **on** both for HCI ACL Data packets and HCI Synchronous Data packets in direction from Controller to Host. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Controller_To_Host_Flow_Control command succeeded. |
| 0x01 to 0xFF | HCI_Set_Controller_To_Host_Flow_Control command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Controller_To_Host_Flow_Control command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.39 Host Buffer Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Host_Buffer_-Size | 0x0033 | Host_ACL_Data_Packet_Length,<br>Host_Synchronous_Data_Packet_Length,<br>Host_Total_Num_ACL_Data_Packets,<br>Host_Total_Num_Synchronous_Data_Packets | Status |

**Description:**

This command is used by the Host to notify the Controller about the maximum size of the data portion of HCI ACL and Synchronous Data packets sent from the Controller to the Host. The Controller shall fragment the data to be transmitted from the Controller to the Host according to these sizes, so that the HCI Data packets will contain data with up to these sizes. The HCI_Host_Buffer_Size command also notifies the Controller about the total number of HCI ACL and Synchronous Data packets that can be stored in the data buffers of the Host. If flow control from the Controller to the Host is turned off, and the HCI_Host_Buffer_Size command has not been issued by the Host, then this means that the Controller can send HCI Data packets to the Host with any length and it is assumed that the data buffer sizes of the Host are unlimited. If flow control from the Controller to the Host is turned on, the HCI_Host_Buffer_Size command shall after a power-on or a reset always be sent by the Host before the first HCI_Host_Number_Of_-Completed_Packets command is sent.

The HCI_Set_Controller_To_Host_Flow_Control command is used to turn flow control on or off. The Host_ACL_Data_Packet_Length parameter will be used to determine the size of the L2CAP fragments contained in ACL Data packets, which are transferred from the Controller to the Host. The Host_Synchronous_Data_Packet_Length parameter is used to determine the maximum size of HCI Synchronous Data packets. Both the Host and the Controller shall support command and event packets, where the data portion (omitting the header) contained in the packets is 255 octets in size.

The Host_Total_Num_ACL_Data_Packets parameter contains the total number of HCI ACL Data packets that can be stored in the data buffers of the Host. The Controller will determine how the buffers are to be divided between different Connection_Handles. The Host_Total_Num_Synchronous_Data_Packets parameter gives the same information for HCI Synchronous Data packets. If the Host does not support SCO or eSCO over HCI, then it shall set Host_Total_Num_Synchronous_Data_Packets to zero, in which case the Controller shall ignore the Host_Synchronous_Data_Packet_Length parameter.

Note: The Host_ACL_Data_Packet_Length and Host_Synchronous_Data_Packet_Length command parameters do not include the length of the HCI ACL Data packet header or the HCI Synchronous Data packet header respectively.

**Command parameters:**

*Host_ACL_Data_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum length (in octets) of the data portion of each HCI ACL Data packet that the Host is able to accept.<br>Range: 0x0001 to 0xFFFF |

*Host_Synchronous_Data_Packet_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Maximum length (in octets) of the data portion of each HCI Synchronous Data packet that the Host is able to accept.<br>Range: 0x01 to 0xFF |

*Host_Total_Num_ACL_Data_Packets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total number of HCI ACL Data packets that can be stored in the data buffers of the Host.<br>Range: 0x0001 to 0xFFFF |

*Host_Total_Num_Synchronous_Data_Packets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | The Host does not support SCO or eSCO over HCI. |
| 0xXXXX | Total number of HCI Synchronous Data packets that can be stored in the data buffers of the Host. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Host_Buffer_Size command succeeded. |
| 0x01 to 0xFF | HCI_Host_Buffer_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Host_Buffer_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.40 Host Number Of Completed Packets command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Host_Number_Of_-Completed_Packets | 0x0035 | Num_Handles,<br>Connection_Handle[i],<br>Host_Num_Completed_Packets[i] | *none* |

**Description:**

This command is used by the Host to indicate to the Controller the number of HCI Data packets that have been completed for each Connection_Handle since the previous HCI_Host_Number_Of_Completed_Packets command was sent to the Controller. This means that the corresponding buffer space has been freed in the Host and is available for new packets to be sent. Based on this information, and the Host_Total_Num_ACL_Data_Packets and Host_Total_Num_Synchronous_Data_Packets command parameters of the HCI_Host_Buffer_Size command, the Controller can determine for which Connection_Handles the following HCI Data packets should be sent to the Host. When the Host has completed one or more HCI Data packet(s) it shall send an HCI_Host_Number_Of_Completed_Packets command to the Controller, until it finally reports that all pending HCI Data packets have been completed. The frequency at which this command is sent is manufacturer specific.

The HCI_Set_Controller_To_Host_Flow_Control command is used to turn flow control on or off. If flow control from the Controller to the Host is turned on, the HCI_Host_Buffer_Size command shall always be sent by the Host after a power-on or a reset before the first HCI_Host_Number_Of_Completed_Packets command is sent.

The HCI_Host_Number_Of_Completed_Packets command may be sent at any time by the Host when there is at least one connection, or if the Controller is in local loopback mode, independent of other commands. If the Host issues this command when neither of these cases applies, the Controller shall ignore it.

**Command parameters:**

*Num_Handles:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Connection_Handles and Host_Num_Completed_Packets parameters pairs contained in this command.<br>Range: 0 to 255 |

*Connection_Handle[i]:* *Size: Num_Handles × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Host_Num_Completed_Packets[i]:* *Size: Num_Handles × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of HCI Data packets that have been completed for the associated Connection_Handle since the previous time the event was returned.<br>Range: 0x0000 to 0xFFFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

Normally, no event is generated after the HCI_Host_Number_Of_Completed_Packets command has completed. However, if the HCI_Host_Number_Of_Completed_Packets command contains one or more invalid parameters, the Controller shall return an HCI_Command_Complete event containing the error code *Invalid HCI Command Parameters* (0x12). The normal flow control for commands is not used for this command.

### 7.3.41 Read Link Supervision Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Supervision_-Timeout | 0x0036 | Handle | Status,<br>Handle,<br>Link_Supervision_Timeout |

**Description:**

This command reads the value for the Link_Supervision_Timeout parameter for the Controller.

The Handle used for this command shall be the ACL connection to the appropriate device. See Section 6.21.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection_Handle's Link Supervision Timeout value is to be read.<br>The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Link_Supervision_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Read_Link_Supervision_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection_Handle's Link Supervision Timeout value was read.<br>The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

*Link_Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No Link_Supervision_Timeout. |
| N = 0xXXXX | Measured in number of BR/EDR Baseband slots<br>Link_Supervision_Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xFFFF<br>Time Range: 0.625 ms to 40.9 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Link_Supervision_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.42 Write Link Supervision Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Link_Supervision_Timeout | 0x0037 | Handle,<br>Link_Supervision_Timeout | Status,<br>Handle |

**Description:**

This command writes the value for the Link_Supervision_Timeout parameter for a BR/EDR Controller.

The Handle used for this command shall be the ACL connection to the appropriate device. This command will set the Link_Supervision_Timeout values for other Synchronous Handles to that device. See Section 6.21.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller is the Peripheral on the ACL connection. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Handle's Link Supervision Timeout value is to be written.<br>The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

*Link_Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No Link_Supervision_Timeout. |
| N = 0xXXXX | Measured in number of BR/EDR Baseband slots<br>Link_Supervision_Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xFFFF<br>Time Range: 0.625 ms to 40.9 s<br>Default: 0x7D00 (20 s)<br>Mandatory Range for Controller: 0x0190 to 0xFFFF; plus 0 for infinite timeout |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Link_Supervision_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Write_Link_Supervision_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Handle's Link Supervision Timeout value was written.<br>The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_Write_Link_Supervision_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.43 Read Number Of Supported IAC command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Number_Of_Supported_IAC | 0x0038 | *none* | Status,<br>Num_Supported_IAC |

**Description:**

This command reads the value for the number of Inquiry Access Codes (IAC) that the local BR/EDR Controller can simultaneous listen for during an Inquiry Scan. All BR/EDR Controllers are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some BR/EDR Controllers support additional IACs.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Number_Of_Supported_IAC command succeeded. |
| 0x01 to 0xFF | HCI_Read_Number_Of_Supported_IAC command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Supported_IAC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of Supported IAC that the local BR/EDR Controller can simultaneous listen for during an Inquiry Scan.<br>Range: 0x01 to 0x40 |

**Event(s) generated (unless masked away):**

When the HCI_Read_Number_Of_Supported_IAC command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.44 Read Current IAC LAP command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Current_IAC_LAP | 0x0039 | *none* | Status,<br>Num_Current_IAC,<br>IAC_LAP[i] |

**Description:**

This command reads the LAP(s) used to create the Inquiry Access Codes (IAC) that the local BR/EDR Controller is simultaneously scanning for during Inquiry Scans. All BR/EDR Controllers shall support at least one IAC, the General Inquiry Access Code (the GIAC). Some BR/EDR Controllers support additional IACs.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Current_IAC_LAP command succeeded. |
| 0x01 to 0xFF | HCI_Read_Current_IAC_LAP command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Current_IAC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of IACs which are currently in use by the local BR/EDR Controller to simultaneously listen for during an Inquiry Scan.<br>Range: 0x01 to 0x40 |

*IAC_LAP[i]:* *Size: Num_Current_IAC × 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | LAP used to create the IAC which is currently in use by the local BR/EDR Controller to simultaneously listen for during an Inquiry Scan.<br>Range: 0x9E8B00 to 0x9E8B3F |

**Event(s) generated (unless masked away):**

When the HCI_Read_Current_IAC_LAP command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.45 Write Current IAC LAP command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Current_IAC_LAP | 0x003A | Num_Current_IAC,<br>IAC_LAP[i] | Status |

**Description:**

This command writes the LAP(s) used to create the Inquiry Access Codes (IAC) that the local BR/EDR Controller is simultaneously scanning for during Inquiry Scans. All BR/EDR Controller shall support at least one IAC, the General Inquiry Access Code (the GIAC). Some BR/EDR Controllers support additional IACs.

This command shall clear any existing IACs and stores Num_Current_IAC and the IAC_LAPs in to the Controller. If Num_Current_IAC is greater than Num_Supported_IAC then only the first Num_Supported_IAC shall be stored in the Controller, and an HCI_Command_Complete event with error code *Success* (0x00) shall be generated.

**Command parameters:**

*Num_Current_IAC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of IACs that will be used by the local BR/EDR Controller to simultaneously listen for during an Inquiry Scan.<br>Range: 0x01 to 0x40 |

*IAC_LAP[i]:* *Size: Num_Current_IAC × 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | LAP that will be used to create the IACs that will be used by the local BR/EDR Controller to simultaneously listen for during an Inquiry Scan.<br>Range: 0x9E8B00 to 0x9E8B3F.<br>The default IAC(s) to be used shall be the GIAC and zero or more other IACs specified by the manufacturer. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Current_IAC_LAP command succeeded. |
| 0x01 to 0xFF | HCI_Write_Current_IAC_LAP command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Current_IAC_LAP command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.46 Set AFH Host Channel Classification command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_AFH_Host_Channel_-Classification | 0x003F | AFH_Host_Channel_-Classification | Status |

**Description:**

This command allows the Host to specify a channel classification based on its "local information". This classification persists until overwritten with a subsequent HCI_Set_AFH_Host_Channel_Classification command or until the BR/EDR Controller is reset.

If this command is used, updates should be sent within 10 seconds of the Host knowing that the channel classification has changed. The interval between two successive commands sent shall be at least 1 second.

**Command parameters:**

*AFH_Host_Channel_Classification:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXX | This parameter contains 80 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>0: channel *n* is bad<br>1: channel *n* is unknown<br>The most significant bit (bit 79) is reserved for future use<br>At least $N_{min}$ channels shall be marked as unknown. (See [Vol 2] Part B, Section 2.3.1). If the device supports Synchronizable mode, then the synchronization train channels (see [Vol 2] Part B, Section 2.6.4.8) shall be ignored when checking this requirement. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_AFH_Host_Channel_Classification command succeeded. |
| 0x01 to 0xFF | HCI_Set_AFH_Host_Channel_Classification command failed. [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_AFH_Host_Channel_Classification command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.47 Read Inquiry Scan Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Scan_Type | 0x0042 | *none* | Status,<br>Inquiry_Scan_Type |

**Description:**

This command reads the Inquiry_Scan_Type configuration parameter from the local BR/EDR Controller. See Section 6.4.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Inquiry_Scan_Type command succeeded |
| 0x01 to 0xFF | HCI_Read_Inquiry_Scan_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Inquiry_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Scan (default) |
| 0x01 | Interlaced Scan |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Inquiry_Scan_Type command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.48 Write Inquiry Scan Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Scan_Type | 0x0043 | Scan_Type | Status |

**Description:**

This command writes the Inquiry Scan Type configuration parameter of the local BR/EDR Controller. See Section 6.4.

**Command parameters:**

*Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Scan (default) |
| 0x01 | Interlaced Scan |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Inquiry_Scan_Type command succeeded |
| 0x01 to 0xFF | HCI_Write_Inquiry_Scan_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Inquiry_Scan_Type command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.49 Read Inquiry Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Mode | 0x0044 | *none* | Status,<br>Inquiry_Mode |

**Description:**

This command reads the Inquiry_Mode configuration parameter of the local BR/EDR Controller. See Section 6.5.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Inquiry_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Read_Inquiry_Mode command failed. See [Vol 1] Part F, Controller Error Codes for list of error codes. |

*Inquiry_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format |
| 0x01 | Inquiry Result format with RSSI |
| 0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Inquiry_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.50 Write Inquiry Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Mode | 0x0045 | Inquiry_Mode | Status |

**Description:**

This command writes the Inquiry_Mode configuration parameter of the local BR/EDR Controller. See Section 6.5.

**Command parameters:**

*Inquiry_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format (default) |
| 0x01 | Inquiry Result format with RSSI |
| 0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Inquiry_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Inquiry_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Inquiry_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.51 Read Page Scan Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Scan_Type | 0x0046 | *none* | Status,<br>Page_Scan_Type |

**Description:**

This command reads the Page Scan Type configuration parameter of the local BR/EDR Controller. See Section 6.11.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Page_Scan_Type command succeeded. |
| 0x01 to 0xFF | HCI_Read_Page_Scan_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Page_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Page_Scan_Type command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.52 Write Page Scan Type command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Scan_Type | 0x0047 | Page_Scan_Type | Status |

**Description:**

This command writes the Page Scan Type configuration parameter of the local BR/EDR Controller. See Section 6.11.

**Command parameters:**

*Page_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Page_Scan_Type command succeeded. |
| 0x01 to 0xFF | HCI_Write_Page_Scan_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Page_Scan_Type command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.53 Read AFH Channel Assessment Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_AFH_Channel_-Assessment_Mode | 0x0048 | *none* | Status,<br>AFH_Channel_Assessment_Mode |

**Description:**

This command reads the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the Controller's channel assessment scheme is enabled or disabled.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_AFH_Channel_Assessment_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Read_AFH_Channel_Assessment_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*AFH_Channel_Assessment_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Controller channel assessment disabled. |
| 0x01 | Controller channel assessment enabled. |
| All other values | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_AFH_Channel_Assessment_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.54 Write AFH Channel Assessment Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_AFH_Channel_-Assessment_Mode | 0x0049 | AFH_Channel_-Assessment_Mode | Status |

**Description:**

This command writes the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the Controller's channel assessment scheme is enabled or disabled.

Disabling channel assessment forces all channels to be unknown in the local classification for the BR/EDR physical transport, but does not affect the AFH_reporting_mode or support for the HCI_Set_AFH_Host_Channel_Classification command. A BR/EDR Peripheral in the AFH_reporting_enabled state shall continue to send LMP channel classification messages for any changes to the channel classification caused by either this command (altering the AFH_Channel_Assessment_Mode) or HCI_Set_AFH_Host_Channel_Classification command (providing a new channel classification from the Host).

Disabling channel assessment also forces all channels to be unknown in the local classification for the LE physical transport. If channel classification reporting is enabled by the Central, then the following rules apply to the Peripheral:

- Irrespective of whether channel assessment is enabled or disabled by the Host, the Controller shall continue to send LL_CHANNEL_STATUS_IND PDUs for any changes to the channel classification caused by the HCI_LE_Set_Host_Channel_Classification command.
- If channel assessment has been enabled by the Host, the Controller shall send LL_CHANNEL_STATUS_IND PDUs for any changes to the channel classification caused by the HCI_LE_Set_Host_Channel_Classification command and for any changes reported by the channel assessment scheme.
- The Controller shall send an LL_CHANNEL_STATUS_IND PDU whenever the channel classification changes because this command changes the channel assessment mode.

If the AFH_Channel_Assessment_Mode parameter is enabled and the Controller does not support a channel assessment scheme, other than via the HCI_Set_AFH_Host_Channel_Classification command (for BR/EDR) or via the HCI_LE_Set_Host_Channel_Classification command (for LE), then a Status parameter of 'Channel Assessment Not Supported' should be returned. See [Vol1] Part F, Controller Error Codes for a list of error codes and descriptions.

If the Controller supports a channel assessment scheme then the default AFH_Channel_Assessment_Mode is enabled, otherwise the default is disabled.

**Command parameters:**

*AFH_Channel_Assessment_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Controller channel assessment disabled. |
| 0x01 | Controller channel assessment enabled. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_AFH_Channel_Assessment_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_AFH_Channel_Assessment_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_AFH_Channel_Assessment_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.55 Read Extended Inquiry Response command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Extended_Inquiry_-Response | 0x0051 | *none* | Status,<br>FEC_Required,<br>Extended_Inquiry_Response |

**Description:**

This command reads the extended inquiry response to be sent during the extended inquiry response procedure. The FEC_Required parameter states if FEC encoding is required.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Extended_Inquiry_Response command succeeded. |
| 0x01 to 0xFF | HCI_Read_Extended_Inquiry_Response command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*FEC_Required:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | FEC is not required |
| 0x01 | FEC is required |
| All other values | Reserved for future use |

*Extended_Inquiry_Response:* *Size: 240 octets*

| Value | Parameter Description |
|---|---|
| | Extended inquiry response data as defined in [Vol 3] Part C, Section 8. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Extended_Inquiry_Response command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.56 Write Extended Inquiry Response command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Extended_Inquiry_-Response | 0x0052 | FEC_Required,<br>Extended_Inquiry_Response | Status |

**Description:**

This command writes the extended inquiry response to be sent during the extended inquiry response procedure. The Controller shall not modify or use the extended inquiry response data for any other purpose except removing some or all of the non-significant part. The FEC_Required parameter states if FEC encoding is required. The extended inquiry response data is not preserved over a reset. The initial value of the inquiry response data is all all-zero octets.

**Command parameters:**

*FEC_Required:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | FEC is not required |
| 0x01 | FEC is required |
| All other values | Reserved for future use |

*Extended_Inquiry_Response:* *Size: 240 octets*

| Value | Parameter Description |
|---|---|
| | Extended inquiry response data as defined in [Vol 3] Part C, Section 8. |
| | All octets zero (default). |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Extended_Inquiry_Response command succeeded |
| 0x01 to 0xFF | HCI_Write_Extended_Inquiry_Response command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Extended_Inquiry_Response command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.57 Refresh Encryption Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Refresh_Encryption_Key | 0x0053 | Connection_Handle | *none* |

**Description:**

This command is used by the Host to cause the BR/EDR Controller to refresh the encryption key on an ACL connection identified by a Connection_Handle by pausing and resuming encryption.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

An HCI_Command_Status event is sent from the BR/EDR Controller to the Host when the Controller has started the Refresh Encryption Key procedure. An HCI_Encryption_Key_Refresh_Complete event shall be generated when the Refresh Encryption Key procedure has completed.

### 7.3.58 Read Simple Pairing Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Simple_Pairing_Mode | 0x0055 | *none* | Status,<br>Simple_Pairing_Mode |

**Description:**

This command reads the Secure Simple Pairing mode parameter in the BR/EDR Controller.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Simple_Pairing_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Read_Simple_Pairing_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Simple_Pairing_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure Simple Pairing disabled (default) |
| 0x01 | Secure Simple Pairing enabled |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Simple_Pairing_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.59 Write Simple Pairing Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Simple_Pairing_Mode | 0x0056 | Simple_Pairing_Mode | Status |

**Description:**

This command writes the Secure Simple Pairing mode in the BR/EDR Controller.

The default value for Secure Simple Pairing mode shall be disabled. A Host shall not set Secure Simple Pairing mode to ‘disabled.’

This command shall be performed before initiating page scan or paging procedures where pairing is intended.

If the command succeeds, then the Controller shall set Secure Simple Pairing mode and the Secure Simple Pairing (Host Support) LMP feature bit to the value of Simple_Pairing_Mode; the new value shall be used in subsequent responses to an LMP_FEATURES_REQ PDU from a remote device.

**Command parameters:**

*Simple_Pairing_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure Simple Pairing disabled (default) |
| 0x01 | Secure Simple Pairing enabled |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Simple_Pairing_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Simple_Pairing_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Simple_Pairing_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.60 Read Local OOB Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_OOB_Data | 0x0057 | *none* | Status,<br>C,<br>R |

**Description:**

This command obtains a Secure Simple Pairing Hash C and Randomizer R which are intended to be transferred to a remote device using an OOB mechanism. The BR/EDR Controller shall create new values for C and R for each invocation of this command. Each random number R shall be created according to [Vol 2] Part H, Section 2.

Note: Each OOB transfer will have unique C and R values.

After each OOB transfer this command shall be used to obtain a new set of values for the next OOB transfer.

Note: The Controller keeps information used to generate these values for later use in the Secure Simple Pairing process. If the BR/EDR Controller is powered off or reset then this information is lost and the values obtained before the power off or reset are invalid.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_OOB_Data command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_OOB_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*C:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Hash C |

*R:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Secure Simple Pairing Randomizer R |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_OOB_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.61 Read Inquiry Response Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Response_Transmit_-Power_Level | 0x0058 | *none* | Status,<br>TX_Power |

**Description:**

This command reads the power level used to transmit the FHS and EIR data packets. This can be used directly in the Tx Power Level EIR data type.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Inquiry_Response_Transmit_Power_Level command succeeded. |
| 0x01 to 0xFF | HCI_Read_Inquiry_Response_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -70 to 20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the HCI_Read_Inquiry_Response_Transmit_Power_Level command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.62 Write Inquiry Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Transmit_Power_Level | 0x0059 | TX_Power | Status |

**Description:**

This command writes the inquiry transmit power level used to transmit the inquiry (ID) data packets. The Controller should use the supported TX power level closest to the TX_Power parameter.

**Command parameters:**

*TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -70 to 20<br>Units: dBm |

**Return parameters:**

*Status:* *Status:*

Size: 1 octet

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Inquiry_Transmit_Power_Level command succeeded |
| 0x01 to 0xFF | HCI_Write_Inquiry_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Inquiry_Transmit_Power_Level command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.63 Send Keypress Notification command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Send_Keypress_Notification | 0x0060 | BD_ADDR,<br>Notification_Type | Status,<br>BD_ADDR |

**Description:**

This command is used during the Passkey Entry protocol by a device with KeyboardOnly IO capabilities. It is used by a Host to inform the remote device when keys have been entered or erased.

**Command parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

*Notification_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 | Passkey entry started |
| 1 | Passkey digit entered |
| 2 | Passkey digit erased |
| 3 | Passkey cleared |
| 4 | Passkey entry completed |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Send_Keypress_Notification command succeeded |
| 0x01 to 0xFF | HCI_Send_Keypress_Notification command failed |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

**Event(s) generated (unless masked away):**

When the HCI_Send_Keypress_Notification command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.64 Read Default Erroneous Data Reporting command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Default_Erroneous_-Data_Reporting | 0x005A | *none* | Status,<br>Erroneous_Data_Reporting |

**Description:**

This command reads the Erroneous_Data_Reporting parameter.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Default_Erroneous_Data_Reporting command succeeded. |
| 0x01 to 0xFF | HCI_Read_Default_Erroneous_Data_Reporting command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Erroneous_Data_Reporting:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous data reporting disabled. |
| 0x01 | Erroneous data reporting enabled. |
| All other values | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Default_Erroneous_Data_Reporting command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.65 Write Default Erroneous Data Reporting command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Default_Erroneous_Data_-Reporting | 0x005B | Erroneous_Data_Reporting | Status |

**Description:**

This command writes the Erroneous_Data_Reporting parameter. The BR/EDR Controller shall set the Packet_Status_Flag as defined in Section 5.4.3, depending on the value of this parameter. The new value for the Erroneous_Data_Reporting parameter shall not apply to existing synchronous connections.

**Command parameters:**

*Erroneous_Data_Reporting:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous Data reporting disabled (default). |
| 0x01 | Erroneous data reporting enabled. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Default_Erroneous_Data_Reporting command succeeded. |
| 0x01 to 0xFF | HCI_Write_Default_Erroneous_Data_Reporting command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Default_Erroneous_Data_Reporting command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.66 Enhanced Flush command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enhanced_Flush | 0x005F | Handle,<br>Packet_Type | *none* |

**Description:**

This command is used to discard all L2CAP packets identified by Packet_Type that are currently pending for transmission in the Controller for the specified Handle, even if there currently are chunks of data that belong to more than one L2CAP packet of the same type in the Controller. The only packet type defined is automatically-flushable. Packets not identified by Packet_Type will not be flushed and will be processed normally by the Controller.

After flushing the packets, all data that is sent to the BR/EDR Controller for the same Handle and packet type shall be discarded by the Controller until an HCI ACL Data packet with the start Packet_Boundary_Flag (0x00 or 0x02) is received. This command allows higher-level software to control how long the Baseband should try to retransmit a Baseband packet of a specific type for a Handle before all data of that type currently pending for transmission in the Controller should be flushed. The HCI_Enhanced_Flush command shall be used for ACL-U connections only. On the BR/EDR Controller, the HCI_Flush command can be used to flush all packets (see Section 7.3.4). In addition to the HCI_Enhanced_Flush and HCI_Flush commands, the automatic flush timers (see Section 7.3.29) can be used to automatically flush an automatically-flushable L2CAP packet that is currently being transmitted after the specified flush timer has expired.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Handle to be used to identify a connection.<br>Range: 0x0000 to 0x0EFF |

*Packet_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Automatically flushable only. |
| All other values | Reserved for future use. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Enhanced_Flush command, the Controller shall send the HCI_Command_Status event to the Host. In addition, when all the packets identified by Packet_Type have been flushed for the specified Handle, the Controller shall send an HCI_Enhanced_Flush_Complete event to the Host. The Controller may send the HCI_Enhanced_Flush_Complete event immediately after flushing all the packets of type Packet_Type for the specified Handle, or it may wait until all packets for the specified Handle, independent of Packet_Type, buffered in the Controller at the time of the receipt of the HCI_Enhanced_Flush command, have been either flushed or transmitted.

### 7.3.67 [This section is no longer used]

### 7.3.68 [This section is no longer used]

### 7.3.69 Set Event Mask Page 2 command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Event_Mask_Page_2 | 0x0063 | Event_Mask_Page_2 | Status |

**Description:**

This command is used to control which events are generated by the HCI for the Host. The Event_Mask_Page_2 is a logical extension to the Event_Mask parameter of the HCI_Set_Event_Mask command. If the bit in the Event_Mask_Page_2 is set to a one, then the event associated with that bit shall be enabled. The event mask allows the Host to control how much it is interrupted.

The Controller shall ignore those bits which are reserved for future use or represent events which it does not support. If the Host sets any of these bits to 1, the Controller shall act as if they were set to 0.

**Command parameters:**

*Event_Mask_Page_2:* *Size: 8 octets*

| Bit | Parameter Description |
|---|---|
| 0 | Previously used |
| 1 | Previously used |
| 2 | Previously used |
| 3 | Previously used |
| 4 | Previously used |
| 5 | Previously used |
| 6 | Previously used |
| 7 | Previously used |
| 8 | HCI_Number_Of_Completed_Data_Blocks event |
| 9 | Previously used |
| 10 | Previously used |
| 11 | Previously used |
| 12 | Previously used |
| 13 | Previously used |
| 14 | HCI_Triggered_Clock_Capture event |
| 15 | HCI_Synchronization_Train_Complete event |
| 16 | HCI_Synchronization_Train_Received event |

| Bit | Parameter Description |
|---|---|
| 17 | HCI_Connectionless_Peripheral_Broadcast_Receive event |
| 18 | HCI_Connectionless_Peripheral_Broadcast_Timeout event |
| 19 | HCI_Truncated_Page_Complete event |
| 20 | HCI_Peripheral_Page_Response_Timeout event |
| 21 | HCI_Connectionless_Peripheral_Broadcast_Channel_Map_Change event |
| 22 | HCI_Inquiry_Response_Notification event |
| 23 | HCI_Authenticated_Payload_Timeout_Expired event |
| 24 | HCI_SAM_Status_Change event |
| 25 | HCI_Encryption_Change event [v2] |
| 60 to 63 | Reserved for specification development purposes |

The value with all bits set to 0 (which is the default) indicates that no events are specified.

All bits not listed in this table are reserved for future use.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Event_Mask_Page_2 command succeeded. |
| 0x01 to 0xFF | HCI_Set_Event_Mask_Page_2 command failed. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Event_Mask_Page_2 command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.70 [This section is no longer used]

### 7.3.71 [This section is no longer used]

### 7.3.72 Read Flow Control Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Flow_Control_Mode | 0x0066 | *none* | Status,<br>Flow_Control_Mode |

**Description:**

This command reads the value for the Flow_Control_Mode configuration parameter. See Section 6.33.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Flow_Control_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Read_Flow_Control_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Flow_Control_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Packet based data flow control mode |
| 0x01 | Data block based data flow control mode |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Flow_Control_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.73 Write Flow Control Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Flow_Control_Mode | 0x0067 | Flow_Control_Mode | Status |

**Description:**

This command writes the value for the Flow_Control_Mode configuration parameter. See Section 6.33.

**Command parameters:**

*Flow_Control_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Packet based data flow control mode (default) |
| 0x01 | Data block based data flow control mode |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Flow_Control_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Flow_Control_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Flow_Control_Mode command has completed, an HCI_Command_Complete event shall be generated. If the set fails then the Controller continues using its current mode.

### 7.3.74 Read Enhanced Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Enhanced_Transmit_-Power_Level | 0x0068 | Connection_Handle,<br>Type | Status,<br>Connection_Handle,<br>TX_Power_Level_GFSK,<br>TX_Power_Level_DQPSK,<br>TX_Power_Level_8DPSK |

**Description:**

This command reads the values for the Enhanced_Transmit_Power_Level parameters for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read Current Transmit Power Level |
| 0x01 | Read Maximum Transmit Power Level |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Enhanced_Transmit_Power_Level command succeeded. |
| 0x01 to 0xFF | HCI_Read_Enhanced_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*TX_Power_Level_GFSK:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -100 to 20<br>Units: dBm |

*TX_Power_Level_DQPSK:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -100 to 20<br>Units: dBm |

*TX_Power_Level_8DPSK:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -100 to 20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the HCI_Read_Enhanced_Transmit_Power_Level command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.75 [This section is no longer used]

### 7.3.76 [This section is no longer used]

### 7.3.77 [This section is no longer used]

### 7.3.78 Read LE Host Support command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_LE_Host_Support | 0x006C | *none* | Status,<br>LE_Supported_Host,<br>Unused |

**Description:**

This command is used to read the LE Supported (Host) Link Manager Protocol feature bit. See [Vol 2] Part C, Section 3.2.

The Unused parameter was previously used.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_LE_Host_Support command succeeded. |
| 0x01 to 0xFF | HCI_Read_LE_Host_Support command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*LE_Supported_Host:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | LE_Supported_Host parameter, see Section 6.34. |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | This value shall be returned by the Controller. |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_LE_Host_Support command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.79 Write LE Host Support command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_LE_Host_Support | 0x006D | LE_Supported_Host, Unused | Status |

**Description:**

This command is used to set the LE Supported (Host) Link Manager Protocol feature bit. See [Vol 2] Part C, Section 3.2.

The default value for this feature bit shall be disabled. When LE_Supported_Host is set to enabled the bit in LMP features mask indicating support for LE Support (Host) shall be set.

The Unused parameter was previously used.

**Command parameters:**

*LE_Supported_Host:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | LE_Supported_Host parameter. See Section 6.34 |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | This value shall be ignored by the Controller. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_LE_Host_Support command succeeded. |
| 0x01 to 0xFF | HCI_Write_LE_Host_Support command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_LE_Host_Support command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.80 Set MWS Channel Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_MWS_Channel_-Parameters | 0x006E | MWS_Channel_Enable, MWS_RX_Center_Frequency, MWS_TX_Center_Frequency, MWS_RX_Channel_Bandwidth, MWS_TX_Channel_Bandwidth, MWS_Channel_Type | Status |

**Description:**

This command is used to inform the Controller of the MWS channel parameters.

The MWS_Channel_Enable parameter is used to enable or disable the MWS channel. If it is set to 0x00, the remaining parameters shall be ignored.

The MWS_RX_Center_Frequency and MWS_TX_Center_Frequency parameters are used to indicate the center frequency of the MWS device's uplink (TX) and downlink (RX) channels. The uplink and downlink channel centers may be the same value or different values.

The MWS_RX_Channel_Bandwidth and MWS_TX_Channel_Bandwidth parameters are used to indicate the bandwidth, in kHz, of the MWS device's uplink and downlink channels.

The MWS_Channel_Type parameter describes the type of channel. The types are defined in Assigned Numbers.

**Command parameters:**

*MWS_Channel_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | MWS channel is disabled. |
| 0x01 | MWS channel is enabled. |
| All other values | Reserved for future use. |

*MWS_RX_Center_Frequency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS RX center frequency in MHz. |

*MWS_TX_Center_Frequency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS TX center frequency in MHz |

*MWS_RX_Channel_Bandwidth:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS RX channel bandwidth in kHz. |

*MWS_TX_Channel_Bandwidth:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS TX channel bandwidth in kHz. |

*MWS_Channel_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_MWS_Channel_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_Set_MWS_Channel_Parameters command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_Set_MWS_Channel_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.81 Set External Frame Configuration command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_External_Frame_-Configuration | 0x006F | MWS_Frame_Duration,<br>MWS_Frame_Sync_Assert_Offset,<br>MWS_Frame_Sync_Assert_Jitter,<br>MWS_Num_Periods,<br>Period_Duration[i],<br>Period_Type[i] | Status |

**Description:**

This command allows the Host to specify a frame configuration for an external collocated MWS system. This frame configuration persists until overwritten with a subsequent Set_External_Frame_Configuration or until the Controller is reset.

This command can be used to allow the Controller to align the piconet clock with an external frame structure.

When the external frame structure is a multiple of 1.25 ms, it can be aligned in a stable manner with the piconet clock.

The start of the external frame structure is defined as an offset from an external frame synchronization signal. This offset is defined by the MWS_Frame_Sync_Assert_Offset parameter. The offset is represented as the time, in microseconds, from the start of the next MWS frame to the FRAME_SYNC signal.

An external frame consists of downlink periods, uplink periods and guard periods. Downlink means the collocated MWS system is receiving, thus may be interfered with by Bluetooth transmissions. Uplink means the collocated MWS system is transmitting, thus may cause interference to Bluetooth receptions. A guard period may be used by the MWS system to compensate for propagation delays; in this case it should be regarded as split equally between downlink and uplink durations.

The number of specified periods is given by MWS_Num_Periods.

The duration of each period, in microseconds, is defined by the Period_Duration[i] parameters.

The Period_Type[i] parameter indicates if the specified period is an uplink, downlink, bi-directional or guard period.

The sum of all Period_Duration[i] parameters shall be equal to the MWS_Frame_Duration parameter.

Upon reception of an HCI_Set_External_Frame_Configuration command and a FRAME_SYNC signal from the MWS Coexistence Logical Interface, the Controller may compute the type 0 submap for local SAM slot maps. The Controller may then initiate the SAM set type 0 and SAM define map LMP sequences with the remote device.

**Command parameters:**

*MWS_Frame_Duration:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | External frame duration in microseconds |

*MWS_Frame_Sync_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | External frame offset, in microseconds (signed integer) |

*MWS_Frame_Sync_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | External frame sync jitter, in microseconds (unsigned integer) |

*MWS_Num_Periods:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of specified periods in an external frame.<br>Valid range: 1 to 32 |

*Period_Duration[i]:* *Size: MWS_Num_Periods × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Duration of the period, in microseconds |

*Period_Type[i]:* *Size: MWS_Num_Periods × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Downlink |
| 0x01 | Uplink |
| 0x02 | Bi-Directional |
| 0x03 | Guard Period |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_External_Frame_Configuration command succeeded. |
| 0x01 to 0xFF | HCI_Set_External_Frame_Configuration command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_Set_External_Frame_Configuration command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.82 Set MWS Signaling command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_-MWS_-Signaling | 0x0070 | MWS_RX_Assert_Offset,<br>MWS_RX_Assert_Jitter,<br>MWS_RX_Deassert_Offset,<br>MWS_RX_Deassert_Jitter,<br>MWS_TX_Assert_Offset,<br>MWS_TX_Assert_Jitter,<br>MWS_TX_Deassert_Offset,<br>MWS_TX_Deassert_Jitter,<br>MWS_Pattern_Assert_Offset,<br>MWS_Pattern_Assert_Jitter,<br>MWS_Inactivity_-Duration_Assert_Offset,<br>MWS_Inactivity_-Duration_Assert_Jitter,<br>MWS_Scan_Frequency_Assert_-Offset,<br>MWS_Scan_Frequency_Assert_-Jitter,<br>MWS_Priority_-Assert_Offset_Request | Status,<br>Bluetooth_RX_Priority_Assert_Offset,<br>Bluetooth_RX_Priority_Assert_Jitter,<br>Bluetooth_RX_-Priority_Deassert_Offset,<br>Bluetooth_RX_-Priority_Deassert_Jitter,<br>802_RX_Priority_Assert_Offset,<br>802_RX_Priority_Assert_Jitter,<br>802_RX_Priority_Deassert_Offset,<br>802_RX_Priority_Deassert_Jitter,<br>Bluetooth_TX_On_Assert_Offset,<br>Bluetooth_TX_On_Assert_Jitter,<br>Bluetooth_TX_On_Deassert_Offset,<br>Bluetooth_TX_On_Deassert_Jitter,<br>802_TX_On_Assert_Offset,<br>802_TX_On_Assert_Jitter,<br>802_TX_On_Deassert_Offset,<br>802_TX_On_Deassert_Jitter |

**Description:**

This command is used to inform the Bluetooth Controller of the MWS signaling interface logical layer parameters.

All signals are defined in [Vol 7] Part A.

**Command parameters:**

*MWS_RX_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_RX signal assert offset in microseconds (signed integer). |

*MWS_RX_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_RX signal assert jitter in microseconds (unsigned integer). |

*MWS_RX_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_RX signal de-assert offset in microseconds (signed integer). |

*MWS_RX_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_RX signal de-assert jitter in microseconds (unsigned integer). |

*MWS_TX_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_TX signal assert offset in microseconds (signed integer). |

*MWS_TX_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_TX signal assert jitter in microseconds (unsigned integer). |

*MWS_TX_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_TX signal de-assert offset in microseconds (signed integer). |

*MWS_TX_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_TX signal de-assert jitter in microseconds (unsigned integer). |

*MWS_Pattern_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_PATTERN signal assert offset in microseconds (signed integer). |

*MWS_Pattern_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_PATTERN signal assert jitter in microseconds (unsigned integer). |

*MWS_Inactivity_Duration_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_INACTIVITY_DURATION signal assert offset in microseconds (signed integer). |

*MWS_Inactivity_Duration_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_INACTIVITY_DURATION signal assert jitter in microseconds (unsigned integer). |

*MWS_Scan_Frequency_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_SCAN_FREQUENCY signal assert offset in microseconds (signed integer). |

*MWS_Scan_Frequency_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | MWS_SCAN_FREQUENCY signal assert jitter in microseconds (unsigned integer). |

*MWS_Priority_Assert_Offset_Request:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum advance notification from the beginning of an MWS Uplink period in microseconds (unsigned integer) before which the BLUETOOTH_RX_PRI or 802_RX_PRI signal shall be asserted to be recognized by the MWS. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_MWS_Signaling command succeeded. |
| 0x01 to 0xFF | HCI_Set_MWS_Signaling command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

*Bluetooth_RX_Priority_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_RX_PRI signal assert offset in microseconds (signed integer). |

*Bluetooth_RX_Priority_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_RX_PRI signal assert jitter in microseconds (unsigned integer). |

*Bluetooth_RX_Priority_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_RX_PRI signal de-assert offset in microseconds (signed integer). |

*Bluetooth_RX_Priority_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_RX_PRI signal de-assert jitter in microseconds (unsigned integer). |

*802_RX_Priority_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_RX_PRI signal assert offset in microseconds (signed integer). |

*802_RX_Priority_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_RX_PRI signal assert jitter in microseconds (unsigned integer). |

*802_RX_Priority_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_RX_PRI signal de-assert offset in microseconds (signed integer). |

*802_RX_Priority_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_RX_PRI signal de-assert jitter in microseconds (unsigned integer). |

*Bluetooth_TX_On_Assert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_TX_ON signal assert offset in microseconds (signed integer). |

*Bluetooth_TX_On_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_TX_ON signal assert jitter in microseconds (unsigned integer). |

*Bluetooth_TX_On_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_TX_ON signal de-assert offset in microseconds (signed integer). |

*Bluetooth_TX_On_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | BLUETOOTH_TX_ON signal de-assert jitter in microseconds (unsigned integer). |

*802_TX_On_Assert_Offset:* Size: 2 octets

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_TX_ON signal assert offset in microseconds (signed integer). |

*802_TX_On_Assert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_TX_ON signal assert jitter in microseconds (unsigned integer). |

*802_TX_On_Deassert_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_TX_ON signal de-assert offset in microseconds (signed integer). |

*802_TX_On_Deassert_Jitter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 802_TX_ON signal de-assert jitter in microseconds (unsigned integer). |

**Event(s) generated (unless masked away):**

When the HCI_Set_MWS_Signaling command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.83 Set MWS Transport Layer command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_MWS_Transport_Layer | 0x0071 | Transport_Layer,<br>To_MWS_Baud_Rate,<br>From_MWS_Baud_Rate | Status |

**Description:**

This command configures the transport layer between the Bluetooth Controller and MWS device.

**Command parameters:**

*Transport_Layer:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers. |

*To_MWS_Baud_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Baud rate in the Bluetooth to MWS direction in Baud. |

*From_MWS_Baud_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Baud rate in the MWS to Bluetooth direction in Baud. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_MWS_Transport_Layer command succeeded. |
| 0x01 to 0xFF | HCI_Set_MWS_Transport_Layer command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_Set_MWS_Transport_Layer command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.84 Set MWS Scan Frequency Table command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_MWS_Scan_Frequency_Table | 0x0072 | Num_Scan_Frequencies,<br>Scan_Frequency_Low[i],<br>Scan_Frequency_High[i] | Status |

**Description:**

This command configures the MWS scan frequency table in the Controller.

The Num_Scan_Frequencies parameter indicates the number of MWS scan frequencies to be set. A Controller shall support at least 8 table entries.

The Scan_Frequency_Low[i] and Scan_Frequency_High[i] parameters indicate the lower and upper edges for each scan frequency. The parameters for a given value of i correspond to a MWS_SCAN_FREQUENCY value of i+1 (see [Vol 7] Part A, Section 2.1.10); e.g., the first value corresponds to MWS_SCAN_FREQUENCY = 1.

**Command parameters:**

*Num_Scan_Frequencies:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N | Number of MWS scan frequencies to be set in the table. |

*Scan_Frequency_Low[i]:* *Size: Num_Scan_Frequencies × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Lower edge of the MWS scan frequency in MHz. |

*Scan_Frequency_High[i]:* *Size: Num_Scan_Frequencies × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Upper edge of the MWS scan frequency in MHz. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_MWS_Scan_Frequency_Table command succeeded. |
| 0x01 to 0xFF | HCI_Set_MWS_Scan_Frequency_Table command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_Set_MWS_Scan_Frequency_Table command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.85 Set MWS_PATTERN Configuration command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_MWS_PATTERN_-Configuration | 0x0073 | MWS_Pattern_Index,<br>MWS_Pattern_Num_Intervals,<br>MWS_Pattern_Interval_Duration[i],<br>MWS_Pattern_Interval_Type[i] | Status |

**Description:**

This command is used by the Host to specify, in conjunction with the HCI_Set_External_Frame_Configuration command, local MWS_PATTERN parameters for an external collocated system.

An MWS_PATTERN configuration shall persist until overwritten by a subsequent Set_MWS_PATTERN_Configuration. All MWS_PATTERN configurations are deleted when an HCI_Set_External_Frame_Configuration command is received or when the Controller is reset.

The sum of the MWS_Pattern_Interval_Duration parameters shall be an integer multiple of the length of a frame as defined by the most recent HCI_Set_External_Frame_Configuration command.

Upon reception of an HCI_Set_MWS_PATTERN_Configuration command, the Controller may compute the local SAM slot map with SAM_Index equal to MWS_Pattern_Index. If the SAM slot map does not exist, it should be created; if the SAM slot map already exists, its parameters should be replaced. The Controller may then initiate the SAM define map LMP sequence with the remote device.

Upon reception of an MWS_PATTERN signal, with a value other than 3, from the MWS Coexistence Logical Interface (see [Vol 7] Part A), the Controller should check the MWS_PATTERN value against the SAM_Index of those SAM slot maps that have been configured by previous HCI_Set_MWS_PATTERN_Configuration commands. It should then take the following course of action:

1. If MWS_PATTERN does not match any configured SAM slot map, it should take no further action.
2. If MWS_PATTERN matches an available SAM slot map that is already active or is being activated, it should take no further action (i.e. let the current or pending active SAM slot map continue).

3. If MWS_PATTERN matches an available SAM slot map that is neither active nor is being activated, then:
   a. If the SAM slot map has been activated previously using the LMP_SAM_SET_TYPE0 (if relevant) and LMP_SAM_DEFINE_MAP LMP sequences, the Controller should start the SAM switch LMP sequence to activate the matched SAM slot map;
   b. Otherwise the Controller should start or complete the SAM set type 0 (if relevant), SAM define map, and SAM switch LMP sequences to activate the matched SAM slot map.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The MWS_Pattern_Interval_Duration for an interval with type 4 is greater than the length of a frame. | *Invalid HCI Command Parameters* (0x12) |
| MC | The sum of the MWS_Pattern_Interval_Duration parameters for the intervals that precede an interval with type 4 is not a multiple of the length of the frame. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*MWS_Pattern_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Index of the MWS_PATTERN instance to be configured. Range: 0 to 2. |

*MWS_Pattern_Num_Intervals:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of intervals in the following arrays. |

*MWS_Pattern_Interval_Duration[i]:* *Size: MWS_Pattern_Num_Intervals × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The duration of this Bluetooth activity interval in microseconds. |

*MWS_Pattern_Interval_Type[i]:* *Size: MWS_Pattern_Num_Intervals × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Neither transmission nor reception is allowed in this interval. |
| 0x01 | Transmission is allowed in this interval. |
| 0x02 | Reception is allowed in this interval. |
| 0x03 | Both transmission and reception are allowed in this interval. |
| 0x04 | Interval for the MWS frame as defined by the HCI_Set_External_Frame_Configuration command. |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_MWS_PATTERN_Configuration command succeeded |
| 0x01 to 0xFF | HCI_Set_MWS_PATTERN_Configuration command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_Set_MWS_PATTERN_Configuration command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.86 Set Reserved LT_ADDR command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Reserved_LT_ADDR | 0x0074 | LT_ADDR | Status,<br>LT_ADDR |

**Description:**

This command allows the Host to request that the BR/EDR Controller reserve a specific LT_ADDR for Connectionless Peripheral Broadcast.

If the command succeeds, then the reserved LT_ADDR shall be used when issuing subsequent HCI_Set_Connectionless_Peripheral_Broadcast_Data and HCI_Set_Connectionless_Peripheral_Broadcast commands.

To ensure that the reserved LT_ADDR is not already allocated, it is recommended that this command be issued at some point after HCI_Reset is issued but before page scanning is enabled or paging is initiated.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The LT_ADDR is already in use | *Connection Already Exists* (0x0B) |
| MC | The LT_ADDR is out of range. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR to reserve for Connectionless Peripheral Broadcast |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Reserved_LT_ADDR command succeeded. |
| 0x01 to 0xFF | HCI_Set_Reserved_LT_ADDR command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR reserved for Connectionless Peripheral Broadcast. This parameter shall have the same value as the command parameter LT_ADDR. |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Set_Reserved_LT_ADDR command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.87 Delete Reserved LT_ADDR command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Delete_Reserved_LT_ADDR | 0x0075 | LT_ADDR | Status,<br>LT_ADDR |

**Description:**

This command requests that the BR/EDR Controller cancel the reservation for a specific LT_ADDR reserved for the purposes of Connectionless Peripheral Broadcast.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The LT_ADDR has not been reserved. | *Unknown Connection Identifier* (0x02) |
| MC | Connectionless Peripheral Broadcast mode is still active. | *Command Disallowed* (0x0C) |

**Command parameters:**

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR currently reserved for Connectionless Peripheral Broadcast and for which reservation is to be cancelled |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Delete_Reserved_LT_ADDR command succeeded. |
| 0x01 to 0xFF | HCI_Delete_Reserved_LT_ADDR command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR whose reservation the Host has requested to cancel. |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Delete_Reserved_LT_ADDR command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.88 Set Connectionless Peripheral Broadcast Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Connectionless_Peripheral_-Broadcast_Data | 0x0076 | LT_ADDR,<br>Fragment,<br>Data_Length,<br>Data | Status,<br>LT_ADDR |

**Description:**

This command provides the ability for the Host to set Connectionless Peripheral Broadcast data in the BR/EDR Controller. This command may be issued at any time after an LT_ADDR has been reserved regardless of whether Connectionless Peripheral Broadcast mode has been enabled or disabled by the Enable parameter in the HCI_Set_Connectionless_Peripheral_Broadcast command.

If Connectionless Peripheral Broadcast mode is disabled, this data shall be kept by the BR/EDR Controller and used once Connectionless Peripheral Broadcast mode is enabled. If Connectionless Peripheral Broadcast mode is enabled, and this command is successful, this data will be sent starting with the next Connectionless Peripheral Broadcast instant.

The Data_Length field may be zero, in which case no data needs to be provided.

The Host may fragment the data using the Fragment field in the command. If the combined length of the fragments exceeds the capacity of the largest allowed packet size specified in the HCI_Set_Connectionless_Peripheral_Broadcast command, all fragments associated with the data being assembled shall be discarded.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The LT_ADDR has not been reserved. | *Unknown Connection Identifier* (0x02) |
| MC | The combined length of the fragments exceeds the capacity of the largest allowed packet size specified in the HCI_Set_Connectionless_Peripheral_Broadcast command. | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR on which to send Connectionless Peripheral Broadcast data |
| All other values | Reserved for future use |

*Fragment:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Continuation fragment |
| 0x01 | Starting fragment |
| 0x02 | Ending fragment |
| 0x03 | No fragmentation (single fragment) |
| All other values | Reserved for future use |

*Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of the Data field |

*Data:* *Size: Data_Length octets*

| Parameter Description |
|---|
| Data to send in future Connectionless Peripheral Broadcast packets. This data will be repeated in future Connectionless Peripheral Broadcast instants until new data is provided.<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Connectionless_Peripheral_Broadcast_Data command succeeded. |
| 0x01 to 0xFF | HCI_Set_Connectionless_Peripheral_Broadcast_Data command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR on which Connectionless Peripheral Broadcast data will be sent |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Set_Connectionless_Peripheral_Broadcast_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.89 Read Synchronization Train Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Synchronization_Train_-Parameters | 0x0077 | *none* | Status,<br>Sync_Train_Interval,<br>Sync_Train_Timeout,<br>Service_Data |

**Description:**

This command returns the currently configured values for the Synchronization Train functionality in the Central's BR/EDR Controller. This command may be issued at any time.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Synchronization_Train_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_Read_Synchronization_Train_Parameters command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*Sync_Train_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Interval in slots between consecutive Synchronization Train events on the same channel.<br>Range: 0x0020 to 0xFFFE; only even values are valid |

*Sync_Train_Timeout:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Duration in slots to continue sending the synchronization train<br>Range: 0x00000002 to 0x07FFFFFE; only even values are valid |

*Service_Data:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Host provided value included in Synchronization Train packet, octet 27; see [Vol 2] Part B, Table 8.11. |

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Read_Synchronization_Train_Parameters command, it shall send an HCI_Command_Complete event to the Host.

### 7.3.90 Write Synchronization Train Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Synchronization_Train_-Parameters | 0x0078 | Interval_Min,<br>Interval_Max,<br>Sync_Train_Timeout,<br>Service_Data | Status,<br>Sync_Train_Interval |

**Description:**

This command configures the Synchronization Train functionality in the BR/EDR Controller. This command may be issued at any time.

Note: The AFH_Channel_Map used in the Synchronization Train packets is configured by the HCI_Set_AFH_Host_Channel_Classification command and the local channel classification in the BR/EDR Controller.

Interval_Min and Interval_Max specify the allowed range of Sync_Train_Interval. Refer to [Vol 2] Part B, Section 2.7.2 for a detailed description of Sync_Train_Interval. The BR/EDR Controller shall select an interval from this range and return it in Sync_Train_Interval.

Once started (via the HCI_Start_Synchronization_Train command) the Synchronization Train will continue until Sync_Train_Timeout slots have passed or Connectionless Peripheral Broadcast has been disabled.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller is unable to select a value in the range Interval_Min to Interval_Max. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum value allowed for the interval Sync_Train_Interval in slots.<br>Range: 0x0020 to 0xFFFE; only even values are valid |

*Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum value allowed for the interval Sync_Train_Interval in slots.<br>Range: 0x0020 to 0xFFFE; only even values are valid |

*Sync_Train_Timeout:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Duration in slots to continue sending the synchronization train<br>Range: 0x00000002 to 0x07FFFFFE; only even values are valid |

*Service_Data:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Host provided value to be included in octet 27 of the Synchronization Train packet payload body; see [Vol 2] Part B, Table 8.11. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Synchronization_Train_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_Write_Synchronization_Train_Parameters command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*Sync_Train_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Interval in slots between consecutive Synchronization Train packets on the same channel.<br>Range: 0x0020 to 0xFFFE; only even values are valid |

**Event(s) generated (unless masked away):**

When the BR/EDR Controller receives the HCI_Write_Synchronization_Train_-Parameters command, it shall send an HCI_Command_Complete event to the Host.

### 7.3.91 Read Secure Connections Host Support command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Secure_-Connections_Host_Support | 0x0079 | *none* | Status,<br>Secure_Connections_Host_Support |

**Description:**

This command reads the Secure_Connections_Host_Support parameter in the BR/EDR Controller. When Secure Connections Host Support is set to 'enabled' the Controller uses the enhanced reporting mechanisms for the Encryption_Enabled parameter in the HCI_Encryption_Change event (see Section 7.7.8) and the Key_Type parameter in the HCI_Link_Key_Notification event (see Section 7.7.24).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Secure_Connections_Host_Support command succeeded. |
| 0x01 to 0xFF | HCI_Read_Secure_Connections_Host_Support command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Secure_Connections_Host_Support:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure_Connections_Host_Support is ‘disabled’. Host does not support Secure Connections (default) |
| 0x01 | Secure_Connections_Host_Support is ‘enabled’. Host supports Secure Connections |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Secure_Connections_Host_Support command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.92 Write Secure Connections Host Support command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Secure_-Connections_Host_Support | 0x007A | Secure_Connections_Host_Support | Status |

**Description:**

This command writes the Secure_Connections_Host_Support parameter in the BR/EDR Controller. When Secure Connections Host Support is set to 'enabled' the Controller shall use the enhanced reporting mechanisms for the Encryption_Enabled parameter in the HCI_Encryption_Change event (see Section 7.7.8) and the Key_Type parameter in the HCI_Link_Key_Notification event (see Section 7.7.24).

The Link Manager Secure Connections (Host Support) feature bit shall be set to the Secure_Connections_Host_Support parameter. The default value for Secure_Connections_Host_Support shall be 'disabled.' When Secure_Connections_Host_Support is set to 'enabled,' the bit in the LMP features mask indicating support for Secure Connections (Host Support) shall be set to enabled in subsequent responses to an LMP_FEATURES_REQ from a remote device.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller is paging, has page scanning enabled, or has an ACL connection. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Secure_Connections_Host_Support:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure_Connections_Host_Support is ‘disabled’. Host does not support Secure Connections (default) |
| 0x01 | Secure_Connections_Host_Support is ‘enabled’. Host supports Secure Connections |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Secure_Connections_Host_Support command succeeded. |
| 0x01 to 0xFF | HCI_Write_Secure_Connections_Host_Support command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Secure_Connections_Host_Support command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.93 Read Authenticated Payload Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Authenticated_-Payload_Timeout | 0x007B | Connection_Handle | Status,<br>Connection_Handle,<br>Authenticated_Payload_Timeout |

**Description:**

This command reads the Authenticated_Payload_Timeout (*authenticatedPayloadTO*, see [Vol 2] Part B, Appendix B for BR/EDR connections and [Vol 6] Part B, Section 5.4 for LE connections) parameter in the Controller on the specified Connection_Handle.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle identifies a BR/EDR synchronous connection or an LE BIS or CIS. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Authenticated_Payload_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Read_Authenticated_Payload_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Authenticated_Payload_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum amount of time specified between packets authenticated by a MIC.<br>Default = 0x0BB8 (30 s)<br>Range: 0x0001 to 0xFFFF<br>Time = N × 10 ms<br>Time Range: 10 ms to 655,350 ms |

**Event(s) generated (unless masked away):**

When the HCI_Read_Authenticated_Payload_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.94 Write Authenticated Payload Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Authenticated_-Payload_Timeout | 0x007C | Connection_Handle,<br>Authenticated_Payload_Timeout | Status,<br>Connection_Handle |

**Description:**

This command writes the Authenticated_Payload_Timeout (*authenticatedPayloadTO*, see [Vol 2] Part B, Appendix B and [Vol 6] Part B, Section 5.4 for the LE connection) parameter in the Controller for the specified Connection_Handle.

When the Connection_Handle identifies a BR/EDR ACL connection:

- If the connection is in Sniff mode, the Authenticated_Payload_Timeout shall be equal to or greater than $T_{sniff}$.
- If the connection is in Sniff Subrating mode, the Authenticated_Payload_Timeout shall be equal to or greater than

  (max subrate)x$T_{sniff}$.
- If the connection is in Hold mode, the Authenticated_Payload_Timeout shall be equal to or greater than the *holdTO* value.

When the Connection_Handle identifies an LE ACL connection, the Authenticated_Payload_Timeout shall be equal to or greater than *connInterval* × *connSubrateFactor* × (1 + *connPeripheralLatency*).

When the Connection_Handle is associated with a BR/EDR ACL connection, the Link Manager will use this parameter to determine when to use the LMP ping sequence.

When the Connection_Handle is associated with an LE ACL connection, the Link Layer will use this parameter to determine when to use the LE ping sequence.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle identifies a BR/EDR synchronous connection or an LE BIS or CIS. | *Command Disallowed* (0x0C) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Authenticated_Payload_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum amount of time specified between packets authenticated by a valid MIC.<br>Range: 0x0001 to 0xFFFF<br>Time = N × 10 ms<br>Time Range: 10 ms to 655,350 ms |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Authenticated_Payload_Timeout command succeeded. |
| 0x01 to 0xFF | HCI_Write_Authenticated_Payload_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_Write_Authenticated_Payload_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.95 Read Local OOB Extended Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_OOB_Extended_Data | 0x007D | *none* | Status,<br>C_192,<br>R_192,<br>C_256,<br>R_256 |

**Description:**

This command obtains the Secure Simple Pairing Hash C_192, Randomizer R_192, Hash C_256, and Randomizer R_256, which are intended to be transferred to a remote device using an OOB mechanism. The BR/EDR Controller shall create new values for C_192, R_192, C_256, and R_256 for each invocation of this command. Each random number (R_192 and R_256) shall be created according to [Vol 2] Part H, Section 2.

Note: Each OOB transfer will have unique C_192, R_192, C_256, and R_256 values.

After each OOB transfer this command shall be used to obtain a new set of values for the next OOB transfer.

Note: The Controller keeps information used to generate these values for later use in the Secure Simple Pairing process. If the BR/EDR Controller is powered off or reset then this information is lost and the values obtained before the power off or reset are invalid.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Neither Secure Connections (Host Support) nor Secure Simple Pairing (Host Support) are enabled. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_OOB_Extended_Data command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_OOB_Extended_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*C_192:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX | Secure Simple Pairing Hash C derived from the P-192 public key. |

*R_192:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX | Secure Simple Pairing Randomizer associated with the P-192 public key. |

*C_256:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX | Secure Simple Pairing Hash C derived from the P-256 public key. |

*R_256:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX XXXXXXXXXXX XXXXXXXXX | Secure Simple Pairing Randomizer associated with the P-256 public key. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_OOB_Extended_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.96 Read Extended Page Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Extended_Page_Timeout | 0x007E | *none* | Status,<br>Extended_Page_Timeout |

**Description:**

This command will read the value for the Extended_Page_Timeout configuration parameter. See Section 6.41.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Extended_Page_Timeout command succeeded. |
| 0x01 to 0x0F | HCI_Read_Extended_Page_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Extended_Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Extended Page Timeout measured in number of Baseband slots. Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 (default) to 0xFFFF<br>Time Range: 0 to 40.9 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Extended_Page_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.97 Write Extended Page Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Extended_Page_Timeout | 0x007F | Extended_Page_Timeout | Status |

**Description:**

This command will write the value for the Extended_Page_Timeout configuration parameter. See Section 6.41.

**Command parameters:**

*Extended_Page_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Extended Page Timeout measured in number of Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 (default) to 0xFFFF<br>Time Range: 0 to 40.9 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Extended_Page_Timeout command succeeded. |
| 0x01 to 0x0F | HCI_Write_Extended_Page_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Extended_Page_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.98 Read Extended Inquiry Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Extended_Inquiry_-Length | 0x0080 | *none* | Status,<br>Extended_Inquiry_Length |

**Description:**

This command will read the value for the Extended_Inquiry_Length configuration parameter. See Section 6.42.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Extended_Inquiry_Length command succeeded. |
| 0x01 to 0x0F | HCI_Read_Extended_Inquiry_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Extended_Inquiry_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Extended_Inquiry_Length measured in number of Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 (default) to 0xFFFF<br>Time Range: 0 to 40.9 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Extended_Inquiry_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.99 Write Extended Inquiry Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Extended_Inquiry_Length | 0x0081 | Extended_Inquiry_Length | Status |

**Description:**

This command will write the value for the Extended_Inquiry_Length configuration parameter. The Extended_Inquiry_Length configuration parameter defines the maximum time after the Inquiry_Length expires that the local Link Manager may wait for a Baseband inquiry response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the inquiry at Baseband level, the inquiry will be considered to have failed.

**Command parameters:**

*Extended_Inquiry_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Extended_Inquiry_Length measured in number of Baseband slots.<br>Interval Length = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 (default) to 0xFFFF<br>Time Range: 0 to 40.9 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Extended_Inquiry_Length command succeeded. |
| 0x01 to 0x0F | HCI_Write_Extended_Inquiry_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Write_Extended_Inquiry_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.100 Set Ecosystem Base Interval command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Ecosystem_Base_Interval | 0x0082 | Interval | Status |

**Description:**

This command provides a hint to the Controller specifying the base communication interval the Controller can expect current and future communications to use. The Controller can assume that future activities will use an interval that is a multiple of the hint and may use that assumption when scheduling future activities. For example, if the Host expects to set up an LE ACL with a connection interval of 15 ms and a periodic advertisement with an interval of 40 ms, it would issue this command with Interval set to 5 ms. If the Host gives a range of possible intervals for an activity, the Controller could use a value in that range that is a multiple of the hint; e.g., if it gives the range 37.5 to 42.5 ms, the Controller could use 40 ms in preference to any other value. Communications being scheduled can include, but are not limited to, (e)SCO connections, BR/EDR ACL connections in Sniff mode, LE ACL connections, periodic advertisements, CISes, and BISes.

The Host may use an interval of zero to indicate that the most recently provided hint is no longer valid.

**Command parameters:**

*Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Ignore any previous hint |
| N = 0xXXXX | Base interval of the ecosystem<br>Range: 0x0002 to 0x7DF0<br>Time = N × 1.25 ms<br>Time Range: 2.5 ms to 40.9 s. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Ecosystem_Base_Interval command succeeded. |
| All other values | HCI_Set_Ecosystem_Base_Interval command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Ecosystem_Base_Interval command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.101 Configure Data Path command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Configure_Data_Path | 0x0083 | Data_Path_Direction,<br>Data_Path_ID,<br>Vendor_Specific_Config_Length,<br>Vendor_Specific_Config | Status |

**Description:**

This command is used to request the Controller to configure the data transport path in a given direction between the Controller and the Host.

The Data_Path_Direction parameter specifies the direction to be configured.

The Data_Path_ID parameter shall indicate the logical transport channel number for the non-HCI transport data path (e.g PCM interface) to be configured. The meaning of these logical transport channel numbers is vendor-specific.

The Vendor_Specific_Config parameter specifies additional vendor-specific configuration information that a Host may provide to the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Data_Path_ID value is unsupported. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Data_Path_Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Input (Host to Controller) |
| 0x01 | Output (Controller to Host) |
| All other values | Reserved for future use |

*Data_Path_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved for future use |
| 0x01 to 0xFE | Logical channel number; the meaning is vendor-specific. |
| 0xFF | Reserved for future use |

*Vendor_Specific_Config_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of the vendor-specific configuration data |

*Vendor_Specific_Config:* *Size: Vendor_Specific_Config_Length octets*

| Parameter Description |
|---|
| Vendor-specific configuration data for the data path being configured<br>Note: This parameter has a variable length. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Configure_Data_Path command succeeded |
| 0x01 to 0xFF | HCI_Configure_Data_Path command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Configure_Data_Path command has completed, an HCI_Command_Complete event shall be generated.

### 7.3.102 Set Min Encryption Key Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Min_Encryption_Key_Size | 0x0084 | Min_Encryption_Key_Size | Status |

**Description:**

This command is used by the Host to configure the minimum encryption key size. The Controller shall not negotiate a key size smaller than this value for any subsequent connection over the BR/EDR transport. This command shall not affect any existing connections.

The Min_Encryption_Key_Size parameter specifies the new minimum encryption key size in octets. See [Vol 3] Part C, Section 5.2.2.8 for recommendations concerning key sizes.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller does not support the Min_Encryption_Key_Size. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Min_Encryption_Key_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Minimum encryption key size in octets.<br>Range: 0x07 to 0x10<br>Default: vendor-specific |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Min_Encryption_Key_Size command succeeded |
| 0x01-0xFF | HCI_Set_Min_Encryption_Key_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Min_Encryption_Key_Size command has completed, an HCI_Command_Complete event shall be generated.

## 7.4 Informational parameters

The informational parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the BR/EDR Controller and the capabilities of the Link Manager and Baseband in the BR/EDR Controller. The Host device cannot modify any of these parameters.

For Informational Parameters commands, the OGF is defined as 0x04.

### 7.4.1 Read Local Version Information command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Version_Information | 0x0001 | *none* | Status,<br>HCI_Version,<br>HCI_Subversion,<br>LMP_Version,<br>Company_Identifier,<br>LMP_Subversion |

**Description:**

This command reads the values for the version information for the local Controller.

The HCI_Version information defines the version information of the HCI layer. The LMP_Version information defines the version of the LMP. The Company_Identifier information indicates the manufacturer of the local device.

The HCI_Subversion and LMP_Subversion are vendor-specific.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Version_Information command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_Version_Information command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*HCI_Version:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Version of the HCI Specification supported by the Controller<br>See Assigned Numbers |

*HCI_Subversion:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Revision of the HCI implementation in the Controller. This value is vendor-specific. |

*LMP_Version:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Version of the current LMP supported by the Controller.<br>See Assigned Numbers |

*Company_Identifier:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Company identifier for the manufacturer of the Controller.<br>See Assigned Numbers |

*LMP_Subversion:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Subversion of the Current LMP in the Controller. This value is vendor-specific. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Version_Information command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.2 Read Local Supported Commands command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Supported_Commands [v2] | 0x0010 | *none* | Status, Supported_Commands |
| HCI_Read_Local_Supported_Commands [v1] | 0x0002 | *none* | Status, Supported_Commands |

**Description:**

This command reads the list of HCI commands supported for the local Controller.

This command shall return the Supported_Commands configuration parameter.

See Section 6.27 for more information.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 | HCI_Read_Local_Supported_Commands command succeeded |
| 0x01 to 0xFF | HCI_Read_Local_Supported_Commands command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Supported_Commands [v1]:* *Size: 64 octets*
*Supported_Commands [v2]:* *Size: 251 octets*

| Value | Parameter Description |
|---|---|
| | Bit mask for each HCI command. If a bit is 1, the Controller supports the corresponding command and the features required for the command. Unsupported or undefined commands shall be set to 0.<br>See Section 6.27. |

The [v1] command returns octets 0 to 63 of the supported commands; the [v2] command returns all octets.

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Supported_Commands command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.3 Read Local Supported Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Supported_Features | 0x0003 | *none* | Status,<br>LMP_Features |

**Description:**

This command requests a list of the supported features for the local BR/EDR Controller. This command will return a list of the LMP features. For details see [Vol 2] Part C, Section 3.2.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Supported_Features command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes. |

*LMP_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit Mask List of LMP features. For details see [Vol 2] Part C, Section 3.2.<br>On a device that does not support BR/EDR, this shall be set to either 0x0000_0060_0000_0000 or 0x8000_0060_0000_0000.<br>Note: bit 37 previously indicated a device that does not support BR/EDR. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Supported_Features command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.4 Read Local Extended Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Extended_-Features | 0x0004 | Page_Number | Status,<br>Page_Number,<br>Max_Page_Number,<br>Extended_LMP_Features |

**Description:**

This command returns the requested page of the extended LMP features.

**Command parameters:**

*Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Requests the normal LMP features as returned by Read_Local_Supported_Features. |
| 0x01 to 0xFF | Return the corresponding page of features. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Extended_Features command succeeded |
| 0x01 to 0xFF | HCI_Read_Local_Extended_Features command failed. See [Vol 1] Part F, Controller Error Codes for list of error codes. |

*Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The normal LMP features as returned by Read_Local_Supported_Features. |
| 0x01 to 0xFF | The page number of the features returned. |

*Max_Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | The highest features page number which contains non-zero bits for the local device. |

*Extended_LMP_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit map of requested page of LMP features.<br>See [Vol 2] Part C, Section 3.3 for details. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Extended_Features command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.5 Read Buffer Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Buffer_Size | 0x0005 | *none* | Status,<br>ACL_Data_Packet_Length,<br>Synchronous_Data_Packet_Length,<br>Total_Num_ACL_Data_Packets,<br>Total_Num_Synchronous_Data_Packets |

**Description:**

This command is used to read the maximum size of the data portion of HCI ACL and Synchronous Data packets sent from the Host to the Controller. The Host will fragment the data to be transmitted from the Host to the Controller according to these sizes, so that the HCI Data packets will contain data with up to these sizes. The HCI_Read_Buffer_Size command also returns the total number of HCI ACL and Synchronous Data packets that can be stored in the data buffers of the Controller. The HCI_Read_Buffer_Size command shall be issued by the Host before it sends any data to the Controller.

For a device supporting BR/EDR and LE, if the HCI_LE_Read_Buffer_Size command returned zero for the number of buffers, then buffers returned by Read_Buffer_Size are shared between BR/EDR and LE.

The ACL_Data_Packet_Length parameter will be used to determine the size of the L2CAP fragments contained in ACL Data packets, which are transferred from the Host to the Controller to be broken up into Baseband packets by the Link Manager. The Synchronous_Data_Packet_Length parameter is used to determine the maximum size of HCI Synchronous Data packets. The Total_Num_ACL_Data_Packets parameter contains the total number of HCI ACL Data packets that can be stored in the data buffers of the Controller. The Host will determine how the buffers are to be divided between different Connection_Handles. The Total_Num_Synchronous_Data_Packets parameter gives the same information but for HCI Synchronous Data packets.
If the Controller does not support SCO or eSCO over HCI, then it shall set Total_Num_Synchronous_Data_Packets to zero, in which case the Host shall ignore the Synchronous_Data_Packet_Length parameter.

The ACL_Data_Packet_Length and Synchronous_Data_Packet_Length return parameters do not include the length of the HCI ACL Data packet header or the HCI Synchronous Data packet header respectively.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Buffer_Size command succeeded. |
| 0x01 to 0xFF | HCI_Read_Buffer_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*ACL_Data_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum length (in octets) of the data portion of each HCI ACL Data packet that the Controller is able to accept.<br>Range: 0x0001 to 0xFFFF |

*Synchronous_Data_Packet_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Maximum length (in octets) of the data portion of each HCI Synchronous Data packet that the Controller is able to accept.<br>Range: 0x01 to 0xFF |

*Total_Num_ACL_Data_Packets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total number of HCI ACL Data packets that can be stored in the data buffers of the Controller.<br>Range: 0x0001 to 0xFFFF |

*Total_Num_Synchronous_Data_Packets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | The Controller does not support SCO or eSCO over HCI. |
| 0xXXXX | Total number of HCI Synchronous Data packets that can be stored in the data buffers of the Controller. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Buffer_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.6 Read BD_ADDR command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_BD_ADDR | 0x0009 | *none* | Status,<br>BD_ADDR |

**Description:**

On a BR/EDR Controller, this command reads the Bluetooth Controller address (BD_ADDR). (See [Vol 2] Part B, Section 1.2 and [Vol 3] Part C, Section 3.2.1).

On an LE Controller, this command shall read the Public Device Address as defined in [Vol 6] Part B, Section 1.3. If this Controller does not have a Public Device Address, the value 0x000000000000 shall be returned.

On a BR/EDR/LE Controller, the public address shall be the same as the BD_ADDR.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_BD_ADDR command succeeded. |
| 0x01 to 0xFF | HCI_Read_BD_ADDR command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device |

**Event(s) generated (unless masked away):**

When the HCI_Read_BD_ADDR command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.7 Read Data Block Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Data_Block_Size | 0x000A | *none* | Status,<br>Max_ACL_Data_Packet_Length,<br>Data_Block_Length,<br>Total_Num_Data_Blocks |

**Description:**

This command is used to read values regarding the maximum permitted data transfers over the Controller and the data buffering available in the Controller.

The Host uses this information when fragmenting data for transmission, and when performing block-based flow control, based on the HCI_Number_Of_Completed_Data_Blocks event. The HCI_Read_Data_Block_Size command shall be issued by the Host before it sends any data to the Controller.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Data_Block_Size command succeeded. |
| 0x01 to 0xFF | HCI_Read_Data_Block_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Max_ACL_Data_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum length (in octets) of the data portion of an HCI ACL Data packet that the Controller is able to accept for transmission. |

*Data_Block_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum length (in octets) of the data portion of each HCI ACL Data packet that the Controller is able to hold in each of its data block buffers. |

*Total_Num_Data_Blocks:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total number of data block buffers available in the Controller for the storage of data packets scheduled for transmission. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Data_Block_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.8 Read Local Supported Codecs command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_-Supported_Codecs [v2] | 0x000D | *none* | Status,<br>Num_Supported_Standard_Codecs,<br>Standard_Codec_ID[i],<br>Standard_Codec_Transport[i],<br>Num_Supported_Vendor_Specific_Codecs,<br>Vendor_Specific_Codec_ID[k],<br>Vendor_Specific_Codec_Transport[k] |
| HCI_Read_Local_-Supported_Codecs [v1] | 0x000B | *none* | Status,<br>Num_Supported_Standard_Codecs,<br>Standard_Codec_ID[i],<br>Num_Supported_Vendor_Specific_Codecs,<br>Vendor_Specific_Codec_ID[k] |

The order of the return parameters in an HCI event packet is:

Status
Num_Supported_Standard_Codecs
Standard_Codec ID[0]
Standard_Codec_Transport[0]
. . .
Standard_Codec_ID[m]
Standard_Codec_Transport[m]
Num_Supported_Vendor_Specific_Codecs
Vendor_Specific_Codec ID[0]
Vendor_Specific_Codec_Transport[0]
. . .
Vendor_Specific_Codec ID[n]
Vendor_Specific_Codec_Transport[n]

**Description:**

This command reads a list of the Bluetooth SIG approved codecs supported by the Controller, as well as vendor specific codecs, which are defined by an individual manufacturer. The [v1] version of this command shall only return codecs supported on the BR/EDR physical transport, while the [v2] version shall return codecs supported on all physical transports.

**Command parameters:**

None

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Supported_Codecs command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_Supported_Codecs command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Supported_Standard_Codecs:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Total number of codecs supported |

*Standard_Codec_ID[i]:* *Size: Num_Supported_Standard_Codecs × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Codec identifier. See Assigned Numbers |

*Standard_Codec_Transport[i]:* *Size: Num_Supported_Standard_Codecs × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Codec supported over BR/EDR ACL |
| 1 | Codec supported over BR/EDR SCO and eSCO |
| 2 | Codec supported over LE CIS |
| 3 | Codec supported over LE BIS |
| All other bits | Reserved for future use |

*Num_Supported_Vendor_Specific_Codecs:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Total number of vendor-specific codecs supported |

*Vendor_Specific_Codec_ID[k]: Size: Num_Supported_Vendor_Specific_Codecs × 4 octets*

| Value | Parameter Description |
|---|---|
| Octets 0 and 1 | Company ID, see Assigned Numbers for Company Identifier |
| Octets 2 and 3 | Vendor-defined codec ID |

*Vendor_Specific_Codec_Transport[k]: Size: Num_Supported_Vendor_Specific_Codecs × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Codec supported over BR/EDR ACL |
| 1 | Codec supported over BR/EDR SCO and eSCO |
| 2 | Codec supported over LE CIS |
| 3 | Codec supported over LE BIS |
| All other bits | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Supported_Codecs command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.9 Read Local Simple Pairing Options command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Simple_Pairing_-Options | 0x000C | *none* | Status,<br>Simple_Pairing_Options,<br>Max_Encryption_Key_Size |

**Description:**

This command is used to read the Secure Simple Pairing options and the maximum encryption key size supported. Bit 0 of the Simple_Pairing_Options parameter shall be set to 1.

Note: If this command is supported, then the Controller must support remote public key validation (see [Vol 2] Part H, Section 7.6).

**Command parameters:**

None

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Simple_Pairing_Options command succeeded. |
| 0x01 to 0xFF | HCI_Read_Local_Simple_Pairing_Options command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Simple_Pairing_Options:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Remote public key validation is always performed. |
| All other bits | Reserved for future use. |

*Max_Encryption_Key_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x07 to 0x10 | Maximum encryption key size (in octets) supported. |
| All other values | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Simple_Pairing_Options command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.10 Read Local Supported Codec Capabilities command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Supported_-Codec_Capabilities | 0x000E | Codec_ID,<br>Logical_Transport_Type,<br>Direction | Status,<br>Num_Codec_Capabilities,<br>Codec_Capability_Length[i],<br>Codec_Capability[i] |

**Description:**

This command returns a list of codec capabilities supported by the Controller for a given codec. Only capabilities for the codec specified by the Codec_ID parameter and that match the transport specified by the Logical_Transport_Type parameter and direction specified by the Direction parameter are returned.

Note: The Controller cannot provide more information than will fit in an HCI Event packet. If more capabilities than that are available, it must select which ones to return. How this is done is not specified.

**Command parameters:**

*Codec_ID:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers |
| Octets 1 to 2 | Company ID, see Assigned Numbers for Company Identifier.<br>Shall be ignored if octet 0 is not 0xFF. |
| Octets 3 to 4 | Vendor-defined codec ID.<br>Shall be ignored if octet 0 is not 0xFF. |

*Logical_Transport_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | BR/EDR ACL |
| 0x01 | BR/EDR SCO or eSCO |
| 0x02 | LE CIS |
| 0x03 | LE BIS |
| All other values | Reserved for future use |

*Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Input (Host to Controller) |
| 0x01 | Output (Controller to Host) |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Supported_Codec_Capabilities command succeeded |
| 0x01 to 0xFF | HCI_Read_Local_Supported_Codec_Capabilities command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Num_Codec_Capabilities:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Total number of codec capabilities returned |

*Codec_Capability_Length[i]:* *Size: Num_Codec_Capabilities × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of the Codec_Capability[i] field |

*Codec_Capability[i]:* *Size: SUM(Codec_Capability_Length[i]) octets*

| Parameter Description |
|---|
| Codec_Capability_Length[i] octets of codec-specific capability data<br>Note: Each element of this array has a variable length. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Supported_Codec_Capabilities command has completed, an HCI_Command_Complete event shall be generated.

### 7.4.11 Read Local Supported Controller Delay command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Supported_-Controller_Delay | 0x000F | Codec_ID,<br>Logical_Transport_Type,<br>Direction,<br>Codec_Configuration_Length,<br>Codec_Configuration | Status,<br>Min_Controller_Delay,<br>Max_Controller_Delay |

**Description:**

This command returns the range of supported Controller delays for the codec specified by the Codec_ID parameter on a given transport type specified by the Logical_Transport_Type parameter, in the direction specified by the Direction parameter, and with the codec configuration specified by the Codec_Configuration parameter.

The Min_Controller_Delay and Max_Controller_Delay parameters returned by the Controller provide a range of allowed values to be used by the Host when issuing the HCI_LE_Setup_ISO_Data_Path command.

The Min_Controller_Delay parameter returned by the Controller shall be greater than or equal to the codec processing delay for the specified direction and codec configuration.

The Max_Controller_Delay parameter returned by the Controller shall be less than or equal to the sum of the codec processing delay and the maximum time the Controller can buffer the data for the specified direction and codec configuration. Max_Controller_Delay shall be greater than or equal to Min_Controller_Delay.

**Command parameters:**

*Codec_ID:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding Format |
| Octets 1 to 2 | Company ID, see Assigned Numbers for Company Identifier.<br>Shall be ignored if octet 0 is not 0xFF. |
| Octets 3 to 4 | Vendor-defined codec ID.<br>Shall be ignored if octet 0 is not 0xFF. |

*Logical_Transport_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | BR/EDR ACL |
| 0x01 | BR/EDR SCO or eSCO |
| 0x02 | LE CIS |
| 0x03 | LE BIS |
| All other values | Reserved for future use |

*Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Input (Host to Controller) |
| 0x01 | Output (Controller to Host) |
| All other values | Reserved for future use |

*Codec_Configuration_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of codec configuration |

*Codec_Configuration:* *Size: Codec_Configuration_Length octets*

| Parameter Description |
|---|
| Codec-specific configuration data<br>Note: This parameter has a variable length. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Supported_Controller_Delay command succeeded |
| 0x01 to 0xFF | HCI_Read_Local_Supported_Controller_Delay command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Min_Controller_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Minimum Controller delay in microseconds for the specified configuration<br>Range: 0x000000 to 0x3D0900<br>Time range: 0 s to 4 s |

*Max_Controller_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Maximum Controller delay in microseconds for the specified configuration<br>Range: 0x000000 to 0x3D0900<br>Time range: 0 s to 4 s |

**Event(s) generated (unless masked away):**

When the HCI_Read_Local_Supported_Controller_Delay command has completed, an HCI_Command_Complete event shall be generated.

## 7.5 Status parameters

The Controller modifies all status parameters. These parameters provide information about the current state of the Link Manager and Baseband in the BR/EDR Controller. The Host device cannot modify any of these parameters other than to reset certain specific parameters.

For the status parameters commands, the OGF is defined as 0x05.

### 7.5.1 Read Failed Contact Counter command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Failed_Contact_Counter | 0x0001 | Handle | Status,<br>Handle,<br>Failed_Contact_Counter |

**Description:**

This command reads the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Handle shall be a Connection_Handle for an ACL connection. See Section 6.15.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the Connection for which the Failed Contact Counter should be read.<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Failed_Contact_Counter command succeeded. |
| 0x01 to 0xFF | HCI_Read_Failed_Contact_Counter command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the Failed Contact Counter has been read.<br>Range: 0x0000 to 0x0EFF |

*Failed_Contact_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of consecutive failed contacts for a connection corresponding to the Handle. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Failed_Contact_Counter command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.2 Reset Failed Contact Counter command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reset_Failed_Contact_Counter | 0x0002 | Handle | Status,<br>Handle |

**Description:**

This command resets the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Handle shall be a Connection_Handle for an ACL connection. See Section 6.15.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the Failed Contact Counter should be reset.<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Reset_Failed_Contact_Counter command succeeded. |
| 0x01 to 0xFF | HCI_Reset_Failed_Contact_Counter command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the Failed Contact Counter has been reset.<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_Reset_Failed_Contact_Counter command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.3 Read Link Quality command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Quality | 0x0003 | Handle | Status,<br>Handle,<br>Link_Quality |

**Description:**

This command returns the value for the Link_Quality for the specified Handle. The Handle shall be a Connection_Handle for an ACL connection. This command shall return a Link_Quality value from 0 to 255, which represents the quality of the link between two Controllers. The higher the value, the better the link quality is. Each Bluetooth module vendor will determine how to measure the link quality.

If the Host specifies a connection handle for an LE ACL connection and the Controller does not support the Connected Isochronous Stream - Central or Connected Isochronous Stream - Peripheral feature (see [Vol 6] Part B, Section 4.6), the Controller shall either begin to execute the command or return an error.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which link quality parameters are to be read.<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Link_Quality command succeeded. |
| 0x01 to 0xFF | HCI_Read_Link_Quality command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the link quality parameter has been read.<br>Range: 0x0000 to 0x0EFF |

*Link_Quality:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The current quality of the Link connection between the local device and the remote device specified by the Handle.<br>Range: 0x00 to 0xFF<br>The higher the value, the better the link quality is. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Link_Quality command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.4 Read RSSI command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_RSSI | 0x0005 | Handle | Status,<br>Handle,<br>RSSI |

**Description:**

This command reads the Received Signal Strength Indication (RSSI) value from a Controller.

For a BR/EDR Controller, the RSSI parameter returns the difference between the measured Received Signal Strength Indication (RSSI) and the limits of a range selected by the Controller. The lower limit shall correspond to a received power not less than -56 dBm and not greater than 6 dB above the actual sensitivity of the receiver. The upper limit shall be 20±6 dB above the lower limit. A positive RSSI value shall indicate how many dB the RSSI is above the upper limit, a negative value shall indicate how many dB the RSSI is below the lower limit, and zero shall indicate that the RSSI is inside the range.

The returned RSSI value is not required to have any specific accuracy provided that it correctly indicates whether the received signal strength was above the upper limit, below the lower limit, or between the limits.

For an LE transport, the RSSI parameter returns the absolute received signal strength value in dBm to ±6 dB accuracy. If the RSSI cannot be read, the RSSI parameter shall be set to 127.

**Command parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the RSSI is to be read.<br>The Handle is a Connection_Handle for an ACL-U or LE-U connection.<br>Range: 0x0000 to 0x0EFF |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_RSSI command succeeded. |
| 0x01 to 0xFF | HCI_Read_RSSI command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Handle for the connection for which the RSSI has been read.<br>Range: 0x0000 to 0x0EFF |

*RSSI:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | BR/EDR<br>Range: -128 to 127<br>Units: dB<br>LE:<br>Range: -127 to 20, 127<br>Units: dBm |

### Event(s) generated (unless masked away):

When the HCI_Read_RSSI command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.5 Read AFH Channel Map command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_AFH_Channel_Map | 0x0006 | Connection_Handle | Status,<br>Connection_Handle,<br>AFH_Mode,<br>AFH_Channel_Map |

**Description:**

This command returns the values for the AFH_Mode and AFH_Channel_Map for the specified Connection_Handle. The Connection_Handle shall be a Connection_Handle for an ACL connection.

The returned values indicate the state of the hop sequence specified by the most recent LMP_SET_AFH message for the specified Connection_Handle, regardless of whether the Central has received the Baseband acknowledgment or whether the AFH_Instant has passed.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_AFH_Channel_Map command succeeded. |
| 0x01 to 0xFF | HCI_Read_AFH_Channel_Map command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*AFH_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AFH is disabled. |
| 0x01 | AFH is enabled. |
| All other values | Reserved for future use. |

*AFH_Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXX | If AFH_Mode is not AFH enabled then the contents of this parameter are reserved for future use. Otherwise:<br>This parameter contains 80 1-bit fields.<br>The $n^{th}$such field (in the range 0 to 78) contains the value for channel *n*:<br>0: channel *n* is unused<br>1: channel *n* is used<br>The most significant bit (bit 79) is reserved for future use |

### Event(s) generated (unless masked away):

When the HCI_Read_AFH_Channel_Map command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.6 Read Clock command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Clock | 0x0007 | Connection_Handle,<br>Which_Clock | Status,<br>Connection_Handle,<br>Clock,<br>Accuracy |

**Description:**

This command reads the estimate of the value of the Bluetooth Clock from the BR/EDR Controller.

If the Which_Clock value is 0, then the Connection_Handle shall be ignored, the local Bluetooth Clock value shall be returned and the accuracy parameter shall be set to 0.

If the Which_Clock value is 1, then the Connection_Handle shall be a valid ACL Connection_Handle. If the current role of this ACL connection is Central, then the Bluetooth Clock of this device shall be returned. If the current role is Peripheral, then an estimate of the Bluetooth Clock of the remote Central and the accuracy of this value shall be returned.

The accuracy reflects the clock drift that might have occurred since the Peripheral last received a valid transmission from the Central.

Note: The Bluetooth Clock has a minimum accuracy of 250 ppm, or about 22 seconds drift in one day.

Note: See [Vol 2] Part B, Section 1.1 for more information about the Bluetooth Clock.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Which_Clock:* *Size 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | 0x00 = Local Clock (Connection_Handle does not have to be valid)<br>0x01 = Piconet Clock (Connection_Handle shall be valid)<br>0x02 to 0xFF = Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Clock command succeeded. |
| 0x01 to 0xFF | HCI_Read_Clock command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets(12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection_Handle for the connection for which the Central's clock has been read. If the Which_Clock parameter was 0, then the Connection_Handle is reserved for future use.<br>Range: 0x0000 to 0x0EFF |

*Clock:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Bluetooth Clock of the device requested. |

*Accuracy:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum (absolute) error in the Bluetooth Clock. Value of 0xFFFF means Unknown.<br>Accuracy = ±N × 0.3125 ms (1 Bluetooth Clock)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 to 20479.375 ms |

**Event(s) generated (unless masked away):**

When the HCI_Read_Clock command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.7 Read Encryption Key Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Encryption_Key_Size | 0x0008 | Connection_Handle | Status,<br>Connection_Handle,<br>Key_Size |

**Description:**

This command reads the current encryption key size associated with the Connection_Handle. The Connection_Handle shall be a Connection_Handle for an active ACL connection.

All BR/EDR Controllers shall implement this command.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Encryption_Key_Size succeeded |
| 0x01 to 0xFF | Read_Encryption_Key_Size failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Key_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Encryption key size in octets. See [Vol 2] Part C, Section 5.2.<br>Range: 0x07 to 0x10 |

**Event(s) generated (unless masked away):**

When the HCI_Read_Encryption_Key_Size command has completed, an HCI_Command_Complete event shall be generated.

If the ACL connection associated with the Connection_Handle is not encrypted, the Controller shall return an HCI_Command_Complete event with the error code *Insufficient Security* (0x2F).

### 7.5.8 [This section is no longer used]

### 7.5.9 [This section is no longer used]

### 7.5.10 [This section is no longer used]

### 7.5.11 Get MWS Transport Layer Configuration command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Get_MWS_Transport_Layer_-Configuration | 0x000C | *none* | Status,<br>Num_Transports,<br>Transport_Layer[i],<br>Num_Baud_Rates[i],<br>To_MWS_Baud_Rate[k],<br>From_MWS_Baud_Rate[k] |

The order of the return parameters in an HCI event packet is:

Status
Num_Transports
Transport_Layer[0]
Num_Baud_Rates[0]
. . .
Transport_Layer[n]
Num_Baud_Rates[n]
To_MWS_Baud_Rate[0]
From_MWS_Baud_Rate[0]
. . .
To_MWS_Baud_Rate[m]
From_MWS_Baud_Rate[m]

**Description:**

This command is used to inform the Host of the Baud rates supported by the Controller for the transport layer.

The Num_Transports parameter is used to indicate the number of MWS coexistence transport interfaces supported by the Controller.

The Num_Baud_Rates[i] parameter indicates the number of supported baud rates for each transport.

The To_MWS_Baud_Rate[k] parameters indicate the supported baud rates in the direction from Bluetooth to MWS for each transport.

The From_MWS_Baud_Rate[k] parameters indicate the supported baud rates in the direction from MWS to Bluetooth for each transport.

If one direction has more supported rates than the other direction, the Controller shall - in the direction with less supported rates - fill with sufficient zeros to produce the same number of values. The rates for the two directions are not necessarily paired.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Get_MWS_Transport_Layer_Configuration command succeeded. |
| 0x01 to 0xFF | HCI_Get_MWS_Transport_Layer_Configuration command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

*Num_Transports:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of supported MWS coexistence transport layers. |

*Transport_Layer[i]:* *Size: Num_Transports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers. |

*Num_Baud_Rates[i]:* *Size: Num_Transports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of different baud rates supported for one transport. |

*To_MWS_Baud_Rate[k]:* *Size: SUM (Num_Baud_Rates [i])× 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | A supported Baud rate in the Bluetooth Controller to MWS Device direction in Baud. The list shall start with the first baud rate for the first transport, followed by the remaining baud rates for the first transport, followed by the baud rates for the second transport (if any), followed by baud rates for subsequent transports (if any). |

*From_MWS_Baud_Rate[k]:* *Size: SUM (Num_Baud_Rates[i]) × 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | A supported Baud rate in the Bluetooth Controller for signals in the MWS to Bluetooth Controller Device direction in Baud. The list shall start with the first baud rate for the first transport, followed by the remaining baud rates for the first transport, followed by the baud rates for the second transport (if any), followed by baud rates for subsequent transports (if any). |

**Event(s) generated (unless masked away):**

When the HCI_Get_MWS_Transport_Layer_Configuration command has completed, an HCI_Command_Complete event shall be generated.

### 7.5.12 Set Triggered Clock Capture command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Triggered_Clock_-Capture | 0x000D | Connection_Handle,<br>Enable,<br>Which_Clock,<br>LPO_Allowed,<br>Num_Clock_Captures_To_Filter | Status |

**Description:**

This command configures the BR/EDR Controller for triggered clock capturing.

Triggered clock capturing is enabled or disabled by the Enable parameter. If Enable is set to 0x00, all the other parameters (including Connection_Handle) shall be ignored.

If the Which_Clock value is 0, then the Connection_Handle shall be ignored. If the Which_Clock value is 1, then the Connection_Handle shall be a valid ACL Connection_Handle.

The LPO_Allowed parameter informs the BR/EDR Controller whether it may use a lower accuracy clock or not.

The Num_Clock_Captures_To_Filter parameter is used to filter triggered clock captures between sending HCI_Triggered_Clock_Capture events to the Host. When set to zero, all triggered clock captures shall result in an HCI_Triggered_Clock_Capture event sent to the Host. When set to a non-zero value, after every HCI_Triggered_Clock_Capture event, Num_Clock_Captures_To_Filter triggered clock captures in a row shall not trigger an event to be sent to the Host.

An implementation should ensure that the rate of triggered clock captures does not overwhelm the HCI event queue and processing.

Note: See [Vol 2] Part B, Section 1.1 for more information about the Bluetooth Clock.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable triggered clock capturing on the specified Connection_Handle (default) |
| 0x01 | Enable triggered clock capturing on the specified Connection_Handle |
| All other values | Reserved for future use |

*Which_Clock:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Local Clock |
| 0x01 | Piconet Clock for the specified connection |
| All other values | Reserved for future use |

*LPO_Allowed:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Controller shall not sleep (that is, clock accuracy shall be equal to or better than ± 20 ppm) |
| 0x01 | Controller may sleep (that is, clock accuracy shall be equal to or better than ± 250 ppm) |
| All other values | Reserved for future use |

*Num_Clock_Captures_To_Filter:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | All triggered clock captures result in an HCI_Triggered_Clock_Capture event sent to the Host |
| 0x01 to 0xFF | Number of triggered clock captures filtered between sending an HCI_Triggered_Clock_Capture event to the Host. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Set_Triggered_Clock_Capture command succeeded. |
| 0x01 to 0xFF | HCI_Set_Triggered_Clock_Capture command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Set_Triggered_Clock_Capture command has completed, an HCI_Command_Complete event shall be sent to the Host.

When Triggered Clock Capturing is enabled, HCI_Triggered_Clock_Capture events are returned until Triggered Clock Capturing is disabled.

## 7.6 Testing commands

The Testing commands are used to provide the ability to test various functional capabilities of the Bluetooth hardware. These commands provide the ability to arrange various conditions for testing.

For the Testing commands, the OGF is defined as 0x06.

### 7.6.1 Read Loopback Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Loopback_Mode | 0x0001 | *none* | Status,<br>Loopback_Mode |

**Description:**

This command reads the value for the setting of the Controller's Loopback mode. The setting of the Loopback_Mode parameter shall determine the path of information. In Non-testing Mode operation, the Loopback_Mode parameter is set to Non-testing Mode and the path of the information is as specified by the Bluetooth specifications. In Local Loopback Mode, every data packet (ACL, SCO and eSCO) and command packet that is sent from the Host to the Controller is sent back with no modifications by the Controller, as shown in Figure 7.1. For details of loopback modes see Section 7.6.2.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Loopback_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Read_Loopback_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Loopback_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Loopback mode enabled (default). |
| 0x01 | Enable Local Loopback. |

| Value | Parameter Description |
|---|---|
| 0x02 | Enable Remote Loopback. |
| All other values | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_Read_Loopback_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.6.2 Write Loopback Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Loopback_Mode | 0x0002 | Loopback_Mode | Status |

**Description:**

This command writes the value for the setting of the BR/EDR Controller's Loopback mode. The setting of the Loopback_Mode parameter shall determine the path of information. In Non-testing mode operation, the Loopback_Mode parameter is set to Non-testing mode and the path of the information as specified by the Bluetooth specifications. In Local Loopback mode, every data packet (ACL, SCO and eSCO) and command packet that is sent from the Host to the BR/EDR Controller is sent back with no modifications by the BR/EDR Controller, as shown in Figure 7.1.

When the BR/EDR Controller enters Local Loopback mode, it shall respond with one to four HCI_Connection_Complete events, one for an ACL connection and zero to three for synchronous connections. The Host should use the Connection_Handles from these events when sending data in Local Loopback mode. The number of Connection_Handles returned for synchronous connections is implementation specific. When in Local Loopback mode, the BR/EDR Controller loops back commands and data to the Host. The HCI_Loopback_Command event is used to loop back commands that the Host sends to the Controller.

If the implementation does not support sending synchronous data over HCI, then it is vendor-specific how this data is sent to the Controller and looped back.

There are some commands that are not looped back in Local Loopback mode: HCI_Reset, HCI_Set_Controller_To_Host_Flow_Control, HCI_Host_Buffer_Size, HCI_Host_Number_Of_Completed_Packets, HCI_Read_Buffer_Size, HCI_Read_Loopback_Mode and HCI_Write_Loopback_Mode. These commands should be executed in the way they are normally executed. The commands HCI_Reset and HCI_Write_Loopback_Mode can be used to exit Local Loopback mode.

If HCI_Write_Loopback_Mode is used to exit Local Loopback mode on a BR/EDR Controller, HCI_Disconnection_Complete events corresponding to the HCI_Connection_Complete events that were sent when entering Local Loopback mode should be sent to the Host. Furthermore, no connections are allowed in Local Loopback mode. If there is a connection, and there is an attempt to set the device to Local Loopback mode, the attempt will be refused. When the device is in Local Loopback mode, the Controller will refuse incoming connection attempts. This allows the Host BR/EDR Controller Transport Layer to be tested without any other variables.

If a BR/EDR Controller is set to Remote Loopback mode, it will send back all data (ACL, SCO and eSCO) that comes over the air. It will only allow a maximum of one

ACL connection and three synchronous connections, and these shall all be to the same remote device. If there are existing connections to a remote device and there is an attempt to set the local device to Remote Loopback mode, the attempt shall be refused.

See Figure 7.2, where the rightmost device is set to Remote Loopback mode and the leftmost device is set to Non-testing mode. This allows the BR/EDR Air link to be tested without any other variables.

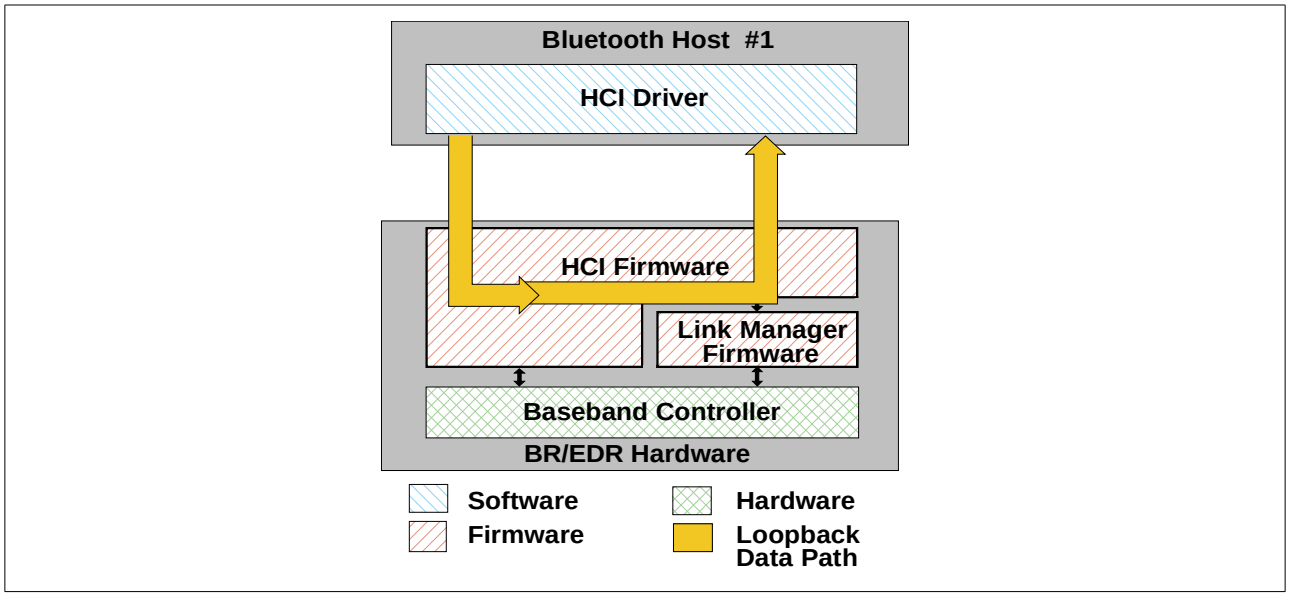



*Figure 7.1: Local Loopback mode*

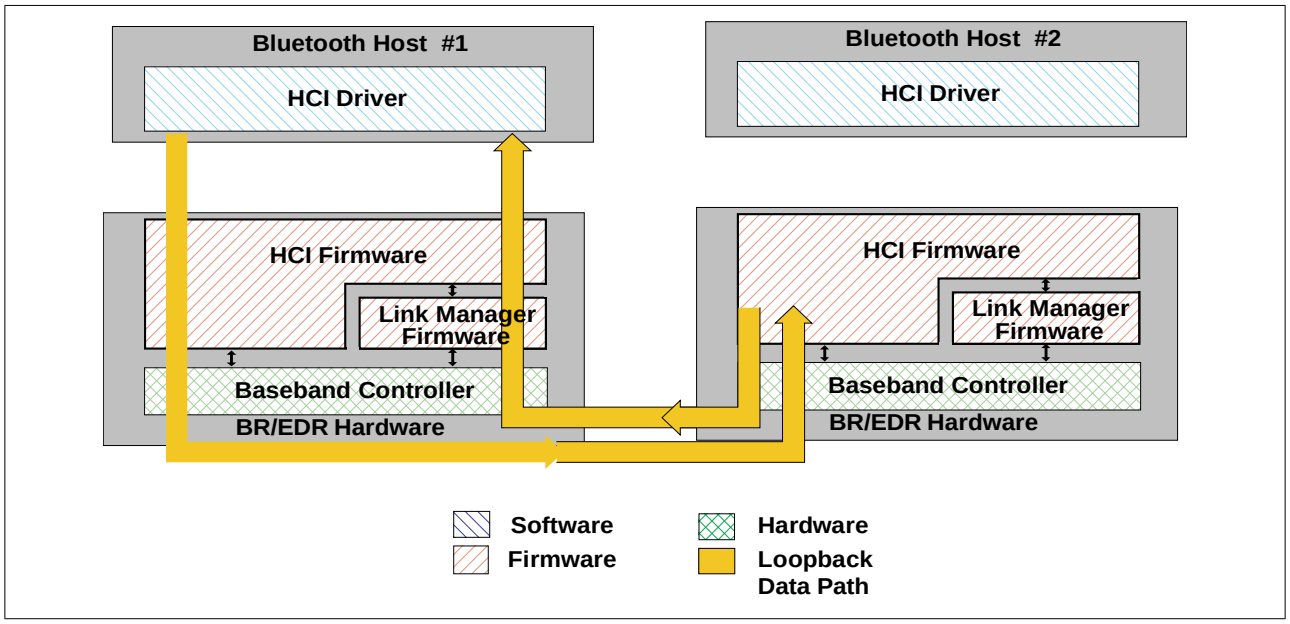



*Figure 7.2: Remote Loopback mode*

### Command parameters:

*Loopback_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Loopback mode enabled (default). |
| 0x01 | Enable Local Loopback. |
| 0x02 | Enable Remote Loopback. |
| All other values | Reserved for future use. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Loopback_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Loopback_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_Write_Loopback_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.6.3 Enable Implementation Under Test Mode command[1]

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enable_-Implementation_Under_Test_Mode | 0x0003 | *none* | Status |

**Description:**

This command allows the local BR/EDR Controller to enter test mode via LMP test commands for BR/EDR Controllers. For details see [Vol 2] Part C, Section 4.7. The Host issues this command to indicate that the local device acts as the IUT for the testing scenarios described in [Vol 3] Part D, Section 1. When the BR/EDR Controller receives this command, it shall complete the command with an HCI_Command_Complete event. The BR/EDR Controller functions as normal until the Lower Tester issues the LMP test command to place the local device into Implementation Under Test mode. To disable and exit the Implementation Under Test Mode, the Host may issue the HCI_Reset command. The local BR/EDR Controller shall not enter test mode, even if instructed by the remote BR/EDR Controller, before this command is issued.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Enable_Implementation_Under_Test_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Enable_Implementation_Under_Test_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_Enable_Implementation_Under_Test_Mode command has completed, an HCI_Command_Complete event shall be generated.

[1]This command was formerly called “Enable Device Under Test Mode”.

### 7.6.4 Write Simple Pairing Debug Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Simple_Pairing_Debug_-Mode | 0x0004 | Simple_Pairing_Debug_Mode | Status |

**Description:**

This command configures the BR/EDR Controller to use a predefined Diffie Hellman private key for Secure Simple Pairing to enable debug equipment to monitor the encrypted connection.

Note: Only one side (initiator or responder) needs to set Secure Simple Pairing debug mode in order for debug equipment to be able to determine the link key and, therefore, be able to monitor the encrypted connection.

When the Simple_Pairing_Debug_Mode parameter is set to enabled the BR/EDR Controller shall use the predefined Diffie Hellman private key. The BR/EDR Controller shall also set the resulting Link_Key type to "Debug Combination Key."

When in Secure Simple Pairing debug mode, the Link Manager shall use the following Diffie Hellman private / public key pairs:

**For P-192:**

Private key: `07915f86918ddc27005df1d6cf0c142b625ed2eff4a518ff`

Public key (X): `15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed`

Public key (Y): `b09d42b81bc5bd009f79e4b59dbbaa857fca856fb9f7ea25`

**For P-256:**

Private key: `3f49f6d4 a3c55f38 74c9b3e3 d2103f50 4aff607b eb40b799 5899b8a6 cd3c1abd`

Public key (X): `20b003d2 f297be2c 5e2c83a7 e9f9a5b9 eff49111 acf4fddb cc030148 0e359de6`

Public key (Y): `dc809c49 652aeb6d 63329abf 5a52155c 766345c2 8fed3024 741c8ed0 1589d28b`

### Command parameters:

*Simple_Pairing_Debug_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure Simple Pairing debug mode disabled (default) |
| 0x01 | Secure Simple Pairing debug mode enabled |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Simple_Pairing_Debug_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Simple_Pairing_Debug_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_Write_Simple_Pairing_Debug_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.6.5 [This section is no longer used]

### 7.6.6 [This section is no longer used]

### 7.6.7 [This section is no longer used]

### 7.6.8 Write Secure Connections Test Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Secure_Connections_Test_-Mode | 0x000A | Connection_Handle,<br>DM1_ACL-U_Mode,<br>eSCO_Loopback_Mode | Status,<br>Connection_Handle |

**Description:**

This command configures the BR/EDR Controller to enable and disable the two test modes used for verifying the Secure Connections feature during qualification. The DM1_ACL-U_Mode parameter enables and disables the use of DM1 packets for transmitting ACL-U data. When DM1 ACL-U Mode is disabled, ACL-U traffic may use DM1 packets. When DM1 ACL-U Mode is enabled, ACL-U traffic shall not use DM1 packets unless the Packet_Type parameter only allows DM1 packets (e.g. set to 0x3306 or 0x330E).

The command is used during testing to help make transmit ACL packet selection predictable.

The eSCO_Loopback_Mode parameter enables and disables the loopback of received eSCO payloads. When the eSCO_Loopback_Mode parameter is set to Enabled, the BR/EDR Controller will send back all eSCO data that comes over the air irrespective of whether the CRC check in the received eSCO packet passes or fails. It will only allow one synchronous connection. If there is more than one synchronous connection and there is an attempt to set the local device to eSCO_Loopback_Mode, the attempt shall be refused.

See Figure 7.3, where the rightmost device has the eSCO_Loopback_Mode parameter set to enabled and the leftmost device is in a normal mode of operation. This allows the encryption and decryption of eSCO packets to be tested without any other variables.

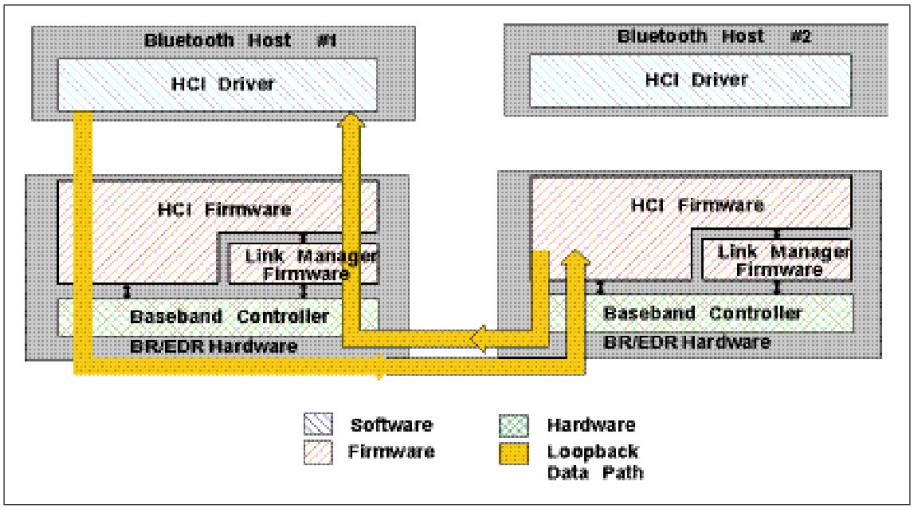



*Figure 7.3: Secure Connections eSCO Loopback*

The Connection_Handle shall be for an ACL connection.

When the eSCO_Loopback_Mode parameter is set to enabled, received eSCO payloads are looped back as subsequent transmitted eSCO payloads. There may be a delay of 0 or more eSCO intervals before the Controller loops back the payload. This is illustrated in Figure 7.4, Figure 7.5, and Figure 7.6.

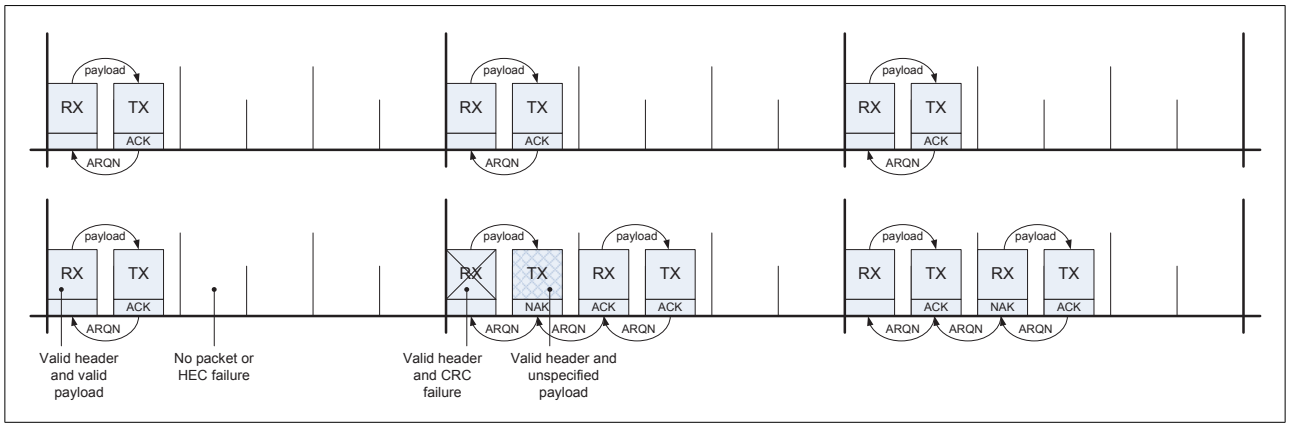



*Figure 7.4: Secure Connections eSCO loopback immediate*

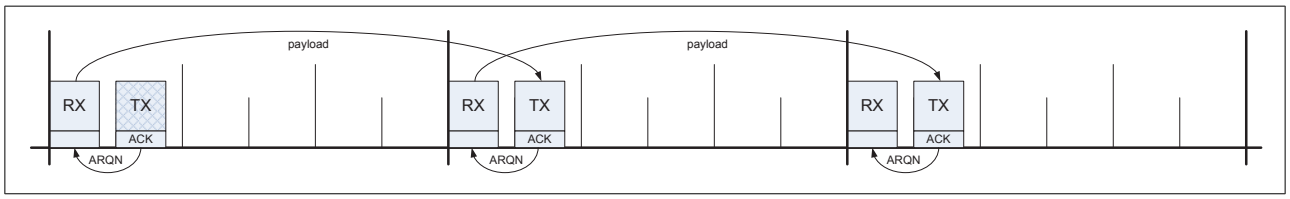



*Figure 7.5: Secure Connections eSCO loopback delayed*

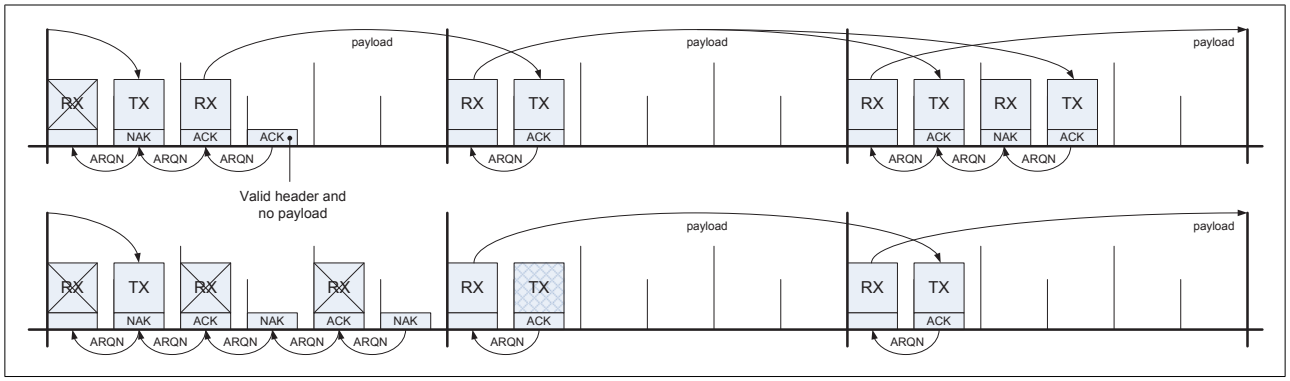



*Figure 7.6: Secure Connections eSCO loopback delayed with retransmissions*

## Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*DM1_ACL-U_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | DM1 ACL-U mode disabled (default) |
| 0x01 | DM1 ACL-U mode enabled |
| All other values | Reserved for future use |

*eSCO_Loopback_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | eSCO loopback mode disabled (default) |
| 0x01 | eSCO loopback mode enabled |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Write_Secure_Connections_Test_Mode command succeeded. |
| 0x01 to 0xFF | HCI_Write_Secure_Connections_Test_Mode command failed.<br>See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_Write_Secure_Connections_Test_Mode command has completed, an HCI_Command_Complete event shall be generated.

## 7.7 Events

### 7.7.1 Inquiry Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Inquiry_Complete | 0x01 | Status |

**Description:**

This event indicates that the Inquiry is finished. This event contains a Status parameter, which is used to indicate if the Inquiry completed successfully or if the Inquiry was not completed.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Inquiry command completed successfully. |
| 0x01 to 0xFF | HCI_Inquiry command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### 7.7.2 Inquiry Result event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Inquiry_Result | 0x02 | Num_Responses,<br>BD_ADDR[i],<br>Page_Scan_Repetition_Mode[i],<br>Reserved[i],<br>Class_Of_Device[i]<br>Clock_Offset[i] |

**Description:**

This event indicates that a BR/EDR Controller or multiple BR/EDR Controllers have responded so far during the current Inquiry process. This event will be sent from the BR/EDR Controller to the Host as soon as an Inquiry Response from a remote device is received if the remote device supports only mandatory paging scheme. The BR/EDR Controller may queue these Inquiry Responses and send multiple BR/EDR Controllers information in one HCI_Inquiry_Result event. The event can be used to return one or more Inquiry responses in one event.

This event is only generated if the Inquiry_Mode parameter of the last HCI_Write_Inquiry_Mode command was set to 0x00 (Standard Inquiry Result event format) or if the HCI_Write_Inquiry_Mode command has not been used.

**Event parameters:**

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of responses from the Inquiry. |

*BD_ADDR[i]:* *Size: Num_Responses × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for a device which responded. |

*Page_Scan_Repetition_Mode[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |

| Value | Parameter Description |
|---|---|
| 0x02 | R2 |
| All other values | Reserved for future use |

*Reserved[i]:* *Size: Num_Responses × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Reserved for future use. |

*Class_Of_Device[i]:* *Size: Num_Responses × 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device |

*Clock_Offset[i]:* *Size: Num_Responses × 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0-14 | Bits 16-2 of CLKNPeripheral - CLK |
| 15 | Reserved for future use |

### 7.7.3 Connection Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connection_Complete | 0x03 | Status,<br>Connection_Handle,<br>BD_ADDR,<br>Link_Type,<br>Encryption_Enabled |

**Description:**

This event indicates to both of the Hosts forming the connection that a new connection has been established. This event also indicates to the Host which issued the HCI_Create_Connection, HCI_Accept_Connection_Request, or HCI_Reject_Connection_Request command, and then received an HCI_Command_Status event, if the issued command failed or was successful.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01 to 0xFF | Connection failed to Complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle for the new connection<br>Range: 0x0000 to 0x0EFF |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the other connected Device forming the connection. |

*Link_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO connection. |
| 0x01 | ACL connection (Data Channels). |
| All other values | Reserved for future use. |

*Encryption_Enabled:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link level encryption disabled. |
| 0x01 | Link level encryption enabled. |
| All other values | Reserved for future use. |

### 7.7.4 Connection Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connection_Request | 0x04 | BD_ADDR,<br>Class_Of_Device,<br>Link_Type |

**Description:**

This event is used to indicate that a new incoming connection is trying to be established. The connection may either be accepted or rejected. When the Host receives this event and the link type parameter is ACL, it should respond with either an HCI_Accept_Connection_Request or HCI_Reject_Connection_Request command before the timer Connection_Accept_Timeout expires. If the link type is SCO or eSCO, the Host should reply with the HCI_Accept_Synchronous_Connection_Request command, the HCI_Enhanced_Accept_Synchronous_Connection_Request command, or the HCI_Reject_Synchronous_Connection_Request command. If the link type is SCO, the Host may respond with the HCI_Accept_Connection_Request command. If the event is responded to with the HCI_Accept_Connection_Request command, then the default parameter settings of the HCI_Accept_Synchronous_Connection_Request command (see Section 7.1.27) should be used by the local Link Manager when negotiating the SCO link parameters. In that case, the HCI_Connection_Complete event and not the HCI_Synchronous_Connection_Complete event, shall be returned on completion of the connection.

Note: See Section 7.3.3 for the behavior when the HCI_Connection_Request event is masked or the connection is auto accepted.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device that requests the connection. |

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device, which requests the connection. |
| 0x000000 | Unknown Class of Device |

*Link_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO connection requested |
| 0x01 | ACL connection requested |
| 0x02 | eSCO connection requested |
| All other values | Reserved for future use. |

### 7.7.5 Disconnection Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Disconnection_Complete | 0x05 | Status,<br>Connection_Handle,<br>Reason |

**Description:**

This event occurs when a connection is terminated. The status parameter indicates if the disconnection was successful or not. The reason parameter indicates the reason for the disconnection if the disconnection was successful. If the disconnection was not successful, then the value of the reason parameter shall be ignored by the Host. For example, this can be the case if the Host has issued the HCI_Disconnect command and there was a parameter error, or the command was not presently allowed, or a Connection_Handle that didn't correspond to a connection was given.

If Connection_Handle identifies a CIS on the Central, then the handle and the associated data paths of that CIS shall remain valid (irrespective of whether the disconnection was successful or not). If Connection_Handle identifies a CIS on the Peripheral and Status is zero, then the handle and the associated data path of that CIS shall be deleted. If Connection_Handle identifies any other kind of connection and Status is zero, then the handle shall be deleted. If Connection_Handle identifies an LE ACL connection and Status is zero, then any associated CS configurations shall also be deleted.

Note: When a physical link fails, one HCI_Disconnection_Complete event will be returned for each logical channel on the physical link with the corresponding Connection_Handle as a parameter.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disconnection has occurred. |
| 0x01 to 0xFF | Disconnection failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle which was disconnected.<br>Range: 0x0000 to 0x0EFF |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason for disconnection. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

### 7.7.6 Authentication Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Authentication_Complete | 0x06 | Status,<br>Connection_Handle |

**Description:**

This event occurs when authentication has been completed for the specified connection. The Connection_Handle shall be a Connection_Handle for an ACL connection.

Note: This event is only generated on the initiator of the authentication and not on the responder.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication Request successfully completed. |
| 0x01 to 0xFF | Authentication Request failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### 7.7.7 Remote Name Request Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Remote_Name_Request_Complete | 0x07 | Status,<br>BD_ADDR,<br>Remote_Name |

**Description:**

This event is used to indicate that a remote name request has been completed. The Remote_Name parameter is a UTF-8 encoded string with the type utf8s{248} defined in [Vol 1] Part E, Section 2.9.3. The BD_ADDR parameter is used to identify which device the user-friendly name was obtained from.

Note: The Remote_Name parameter is a string parameter. Endianness does therefore not apply to the Remote_Name parameter. The first octet of the name is received first.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Remote_Name_Request command succeeded. |
| 0x01 to 0xFF | HCI_Remote_Name_Request command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device whose name was requested. |

*Remote_Name:* *Size: 248 octets*

| Value | Parameter Description |
|---|---|
| Name[248] | A UTF-8 encoded user-friendly descriptive name for the remote device with type utf8s{248}. |

### 7.7.8 Encryption Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Encryption_Change [v2] | 0x59 | Status,<br>Connection_Handle,<br>Encryption_Enabled,<br>Encryption_Key_Size |
| HCI_Encryption_Change [v1] | 0x08 | Status,<br>Connection_Handle,<br>Encryption_Enabled |

**Description:**

This event is used to indicate that the change of the encryption mode has been completed. The Connection_Handle parameter will be a Connection_Handle for an ACL connection and is used to identify the remote device. The Encryption_Enabled parameter specifies the new encryption state for the connection specified by Connection_Handle. The Encryption_Key_Size parameter specifies the size, in octets, of the key used to encrypt the link. This event will occur on both devices to notify the Hosts when encryption has changed for all connections between the two devices. This event shall not be generated if encryption is paused or resumed; during a role switch, for example.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | An encryption change has occurred. |
| 0x01 to 0xFF | An attempt to change encryption failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Encryption_Enabled:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link Level Encryption is OFF. |
| 0x01 | Link Level Encryption is ON with E0 for BR/EDR.<br>Link Level Encryption is ON with AES-CCM for LE. |
| 0x02 | Link Level Encryption is ON with AES-CCM for BR/EDR. |
| All other values | Reserved for future use. |

*Encryption_Key_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Encryption key size in octets.<br>This parameter shall be ignored for LE connections and shall be ignored when Link Level Encryption is OFF.<br>Range: 0x07 to 0x10 |

### 7.7.9 Change Connection Link Key Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Change_Connection_Link_Key_Complete | 0x09 | Status,<br>Connection_Handle |

**Description:**

This event is used to indicate that the change in the Link Key for all connections to a given remote BR/EDR Controller has been completed.

The Connection_Handle will be a Connection_Handle for an ACL connection to the remote Controller. The HCI_Change_Connection_Link_Key_Complete event is sent only to the Host which issued the HCI_Change_Connection_Link_Key command.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Change_Connection_Link_Key command succeeded. |
| 0x01 to 0xFF | HCI_Change_Connection_Link_Key command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### 7.7.10 Link Key Type Changed event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Link_Key_Type_Changed | 0x0A | Status,<br>Connection_Handle,<br>Key_Flag |

**Description:**

This event is used to indicate that the Link Key managed by the Central of the piconet has been changed. The Connection_Handle will be a Connection_Handle for an ACL connection within that piconet. The link key used for the connection will be the temporary link key or the semi-permanent link key indicated by the Key_Flag. The Key_Flag parameter is used to indicate which Link Key (temporary link key or the semi-permanent link keys) is now being used in the piconet.

This event is also generated on the local Controller when the HCI_Link_Key_Selection command finishes because no change was requested or an error occurred. If a key change was attempted but failed, the remote Controller may generate the event.

Note: For a Central, the change from a semi-permanent Link Key to temporary Link Key will affect all Connection_Handles related to the piconet. For a Peripheral, this change affects only this particular Connection_Handle. A temporary link key must be used when both broadcast and point-to-point traffic are being encrypted (see [Vol 2] Part H, Section 3.2.6 and [Vol 2] Part H, Section 4.2).

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Link_Key_Selection command succeeded. |
| 0x01 to 0xFF | HCI_Link_Key_Selection command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Key_Flag:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Using Semi-permanent Link Key. |
| 0x01 | Using Temporary Link Key. |

### 7.7.11 Read Remote Supported Features Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Read_Remote_Supported_Features_Complete | 0x0B | Status,<br>Connection_Handle,<br>LMP_Features |

**Description:**

This event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote BR/EDR Controller specified by the Connection_Handle parameter. Connection_Handle will be a Connection_Handle for an ACL connection. The LMP_Features parameter specifies the LMP features supported by the remote Controller. For details see [Vol 2] Part C, Section 3.3.

If the features are requested more than once while a connection exists between the two devices, the second and subsequent requests may report a cached copy of the features rather than fetching the feature mask again.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Remote_Supported_Features command succeeded. |
| 0x01 to 0xFF | HCI_Read_Remote_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*LMP_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit Mask List of LMP features. See [Vol 2] Part C, Section 3.3. |

### 7.7.12 Read Remote Version Information Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Read_Remote_Version_Information_Complete | 0x0C | Status,<br>Connection_Handle,<br>Version,<br>Company_Identifier,<br>Subversion |

**Description:**

This event is used to indicate the completion of the process obtaining the version information of the remote Controller specified by the Connection_Handle parameter. Connection_Handle shall be for an ACL connection.

The Version parameter defines the specification version of the BR/EDR or LE Controller. The Company_Identifier parameter indicates the manufacturer of the remote Controller. The Subversion parameter is controlled by the manufacturer and is vendor-specific. These parameters shall contain the same values as the CompanyID and SubVersion parameters in [Vol 2] Part C, Section 4.3.3 and [Vol 6] Part B, Section 2.4.2.13.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Remote_Version_Information command succeeded. |
| 0x01 to 0xFF | HCI_Read_Remote_Version_Information command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Version:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Version of the Current LMP or Link Layer supported by the remote Controller. See Assigned Numbers. |

*Company_Identifier:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Company identifier for the manufacturer of the remote Controller. See Assigned Numbers. |

*Subversion:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Revision of the LMP or Link Layer implementation in the remote Controller. This value is vendor-specific. |

### 7.7.13 QoS Setup Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_QoS_Setup_Complete | 0x0D | Status,<br>Connection_Handle,<br>Unused,<br>Service_Type,<br>Token_Rate,<br>Peak_Bandwidth,<br>Latency,<br>Delay_Variation |

**Description:**

This event is used to indicate the completion of the process of the Link Manager setting up QoS with the remote BR/EDR Controller specified by the Connection_Handle parameter. Connection_Handle will be a Connection_Handle for an ACL connection. For more detail see [Vol 3] Part A, Section 5.3.

The Unused parameter is reserved for future use.

This event is generated following an HCI_QoS_Setup command issued by the local Host.

Note: This event or the HCI_Flow_Specification_Complete event can be generated if the remote device performs an LMP transaction involving the flow parameter values.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_QoS_Setup command succeeded. |
| 0x01 to 0xFF | HCI_QoS_Setup command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | This value shall be used by the Controller. |
| All other values | Reserved for future use. |

*Service_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic Available. |
| 0x01 | Best Effort Available. |
| 0x02 | Guaranteed Available. |
| All other values | Reserved for future use. |

*Token_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Token Rate, in octets per second. |

*Peak_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Peak Bandwidth, in octets per second. |

*Latency:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Latency, in microseconds. |

*Delay_Variation:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Delay Variation, in microseconds. |

### 7.7.14 Command Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Command_Complete | 0x0E | Num_HCI_Command_Packets,<br>Command_Opcode,<br>Return_Parameters |

**Description:**

This event is used by the Controller for most commands to transmit return status of a command and the other event parameters that are specified for the issued HCI command.

The Num_HCI_Command_Packets parameter allows the Controller to indicate the number of HCI command packets the Host can send to the Controller. If the Controller requires the Host to stop sending commands, Num_HCI_Command_Packets will be set to zero. To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller generates an HCI_Command_Complete event with the Command_Opcode parameter set to 0x0000 and Num_HCI_Command_Packets set to 1 or more. Command_Opcode 0x0000 is a special value indicating that this event is not associated with a command sent by the Host. The Controller can send an HCI_Command_Complete event with Command Opcode 0x0000 at any time to change the number of outstanding HCI command packets that the Host can send before waiting. See each command for the parameters that are returned by this event.

**Event parameters:**

*Num_HCI_Command_Packets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The Number of HCI Command packets which are allowed to be sent to the Controller from the Host.<br>Range: 0 to 255 |

*Command_Opcode:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No associated command |
| 0xXXXX | (non-zero) Opcode of the command which caused this event. |

*Return_Parameters:* *Size: Depends on command*

| Parameter Description |
| --- |
| This is the return parameter(s) for the command specified in the Command_Opcode event parameter. See each command's definition for the list of return parameters associated with that command.<br><br>Note: This parameter has a variable length. |

### 7.7.15 Command Status event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Command_Status | 0x0F | Status,<br>Num_HCI_Command_Packets,<br>Command_Opcode |

**Description:**

This event is used to indicate that the command described by the Command_Opcode parameter has been received, and that the Controller is currently performing the task for this command. This event is needed to provide mechanisms for asynchronous operation, which avoids the need for the Host to wait for a command to finish. If the command cannot begin to execute (a parameter error may have occurred, or the command may currently not be allowed), the Status parameter will contain the corresponding error code, and no complete event will follow since the command was not started. The Num_HCI_Command_Packets parameter allows the Controller to indicate the number of HCI command packets the Host can send to the Controller. If the Controller requires the Host to stop sending commands, Num_HCI_Command_Packets will be set to zero. To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller generates an HCI_Command_Status event with Status 0x00 and Command_Opcode 0x0000 and Num_HCI_Command_Packets set to 1 or more. Command_Opcode 0x0000 is a special value indicating that this event is not associated with a command sent by the Host. The Controller can send an HCI_Command_Status event with Command Opcode 0x0000 at any time to change the number of outstanding HCI command packets that the Host can send before waiting.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Command now pending. |
| 0x01 to 0xFF | Command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_HCI_Command_Packets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The Number of HCI Command packets which are allowed to be sent to the Controller from the Host.<br>Range: 0 to 255 |

*Command_Opcode:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No associated command |
| 0xXXXX | (non-zero) Opcode of the command which caused this event and is now pending. |

### 7.7.16 Hardware Error event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Hardware_Error | 0x10 | Hardware_Code |

**Description:**

This event is used to notify the Host that a hardware failure has occurred in the Controller.

**Event parameters:**

*Hardware_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | These Hardware_Codes will be implementation-specific, and can be assigned to indicate various hardware problems. |

### 7.7.17 Flush Occurred event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Flush_Occurred | 0x11 | Handle |

**Description:**

This event is used to indicate that, for the specified Handle, the current user data to be transmitted has been removed. The Handle shall be a Connection_Handle for an ACL connection. This could result from the HCI_Flush command, or be due to the automatic flush. Multiple blocks of an L2CAP packet could have been pending in the Controller. If one Baseband packet part of an L2CAP PDU is flushed, then the rest of the HCI ACL Data packets for the L2CAP PDU shall also be flushed.

**Event parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Handle that was flushed.<br>Range: 0x0000 to 0x0EFF |

### 7.7.18 Role Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Role_Change | 0x12 | Status,<br>BD_ADDR,<br>New_Role |

**Description:**

This event is used to indicate that the current role of the BR/EDR Controller related to the particular connection has changed. This event occurs (with Status set to zero) when both the remote and local BR/EDR Controllers have completed their role change for the BR/EDR Controller associated with the BD_ADDR parameter, allowing both affected Hosts to be notified when the Role has been changed.

This event is also generated on the local Controller when the HCI_Switch_Role command finishes because no change was requested or an error occurred. If a Baseband role switch was attempted but failed, the remote Controller may generate the event.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | A role change has occurred. |
| 0x01 to 0xFF | A role change failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which a role change has completed. |

*New_Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Currently the Central for specified BD_ADDR. |
| 0x01 | Currently the Peripheral for specified BD_ADDR. |

### 7.7.19 Number Of Completed Packets event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Number_Of_Completed_Packets | 0x13 | Num_Handles,<br>Handle[i],<br>Num_Completed_Packets[i] |

**Description:**

This event is used by the Controller to indicate to the Host how many HCI Data packets or HCI ISO Data packets have been completed for each Handle since the previous HCI_Number_Of_Completed_Packets event was sent to the Host. This means that the corresponding buffer space has been freed in the Controller and is available for new packets to be sent. Based on this information and the return parameters of the HCI_Read_Buffer_Size and HCI_LE_Read_Buffer_Size commands, the Host can determine for which Handles the following HCI packets should be sent to the Controller. The HCI_Number_Of_Completed_Packets event shall not specify a given Handle before the Controller has sent the event indicating that the corresponding connection or BIG has been created or after it has sent the event indicating disconnection of the corresponding connection or indicating that the BIG has been terminated. While the Controller has HCI Data packets or HCI ISO Data packets in its buffer, it shall keep sending the HCI_Number_Of_Completed_Packets event to the Host at least periodically, until it finally reports that all the pending packets have been completed. The rate with which this event is sent is manufacturer specific.

Note: HCI_Number_Of_Completed_Packets events will not report on synchronous Connection_Handles if synchronous Flow Control is disabled. (See Section 7.3.36 and Section 7.3.37.)

**Event parameters:**

*Num_Handles:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Handles and Num_HCI_Data_Packets parameters pairs contained in this event.<br>Range: 0 to 255 |

*Handle[i]:* *Size: Num_Handles × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle or BIS_Handle<br>Range: 0x0000 to 0x0EFF |

*Num_Completed_Packets[i]:* *Size: Num_Handles × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of packets that have been completed for the associated Connection_Handle since the previous time the event was returned.<br>Range: 0x0000 to 0xFFFF |

### 7.7.20 Mode Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Mode_Change | 0x14 | Status,<br>Connection_Handle,<br>Current_Mode,<br>Interval |

**Description:**

This event is used to indicate when the device associated with the Connection_Handle changes between Active mode, Hold mode, and Sniff mode. The Connection_Handle parameter will be a Connection_Handle for an ACL connection. Connection_Handle is used to indicate which connection the HCI_Mode_Change event is for. The Current_Mode event parameter is used to indicate which state the connection is currently in. The Interval parameter is used to specify a time amount specific to each state. Each Controller that is associated with the Connection_Handle which has changed modes shall send the HCI_Mode_Change event to its Host.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | A Mode Change has occurred. |
| 0x01 to 0xFF | HCI_Hold_Mode, HCI_Sniff_Mode, or HCI_Exit_Sniff_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle.<br>Range: 0x0000 to 0x0EFF |

*Current_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Active mode. |
| 0x01 | Hold mode. |
| 0x02 | Sniff mode. |
| All other values | Reserved for future use. |

*Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Hold:<br>Number of Baseband slots to wait in Hold mode.<br>Hold Interval = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0002 to 0xFFFE<br>Time Range: 1.25 ms to 40.9 s<br>Sniff:<br>Number of Baseband slots between sniff anchor points.<br>Time between sniff anchor points = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0002 to 0xFFFE<br>Time Range: 1.25 ms to 40.9 s |

### 7.7.21 Return Link Keys event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Return_Link_Keys | 0x15 | Num_Keys,<br>BD_ADDR[i],<br>Link_Key[i] |

**Description:**

This event is used by the BR/EDR Controller to send the Host the BD_ADDRs associated with one or more stored Link Keys. Zero or more instances of this event will occur after the HCI_Read_Stored_Link_Key command. When there are no link keys stored, no HCI_Return_Link_Keys events shall be returned. When there are link keys stored, the number of link keys returned in each HCI_Return_Link_Keys event is implementation specific. This event shall never return the value of the link keys. The link keys value parameter shall always contain the value of zero.

**Event parameters:**

*Num_Keys:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys contained in this event.<br>Range: 0x01 to 0x0B |

*BD_ADDR[i]:* *Size: Num_Keys × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the associated Link Key. |

*Link_Key[i]:* *Size: Num_Keys × 16 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000000000000000000000000000 | Shall be zero. |

### 7.7.22 PIN Code Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_PIN_Code_Request | 0x16 | BD_ADDR |

**Description:**

This event is used to indicate that a PIN code is required to create a new link key. The Host shall respond using either the HCI_PIN_Code_Request_Reply or the HCI_PIN_Code_Request_Negative_Reply command, depending on whether the Host can provide the Controller with a PIN code or not.

Note: If the HCI_PIN_Code_Request event is masked away, then the BR/EDR Controller will assume that the Host has no PIN Code.

When the BR/EDR Controller generates an HCI_PIN_Code_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_PIN_Code_Request_Reply or HCI_PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which a new link key is being created. |

### 7.7.23 Link Key Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Link_Key_Request | 0x17 | BD_ADDR |

**Description:**

This event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR. If the Host has the requested stored Link Key, then the Host shall pass the requested Key to the Controller using the HCI_Link_Key_Request_Reply command. If the Host does not have the requested stored Link Key, or the stored Link Key does not meet the security requirements for the requested service, then the Host shall use the HCI_Link_Key_Request_Negative_Reply command to indicate to the Controller that the Host does not have the requested key.

Note: If the HCI_Link_Key_Request event is masked away, then the BR/EDR Controller will assume that the Host has no additional link keys.

If the Host uses the HCI_Link_Key_Request_Negative_Reply command when the requested service requires an authenticated Link Key and the current Link Key is unauthenticated, the Host should set the Authentication_Requirements parameter one of the MITM Protection Required options.

When the Controller generates an HCI_Link_Key_Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of an HCI_Create_Connection or HCI_Authentication_Requested command from the remote Host), the local Host shall respond with either an HCI_Link_Key_Request_Reply or HCI_Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which a stored link key is being requested. |

### 7.7.24 Link Key Notification event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Link_Key_Notification | 0x18 | BD_ADDR,<br>Link_Key,<br>Key_Type |

**Description:**

This event is used to indicate to the Host that a new Link Key has been created for the connection with the device specified in BD_ADDR. The Host can save this new Link Key in its own storage for future use. Also, the Host can decide to store the Link Key in the BR/EDR Controller's Link Key Storage by using the HCI_Write_Stored_Link_Key command. The Key_Type parameter informs the Host about which key type (combination key, debug combination key, unauthenticated combination key, authenticated combination key or changed combination key) was used during pairing. If the key type is not supported or is reserved for future use, the Host may discard the key or disconnect the link.

The combination key Key_Type is used when standard pairing was used. The debug combination key Key_Type is used when Secure Simple Pairing was used and the debug public key is sent or received. The unauthenticated combination key Key_Type is used when the Just Works Secure Simple Pairing association model was used. The authenticated combination key Key_Type is used when Secure Simple Pairing was used and the Just Works association mode was not used. The changed combination key Key_Type is used when the link key has been changed using the Change Connection Link Key procedure and Secure Simple Pairing Mode is set to enabled.

Note: It is the responsibility of the Host to remember the Key_Type (combination, debug combination, unauthenticated combination, or authenticated combination) prior to changing the link key.

When the unauthenticated or authenticated combination key Key_Type is used, the Controller shall use values 0x04 and 0x05 to indicate keys created with the P-192 elliptic curve and values 0x07 and 0x08 to indicate keys created with the P-256 elliptic curve. The values 0x07 and 0x08 shall only be used when the Host has indicated support for Secure Connections in the Secure_Connections_Host_Support parameter.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which the new link key has been generated. |

*Link_Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Link Key for the associated BD_ADDR. |

*Key_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Combination Key |
| 0x03 | Debug Combination Key |
| 0x04 | Unauthenticated Combination Key generated from P-192 |
| 0x05 | Authenticated Combination Key generated from P-192 |
| 0x06 | Changed Combination Key |
| 0x07 | Unauthenticated Combination Key generated from P-256 |
| 0x08 | Authenticated Combination Key generated from P-256 |
| All other values | Reserved for future use |

### 7.7.25 Loopback Command event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Loopback_Command | 0x19 | HCI_Command_Packet |

**Description:**

When in Local Loopback mode, the BR/EDR Controller loops back commands and data to the Host. This event is used to loop back all commands that the Host sends to the Controller with some exceptions. See Section 7.6.1 for a description of which commands that are not looped back. The HCI_Command_Packet parameter contains the entire HCI Command Packet including the header.

Note: The event packet is limited to a maximum of 255 octets in the payload; since an HCI Command packet has 3 octets of header data, only the first 252 octets of the command parameters will be returned.

**Event parameters:**

*HCI_Command_Packet:* *Size: Depends on command*

| Parameter Description |
|---|
| HCI Command packet, including header.<br>Note: This parameter has a variable length. |

### 7.7.26 Data Buffer Overflow event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Data_Buffer_Overflow | 0x1A | Link_Type |

**Description:**

This event is used to indicate that the Controller's data buffers have been overflowed. This can occur if the Host has sent more packets than allowed. The Link_Type parameter is used to indicate the type of data whose buffers overflowed.

**Event parameters:**

*Link_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronous Data packet buffers |
| 0x01 | ACL Data packet buffers |
| 0x02 | ISO Data packet buffers |
| All other values | Reserved for future use. |

### 7.7.27 Max Slots Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Max_Slots_Change | 0x1B | Connection_Handle,<br>LMP_Max_Slots |

**Description:**

This event is used to notify the Host about the LMP_Max_Slots parameter when the value of this parameter changes. It shall be sent each time the maximum allowed length, in number of slots, for Baseband packets transmitted by the local device, changes. The Connection_Handle will be a Connection_Handle for an ACL connection.

**Event parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle.<br>Range: 0x0000 to 0x0EFF |

*LMP_Max_Slots:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01, 0x03, 0x05 | Maximum number of slots allowed to use for Baseband packets, see [Vol 2] Part C, Section 4.1.10 and Section 5.2. |

### 7.7.28 Read Clock Offset Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Read_Clock_Offset_Complete | 0x1C | Status,<br>Connection_Handle,<br>Clock_Offset |

**Description:**

This event is used to indicate the completion of the process of the Link Manager obtaining the Clock Offset information of the BR/EDR Controller specified by the Connection_Handle parameter. Connection_Handle will be a Connection_Handle for an ACL connection.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Clock_Offset command succeeded. |
| 0x01 to 0xFF | HCI_Read_Clock_Offset command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Clock_Offset:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0-14 | Bits 16-2 of CLKNPeripheral - CLK |
| 15 | Reserved for future use. |

### 7.7.29 Connection Packet Type Changed event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connection_Packet_Type_Changed | 0x1D | Status,<br>Connection_Handle,<br>Packet_Type |

**Description:**

This event is used to indicate that the process has completed of the Link Manager changing which packet types can be used for the connection. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type parameter specifies which packet types the Link Manager can use for the connection identified by the Connection_Handle parameter for sending L2CAP data or voice. Packet_Type does not decide which packet types the LM is allowed to use for sending LMP PDUs.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Change_Connection_Packet_Type command succeeded. |
| 0x01 to 0xFF | HCI_Change_Connection_Packet_Type command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle.<br>Range: 0x0000 to 0x0EFF |

*Packet_Type:* *Size: 2 octets*

*For ACL_Link_Type*

| Bit Number | Parameter Description |
|---|---|
| 1 | 2-DH1 shall not be used. |
| 2 | 3-DH1 shall not be used. |
| 3 | Ignored; DM1 may be used whether or not this bit is set. |
| 4 | DH1 may be used. |
| 8 | 2-DH3 shall not be used. |

| Bit Number | Parameter Description |
|---|---|
| 9 | 3-DH3 shall not be used. |
| 10 | DM3 may be used. |
| 11 | DH3 may be used. |
| 12 | 2-DH5 shall not be used. |
| 13 | 3-DH5 shall not be used. |
| 14 | DM5 may be used. |
| 15 | DH5 may be used. |
| All other bits | Reserved for future use. |

*For SCO_Link_Type*

| Bit Number | Parameter Description |
|---|---|
| 5 | HV1 may be used. |
| 6 | HV2 may be used. |
| 7 | HV3 may be used. |
| All other bits | Reserved for future use. |

### 7.7.30 QoS Violation event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_QoS_Violation | 0x1E | Handle |

**Description:**

This event is used to indicate the Controller is unable to provide the current QoS requirement for the Connection identified by the Handle. This event indicates that the Controller is unable to provide one or more of the agreed QoS parameters.

The Host chooses what action should be done; for example, it can reissue the HCI_QoS_Setup command to renegotiate the QoS setting for Connection_Handle.

**Event parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Handle for the link that the Controller cannot provide the current QoS requested. The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

### 7.7.31 Page Scan Repetition Mode Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Page_Scan_Repetition_Mode_Change | 0x20 | BD_ADDR,<br>Page_Scan_Repetition_Mode |

**Description:**

This event indicates that the remote BR/EDR Controller with the specified BD_ADDR has successfully changed the Page Scan Repetition Mode (see [Vol 2] Part B, Section 8.3.1).

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the remote device. |

*Page_Scan_Repetition_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use |

### 7.7.32 Flow Specification Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Flow_Specification_Complete | 0x21 | Status,<br>Connection_Handle,<br>Unused,<br>Flow_Direction,<br>Service_Type,<br>Token_Rate,<br>Token_Bucket_Size,<br>Peak_Bandwidth,<br>Access_Latency |

**Description:**

This event is used to inform the Host about the Quality of Service for the ACL connection the Controller is able to support. The Connection_Handle will be a Connection_Handle for an ACL connection. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_Direction field. The flow parameters are defined in the L2CAP specification [Vol 3] Part A, Section 5.3. When the Status parameter indicates a successful completion, the flow parameters specify the agreed values by the Controller. When the Status parameter indicates a failed completion with the error code *QoS Unacceptable Parameter* (0x2C), the flow parameters specify the acceptable values of the Controller. This enables the Host to continue the 'QoS negotiation' with a new HCI_Flow_Specification command with flow parameter values that are acceptable for the Controller. When the Status parameter indicates a failed completion with the error code *QoS Rejected* (0x2D), this indicates a request of the Controller to discontinue the 'QoS negotiation'. When the Status parameter indicates a failed completion, the flow parameter values of the most recently successful completion shall be assumed (or the default values when there was no success completion).

The Unused parameter is reserved for future use.

This event is generated following an HCI_Flow_Specification command issued by the local Host.

Note: This event or the HCI_QoS_Setup_Complete event can be generated if the remote device performs an LMP transaction involving the flow parameter values.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Flow_Specification command succeeded |
| 0x01 to 0xFF | HCI_Flow_Specification command failed. See [Vol 1] Part F, Controller Error Codes for list of error codes |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Unused:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | This value shall be used by the Controller. |
| All other values | Reserved for future use. |

*Flow_Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Outgoing Flow i.e., traffic sent over the ACL connection. |
| 0x01 | Incoming Flow i.e., traffic received over the ACL connection. |
| All other values | Reserved for future use. |

*Service_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic |
| 0x01 | Best Effort |
| 0x02 | Guaranteed |
| All other values | Reserved for future use |

*Token_Rate:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second |

*Token_Bucket_Size:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Bucket Size in octets |

*Peak_Bandwidth:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second |

*Access_Latency:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Access Latency in microseconds |

### 7.7.33 Inquiry Result with RSSI event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Inquiry_Result_with_RSSI | 0x22 | Num_Responses,<br>BD_ADDR[i],<br>Page_Scan_Repetition_Mode[i],<br>Reserved[i],<br>Class_Of_Device[i],<br>Clock_Offset[i],<br>RSSI[i] |

**Description:**

This event indicates that a BR/EDR Controller or multiple BR/EDR Controllers have responded so far during the current Inquiry process. This event will be sent from the BR/EDR Controller to the Host as soon as an Inquiry Response from a remote device is received if the remote device supports only mandatory paging scheme. This BR/EDR Controller may queue these Inquiry Responses and send multiple BR/EDR Controllers information in one HCI_Inquiry_Result event. The event can be used to return one or more Inquiry responses in one event. The RSSI parameter is measured during the FHS packet returned by each responding Peripheral.

This event shall only be generated if the Inquiry Mode parameter of the last HCI_Write_Inquiry_Mode command was set to 0x01 (Inquiry Result format with RSSI), or was set to 0x02 (Inquiry Result with RSSI format or Extended Inquiry Result format) and the inquiry response packet had the EIR field set to 0.

**Event parameters:**

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of responses from the Inquiry. |

*BD_ADDR[i]:* *Size: Num_Responses × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for a device which responded. |

*Page_Scan_Repetition_Mode[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use |

*Reserved[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reserved for future use. |

*Class_Of_Device[i]:* *Size: Num_Responses × 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device |

*Clock_Offset[i]:* *Size: Num_Responses × 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0-14 | Bits 16-2 of CLKNPeripheral - CLK |
| 15 | Reserved for future use |

*RSSI[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

### 7.7.34 Read Remote Extended Features Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Read_Remote_Extended_Features_Complete | 0x23 | Status,<br>Connection_Handle,<br>Page_Number,<br>Max_Page_Number,<br>Extended_LMP_Features |

**Description:**

This event is used to indicate the completion of the process of the Link Manager obtaining the remote extended LMP features of the remote device specified by the Connection_Handle parameter. Connection_Handle will be a Connection_Handle for an ACL connection. The parameters include a page of the remote devices extended LMP features. For details see [Vol 2] Part C, Section 3.3.

If a feature page is requested more than once while a connection exists between the two devices, the second and subsequent requests may report a cached copy of that page rather than fetching it again.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Request for remote extended features succeeded |
| 0x01 to 0xFF | Request for remote extended features failed. See [Vol 1] Part F, Controller Error Codes for error codes. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The normal LMP features as returned by HCI_Read_Remote_Supported_Features command |
| 0x01 to 0xFF | The page number of the features returned |

*Max_Page_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | The highest features page number which contains non-zero bits for the remote device |

*Extended_LMP_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit map of requested page of LMP features.<br>See [Vol 2] Part C, Section 3.3 for details. |

### 7.7.35 Synchronous Connection Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Synchronous_Connection_Complete | 0x2C | Status,<br>Connection_Handle,<br>BD_ADDR,<br>Link_Type,<br>Transmission_Interval,<br>Retransmission_Window,<br>RX_Packet_Length,<br>TX_Packet_Length,<br>Air_Mode |

**Description:**

This event is sent to indicate completion of any of the following commands:

- HCI_Setup_Synchronous_Connection
- HCI_Accept_Synchronous_Connection_Request
- HCI_Reject_Synchronous_Connection_Request
- HCI_Enhanced_Setup_Synchronous_Connection
- HCI_Enhanced_Accept_Synchronous_Connection_Request

This event returns the completion status for the command.

If the HCI_Synchronous_Connection_Complete event was triggered by the HCI_Enhanced_Setup_Synchronous_Connection or HCI_Enhanced_Accept_Synchronous_Connection_Request commands, then the Controller shall set the Air_Mode parameter to the first octet of the Transmit_Coding_Format parameter of the original command. Otherwise the Controller should set the Air_Mode parameter to the Air_Mode that was negotiated over LMP for the connection.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01 to 0xFF | Connection failed to complete. See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the other connected device forming the connection. |

*Link_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO connection |
| 0x01 | Reserved for future use |
| 0x02 | eSCO connection |
| All other values | Reserved for future use |

*Transmission_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Time between two consecutive eSCO instants measured in slots. Shall be zero for SCO links. |

*Retransmission_Window:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The size of the retransmission window measured in slots. Shall be zero for SCO links. |

*RX_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the eSCO payload in the receive direction. Shall be zero for SCO links. |

*TX_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the eSCO payload in the transmit direction. Shall be zero for SCO links. |

*Air_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | See Assigned Numbers for Coding_Format |

### 7.7.36 Synchronous Connection Changed event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Synchronous_Connection_Changed | 0x2D | Status,<br>Connection_Handle,<br>Transmission_Interval,<br>Retransmission_Window,<br>RX_Packet_Length,<br>TX_Packet_Length |

**Description:**

This event indicates to the Host that an existing synchronous connection has been reconfigured. This event also indicates to the initiating Host (if the change was Host initiated) if the issued command failed or was successful.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully reconfigured. |
| 0x01 to 0xFF | Reconfiguration failed to complete.<br>See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Transmission_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Time between two consecutive SCO/eSCO instants measured in slots. |

*Retransmission_Window:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The size of the retransmission window measured in slots. Shall be zero for SCO links. |

*RX_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the SCO/eSCO payload in the receive direction. |

*TX_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the SCO/eSCO payload in the transmit direction. |

### 7.7.37 Sniff Subrating event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Sniff_Subrating | 0x2E | Status,<br>Connection_Handle,<br>Max_TX_Latency,<br>Max_RX_Latency,<br>Min_Remote_Timeout,<br>Min_Local_Timeout |

**Description:**

This event indicates that the device associated with the Connection_Handle has either enabled sniff subrating or the sniff subrating parameters have been renegotiated by the link manager. The Connection_Handle parameter will be a Connection_Handle for an ACL connection. Connection_Handle indicates which connection the HCI_Sniff_Subrating event is for.

Each BR/EDR Controller that is associated with the Connection_Handle that has changed its subrating parameters will send the Sniff Subrating event to its Host.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Sniff_Subrating command succeeded |
| 0x01 to 0xFF | HCI_Sniff_Subrating command failed to complete.<br>See [Vol 1] Part F, Controller Error Codes for error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle.<br>Range: 0x0000 to 0x0EFF |

*Max_TX_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum latency for data being transmitted from the local device to the remote device.<br>Latency = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

*Max_RX_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum latency for data being received by the local device from the remote device.<br>Latency = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

*Min_Remote_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The base sniff subrate timeout in Baseband slots that the remote device shall use.<br>Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

*Min_Local_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The base sniff subrate timeout in Baseband slots that the local device will use.<br>Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0000 to 0xFFFE<br>Time Range: 0 s to 40.9 s |

### 7.7.38 Extended Inquiry Result event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Extended_Inquiry_Result | 0x2F | Num_Responses,<br>BD_ADDR,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Class_Of_Device,<br>Clock_Offset,<br>RSSI,<br>Extended_Inquiry_Response |

**Description:**

This event indicates that a BR/EDR Controller has responded during the current inquiry process with extended inquiry response data. This event will be sent from the BR/EDR Controller to the Host upon reception of an Extended Inquiry Response from a remote device. One single Extended Inquiry Response is returned per event. This event contains RSSI and inquiry response data for the BR/EDR Controller that responded to the latest inquiry. The RSSI parameter is measured during the FHS packet returned by each responding discoverable device. The Num_Responses parameter shall be set to one.

This event is only generated if the Inquiry_Mode parameter of the last HCI_Write_Inquiry_Mode command was set to 0x02 (Inquiry Result with RSSI format or Extended Inquiry Result format).

Note: This ensures that a Host that does not support Extended Inquiry Results will never receive the HCI_Extended_Inquiry_Result event.

If an inquiry response packet with the EIR field set to zero is received, the HCI_Inquiry_Result_with_RSSI event format shall be used. If the EIR bit is set to one the HCI_Extended_Inquiry_Result event format shall be used. If the EIR bit is set to one but the Controller failed to receive the extended inquiry response packet, the Extended_Inquiry_Response parameter is set to zeros. If an extended inquiry response packet from the same device is correctly received in a later response, another event shall be generated.

Note: The only difference between the HCI_Extended_Inquiry_Result event and the HCI_Inquiry_Result_with_RSSI event is the additional Extended_Inquiry_Response parameter.

The Extended_Inquiry_Response parameter is not interpreted by the Controller. The tagged data set by the other Host shall be passed unaltered if it has been correctly received.

**Event parameters:**

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Number of responses from the inquiry. The HCI_Extended_Inquiry_Result event always contains a single response. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device that responded. |

*Page_Scan_Repetition_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| All other values | Reserved for future use |

*Reserved:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved, shall be set to 0x00. |

*Class_Of_Device:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device that responded. |

*Clock_Offset:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0-14 | Bits 16-2 of CLKNPeripheral - CLK |
| 15 | Reserved for future use. |

*RSSI:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

*Extended_Inquiry_Response:* *Size: 240 octets*

| Value | Parameter Description |
|---|---|
| *none* | Extended Inquiry Response data as defined in [Vol 3] Part C, Section 8 |

### 7.7.39 Encryption Key Refresh Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Encryption_Key_Refresh_Complete | 0x30 | Status,<br>Connection_Handle |

**Description:**

This event is used to indicate to the Host that the encryption key was refreshed on the given Connection_Handle any time encryption is paused and then resumed. The Controller shall send this event when the encryption key has been refreshed due to encryption being started or resumed.

If the HCI_Encryption_Key_Refresh_Complete event was generated due to an encryption pause and resume operation embedded within a change connection link key procedure, the HCI_Encryption_Key_Refresh_Complete event shall be sent prior to the HCI_Change_Connection_Link_Key_Complete event.

If the HCI_Encryption_Key_Refresh_Complete event was generated due to an encryption pause and resume operation embedded within a role switch procedure, the HCI_Encryption_Key_Refresh_Complete event shall be sent prior to the HCI_Role_Change event.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Encryption key refresh completed successfully |
| 0x01 to 0xFF | Encryption key refresh failed. See [Vol 1] Part F, Controller Error Codes for list of error codes |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### 7.7.40 IO Capability Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_IO_Capability_Request | 0x31 | BD_ADDR |

**Description:**

This event is used to indicate that the IO capabilities of the Host are required for a Secure Simple Pairing process. The Host shall respond with an HCI_IO_Capability_Request_Reply command or HCI_IO_Capability_Request_Negative_Reply command. This event shall only be generated if Secure Simple Pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

### 7.7.41 IO Capability Response event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_IO_Capability_Response | 0x32 | BD_ADDR,<br>IO_Capability,<br>OOB_Data_Present,<br>Authentication_Requirements |

**Description:**

This event is used to indicate to the Host that IO capabilities from a remote device specified by BD_ADDR have been received during a Secure Simple Pairing process. This event will only be generated if Secure Simple Pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR identifying the device to which the IO capabilities apply. |

*IO_Capability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | DisplayOnly |
| 0x01 | DisplayYesNo |
| 0x02 | KeyboardOnly |
| 0x03 | NoInputNoOutput |
| All other values | Reserved for future use |

*OOB_Data_Present:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | OOB authentication data not present |
| 0x01 | OOB authentication data from remote device present |
| All other values | Reserved for future use |

*Authentication_Requirements:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | MITM Protection Not Required – No Bonding. Numeric comparison with automatic accept allowed. |
| 0x01 | MITM Protection Required – No Bonding. Use IO Capabilities to determine authentication procedure |
| 0x02 | MITM Protection Not Required – Dedicated Bonding. Numeric comparison with automatic accept allowed. |
| 0x03 | MITM Protection Required – Dedicated Bonding. Use IO Capabilities to determine authentication procedure |
| 0x04 | MITM Protection Not Required – General Bonding. Numeric Comparison with automatic accept allowed. |
| 0x05 | MITM Protection Required – General Bonding. Use IO capabilities to determine authentication procedure. |
| All other values | Reserved for future use |

### 7.7.42 User Confirmation Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_User_Confirmation_Request | 0x33 | BD_ADDR,<br>Numeric_Value |

**Description:**

This event is used to indicate that user confirmation of a numeric value is required. The Host shall reply with either the HCI_User_Confirmation_Request_Reply or the HCI_User_Confirmation_Request_Negative_Reply command. If the Host has output capability (DisplayYesNo or KeyboardOnly), it shall display the Numeric_Value until the HCI_Simple_Pairing_Complete event is received. It shall reply based on the yes/no response from the user. If the Host has no input and no output it shall reply with the HCI_User_Confirmation_Request_Reply command. When the Controller generates an HCI_User_Confirmation_Request event, in order for the local Link Manager to respond to the request from the remote Link Manager, the local Host shall respond with either an HCI_User_Confirmation_Request_Reply or an HCI_User_Confirmation_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of device involved in the Secure Simple Pairing process. |

*Numeric_Value:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0x000F423F | Numeric value to be displayed. Valid values are decimal 000000 to 999999. |

### 7.7.43 User Passkey Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_User_Passkey_Request | 0x34 | BD_ADDR |

**Description:**

This event is used to indicate that a passkey is required as part of a Secure Simple Pairing process. The Host shall respond with either an HCI_User_Passkey_Request_Reply or HCI_User_Passkey_Request_Negative_Reply command. This event will only be generated if Secure Simple Pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command. When the Controller generates an HCI_User_Passkey_Request event, in order for the local Link Manager to respond to the request from the remote Link Manager, the local Host shall respond with either an HCI_User_Passkey_Request_Reply or HCI_User_Passkey_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See [Vol 2] Part C, Section 2.4.1.)

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in the Secure Simple Pairing process. |

### 7.7.44 Remote OOB Data Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Remote_OOB_Data_Request | 0x35 | BD_ADDR |

**Description:**

This event is used to indicate that the Secure Simple Pairing Hash C and Randomizer R are required for the Secure Simple Pairing process involving the device identified by BD_ADDR. The C and R values were transferred to the Host from the remote device via an OOB mechanism. This event is sent by the Controller because the Host previously set the OOB Data Present parameter to "OOB authentication data from remote device present" in an HCI_IO_Capability_Request_Reply command. If both the Host and Controller support Secure Connections the Host shall respond with the values using the HCI_Remote_OOB_Extended_Data_Request_Reply command. Otherwise, the Host shall respond with the values using the HCI_Remote_OOB_Data_Request_Reply command. In either case, if the Host does not have the C and R values for the device, it shall respond with the HCI_Remote_OOB_Data_Request_Negative_Reply command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device from which the C and R values were received. |

### 7.7.45 Simple Pairing Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Simple_Pairing_Complete | 0x36 | Status,<br>BD_ADDR |

**Description:**

This event is used to indicate that the Secure Simple Pairing process has completed. A Host that is displaying a numeric value can use this event to change its UI.

When the LMP Secure Simple Pairing sequences fail for any reason, the HCI_Simple_Pairing_Complete event shall be sent to the Host. When HCI_Simple_Pairing_Complete event is sent in response to the IO capability exchange failing, the Status parameter shall be set to the error code received from the remote device. Otherwise, the Status shall be set to the error code *Authentication Failure* (0x05).

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Secure Simple Pairing succeeded |
| 0x01 to 0xFF | Secure Simple Pairing failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in the Secure Simple Pairing process. |

### 7.7.46 Link Supervision Timeout Changed event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Link_Supervision_Timeout_Changed | 0x38 | Connection_Handle,<br>Link_Supervision_Timeout |

**Description:**

This event is used to notify the Peripheral's Host when the Link_Supervision_Timeout parameter is changed in the Peripheral's Controller. This event shall only be sent to the Host by the Peripheral's Controller upon receiving an LMP_SUPERVISION_TIMEOUT PDU from the Central.

The Connection_Handle used for this command shall be the ACL connection of the appropriate device.

**Event parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Link_Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Measured in number of Baseband slots<br>Link_Supervision_Timeout = N × 0.625 ms (1 Baseband slot)<br>Range: 0x0001 to 0xFFFF<br>Time Range: 0.625 ms to 40.9 s<br>(0 means infinite timeout) |

### 7.7.47 Enhanced Flush Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Enhanced_Flush_Complete | 0x39 | Handle |

**Description:**

This event is used to indicate that an Enhanced Flush is complete for the specified Handle.

**Event parameters:**

*Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Handle of the connection for which the Enhanced Flush was done. The Handle is a Connection_Handle for a BR/EDR Controller.<br>Range: 0x0000 to 0x0EFF |

### 7.7.48 User Passkey Notification event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_User_Passkey_Notification | 0x3B | BD_ADDR,<br>Passkey |

**Description:**

This event is used to provide a passkey for the Host to display to the user as required as part of a Secure Simple Pairing process. The Passkey parameter shall be a random number created according to [Vol 2] Part H, Section 2, *mod* 1,000,000.

This event will be generated if the IO capabilities of the local device are DisplayOnly or DisplayYesNo and the IO capabilities of the remote device are KeyboardOnly.

This event shall only be generated if Secure Simple Pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in the Secure Simple Pairing process. |

*Passkey:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0x000F423F | Passkey to be displayed. Valid values are decimal 000000 to 999999. |

### 7.7.49 Keypress Notification event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Keypress_Notification | 0x3C | BD_ADDR,<br>Notification_Type |

**Description:**

This event is sent to the Host after a passkey notification has been received by the Link Manager on the given BD_ADDR. The Notification_Type parameter may be used by the Host's user interface.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Secure Simple Pairing process |

*Notification_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 | Passkey entry started |
| 1 | Passkey digit entered |
| 2 | Passkey digit erased |
| 3 | Passkey cleared |
| 4 | Passkey entry completed |
| All other values | Reserved for future use |

### 7.7.50 Remote Host Supported Features Notification event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Remote_Host_Supported_Features_Notification | 0x3D | BD_ADDR,<br>Host_Supported_Features |

**Description:**

This event is used to return the LMP extended features page containing the Host features. The BD_ADDR shall be the address of the remote device.

This event shall only be generated after the LMP extended features are read from the remote device during a connection initiated by an HCI_Remote_Name_Request command.

Note: This event is not generated during a connection initiated by the HCI_Create_Connection command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device. |

*Host_Supported_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit map of Host Supported Features page of LMP extended features.<br>For more information, see [Vol 2] Part C, Section 3.3. |

### 7.7.51 [This section is no longer used]

### 7.7.52 [This section is no longer used]

### 7.7.53 [This section is no longer used]

### 7.7.54 [This section is no longer used]

### 7.7.55 [This section is no longer used]

### 7.7.56 [This section is no longer used]

### 7.7.57 [This section is no longer used]

### 7.7.58 [This section is no longer used]

### 7.7.59 Number Of Completed Data Blocks event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Number_Of_Completed_Data_Blocks | 0x48 | Total_Num_Data_Blocks,<br>Num_Handles,<br>Connection_Handle[i],<br>Num_Completed_Packets[i],<br>Num_Completed_Blocks[i] |

**Description:**

This event is used by the Controller to indicate to the Host how many HCI ACL Data packets have been completed (transmitted or flushed), and how many data block buffers have been freed, for each Connection_Handle since the previous HCI_Number_Of_Completed_Data_Blocks event was sent to the Host. This means that the corresponding buffer space has been freed in the Controller. Based on this information, and the Total_Num_Data_Blocks parameter, the Host can determine for which Handles the following HCI ACL Data packets should be sent to the Controller.

The Host should determine the number of blocks occupied by each ACL data packet by dividing the ACL data packet size by the Data_Block_Length parameter of the HCI_Read_Data_Block_Size command.

The Total_Num_Data_Blocks parameter indicates the total number of buffer blocks available in the Controller. Before any HCI_Number_Of_Completed_Data_Blocks event is received, the value of Total_Num_Data_Blocks from the HCI_Read_Data_Block_Size command is used. This allows the value to be updated at any time, which provides the Controller with some flexibility on its buffer allocation.

If the Controller were permitted to reduce its buffer pool in an arbitrary way then there is a potential race condition, in the case where the Host has just started to transmit a new packet. In order to prevent this race condition, the Total_Num_Data_Blocks parameter shall not indicate a reduction in the pool of blocks greater than the sum of the Num_Completed_Blocks values in this event. If a greater reduction in the block pool is required then the value 0 shall be indicated here. The value 0 has a special meaning and indicates that the Host shall re-issue the HCI_Read_Data_Block_Size command in order to find the new buffer pool size. The Host shall wait for any outstanding TX to complete and shall defer further TX until the HCI_Read_Data_Block_Size command has been issued and completed. The Controller shall reduce its allocation only after the HCI_Read_Data_Block_Size command has been issued and completed. This ensures that the race condition described above is avoided.

**Event parameters:**

*Total_Num_Data_Blocks:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | The size of the buffer pool may have changed. The Host is requested to issue an HCI_Read_Data_Block_Size command in order to determine the new value of Total_Num_Data_Blocks. |
| 0xXXXX | Total number of data block buffers available in the Controller for the storage of data packets scheduled for transmission. This indicates the existing value is unchanged, or increased, or reduced by up to the sum of the Num_Completed_Blocks values in this command. |

*Num_Handles:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Handles and Num_Completed_Packets and Num_Completed_Blocks parameter triples contained in this event.<br>Range: 0 to 255 |

*Connection_Handle[i]:* *Size: Num_Handles × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle for a BR/EDR connection.<br>Range: 0x0000 to 0x0EFF |

*Num_Completed_Packets[i]:* *Size: Num_Handles × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of HCI ACL Data packets that have been completed (transmitted or flushed) for the associated Handle since the previous time that an HCI_Number_Of_Completed_Data_Blocks event provided information about this Handle.<br>Range: 0x0000 to 0xFFFF |

*Num_Completed_Blocks[i]:* *Size: Num_Handles × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of data blocks that have been freed for the associated Handle since the previous time that an HCI_Number_Of_Completed_Data_Blocks event provided information about this Handle.<br>Range: 0x0000 to 0xFFFF |

### 7.7.60 [This section is no longer used]

### 7.7.61 [This section is no longer used]

### 7.7.62 [This section is no longer used]

### 7.7.63 [This section is no longer used]

### 7.7.64 [This section is no longer used]

### 7.7.65 LE Meta event

**Description:**

The LE Meta event is used to encapsulate all LE Controller specific events. The Event Code of all LE Meta events shall be 0x3E. The Subevent_Code is the first octet of the event parameters. The Subevent_Code shall be set to one of the valid Subevent_Codes from an LE specific event. All other parameters are defined in the LE Controller specific events.

#### 7.7.65.1 LE Connection Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Connection_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Role,<br>Peer_Address_Type,<br>Peer_Address,<br>Connection_Interval,<br>Peripheral_Latency,<br>Supervision_Timeout,<br>Central_Clock_Accuracy |

**Description:**

This event indicates to both of the Hosts forming the connection that a new connection has been created. Upon the creation of the connection a Connection_Handle shall be assigned by the Controller, and passed to the Host in this event. If the connection creation fails this event shall be provided to the Host that had issued the HCI_LE_Create_Connection command.

This event indicates to the Host which issued an HCI_LE_Create_Connection command and received an HCI_Command_Status event if the connection creation failed or was successful.

The Central_Clock_Accuracy parameter is only valid for a Peripheral. On a Central, this parameter shall be set to 0x00.

Note: This event is not sent if the HCI_LE_Enhanced_Connection_Complete event (see Section 7.7.65.10) is unmasked.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Subevent code for the HCI_LE_Connection_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01 to 0xFF | Connection failed to complete. [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection is Central |
| 0x01 | Connection is Peripheral |
| All other values | Reserved for future use |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Peer is using a Public Device Address |
| 0x01 | Peer is using a Random Device Address |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address or Random Device Address of the peer device |

*Connection_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection interval used on this connection.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4000 ms. |

*Peripheral_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Peripheral latency for the connection in number of connection events.<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection supervision timeout.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Central_Clock_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 500 ppm |
| 0x01 | 250 ppm |
| 0x02 | 150 ppm |
| 0x03 | 100 ppm |
| 0x04 | 75 ppm |
| 0x05 | 50 ppm |
| 0x06 | 30 ppm |
| 0x07 | 20 ppm |
| All other values | Reserved for future use |

### 7.7.65.2 LE Advertising Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Advertising_Report | 0x3E | Subevent_Code,<br>Num_Reports,<br>Event_Type[i],<br>Address_Type[i],<br>Address[i],<br>Data_Length[i],<br>Data[i],<br>RSSI[i] |

**Description:**

This event indicates that one or more Bluetooth devices have responded to an active scan or have broadcast advertisements that were received during a passive scan. The Controller may queue these advertising reports and send information from multiple devices in one HCI_LE_Advertising_Report event.

This event shall only be generated if scanning was enabled using the HCI_LE_Set_Scan_Enable command. It only reports advertising events that used legacy advertising PDUs.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 | Subevent code for the HCI_LE_Advertising_Report event |

*Num_Reports:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x19 | Number of responses in event. |
| All other values | Reserved for future use |

*Event_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connectable and scannable undirected advertising (ADV_IND) |
| 0x01 | Connectable directed advertising (ADV_DIRECT_IND) |

| Value | Parameter Description |
|---|---|
| 0x02 | Scannable undirected advertising (ADV_SCAN_IND) |
| 0x03 | Non connectable undirected advertising (ADV_NONCONN_IND) |
| 0x04 | Scan Response (SCAN_RSP) |
| All other values | Reserved for future use |

*Address_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (Corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (Corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Address[i]:* *Size: Num_Reports × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address or Random (static) Identity Address of the advertising device. |

*Data_Length[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | Length of the Data[i] field for the device which responded. |
| All other values | Reserved for future use. |

*Data[i]:* *Size: SUM (Data_Length[i]) octets*

| Parameter Description |
|---|
| Data_Length[i] octets of advertising or scan response data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: Each element of this array has a variable length. |

*RSSI[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |

#### 7.7.65.3 LE Connection Update Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Connection_Update_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Connection_Interval,<br>Peripheral_Latency,<br>Supervision_Timeout |

**Description:**

This event is used to indicate that the Connection Update procedure has completed.

This event shall be issued if the HCI_LE_Connection_Update command was issued by the Host or if the connection parameters are updated following a request from the peer device. If no parameters are updated following a request from the peer device or the parameters were changed using the Connection Subrate Update procedure or the Connection Rate Update procedure, then this event shall not be issued.

If the Status parameter is zero and the connection interval has changed, then the Link Layer must have set the subrating factor to 1 and the continuation number to 0 (see [Vol 6] Part B, Section 5.1.1).

Note: This event can be issued autonomously by the Central's Controller if it decides to change the connection interval based on the range of allowable connection intervals for that connection.

Note: The parameter values returned in this event can be different from the parameter values provided by the Host through the HCI_LE_Connection_Update command (Section 7.8.18) or the HCI_LE_Remote_Connection_Parameter_Request_Reply command (Section 7.8.31).

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x03 | Subevent code for the HCI_LE_Connection_Update_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Connection_Update command successfully completed. |
| 0x01 to 0xFF | HCI_LE_Connection_Update command failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Connection_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection interval used on this connection.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4000 ms. |

*Peripheral_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Peripheral latency for the connection in number of subrated connection events.<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Supervision timeout for this connection.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32000 ms |

### 7.7.65.4 LE Read Remote Features Page 0 Complete event[1]

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Read_Remote_Features_Page_0_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>LE_Features |

**Description:**

This event is used to indicate the completion of the process of the Controller obtaining page 0 of the features used on the connection and the features supported by the remote Bluetooth device specified by the Connection_Handle parameter.

If page 0 is requested more than once while a connection exists between the two devices, then the second and subsequent requests may report a cached copy of page 0 rather than fetching the feature mask again.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x04 | Subevent code for the HCI_LE_Read_Remote_Features_Page_0_Complete event. |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Remote_Features_Page_0 command successfully completed. |
| 0x01 to 0xFF | HCI_LE_Read_Remote_Features_Page_0 command failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

[1]This event was formerly called “LE Read Remote Features Complete”.

*LE_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit Mask List of page 0 of the LE features. See [Vol 6] Part B, Section 4.6. |

#### 7.7.65.5 LE Long Term Key Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Long_Term_Key_Request | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Random_Number,<br>Encrypted_Diversifier |

**Description:**

This event indicates that the peer device, in the Central role, is attempting to encrypt or re-encrypt the link and is requesting the Long Term Key from the Host. (See [Vol 6] Part B, Section 5.1.3).

This event shall only be generated when the local device's role is Peripheral.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x05 | Subevent code for the HCI_LE_Long_Term_Key_Request event |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Random_Number:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | 64-bit random number. |

*Encrypted_Diversifier:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 16-bit encrypted diversifier. |

### 7.7.65.6 LE Remote Connection Parameter Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Remote_Connection_Parameter_Request | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Interval_Min,<br>Interval_Max,<br>Max_Latency,<br>Timeout |

**Description:**

This event indicates to the Central's Host or the Peripheral's Host that the remote device is requesting a change in the connection parameters using the Connection Update procedure. The Host replies either with the HCI_LE_Remote_Connection_Parameter_Request_Reply command or the HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply command.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x06 | Subevent code for the HCI_LE_Remote_Connection_Parameter_Request event. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum value of the connection interval requested by the remote device.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |

*Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum value of the connection interval requested by the remote device.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum allowed Peripheral latency for the connection specified as the number of subrated connection events requested by the remote device.<br>Range: 0x0000 to 0x01F3 (499) |

*Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for the connection requested by the remote device.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

#### 7.7.65.7 LE Data Length Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Data_Length_Change | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Max_TX_Octets,<br>Max_TX_Time,<br>Max_RX_Octets,<br>Max_RX_Time |

**Description:**

This event notifies the Host of a change to either the maximum LL Data PDU Payload length or the maximum transmission time of packets containing LL Data PDUs in either direction. The values reported are the limits imposed on the connection by the Link Layer following the change (see [Vol 6] Part B, Section 4.5.10); the actual maximum used on the connection may be less for other reasons. This event shall not be generated if the values have not changed.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x07 | Subevent code for the HCI_LE_Data_Length_Change event |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Max_TX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum number of payload octets in a LLData PDU that the local Controller will send on this connection (*connEffectiveMaxTxOctets* defined in [Vol 6] Part B, Section 4.5.10).<br>Range 0x001B to 0x00FB |

*Max_TX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum time that the local Controller will take to send a Link Layer packet containing an LL Data PDU on this connection (*connEffectiveMaxTxTime* defined in [Vol 6] Part B, Section 4.5.10).<br>Range 0x0148 to 0x4290 |

*Max_RX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum number of payload octets in a Link Layer packet that the local Controller expects to receive on this connection (*connEffectiveMaxRxOctets* defined in [Vol 6] Part B, Section 4.5.10).<br>Range 0x001B to 0x00FB |

*Max_RX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum time that the local Controller expects to take to receive a Link Layer packet on this connection *(connEffectiveMaxRxTime* defined in [Vol 6] Part B, Section 4.5.10).<br>Range 0x0148 to 0x4290 |

#### 7.7.65.8 LE Read Local P-256 Public Key Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Read_Local_P-256_Public_Key_Complete | 0x3E | Subevent_Code,<br>Status,<br>Key_X_Coordinate,<br>Key_Y_Coordinate |

**Description:**

This event is generated when local P-256 key generation is complete.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x08 | Subevent code for the HCI_LE_Read_Local_P-256_Public_Key_Complete event. |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Local_P-256_Public_Key command completed successfully. |
| 0x01 to 0xFF | HCI_LE_Read_Local_P-256_Public_Key command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Key_X_Coordinate:* *Size: 32 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX | Local P-256 public key X coordinate. |

*Key_Y_Coordinate:* *Size: 32 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX | Local P-256 public key Y coordinate. |

#### 7.7.65.9 LE Generate DHKey Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Generate_DHKey_Complete | 0x3E | Subevent_Code,<br>Status,<br>DH_Key |

**Description:**

This event indicates that LE Diffie Hellman key generation has been completed by the Controller.

If the remote P-256 public key parameter of the HCI_LE_Generate_DHKey command (see Section 7.8.37) was invalid (see [Vol 3] Part H, Section 2.3.5.6.1), then all octets of the DH_Key parameter shall be set to 0xFF.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x09 | Subevent code for the HCI_LE_Generate_DHKey_Complete event. |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Generate_DHKey command completed successfully. |
| 0x01 to 0xFF | HCI_LE_Generate_DHKey command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*DH_Key:* *Size: 32 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX<br>XXXXXXXX<br>XXXXXXXX<br>XXXXXXXX<br>XXXXXXXX<br>XXXXXXXX<br>XXXXXXXX<br>XXXXXXXX | Diffie Hellman Key. |

#### 7.7.65.10 LE Enhanced Connection Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Enhanced_Connection_Complete [v2] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Role,<br>Peer_Address_Type,<br>Peer_Address,<br>Local_Resolvable_Private_Address,<br>Peer_Resolvable_Private_Address,<br>Connection_Interval,<br>Peripheral_Latency,<br>Supervision_Timeout,<br>Central_Clock_Accuracy,<br>Advertising_Handle,<br>Sync_Handle |
| HCI_LE_Enhanced_Connection_Complete [v1] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Role,<br>Peer_Address_Type,<br>Peer_Address,<br>Local_Resolvable_Private_Address,<br>Peer_Resolvable_Private_Address<br>Connection_Interval,<br>Peripheral_Latency,<br>Supervision_Timeout,<br>Central_Clock_Accuracy |

**Description:**

This event indicates to both of the Hosts forming the connection that a new connection has been created. Upon the creation of the connection a Connection_Handle shall be assigned by the Controller, and passed to the Host in this event. If the connection creation fails, this event shall be provided to the Host that had issued the HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command.

If this event is unmasked and the HCI_LE_Connection_Complete event is unmasked, only the HCI_LE_Enhanced_Connection_Complete event is sent when a new connection has been created.

This event indicates to the Host that issued an HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command and received an HCI_Command_Status event if the connection creation failed or was successful.

The Peer_Address, Peer_Resolvable_Private_Address, and Local_Resolvable_Private_Address shall always reflect the most recent packet sent and received on air.

The Central_Clock_Accuracy parameter is only valid for a Peripheral. On a Central, this parameter shall be set to 0x00.

If the connection is established from periodic advertising with responses and Role is 0x00, then the Advertising_Handle parameter shall be set according to the periodic advertising train the connection was established from. If the connection is established from periodic advertising with responses and Role is 0x01, then the Sync_Handle parameter shall be set according to the periodic advertising train the connection was established from. In all other circumstances, Advertising_Handle and Sync_Handle shall be set to No Advertising_Handle and No Sync_Handle and shall be ignored by the Host.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x29 | Subevent code for the HCI_LE_Enhanced_Connection_Complete event [v2] |
| 0x0A | Subevent code for the HCI_LE_Enhanced_Connection_Complete event [v1] |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01 to 0xFF | Connection failed to complete. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection is Central |
| 0x01 | Connection is Peripheral |
| All other values | Reserved for future use |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address (default) |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (Corresponds to a resolved RPA) |
| 0x03 | Random (Static) Identity Address (Corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, or Random Device Address, Public Identity Address or Random (static) Identity Address of the device to be connected. |

*Local_Resolvable_Private_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Resolvable Private Address being used by the local device for this connection. This is only valid when the Own_Address_Type (from the HCI_LE_Create_Connection, HCI_LE_Set_Advertising_Parameters, HCI_LE_Set_Extended_Advertising_Parameters, or HCI_LE_Extended_Create_Connection commands) is set to 0x02 or 0x03, and the Controller generated a resolvable private address for the local device using a non-zero local IRK. For other Own_Address_Type values, the Controller shall return all zeros. |

*Peer_Resolvable_Private_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Resolvable Private Address being used by the peer device for this connection. This is only valid for Peer_Address_Type 0x02 and 0x03. For other Peer_Address_Type values, the Controller shall return all zeros. |

*Connection_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection interval used on this connection.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4000 ms. |

*Peripheral_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Peripheral latency for the connection in number of connection events.<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection supervision timeout.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Central_Clock_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 500 ppm |
| 0x01 | 250 ppm |
| 0x02 | 150 ppm |
| 0x03 | 100 ppm |
| 0x04 | 75 ppm |
| 0x05 | 50 ppm |
| 0x06 | 30 ppm |
| 0x07 | 20 ppm |
| All other values | Reserved for future use |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |
| 0xFF | No Advertising_Handle |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train<br>Range: 0x0000 to 0x0EFF |
| 0xFFFF | No Sync_Handle |

#### 7.7.65.11 LE Directed Advertising Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Directed_Advertising_Report | 0x3E | Subevent_Code,<br>Num_Reports,<br>Event_Type[i],<br>Address_Type[i],<br>Address[i],<br>Direct_Address_Type[i],<br>Direct_Address[i],<br>RSSI[i] |

**Description:**

This event indicates that directed advertisements have been received where the advertiser is using a resolvable private address for the TargetA field of the advertising PDU which the Controller is unable to resolve and the Scanning_Filter_Policy is equal to 0x02 or 0x03, see Section 7.8.10. Direct_Address_Type and Direct_Address specify the address the directed advertisements are being directed to. Address_Type and Address specify the address of the advertiser sending the directed advertisements. The Controller may queue these advertising reports and send information from multiple advertisers in one HCI_LE_Directed_Advertising_Report event.

This event shall only be generated if scanning was enabled using the HCI_LE_Set_Scan_Enable command. It only reports advertising events that used legacy advertising PDUs.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0B | Subevent code for the HCI_LE_Directed_Advertising_Report event |

*Num_Reports:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x19 | Number of responses in event |
| All other values | Reserved for future use |

*Event_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Connectable directed legacy advertising (ADV_DIRECT_IND) |
| All other values | Reserved for future use |

*Address_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address (default) |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (Corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (Corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Address[i]:* *Size: Num_Reports × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address or Random (static) Identity Address of the advertising device. |

*Direct_Address_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Random Device Address (default) |
| All other values | Reserved for future use |

*Direct_Address[i]:* *Size: Num_Reports × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Random Device Address |

*RSSI[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |

### 7.7.65.12 LE PHY Update Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_PHY_Update_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>TX_PHY,<br>RX_PHY |

**Description:**

This event is used to indicate that the Controller has changed the transmitter PHY or receiver PHY in use.

If the Controller changes the transmitter PHY, the receiver PHY, or both PHYs, this event shall be issued.

If an HCI_LE_Set_PHY command was sent and the Controller determines that neither PHY will change as a result, it issues this event immediately.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0C | Subevent code for the HCI_LE_PHY_Update_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_PHY command succeeded or autonomous PHY update made by the Controller. |
| 0x01 to 0xFF | HCI_LE_Set_PHY command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*TX_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The transmitter PHY for the connection is LE 1M |
| 0x02 | The transmitter PHY for the connection is LE 2M |
| 0x03 | The transmitter PHY for the connection is LE Coded |
| All other values | Reserved for future use |

*RX_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The receiver PHY for the connection is LE 1M |
| 0x02 | The receiver PHY for the connection is LE 2M |
| 0x03 | The receiver PHY for the connection is LE Coded |
| All other values | Reserved for future use |

#### 7.7.65.13 LE Extended Advertising Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Extended_Advertising_Report | 0x3E | Subevent_Code,<br>Num_Reports,<br>Event_Type[i],<br>Address_Type[i],<br>Address[i],<br>Primary_PHY[i],<br>Secondary_PHY[i],<br>Advertising_SID[i],<br>TX_Power[i],<br>RSSI[i],<br>Periodic_Advertising_Interval[i],<br>Direct_Address_Type[i],<br>Direct_Address[i],<br>Data_Length[i],<br>Data[i] |

**Description:**

This event indicates that one or more Bluetooth devices have responded to an active scan or have broadcast advertisements that were received during a passive scan. The Controller may coalesce multiple advertising reports from the same or different advertisers into a single HCI_LE_Extended_Advertising_Report event, provided all the parameters from all the advertising reports fit in a single HCI event.

This event shall only be generated if scanning was enabled using the HCI_LE_Set_-Extended_Scan_Enable command. It reports advertising events using either legacy or extended advertising PDUs.

The Controller may split the data from a single advertisement or scan response (whether one PDU or several) into several reports. If so, each report except the last shall have an Event_Type with a data status field of "incomplete, more data to come", while the last shall have the value "complete"; the Address_Type, Address, Advertising_SID, Primary_PHY, and Secondary_PHY fields shall be the same in all the reports. No further reports shall be sent for a given advertisement or scan response after one with a Data_Status other than "incomplete, more data to come".

When a scan response is received, bits 0 to 2 and 4 of the event type shall indicate the properties of the original advertising event and the Advertising_SID field should be set to the value in the original scannable advertisement.

An Event_Type with a data status field of "incomplete, data truncated" shall indicate that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful or received it but was unable to store the data.

Where the event being reported used a legacy advertising PDU, the Controller shall set the Event_Type to the value specified in Table 7.1.

| PDU Type | Event_Type |
|---|---|
| ADV_IND | 0b0010011 |
| ADV_DIRECT_IND | 0b0010101 |
| ADV_SCAN_IND | 0b0010010 |
| ADV_NONCONN_IND | 0b0010000 |
| SCAN_RSP to an ADV_IND | 0b0011011 |
| SCAN_RSP to an ADV_SCAN_IND | 0b0011010 |

*Table 7.1: Event_Type values for legacy PDUs*

If the Event_Type indicates a legacy PDU (bit 4 = 1), the Primary_PHY parameter shall indicate the LE 1M PHY and the Secondary_PHY parameter shall be set to 0x00. Otherwise, the Primary_PHY parameter shall indicate the PHY used to send the advertising PDU on the primary advertising physical channel and the Secondary_PHY parameter shall indicate the PHY used to send the advertising PDU(s), if any, on the secondary advertising physical channel. If the Advertising Coding Selection (Host Support) Link Layer feature bit is set (see [Vol 6] Part B, Section 4.6) and the Primary_PHY or Secondary_PHY parameter indicates that the LE Coded PHY was used, then the parameter shall also indicate which coding was used.

The Periodic_Advertising_Interval parameter shall be set to zero when no periodic advertising exists as part of the advertising set.

The Direct_Address_Type and Direct_Address parameters shall contain the TargetA address in the advertising PDU for directed advertising event types (bit 2 = 1). These parameters shall be ignored for undirected advertising event types (bit 2 = 0). If the TargetA address is a resolvable private address that the Controller successfully resolved, then the value of Direct_Address_Type shall depend on the value of the Own_Address_Type parameter of the command that set the extended scan parameters. Direct_Address shall be set as follows:

- If Direct_Address_Type equals 0x02, then Direct_Address shall be set to either the TargetA field in the received advertisement or to the public device address of the scanning device.
- If Direct_Address_Type equals 0x03, then Direct_Address shall be set to either the TargetA field in the received advertisement or to the address set by the HCI_LE_Set_Random_Address command.

- Otherwise Direct_Address shall be set to the TargetA field in the received advertisement.

When multiple advertising packets are used to complete a single advertising report (e.g., a packet containing an ADV_EXT_IND PDU combined with one containing an AUX_ADV_IND PDU), the RSSI parameter shall be set based on the last packet received and the TX_Power parameter shall be based on the last packet of the current advertisement or scan response received that contains a TxPower field. If there is no packet containing a TxPower field, then TX_Power shall be set to 0x7F. However, if an event has been sent with a TX_Power value other than 0x7F and a Data_Status of "incomplete, more data to come", and if no subsequent PDU with a TxPower field has been received, then subsequent events may instead have a TX_Power value of 0x7F.

If the Controller receives an AUX_CHAIN_IND with no AdvData, it should send the report (or the last report if it has split the data) immediately without waiting for any subsequent AUX_CHAIN_IND PDUs.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0D | Subevent code for the HCI_LE_Extended_Advertising_Report event |

*Num_Reports:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0A | Number of separate reports in the event |
| All other values | Reserved for future use |

*Event_Type[i]:* *Size: Num_Reports × 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Connectable advertising |
| 1 | Scannable advertising |
| 2 | Directed advertising |
| 3 | Scan response |
| 4 | Legacy advertising PDUs used |

| Bit Number | Parameter Description |
|---|---|
| 5 to 6 | Data status:<br>0b00 = Complete<br>0b01 = Incomplete, more data to come<br>0b10 = Incomplete, data truncated, no more to come<br>0b11 = Reserved for future use |
| All other bits | Reserved for future use |

*Address_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (corresponds to a resolved RPA) |
| 0xFF | No address provided (anonymous advertisement) |
| All other values | Reserved for future use |

*Address[i]:* *Size: Num_Reports × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address or Random (static) Identity Address of the advertising device. |

*Primary_PHY[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Advertiser PHY is LE 1M |
| 0x03 | If the Advertising Coding Selection (Host Support) feature bit is set:<br>Advertising PHY is LE Coded with S=8 data coding<br>Otherwise:<br>Advertiser PHY is LE Coded |
| 0x04 | If the Advertising Coding Selection (Host Support) feature bit is set:<br>Advertising PHY is LE Coded with S=2 data coding<br>Otherwise:<br>Reserved for future use |
| All other values | Reserved for future use |

*Secondary_PHY[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No packets on the secondary advertising physical channel |
| 0x01 | Advertiser PHY is LE 1M |
| 0x02 | Advertiser PHY is LE 2M |
| 0x03 | If the Advertising Coding Selection (Host Support) feature bit is set:<br>Advertising PHY is LE Coded with S=8 data coding<br>Otherwise:<br>Advertiser PHY is LE Coded |
| 0x04 | If the Advertising Coding Selection (Host Support) feature bit is set:<br>Advertising PHY is LE Coded with S=2 data coding<br>Otherwise:<br>Reserved for future use |
| All other values | Reserved for future use |

*Advertising_SID[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Value of the Advertising SID subfield in the ADI field of the PDU or, for scan responses, in the ADI field of the original scannable advertisement |
| 0xFF | No ADI field provided |
| All other values | Reserved for future use |

*TX_Power[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | Tx Power information not available |

*RSSI[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |

*Periodic_Advertising_Interval[i]:* *Size: Num_Reports × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No periodic advertising |
| N = 0xXXXX | Interval of the periodic advertising<br>Range: 0x0006 to 0xFFFF<br>Time = N × 1.25 ms<br>Time Range<br>: 7.5 ms to 81,918.75 s |

*Direct_Address_Type[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Non-resolvable Private Address or Static Device Address |
| 0x02 | Resolvable Private Address (resolved by Controller; Own_Address_Type was 0x00 or 0x02) |
| 0x03 | Resolvable Private Address (resolved by Controller; Own_Address_Type was 0x01 or 0x03) |
| 0xFE | Resolvable Private Address (Controller unable to resolve) |
| All other values | Reserved for future use |

*Direct_Address[i]:* *Size: Num_Reports × 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | TargetA field in the advertisement or either Public Identity Address or Random (static) Identity Address of the target device |

*Data_Length[i]:* *Size: Num_Reports × 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 229 | Length of the Data[i] field for each device which responded |
| All other values | Reserved for future use |

*Data[i]:* *Size: SUM (Data_Length[i]) octets*

| Parameter Description |
|---|
| Data_Length[i] octets of advertising or scan response data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: Each element of this array has a variable length. |

#### 7.7.65.14 LE Periodic Advertising Sync Established event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Sync_Established [v2] | 0x3E | Subevent_Code,<br>Status,<br>Sync_Handle,<br>Advertising_SID,<br>Advertiser_Address_Type,<br>Advertiser_Address,<br>Advertiser_PHY,<br>Periodic_Advertising_Interval,<br>Advertiser_Clock_Accuracy,<br>Num_Subevents,<br>Subevent_Interval,<br>Response_Slot_Delay,<br>Response_Slot_Spacing |
| HCI_LE_Periodic_Advertising_Sync_Established [v1] | 0x3E | Subevent_Code,<br>Status,<br>Sync_Handle,<br>Advertising_SID,<br>Advertiser_Address_Type,<br>Advertiser_Address,<br>Advertiser_PHY,<br>Periodic_Advertising_Interval,<br>Advertiser_Clock_Accuracy |

**Description:**

This event indicates that the Controller has received the first periodic advertising packet from an advertiser after the HCI_LE_Periodic_Advertising_Create_Sync command has been sent to the Controller.

The Sync_Handle parameter identifies the periodic advertising train in subsequent commands and events and shall be assigned by the Controller.

The Advertising_SID parameter is set to the value of the Advertising SID subfield in the ADI field of the advertising PDU referring to the periodic advertising train.

The Advertiser_Address_Type and Advertiser_Address parameters specify the address of the periodic advertiser.

The Advertiser_PHY parameter specifies the PHY used for the periodic advertising.

The Periodic_Advertising_Interval parameter specifies the interval between the periodic advertising events.

The Advertiser_Clock_Accuracy parameter specifies the accuracy of the periodic advertiser's clock.

If the periodic advertising is with responses, then the Num_Subevents, Subevent_Interval, Response_Slot_Delay, and Response_Slot_Spacing specify the parameters for these subevents. If the periodic advertising is without responses, then these parameters shall be set to 0x00.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x24 | Subevent code for the HCI_LE_Periodic_Advertising_Sync_Established event [v2] |
| 0x0E | Subevent code for the HCI_LE_Periodic_Advertising_Sync_Established event [v1] |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising sync successful |
| 0x01 to 0xFF | Periodic advertising sync failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train<br>Range: 0x0000 to 0x0EFF |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Value of the Advertising SID subfield in the ADI field of the PDU |

*Advertiser_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Advertiser_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser |

*Advertiser_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Advertiser PHY is LE 1M |
| 0x02 | Advertiser PHY is LE 2M |
| 0x03 | Advertiser PHY is LE Coded |
| All other values | Reserved for future use |

*Periodic_Advertising_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Periodic advertising interval<br>Range: 0x0006 to 0xFFFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 81.91875 s |

*Advertiser_Clock_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 500 ppm |
| 0x01 | 250 ppm |
| 0x02 | 150 ppm |
| 0x03 | 100 ppm |
| 0x04 | 75 ppm |

| Value | Parameter Description |
|---|---|
| 0x05 | 50 ppm |
| 0x06 | 30 ppm |
| 0x07 | 20 ppm |
| All other values | Reserved for future use |

*Num_Subevents:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising without responses |
| N = 0xXX | Number of subevents.<br>Range: 0x01 to 0x80 |

*Subevent_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising without responses |
| N = 0xXX | Subevent interval.<br>Range: 0x06 to 0xFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 318.75 ms |

*Response_Slot_Delay:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising without responses |
| N = 0xXX | Response slot delay.<br>Range: 0x01 to 0xFE<br>Time = N × 1.25 ms<br>Time Range: 1.25 ms to 317.5 ms |

*Response_Slot_Spacing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising without responses |
| N = XX | Response slot spacing<br>Range: 0x02 to 0xFF<br>Time = N × 0.125 ms<br>Time Range: 0.25 ms to 31.875 |

#### 7.7.65.15 LE Periodic Advertising Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Report [v2] | 0x3E | Subevent_Code,<br>Sync_Handle,<br>TX_Power,<br>RSSI,<br>CTE_Type,<br>Periodic_Event_Counter,<br>Subevent,<br>Data_Status,<br>Data_Length,<br>Data |
| HCI_LE_Periodic_Advertising_Report [v1] | 0x3E | Subevent_Code,<br>Sync_Handle,<br>TX_Power,<br>RSSI,<br>CTE_Type,<br>Data_Status,<br>Data_Length,<br>Data |

**Description:**

This event indicates that the Controller has received a periodic advertisement or has failed to receive an AUX_SYNC_SUBEVENT_IND PDU.

The Sync_Handle parameter identifies the periodic advertising train that the report relates to.

The RSSI parameter contains the RSSI value, ignoring any Constant Tone Extension. If the Controller supports the Connectionless CTE Receiver feature, RSSI shall not be set to 0x7F. When multiple advertising packets are used to complete a periodic advertising report (e.g., a packet containing an AUX_SYNC_IND PDU combined with one containing an AUX_CHAIN PDU), the RSSI parameter shall be set based on the last packet received and the TX_Power parameter shall be set based on the AUX_SYNC_IND PDU if that contains a TxPower field and shall be set to 0x7F otherwise. However, the second or subsequent events for the same periodic advertisement may instead have a TX_Power value of 0x7F.

The Controller may split the data from a single periodic advertisement (whether one PDU or several) into several reports. If so, each report except the last shall have a

Data_Status of "incomplete, more data to come", while the last shall have the value "complete". No further reports shall be sent for a given periodic advertisement after one with a Data_Status other than "incomplete, more data to come".

A Data_Status of "incomplete, data truncated" indicates that the Controller attempted to receive an AUX_CHAIN_IND PDU but was not successful or received it but was unable to store the data.

The CTE_Type parameter indicates the type of Constant Tone Extension in the periodic advertising packets.

The Periodic_Event_Counter parameter indicates the periodic advertising event counter (*paEventCounter*) of the event that the periodic advertising packet was received in.

The Subevent parameter indicates the Periodic Advertising with Responses subevent that the periodic advertising packet was received in. If the Periodic Advertising does not have subevents, then Subevent shall be set to 0xFF.

If the Controller receives an AUX_CHAIN_IND PDU with no AdvData, it should send the report (or the last report if it has split the data) immediately without waiting for any subsequent AUX_CHAIN_IND PDUs.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x25 | Subevent code for the HCI_LE_Periodic_Advertising_Report event [v2] |
| 0x0F | Subevent code for the HCI_LE_Periodic_Advertising_Report event [v1] |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

*TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | Tx Power information not available |

*RSSI:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| 0xFF | No Constant Tone Extension |
| All other values | Reserved for future use |

*Periodic_Event_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The value of paEventCounter (see [Vol 6] Part B, Section 4.4.2.1) for the reported periodic advertising packet |

*Subevent:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The subevent number.<br>Range: 0x00 to 0x7F |
| 0xFF | No subevents |

*Data_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Data complete |
| 0x01 | Data incomplete, more data to come |
| 0x02 | Data incomplete, data truncated, no more to come |
| 0xFF | Failed to receive an AUX_SYNC_SUBEVENT_IND PDU |
| All other values | Reserved for future use |

*Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of the Data field |

*Data:* *Size: Data_Length octets*

| Parameter Description |
|---|
| Data received from a Periodic Advertising packet<br>Note: This parameter has a variable length. |

#### 7.7.65.16 LE Periodic Advertising Sync Lost event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Sync_Lost | 0x3E | Subevent_Code,<br>Sync_Handle |

**Description:**

This event indicates that the Controller has not received a Periodic Advertising packet from the train identified by Sync_Handle within the timeout period.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x10 | Subevent code for the HCI_LE_Periodic_Advertising_Sync_Lost event |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

#### 7.7.65.17 LE Scan Timeout event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Scan_Timeout | 0x3E | Subevent_Code |

**Description:**

This event indicates that scanning has ended because the duration has expired.

This event shall only be generated if scanning was enabled using the HCI_LE_Set_Extended_Scan_Enable command.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x11 | Subevent code for the HCI_LE_Scan_Timeout event |

#### 7.7.65.18 LE Advertising Set Terminated event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Advertising_Set_-Terminated | 0x3E | Subevent_Code,<br>Status,<br>Advertising_Handle,<br>Connection_Handle,<br>Num_Completed_Extended_Advertising_Events |

**Description:**

This event indicates that the Controller has terminated advertising in the advertising sets specified by the Advertising_Handle parameter.

This event shall be generated every time connectable advertising in an advertising set results in a connection being created or because the advertising duration or the maximum number of extended advertising events has been reached. It is generated for both legacy and extended advertising. It shall not be generated if the Host disables the advertising set.

This event shall only be generated if advertising was enabled using the HCI_LE_Set_-Extended_Advertising_Enable command.

The Connection_Handle parameter is only valid when advertising ends because a connection was created.

If the Max_Extended_Advertising_Events parameter in the HCI_LE_Set_Extended_-Advertising_Enable command was non-zero, the Num_Completed_Extended_-Advertising_Events parameter shall be set to the number of completed extended advertising events the Controller had transmitted when either the duration elapsed or the maximum number of extended advertising events was reached; otherwise it shall be set to zero.

If advertising has terminated as a result of the advertising duration elapsing, the Status parameter shall be set to the error code *Advertising Timeout* (0x3C).

If advertising has terminated because the Max_Extended_Advertising_Events was reached, the Status parameter shall be set to the error code *Limit Reached* (0x43).

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x12 | Subevent code for the HCI_LE_Advertising_Set_Terminated event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Advertising successfully ended with a connection being created |
| 0x01 to 0xFF | Advertising ended for another reason. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Advertising_Handle in which advertising has ended<br>Range: 0x00 to 0xEF |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle of the connection whose creation ended the advertising<br>Range: 0x0000 to 0x0EFF |

*Num_Completed_Extended_Advertising_Events:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of completed extended advertising events transmitted by the Controller |

#### 7.7.65.19 LE Scan Request Received event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Scan_Request_Received | 0x3E | Subevent_Code,<br>Advertising_Handle,<br>Scanner_Address_Type,<br>Scanner_Address |

**Description:**

This event indicates that a SCAN_REQ PDU or an AUX_SCAN_REQ PDU has been received by the advertiser. The request contains a device address from a scanner that is allowed by the advertising filter policy. The advertising set is identified by Advertising_Handle.

This event shall only be generated if advertising was enabled using the HCI_LE_Set_Extended_Advertising_Enable command.

The Scanner_Address_Type and Scanner_Address indicates the type of the address and the address of the scanner device.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x13 | Subevent code for the HCI_LE_Scan_Request_Received event |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Scanner_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Scanner_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address or Random (static) Identity Address of the advertising device |

#### 7.7.65.20 LE Channel Selection Algorithm event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Channel_Selection_Algorithm | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Channel_Selection_Algorithm |

**Description:**

The HCI_LE_Channel_Selection_Algorithm event indicates which channel selection algorithm is used on a data physical channel connection (see [Vol 6] Part B, Section 4.5.8).

This event is only generated following an HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event as specified in Section 7.8.12, Section 7.8.56, or Section 7.8.66.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x14 | Subevent code for the HCI_LE_Channel_Selection_Algorithm event |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Channel_Selection_Algorithm:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE Channel Selection Algorithm #1 is used |
| 0x01 | LE Channel Selection Algorithm #2 is used |
| All other values | Reserved for future use |

#### 7.7.65.21 LE Connectionless IQ Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Connectionless_IQ_Report | 0x3E | Subevent_Code,<br>Sync_Handle,<br>Channel_Index,<br>RSSI,<br>RSSI_Antenna_ID,<br>CTE_Type,<br>Slot_Durations,<br>Packet_Status,<br>Periodic_Event_Counter,<br>Sample_Count,<br>I_Sample[i],<br>Q_Sample[i] |

**Description:**

This event is used by the Controller to report IQ information from the Constant Tone Extension of a received advertising packet forming part of the periodic advertising train identified by Sync_Handle and to report IQ information from the Constant Tone Extension of a received Test Mode packet (see Section 7.8.28).

The index of the channel on which the packet was received, the RSSI of the packet (ignoring the Constant Tone Extension), the ID of the antenna on which this was measured, the type of Constant Tone Extension, the value of *paEventCounter*, and the IQ samples of the Constant Tone Extension of the advertisement are reported in the corresponding parameters. For any given sample, either both or neither of I_Sample[i] and Q_Sample[i] shall equal 0x80.

The Slot_Durations parameter specifies the sampling rate used by the Controller.

The Packet_Status parameter indicates whether the received packet had a valid CRC and, if not, whether the Controller has determined the position and size of the Constant Tone Extension using the Length and CTETime fields.

Note: A Controller is not required to generate this event for packets that have a bad CRC.

The Constant Tone Extension format is defined in [Vol 6] Part B, Section 2.5.1.

If the PDU contains AdvData, then any HCI_LE_Periodic_Advertising_Report event triggered by this PDU shall be generated before this event.

Not all PDUs in a periodic advertisement will necessarily trigger an HCI_LE_Periodic_Advertising_Report event. For example, PDUs without AdvData might not trigger that event (see [Vol 6] Part B, Section 4.4.5.1).

The Controller is not required to generate this event for a Constant Tone Extension with a type that it does not support.

This event is also used by the Controller to report that it has insufficient resources to report IQ samples for all received Constant Tone Extensions and has failed to sample at least once. In this case Packet_Status shall be set to 0xFF and Sample_Count to 0x00.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x15 | Subevent code for HCI_LE_Connectionless_IQ_Report event |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |
| 0x0FFF | Receiver Test |

*Channel_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x27 | The index of the channel on which the packet was received.<br>Note: 0x25 to 0x27 can be used only for packets generated during test modes. |
| All other values | Reserved for future use |

*RSSI:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | RSSI of the packet<br>Range: -1270 to +200<br>Units: 0.1 dBm |

*RSSI_Antenna_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID |

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Switching and sampling slots are 1 µs each |
| 0x02 | Switching and sampling slots are 2 µs each |
| All other values | Reserved for future use |

*Packet_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CRC was correct |
| 0x01 | CRC was incorrect and the Length and CTETime fields of the packet were used to determine sampling points |
| 0x02 | CRC was incorrect but the Controller has determined the position and length of the Constant Tone Extension in some other way |
| 0xFF | Insufficient resources to sample (Channel_Index, CTE_Type, and Slot_Durations invalid). |
| All other values | Reserved for future use |

*Periodic_Event_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The value of paEventCounter (see [Vol 6] Part B, Section 4.4.2.1) for the reported AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU |

*Sample_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No samples provided (only permitted if Packet_Status is 0xFF). |
| 0x09 to 0x52 | Total number of sample pairs (there shall be the same number of I samples and Q samples).<br>Note: This number is dependent on the switch and sample slot durations used. |
| All other values | Reserved for future use |

*I_Sample[i]:* *Size: Sample_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | I sample for the reported packet (signed integer).<br>The list is in the order of the sampling points within the packet. |

*Q_Sample[i]:* *Size: Sample Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | Q sample for the reported packet (signed integer).<br>The list is in the order of the sampling points within the packet. |

#### 7.7.65.22 LE Connection IQ Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Connection_IQ_Report | 0x3E | Subevent_Code,<br>Connection_Handle,<br>RX_PHY,<br>Data_Channel_Index,<br>RSSI,<br>RSSI_Antenna_ID,<br>CTE_Type,<br>Slot_Durations,<br>Packet_Status,<br>Connection_Event_Counter,<br>Sample_Count,<br>I_Sample[i],<br>Q_Sample[i] |

**Description:**

This event is used by the Controller to report the IQ samples from the Constant Tone Extension of a received packet (see [Vol 6] Part B, Section 2.4).

The Connection_Handle parameter identifies the connection that corresponds to the reported information.

The receiver PHY, the index of the data channel, the RSSI value of the packet (ignoring the Constant Tone Extension), the ID of the antenna on which this was measured, the type of Constant Tone Extension, the value of *connEventCounter*, and the IQ samples of the Constant Tone Extension of the received packet are reported in the corresponding parameters. For any given sample, either both or neither of I_Sample[i] and Q_Sample[i] shall equal 0x80.

The Slot_Durations parameter specifies the sampling rate used by the Controller.

The Packet_Status parameter indicates whether the received packet had a valid CRC and, if not, whether the Controller has determined the position and size of the Constant Tone Extension using the Length and CTETime fields.

Note: A Controller is not required to generate this event for packets that have a bad CRC.

This event is also used by the Controller to report that it has insufficient resources to report IQ samples for all received Constant Tone Extensions and has failed to sample at least once. In this case Packet_Status shall be set to 0xFF and Sample_Count to 0x00.

The Constant Tone Extension format is defined in [Vol 6] Part B, Section 2.1.5.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x16 | Subevent code for HCI_LE_Connection_IQ_Report event |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*RX_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The receiver PHY for the connection is LE 1M |
| 0x02 | The receiver PHY for the connection is LE 2M |
| All other values | Reserved for future use |

*Data_Channel_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x24 | The index of the data channel on which the Data Physical Channel PDU was received. |
| All other values | Reserved for future use |

*RSSI:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | RSSI of the packet<br>Range: -1270 to +200<br>Units: 0.1 dBm |

*RSSI_Antenna_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | ID of the antenna on which the RSSI is measured |

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Switching and sampling slots are 1 µs each |
| 0x02 | Switching and sampling slots are 2 µs each |
| All other values | Reserved for future use |

*Packet_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CRC was correct |
| 0x01 | CRC was incorrect and the Length and CTETime fields of the packet were used to determine sampling points |
| 0x02 | CRC was incorrect but the Controller has determined the position and length of the Constant Tone Extension in some other way |
| 0xFF | Insufficient resources to sample (Data_Channel_Index, CTE_Type, and Slot_Durations invalid). |
| All other values | Reserved for future use |

*Connection_Event_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The value of connEventCounter (see [Vol 6] Part B, Section 4.5.1) for the reported PDU |

*Sample_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No samples provided (only permitted if Packet_Status is 0xFF). |
| 0x09 to 0x52 | Total number of sample pairs (there shall be the same number of I samples and Q samples).<br>Note: This number is dependent on the switch and sample slot durations used. |
| All other values | Reserved for future use |

*I_Sample[i]:* *Size: Sample_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | I sample for the reported PDU (signed integer).<br>The list is in the order of the sampling points within the PDU. |

*Q_Sample[i]:* *Size: Sample_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | Q sample for the reported PDU (signed integer).<br>The list is in the order of the sampling points within the PDU. |

#### 7.7.65.23 LE CTE Request Failed event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CTE_Request_Failed | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle |

**Description:**

This event is used by the Controller to report an issue following a request to a peer device to reply with a packet containing an LL_CTE_RSP PDU and a Constant Tone Extension. It shall be generated if the packet containing the LL_CTE_RSP PDU sent in response did not contain a Constant Tone Extension or if the peer rejected the request. It shall not be generated if the packet containing the LL_CTE_RSP PDU had a CRC error or if the procedure response timeout timer (see [Vol 6] Part B, Section 5.2) expired.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x17 | Subevent code for HCI_LE_CTE_Request_Failed event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LL_CTE_RSP PDU received successfully but without a Constant Tone Extension field. |
| 0x01 to 0xFF | Peer rejected the request. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

#### 7.7.65.24 LE Periodic Advertising Sync Transfer Received event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Sync_Transfer_Received [v2] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Service_Data,<br>Sync_Handle,<br>Advertising_SID,<br>Advertiser_Address_Type,<br>Advertiser_Address,<br>Advertiser_PHY,<br>Periodic_Advertising_Interval,<br>Advertiser_Clock_Accuracy,<br>Num_Subevents,<br>Subevent_Interval,<br>Response_Slot_Delay,<br>Response_Slot_Spacing |
| HCI_LE_Periodic_Advertising_Sync_Transfer_Received [v1] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Service_Data,<br>Sync_Handle,<br>Advertising_SID,<br>Advertiser_Address_Type,<br>Advertiser_Address,<br>Advertiser_PHY,<br>Periodic_Advertising_Interval,<br>Advertiser_Clock_Accuracy |

**Description:**

This event is used by the Controller to report that it has received periodic advertising synchronization information from the device referred to by the Connection_Handle parameter and either successfully synchronized to the periodic advertising train or timed out while attempting to synchronize. The Status will be zero if it successfully synchronized and non-zero otherwise.

The Service_Data value is provided by the Host of the device sending the information.

The Sync_Handle identifies the periodic advertising in subsequent commands and events and shall be assigned by the Controller.

The remaining parameters provide information about the periodic advertising (see Section 7.7.65.14). If the transfer used an LL_PERIODIC_SYNC_IND PDU and so the periodic advertising is without responses, then the Controller shall set the Num_Subevents parameter to zero and the Host shall ignore the Subevent_Interval, Response_Slot_Delay, and Response_Slot_Spacing parameters. If the transfer used an LL_PERIODIC_SYNC_WR_IND PDU and so the periodic advertising is with responses, then the Controller shall set Num_Subevents to a non-zero value. If Num_Subevents equals 1, then the Host shall ignore Subevent_Interval.

If Status is non-zero, all parameter values are valid except Sync_Handle, which the Host shall ignore.

Note: If the Controller is already synchronized to the periodic advertising train described in the received information, no event will be generated.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x26 | Subevent code for the HCI_LE_Periodic_Advertising_Sync_Transfer_Received event [v2] |
| 0x18 | Subevent code for the HCI_LE_Periodic_Advertising_Sync_Transfer_Received event [v1] |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronization to the periodic advertising train succeeded. |
| 0x01 to 0xFF | Synchronization to the periodic advertising train failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Service_Data:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | A value provided by the peer device |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Value of the Advertising SID used to advertise the periodic advertising |
| All other values | Reserved for future use |

*Advertiser_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (corresponds to a resolved RPA) |
| 0x03 | Random (static) Identity Address (corresponds to a resolved RPA) |
| All other values | Reserved for future use |

*Advertiser_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser |

*Advertiser_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Advertiser PHY is LE 1M |
| 0x02 | Advertiser PHY is LE 2M |
| 0x03 | Advertiser PHY is LE Coded |
| All other values | Reserved for future use |

*Periodic_Advertising_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Periodic advertising interval<br>Range: 0x0006 to 0xFFFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 81.91875 s |

*Advertiser_Clock_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 500 ppm |
| 0x01 | 250 ppm |
| 0x02 | 150 ppm |
| 0x03 | 100 ppm |
| 0x04 | 75 ppm |
| 0x05 | 50 ppm |
| 0x06 | 30 ppm |
| 0x07 | 20 ppm |
| All other values | Reserved for future use |

*Num_Subevents:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic advertising without responses |
| N=0xXX | Number of subevents.<br>Range: 0x01 to 0x80 |

*Subevent_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Subevent interval.<br>Range: 0x06 to 0xFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 318.75 ms |

*Response_Slot_Delay:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Response slot delay.<br>Range: 0x01 to 0xFE<br>Time = N × 1.25 ms<br>Time Range: 1.25 ms to 317.5 ms |

*Response_Slot_Spacing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Response slot spacing<br>Range: 0x02 to 0xFF<br>Time = N × 0.125 ms<br>Time Range: 0.25 ms to 31.875 ms |

#### 7.7.65.25 LE CIS Established event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CIS_Established [v2] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>CIG_Sync_Delay,<br>CIS_Sync_Delay,<br>Transport_Latency_C_To_P,<br>Transport_Latency_P_To_C,<br>PHY_C_To_P,<br>PHY_P_To_C,<br>NSE,<br>BN_C_To_P,<br>BN_P_To_C,<br>FT_C_To_P,<br>FT_P_To_C,<br>Max_PDU_C_To_P,<br>Max_PDU_P_To_C,<br>ISO_Interval,<br>Sub_Interval,<br>Max_SDU_C_To_P,<br>Max_SDU_P_To_C,<br>SDU_Interval_C_To_P,<br>SDU_Interval_P_To_C,<br>Framing |

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CIS_Established [v1] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>CIG_Sync_Delay,<br>CIS_Sync_Delay,<br>Transport_Latency_C_To_P,<br>Transport_Latency_P_To_C,<br>PHY_C_To_P,<br>PHY_P_To_C,<br>NSE,<br>BN_C_To_P,<br>BN_P_To_C,<br>FT_C_To_P,<br>FT_P_To_C,<br>Max_PDU_C_To_P,<br>Max_PDU_P_To_C,<br>ISO_Interval |

**Description:**

This event indicates that a CIS has been established, was considered lost before being established, or—on the Central—was rejected by the Peripheral. It is generated by the Controller in the Central and Peripheral. The Connection_Handle parameter shall be set to the value provided in the HCI_LE_Create_CIS command on the Central and in the HCI_LE_CIS_Request event on the Peripheral.

The CIG_Sync_Delay parameter is the maximum time, in microseconds, for transmission of PDUs of all CISes in a CIG event (see [Vol 6] Part B, Section 4.5.14.1).

The CIS_Sync_Delay parameter is the maximum time, in microseconds, for transmission of PDUs of the specified CIS in a CIG event (see [Vol 6] Part B, Section 4.5.14.1).

The Transport_Latency_C_To_P and Transport_Latency_P_To_C parameters are the actual transport latencies, in microseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2.

The PHY_C_To_P parameter indicates the PHY selected for packets from the Central to Peripheral.

The PHY_P_To_C parameter indicates the PHY selected for packets from the Peripheral to Central.

The remaining parameters are the corresponding parameters of the CIS (see [Vol 6] Part B, Section 4.5.13.1).

If this event is generated on the Peripheral with a non-zero status, the Controller shall delete the Connection_Handle and any associated ISO data paths.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x2A | Subevent Code for HCI_LE_CIS_Established [v2] event |
| 0x19 | Subevent Code for HCI_LE_CIS_Established [v1] event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The CIS is successfully established. |
| 0x01 to 0xFF | The CIS failed to be established. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS<br>Range: 0x0000 to 0x0EFF |

*CIG_Sync_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000F2 to 0x7FFFFF | The maximum time, in microseconds, for transmission of PDUs of all CISes in a CIG event |
| All other values | Reserved for future use |

*CIS_Sync_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000F2 to 0x7FFFFF | The maximum time, in microseconds, for transmission of PDUs of the specified CIS in a CIG event |
| All other values | Reserved for future use |

*Transport_Latency_C_To_P:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000F2 to 0x7FFFFF | The actual transport latency, in microseconds, from Central to Peripheral |
| All other values | Reserved for future use |

*Transport_Latency_P_To_C:* *Size: 3 octets*

| Bit Number | Parameter Description |
|---|---|
| 0x0000F2 to 0x7FFFFF | The actual transport latency, in microseconds, from Peripheral to Central |
| All other values | Reserved for future use |

*PHY_C_To_P:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The transmitter PHY of packets from the Central is LE 1M. |
| 0x02 | The transmitter PHY of packets from the Central is LE 2M. |
| 0x03 | The transmitter PHY of packets from the Central is LE Coded. |
| All other values | Reserved for future use. |

*PHY_P_To_C:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The transmitter PHY of packets from the Peripheral is LE 1M. |
| 0x02 | The transmitter PHY of packets from the Peripheral is LE 2M. |
| 0x03 | The transmitter PHY of packets from the Peripheral is LE Coded. |
| All other values | Reserved for future use. |

*NSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Maximum number of subevents in each CIS event |
| All other values | Reserved for future use |

*BN_C_To_P:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No isochronous data from the Central to the Peripheral |
| 0x01 to 0x0F | The burst number for Central to Peripheral transmission |
| All other values | Reserved for future use |

*BN_P_To_C:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No isochronous data from the Peripheral to the Central |
| 0x01 to 0x0F | The burst number for Peripheral to Central transmission |
| All other values | Reserved for future use |

*FT_C_To_P:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The flush timeout, in multiples of the ISO_Interval for the CIS, for each payload sent from the Central to the Peripheral.<br>Range: 0x01 to 0xFF |

*FT_P_To_C:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The flush timeout, in multiples of the ISO_Interval for the CIS, for each payload sent from the Peripheral to the Central.<br>Range: 0x01 to 0xFF |

*Max_PDU_C_To_P:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x00FB | Maximum size, in octets, of the payload from Central to Peripheral |
| All other values | Reserved for future use |

*Max_PDU_P_To_C:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x00FB | Maximum size, in octets, of the payload from Peripheral to Central |
| All other values | Reserved for future use |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The time between two consecutive CIS anchor points.<br>Range: 0x0004 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 5 ms to 4 s. |

*Sub_Interval:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000000 | NSE = 1 (meaning there is no Sub_Interval) |
| 0xXXXXXX | Time, in microseconds, between the start of consecutive subevents in a CIS event<br>Range: 0x000190 to ISO_Interval×1250 – 1 |

*Max_SDU_C_To_P:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum size, in octets, of the payload from the Central's Host<br>Range: 0 to 0x0FFF |

*Max_SDU_P_To_C:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum size, in octets, of the payload from the Peripheral's Host<br>Range: 0 to 0x0FFF |

*SDU_Interval_C_To_P:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Time, in microseconds, between the start of consecutive SDUs sent by the Central<br>Range: 0x0000FF to 0x0FFFFF |

*SDU_Interval_P_To_C:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Time, in microseconds, between the start of consecutive SDUs sent by the Peripheral<br>Range: 0x0000FF to 0x0FFFFF |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

### 7.7.65.26 LE CIS Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CIS_Request | 0x3E | Subevent_Code,<br>ACL_Connection_Handle,<br>CIS_Connection_Handle,<br>CIG_ID,<br>CIS_ID |

**Description:**

This event indicates that a Controller has received a request to establish a CIS. If the Controller receives such a request while the HCI_LE_CIS_Request event is masked away, it shall reject it. Otherwise the Controller shall assign a connection handle for the requested CIS and send the handle in the CIS_Connection_Handle parameter of the event.

When the Host receives this event it shall respond with either an HCI_LE_Accept_CIS_Request command or an HCI_LE_Reject_CIS_Request command before the timer Connection_Accept_Timeout expires. If it does not, the Controller shall reject the request and generate an HCI_LE_CIS_Established event with the status *Connection Accept Timeout Exceeded* (0x10).

The ACL_Connection_Handle is the connection handle of the ACL connection that is associated with the requested CIS.

The CIG_ID parameter contains the identifier of the CIG that contains the requested CIS. This parameter is sent by the Central in the request to establish the CIS.

The CIS_ID parameter contains the identifier of the requested CIS. This parameter is sent by the Central in the request to establish the CIS.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1A | Subevent Code for the HCI_LE_CIS_Request event |

*ACL_Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the ACL<br>Range: 0x0000 to 0x0EFF |

*CIS_Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS<br>Range: 0x0000 to 0x0EFF |

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifier of the CIG |
| All other values | Reserved for future use |

*CIS_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifier of the CIS |
| All other values | Reserved for future use |

#### 7.7.65.27 LE Create BIG Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Create_BIG_Complete | 0x3E | Subevent_Code,<br>Status,<br>BIG_Handle,<br>BIG_Sync_Delay,<br>Transport_Latency_BIG,<br>PHY,<br>NSE,<br>BN,<br>PTO,<br>IRC,<br>Max_PDU,<br>ISO_Interval,<br>Num_BIS,<br>Connection_Handle[i] |

**Description:**

This event indicates that the HCI_LE_Create_BIG or HCI_LE_Create_BIG_Test command has completed and, if successful, the Link Layer has entered the Isochronous Broadcasting state.

The BIG_Handle parameter shall be the same as the one specified in the command that has completed.

The BIG_Sync_Delay parameter is the maximum time, in microseconds, for transmission of PDUs of all BISes in a BIG event (see [Vol 6] Part B, Section 4.4.6.4).

The Transport_Latency_BIG parameter is the actual transport latency, in microseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2.

The Num_BIS parameter is the total number of BISes in the BIG. This parameter shall be the same as the parameter that is provided by the Host in the command that has completed.

The PHY parameter is the PHY used to create the BIG.

The NSE, BN, PTO, IRC, Max_PDU, and ISO_Interval parameters are the corresponding parameters of the BIS (see [Vol 6] Part B, Section 4.4.6.3).

The Connection_Handle arrayed parameter contains the connection handles of all the BIS in the BIG.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1B | Subevent Code for the HCI_LE_Create_BIG_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The BIG was successfully created. |
| 0x01 to 0xFF | There was an error creating the BIG. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | The identifier of the BIG |
| All other values | Reserved for future use |

*BIG_Sync_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000030 to 0x7FFFFF | The maximum time, in microseconds, for transmission of PDUs of all BISes in a BIG event |
| All other values | Reserved for future use. |

*Transport_Latency_BIG:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000030 to 0x7FFFFF | The actual transport latency, in microseconds |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The PHY used to create the BIG is LE 1M. |
| 0x02 | The PHY used to create the BIG is LE 2M. |

| Value | Parameter Description |
|---|---|
| 0x03 | The PHY used to create the BIG is LE Coded. |
| All other values | Reserved for future use. |

*NSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | The number of subevents in each BIS event in the BIG |
| All other values | Reserved for future use |

*BN:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | The number of new payloads in each BIS event |
| All other values | Reserved for future use |

*PTO:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Offset used for pre-transmissions |
| All other values | Reserved for future use |

*IRC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0F | The number of times a payload is transmitted in a BIS event |
| All other values | Reserved for future use |

*Max_PDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x00FB | Maximum size, in octets, of the payload |
| All other values | Reserved for future use |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The time between two consecutive BIG anchor points.<br>Range: 0x0004 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 5 ms to 4 s. |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of BISes in the BIG |
| All other values | Reserved for future use |

*Connection_Handle[i]:* *Size: 2 octets (12 bits meaningful) × Num_BIS*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a BIS<br>Range: 0x0000 to 0x0EFF |

#### 7.7.65.28 LE Terminate BIG Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Terminate_BIG_Complete | 0x3E | Subevent_Code,<br>BIG_Handle,<br>Reason |

**Description:**

This event indicates that the transmission of all the BISes in the BIG are terminated.

The BIG_Handle parameter is used to identify the BIG that is terminated. This parameter is provided by the Host in the HCI_LE_Terminate_BIG command.

If the BIG is terminated by the local Host, the Reason parameter shall be set to the error code *Connection Terminated by Local Host* (0x16).

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1C | Subevent Code for the HCI_LE_Terminate_BIG_Complete event |

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | The identifier of the BIG |
| All other values | Reserved for future use |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason for termination. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

#### 7.7.65.29 LE BIG Sync Established event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_BIG_Sync_Established | 0x3E | Subevent_Code,<br>Status,<br>BIG_Handle,<br>Transport_Latency_BIG,<br>NSE,<br>BN,<br>PTO,<br>IRC,<br>Max_PDU,<br>ISO_Interval,<br>Num_BIS,<br>Connection_Handle[i] |

**Description:**

This event indicates that the HCI_LE_BIG_Create_Sync command has completed.

The BIG_Handle parameter is used to identify the BIG. This parameter is provided by the Host in the HCI_LE_BIG_Create_Sync command.

The Transport_Latency_BIG parameter is the actual transport latency, in microseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2.

The NSE, BN, PTO, IRC, Max_PDU, and ISO_Interval parameters are the corresponding parameters of the BIS (see [Vol 6] Part B, Section 4.4.6.3).

The Num_BIS parameter indicates the number of BISes in the synchronized BIG specified by the HCI_LE_BIG_Create_Sync command. This parameter shall be the same as the parameter that is provided by the Host in the HCI_LE_BIG_Create_Sync command.

The Connection_Handle arrayed parameter is the list of connection handle(s) of the BIS(es) that are requested in the HCI_LE_BIG_Create_Sync command. The order of the connection handle(s) shall correspond to the order of the BIS(s) that are requested in the BIS arrayed parameter field of the HCI_LE_BIG_Create_Sync command.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1D | Subevent Code for the HCI_LE_BIG_Sync_Established event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronization to the BIG is completed. |
| 0x01 to 0xFF | Synchronization to the BIG failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | The identifier of the BIG |
| All other values | Reserved for future use |

*Transport_Latency_BIG:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000030 to 0x7FFFFF | The actual transport latency, in microseconds |
| All other values | Reserved for future use |

*NSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | The number of subevents in each BIS event in the BIG |
| All other values | Reserved for future use |

*BN:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | The number of new payloads in each BIS event |
| All other values | Reserved for future use |

*PTO:* Size: 1 octet

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Offset used for pre-transmissions |
| All other values | Reserved for future use |

*IRC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0F | The number of times a payload is transmitted in a BIS event |
| All other values | Reserved for future use |

*Max_PDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x00FB | Maximum size, in octets, of the payload |
| All other values | Reserved for future use |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The time between two consecutive BIG anchor points.<br>Range: 0x0004 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 5 ms to 4 s. |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of BISes in the BIG |
| All other values | Reserved for future use |

*Connection_Handle[i]:* *Size: Num_BIS × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a BIS in the BIG<br>Range: 0x0000 to 0x0EFF |

#### 7.7.65.30 LE BIG Sync Lost event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_BIG_Sync_Lost | 0x3E | Subevent_Code,<br>BIG_Handle,<br>Reason |

**Description:**

This event indicates that the Controller has not received any PDUs on a BIG within the timeout period BIG_Sync_Timeout or the BIG has been terminated by the remote device.

The BIG_Handle parameter is used to identify the BIG. This parameter is provided by the Host in the HCI_LE_BIG_Create_Sync command.

The Reason parameter is used to indicate the reason why the synchronization was lost or terminated. If synchronization was terminated due to the Broadcaster terminating the BIG, the Reason parameter shall be set to the error code *Remote User Terminated Connection* (0x13). If synchronization was terminated due to a timeout, the Reason parameter shall be set to the error code *Connection Timeout* (0x08). If synchronization was terminated due to a MIC failure, the Reason parameter shall be set to the error code *Connection Terminated due to MIC Failure* (0x3D).

When the HCI_LE_BIG_Sync_Lost event occurs, the Controller shall remove the connection handle(s) and data paths of all BIS(s) in the BIG with which the Controller was synchronized.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1E | Subevent Code for the HCI_LE_BIG_Sync_Lost event |

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | The identifier of a BIG |
| All other values | Reserved for future use |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The synchronization to BIG is terminated. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

#### 7.7.65.31 LE Request Peer SCA Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Request_Peer_SCA_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Peer_Clock_Accuracy |

**Description:**

This event indicates that the HCI_LE_Request_Peer_SCA command has been completed.

The Peer_Clock_Accuracy parameter contains the sleep clock accuracy of the peer.

The Connection_Handle is the connection handle of the ACL connection in which the HCI_LE_Request_Peer_SCA command is issued.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x1F | Subevent Code for the HCI_LE_Request_Peer_SCA_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Peer_Clock_Accuracy parameter is successfully received. |
| 0x01 to 0xFF | The reception of Peer_Clock_Accuracy parameter failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the ACL<br>Range: 0x0000 to 0x0EFF |

*Peer_Clock_Accuracy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 251 ppm to 500 ppm |
| 0x01 | 151 ppm to 250 ppm |
| 0x02 | 101 ppm to 150 ppm |
| 0x03 | 76 ppm to 100 ppm |
| 0x04 | 51 ppm to 75 ppm |
| 0x05 | 31 ppm to 50 ppm |
| 0x06 | 21 ppm to 30 ppm |
| 0x07 | 0 ppm to 20 ppm |
| All other values | Reserved for future use |

#### 7.7.65.32 LE Path Loss Threshold event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Path_Loss_Threshold | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Current_Path_Loss,<br>Zone_Entered |

**Description:**

This event is used to report a path loss threshold crossing (see [Vol 6] Part B, Section 4.5.16) on the ACL connection identified by the Connection_Handle parameter.

The Current_Path_Loss parameter indicates the current path loss value as calculated by the Controller.

The Zone_Entered parameter indicates which zone was entered. If Current_Path_Loss is set to 0xFF then Zone_Entered shall be ignored.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x20 | Subevent code for the HCI_LE_Path_Loss_Threshold event |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Current_Path_Loss:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Current path loss (always zero or positive)<br>Units: dB |
| 0xFF | Unavailable |

*Zone_Entered:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Entered low zone |
| 0x01 | Entered middle zone |
| 0x02 | Entered high zone |
| All other values | Reserved for future use |

#### 7.7.65.33 LE Transmit Power Reporting event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Transmit_Power_Reporting | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Reason,<br>PHY,<br>TX_Power_Level,<br>TX_Power_Level_Flag,<br>Delta |

**Description:**

This event is used to report the transmit power level on the ACL connection identified by the Connection_Handle parameter.

The Reason parameter indicates why the event was sent and the device whose transmit power level is being reported.

Whenever the transmit power changes and local reporting has been enabled by the HCI_LE_Set_Transmit_Power_Reporting_Enable command, the Controller shall generate this event with Reason set to 0x00. In this case, the PHY, TX_Power_Level, TX_Power_Level_Flag and Delta parameters shall refer to the local device and the Status parameter shall be ignored.

Whenever the Controller becomes aware that the peer's transmitter power has changed other than through an HCI_LE_Read_Remote_Transmit_Power_Level command and remote reporting has been enabled by the HCI_LE_Set_Transmit_Power_Reporting_Enable command, the Controller shall generate this event with Reason set to 0x01. In this case, the PHY, TX_Power_Level, TX_Power_Level_Flag and Delta parameters shall refer to the remote device and the Status parameter shall be ignored.

When the Reason is set to 0x02, this event indicates completion of an HCI_LE_Read_Remote_Transmit_Power_Level command. In this case, the PHY, TX_Power_Level, TX_Power_Level_Flag and Delta parameters shall refer to the remote device.

The PHY parameter shall indicate the PHY involved (which might not be the current transmit PHY for the relevant device).

The TX_Power_Level parameter shall indicate the transmit power level for the PHY.

The TX_Power_Level_Flag parameter shall indicate whether the transmit power level that is being reported has reached its minimum and/or maximum level. TX_Power_Level_Flag shall be ignored if the TX_Power_Level parameter is set to 0x7E or 0x7F.

The Delta parameter shall be set to the change in power level for the transmitter being reported, whenever it changes its transmit power level. When this event is generated with Reason set to 0x02, Delta shall be set to zero. Delta shall be ignored if the TX_Power_Level parameter is set to 0x7E.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x21 | Subevent code for the HCI_LE_Transmit_Power_Reporting event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Remote_Transmit_Power_Level command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Remote_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Local transmit power changed |
| 0x01 | Remote transmit power changed |
| 0x02 | HCI_LE_Read_Remote_Transmit_Power_Level command completed |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |
| 0x04 | LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

*TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Transmit power level<br>Range: -127 to 20<br>Units: dBm |
| 0x7E | Remote device is not managing power levels on this PHY. |
| 0x7F | Transmit power level is not available |

*TX_Power_Level_Flag:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Transmit power level is at minimum level |
| 1 | Transmit power level is at maximum level |
| All other bits | Reserved for future use |

*Delta:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Change in transmit power level (positive indicates increased power, negative indicates decreased power, zero indicates unchanged)<br>Units: dB |
| 0x7F | Change is not available or is out of range. |

#### 7.7.65.34 LE BIGInfo Advertising Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_BIGInfo_Advertising_Report | 0x3E | Subevent_Code,<br>Sync_Handle<br>Num_BIS,<br>NSE,<br>ISO_Interval,<br>BN,<br>PTO,<br>IRC,<br>Max_PDU,<br>SDU_Interval,<br>Max_SDU,<br>PHY,<br>Framing,<br>Encryption |

**Description:**

This event indicates that the Controller has received an Advertising PDU that contained a BIGInfo field. If the Controller also generates an HCI_LE_Periodic_Advertising_Report event, the HCI_LE_BIGInfo_Advertising_Report event shall immediately follow that event.

An HCI_LE_BIGInfo_Advertising_Report event shall be generated even if the Controller is already synchronized to the BIG.

The Sync_Handle parameter shall identify the periodic advertising train containing the BIGInfo field and shall be the same as the corresponding field in the HCI_LE_Periodic_Advertising_Report event if one is generated.

The Num_BIS, NSE, ISO_Interval, BN, PTO, IRC, Max_PDU, SDU_Interval, Max_SDU, and PHY parameters correspond to the associated fields in the BIGInfo field of the Advertising PDU. The Framing parameter corresponds to both the Framed and Framing_Mode fields in the BIGInfo field.

If the BIGInfo field indicates that the corresponding BIG is encrypted, the Encryption parameter shall be set to 0x01. Otherwise, the Encryption parameter shall be set to 0x00.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x22 | Subevent code for the HCI_LE_BIGInfo_Advertising_Report event |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Value of the Num_BIS subfield of the BIGInfo field |
| All other values | Reserved for future use |

*NSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Value of the NSE subfield of the BIGInfo field |
| All other values | Reserved for future use |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Value of the ISO_Interval subfield of the BIGInfo field |

*BN:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | Value of the BN subfield of the BIGInfo field |
| All other values | Reserved for future use |

*PTO:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Value of the PTO subfield of the BIGInfo field |
| All other values | Reserved for future use |

*IRC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0F | Value of the IRC subfield of the BIGInfo field |
| All other values | Reserved for future use |

*Max_PDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x00FB | Value of the Max_PDU subfield of the BIGInfo |
| All other values | Reserved for future use |

*SDU_Interval:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | Value of the SDU_Interval subfield of the BIGInfo field |
| All other values | Reserved for future use |

*Max_SDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x0FFF | Value of the Max_SDU subfield of the BIGInfo field in the Advertising PDU |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The BIG is transmitted on the LE 1M PHY |
| 0x02 | The BIG is transmitted on the LE 2M PHY |
| 0x03 | The BIG is transmitted on the LE Coded PHY |
| All other values | Reserved for future use |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

*Encryption:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | BIG carries unencrypted data |
| 0x01 | BIG carries encrypted data |
| All other values | Reserved for future use |

#### 7.7.65.35 LE Subrate Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Subrate_Change | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Subrate_Factor,<br>Peripheral_Latency,<br>Continuation_Number,<br>Supervision_Timeout |

**Description:**

This event is used to indicate that a Connection Subrate Update procedure has completed and some parameters of the specified connection have changed.

This event shall be issued if the HCI_LE_Subrate_Request command was issued by the Host or the parameters are updated successfully following a request from the peer device. If no parameters are updated following a request from the peer device or the parameters were changed using the Connection Update procedure or the Connection Rate Update procedure, then this event shall not be issued.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x23 | Subevent Code for the HCI_LE_Subrate_Change event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Subrate_Request command succeeded or this event was generated following a request from the peer device. |
| 0x01 to 0xFF | The HCI_LE_Subrate_Request command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle<br>Range: 0x0000 to 0x0EFF |

*Subrate_Factor:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | New subrate factor applied to the specified underlying connection interval<br>Range 0x0001 to 0x01F4 |
| All other values | Reserved for future use |

*Peripheral_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | New Peripheral latency for the connection in number of subrated connection events<br>Range: 0x0000 to 0x01F3 |
| All other values | Reserved for future use |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received<br>Range: 0x0000 to 0x01F3 |
| All other values | Reserved for future use |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | New supervision timeout for this connection.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

#### 7.7.65.36 LE Periodic Advertising Subevent Data Request event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Subevent_Data_Request | 0x3E | Subevent_Code,<br>Advertising_Handle,<br>Subevent_Start,<br>Subevent_Data_Count |

**Description:**

This event is used to allow the Controller to indicate that it is ready to transmit one or more subevents and is requesting the advertising data for these subevents. The Subevent_Data_Count parameter shall be less than or equal to the number of subevents. The Subevent_Start parameter is the first subevent being requested and the Subevent_Data_Count parameter determines the subsequent subevents being requested. The subevent numbers wrap from one less than the number of subevents to zero.

This event should be sent from the Controller when it has no data for upcoming subevents. The Controller should request data for as many subevents as it has memory to accept to minimize the number of events generated by the Controller.

The Controller shall not send this event more than once for the same subevent of the same periodic advertising event.

This event shall override any outstanding request for subevent data that the Controller sent in the past for this Advertising_Handle.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x27 | Subevent code for the HCI_LE_Periodic_Advertising_Subevent_Data_Request event |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

*Subevent_Start:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The first subevent that data is requested for.<br>Range: 0x00 to 0x7F |

*Subevent_Data_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of subevents that data is requested for.<br>Range: 0x01 to 0x80 |

### 7.7.65.37 LE Periodic Advertising Response Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Periodic_Advertising_Response_Report | 0x3E | Subevent_Code,<br>Advertising_Handle,<br>Subevent,<br>TX_Status,<br>Num_Responses,<br>TX_Power[i],<br>RSSI[i],<br>CTE_Type[i],<br>Response_Slot[i]<br>Data_Status[i],<br>Data_Length[i],<br>Data[i] |

**Description:**

This event indicates that one or more Bluetooth devices have responded to a periodic advertising subevent during a PAwR train. The Controller may queue these advertising reports and send information from multiple devices in one HCI_LE_Periodic_Advertising_Response_Report event. The Controller shall only send reports for response slots that the Host requested it listen to.

The Controller may fail to transmit the synchronization packet required to enable the response packets to be sent. If this happens, the Controller shall report this to the Host using the TX_Status parameter.

The Controller may split the data from a single response into several reports. If so, each report except the last shall have a Data_Status of "incomplete, more data to come", while the last shall have the value "complete". No further reports shall be sent for a given response slot after one with a Data_Status other than "incomplete, more data to come".

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x28 | Subevent code for the HCI_LE_Periodic_Advertising_Response_Report event |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

*Subevent:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The subevent number.<br>Range: 0x00 to 0x7F |

*TX_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AUX_SYNC_SUBEVENT_IND packet was transmitted. |
| 0x01 | AUX_SYNC_SUBEVENT_IND packet was not transmitted. |
| All other values | Reserved for future use |

*Num_Responses:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x19 | Number of responses in event. |
| All other values | Reserved for future use |

*TX_Power[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | Tx Power information not available |

*RSSI[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |

*CTE_Type[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| 0xFF | No Constant Tone Extension |
| All other values | Reserved for future use |

*Response_Slot[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The response slot the data was received in.<br>Range: 0x00 to 0xFF |

*Data_Status[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Data complete |
| 0x01 | Data incomplete, more data to come |
| 0xFF | Failed to receive or listen for an AUX_SYNC_SUBEVENT_RSP PDU |
| All other values | Reserved for future use |

*Data_Length[i]:* *Size: Num_Responses × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of the Data field |

*Data[i]:* *Size: SUM (Data_Length[i]) octets*

| Parameter Description |
|---|
| Periodic advertising response data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: Each element of this array has a variable length. |

### 7.7.65.38 LE Read All Remote Features Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Read_All_Remote_Features_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Max_Remote_Page,<br>Max_Valid_Page,<br>LE_Features |

**Description:**

This event is used to indicate the completion of the process of the Controller obtaining the features supported by the remote Bluetooth device specified by the Connection_Handle event parameter.

The Max_Remote_Page parameter specifies the highest-numbered page of the remote device's supported LE features that contains at least one bit set to 1; all higher-number pages therefore only contain zeroes. The Max_Valid_Page parameter specifies the highest-numbered page of features that the Controller has obtained from the remote device or, if it has obtained all pages from 1 to Max_Remote_Page, then any value greater than or equal to Max_Remote_Page.

The LE_Features parameter contains the LE features. The Controller shall set all pages between 0 and Max_Valid_Page to valid data and shall set all higher-numbered pages to all zero bits.

Note: If Max_Valid_Page ≥ Max_Remote_Page, then all pages will contain valid data, which will be all zero bits for pages numbered greater than Max_Remote_Page.

If the feature mask is requested more than once while a connection exists between the two devices, then the second and subsequent requests may report a cached copy of the feature mask rather than fetching the feature mask again.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x2B | Subevent code for the HCI_LE_Read_All_Remote_Features_Complete event. |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_All_Remote_Features command successfully completed. |
| 0x01 to 0xFF | HCI_LE_Read_All_Remote_Features command failed to complete. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Max_Remote_Page:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of the highest-numbered page of the remote device's supported LE features that contains at least one bit set to 1.<br>Range: 0x00 to 0x0A |

*Max_Valid_Page:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of the highest-numbered page of LE_Features that contains valid data.<br>Range: 0x00 to 0x0A |

*LE_Features:* *Size: 248 octets*

| Value | Parameter Description |
|---|---|
| 0xXX...XX | Bit Mask List of the LE features. See [Vol 6] Part B, Section 4.6. |

#### 7.7.65.39 LE CS Read Remote Supported Capabilities Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Read_Remote_Supported_Capabilities_Complete [v2] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability,<br>T_IP2_IPT_Times_Supported,<br>T_SW_IPT_Times_Supported,<br>RTT_2M_AA_Only_N,<br>RTT_2M_Sounding_N,<br>RTT_2M_Random_Sequence_N |

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Read_Remote_Supported_-Capabilities_Complete [v1] | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability |

**Description:**

This event shall be generated when a locally initiated CS Capabilities Exchange procedure has completed or when the local Controller has received an LL_CS_CAPABILITIES_REQ from the remote Controller.

The Num_Config_Supported parameter indicates the number of CS configurations that are supported per connection by the remote Controller.

The Max_Consecutive_Procedures_Supported parameter indicates the maximum number of consecutive CS procedures that are supported by the remote Controller.

The Num_Antennae_Supported parameter indicates the number of antenna elements that are available for CS tone exchanges.

The Max_Antenna_Paths_Supported parameter indicates the maximum number of antenna paths that are supported by the remote Controller for CS tone exchanges.

The Roles_Supported parameter indicates the CS roles that are supported by the remote Controller.

The Modes_Supported parameter indicates the optional CS modes that are supported by the remote Controller.

The RTT_Capability, RTT_AA_Only_N, RTT_2M_AA_Only_N, RTT_Sounding_N, RTT_2M_Sounding_N, RTT_Random_Sequence_N, and RTT_2M_Random_Sequence_N parameters indicate the time-of-flight accuracy as described in [Vol 6] Part B, Section 2.4.2.44.

The NADM_Sounding_Capability and NADM_Random_Capability parameters indicate the support by the remote Controller for reporting Normalized Attack Detector Metric (NADM) when a CS_SYNC with a sounding sequence or random sequence is received.

The CS_SYNC_PHYs_Supported parameter indicates the optional transmit and receive PHYs supported by the remote Controller for CS_SYNC exchanges as described in [Vol 6] Part H, Section 4.3.

The Subfeatures_Supported parameter indicates which of the following optional subfeatures are supported by the remote Controller:

- A Frequency Actuation Error of zero for all allowed CS channels as described in [Vol 6] Part A, Section 3.5.
- Channel Selection Algorithm #3c as described in [Vol 6] Part H, Section 4.1.4.2.
- Phase-based ranging from a sounding sequence as described in [Vol 6] Part H, Section 3.3.1.
- IPT in the CS reflector as described in [Vol 6] Part H, Section 4.3.3.
- RTT_PHY support denoting the setting of the RTT accuracy requirements for optional LE 2M and LE 2M 2BT PHYs separate from the RTT accuracy requirements for the mandatory LE 1M PHY as described in [Vol 6] Part B, Section 2.4.2.44. This bit shall only be set to 1 if [v2] or above of this event is unmasked.

The T_IP1_Times_Supported, T_IP2_Times_Supported, T_FCS_Times_Supported, T_PM_Times_Supported, and T_SW_Time_Supported parameters indicate the supported optional time durations used in CS steps as described in [Vol 6] Part H, Section 4.3.

The TX_SNR_Capability parameter indicated the supported SNR levels used for the CS_SYNC packets used in mode-1 and mode-3 steps as described in [Vol 6] Part A, Section 3.1.3.

The T_IP2_IPT_Times_Supported and T_SW_IPT_Times_Supported parameters indicate the supported optional time durations used in CS steps if the CS reflector supports the Channel Sounding Enhancement #1 feature and IPT capability (see [Vol 6] Part H, Section 4.3).

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x38 | Subevent code for the HCI_LE_CS_Read_Remote_Supported_Capabilities_Complete event [v2] |
| 0x2C | Subevent code for the HCI_LE_CS_Read_Remote_Supported_-Capabilities_Complete event [v1] |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE_CS_Read_Remote_Supported_Capabilities command successfully completed |
| 0x01 to 0xFF | LE_CS_Read_Remote_Supported_Capabilities command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Num_Config_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of CS configurations supported per connection |
| All other values | Reserved for future use |

*Max_Consecutive_Procedures_Supported:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Support for both a fixed number of consecutive CS procedures and for an indefinite number of CS procedures until termination. |
| 0x0001 to 0xFFFF | Maximum number of consecutive CS procedures supported. |

*Num_Antennae_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of antennae supported |
| All other values | Reserved for future use |

*Max_Antenna_Paths_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Maximum number of antenna paths supported |
| All other values | Reserved for future use |

*Roles_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Initiator |
| 1 | Reflector |
| All other values | Reserved for future use |

*Modes_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Mode-3 |
| All other values | Reserved for future use |

*RTT_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | If the RTT_AA_Only_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |

| Bit Number | Parameter Description |
|---|---|
| 1 | If the RTT_Sounding_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 2 | If the RTT_Random_Sequence_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 3 | If the RTT_2M_AA_Only_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement. |
| 4 | If the RTT_2M_Sounding_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement. |
| 5 | If the RTT_2M_Random_Sequence_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement. |
| All other bits | Reserved for future use |

*RTT_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA Only not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*NADM_Sounding_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| All other bits | Reserved for future use |

*NADM_Random_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| All other bits | Reserved for future use |

*CS_SYNC_PHYs_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 1 | LE 2M PHY |
| 2 | LE 2M 2BT PHY |
| All other values | Reserved for future use |

*Subfeatures_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | RFU |
| 1 | CS with no transmitter Frequency Actuation Error |
| 2 | CS Channel Selection Algorithm #3c |
| 3 | CS phase-based ranging from RTT sounding sequence |
| 4 | IPT in the CS reflector |
| 5 | CS RTT accuracy specified on a per PHY basis |
| All other bits | Reserved for future use |

*T_IP1_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other values | Reserved for future use |

*T_IP2_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |

| Bit Number | Parameter Description |
|---|---|
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other values | Reserved for future use |

*T_FCS_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 15 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| 7 | 100 µs supported |
| 8 | 120 µs supported |
| All other values | Reserved for future use |

*T_PM_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| All other values | Reserved for future use |

*T_SW_Time_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*TX_SNR_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 18 dB supported |
| 1 | 21 dB supported |
| 2 | 24 dB supported |
| 3 | 27 dB supported |
| 4 | 30 dB supported |
| All other values | Reserved for future use |

*T_IP2_IPT_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other values | Reserved for future use |

*T_SW_IPT_Times_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*RTT_2M_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA Only not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

*RTT_2M_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the precision accuracy on the LE 2M or the LE 2 M 2BT PHY |

*RTT_2M_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

#### 7.7.65.40 LE CS Read Remote FAE Table Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Read_Remote_FAE_Table_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Remote_FAE_Table |

**Description:**

This event shall be generated when a locally initiated CS Mode-0 Frequency Actuation Table Update procedure has completed.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x2D | Subevent code for the HCI_LE_CS_Read_Remote_FAE_Table_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE_CS_Read_Remote_FAE_Table command successfully completed |
| 0x01 to 0xFF | LE_CS_Read_Remote_FAE_Table command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Remote_FAE_Table:* *Size: 72 octets*

| Value | Parameter Description |
|---|---|
| Variable | Per-channel mode-0 Frequency Actuation Error table of the remote Controller as described in [Vol 6] Part B, Section 2.4.2.52. |

#### 7.7.65.41 LE CS Security Enable Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Security_Enable_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle |

**Description:**

This event shall be generated when a locally initiated CS Security Start procedure has completed or when the local Controller has responded to a CS security request from the remote Controller.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x2E | Subevent code for the HCI_LE_CS_Security_Enable_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CS security parameters successfully exchanged |
| 0x01 to 0xFF | CS security parameter exchange failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

#### 7.7.65.42 LE CS Config Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Config_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Config_ID,<br>Action,<br>Main_Mode_Type,<br>Sub_Mode_Type,<br>Min_Main_Mode_Steps,<br>Max_Main_Mode_Steps,<br>Main_Mode_Repetition,<br>Mode_0_Steps,<br>Role,<br>RTT_Type,<br>CS_SYNC_PHY,<br>Channel_Map,<br>Channel_Map_Repetition,<br>Channel_Selection_Type,<br>Ch3c_Shape,<br>Ch3c_Jump,<br>CS_Enhancements,<br>T_IP1_Time,<br>T_IP2_Time,<br>T_FCS_Time,<br>T_PM_Time |

**Description:**

This event shall be generated when a locally initiated Channel Sounding Configuration procedure has completed or when the local Controller has responded to a CS configuration request from the remote Controller for the CS configuration identified by Config_ID or when a CS configuration is created only with local context. The Action parameter indicates if a CS configuration was requested to be created or removed. The Status parameter indicates whether the request indicated by the Action parameter was successful.

When the Action parameter is set to 0x00, all the remaining event parameters are ignored.

The Main_Mode_Type and Sub_Mode_Type parameters indicate the CS modes used during the CS procedure for the specified CS configuration. The Min_Main_Mode_Steps and Max_Main_Mode_Steps parameters indicate the range of main mode CS steps executed before a submode CS step is executed during the CS procedure. The Main_Mode_Repetition parameter indicates the number of main mode CS steps repeated from the last CS subevent at the beginning of the current CS subevent. The Mode_0_Steps parameter indicates the number of mode-0 CS steps included at the beginning of each CS subevent.

The Role parameter indicates the CS role for the local Controller for the specified CS configuration. The RTT_Type parameter indicates the RTT variant to be used during the CS procedure, and the CS_SYNC_PHY parameter indicates the PHY to be used for CS_SYNC exchanges during the CS procedure for the specified CS configuration.

The Channel_Map parameter indicates the channels to be used or unused during the CS procedure, and the Channel_Map_Repetition parameter indicates the number of times the channels specified by Channel_Map will be repeated for non-mode-0 steps during the CS procedure (see [Vol 6] Part H, Section 4.1.4).

The Channel_Selection_Type parameter indicates the Channel Selection Algorithm to be used during the CS procedure for non-mode-0 steps. When the Channel_Selection_Type parameter is set to 0x01, the Ch3c_Shape and the Ch3c_Jump parameters indicate the selected shape and channels to be skipped as described in [Vol 6] Part H, Section 4.1.4.2.

The CS_Enhancements parameter indicates which CS enhancements are enabled. For each enhancement that is not enabled, the corresponding bit shall be set to 0.

The T_IP1_Time, T_IP2_Time, T_FCS_Time, T_PM_Time, and T_SW_Time parameters indicate the time durations used in CS steps as described in [Vol 6] Part H, Section 4.3.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x2F | Subevent code for the HCI_LE_CS_Config_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Channel Sounding Configuration procedure succeeded |
| 0x01 to 0xFF | Channel Sounding Configuration procedure failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*Action:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CS configuration is removed |
| 0x01 | CS configuration is created |
| All other values | Reserved for future use |

*Main_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| All other values | Reserved for future use |

*Sub_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |

| Value | Parameter Description |
|---|---|
| 0x03 | Mode-3 |
| 0xFF | Unused |
| All other values | Reserved for future use |

*Min_Main_Mode_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Minimum number of CS main mode steps to be executed before a submode step is executed |
| All other values | Reserved for future use |

*Max_Main_Mode_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Maximum number of CS main mode steps to be executed before a submode step is executed |
| All other values | Reserved for future use |

*Main_Mode_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x03 | Number of main mode steps taken from the end of the last CS subevent to be repeated at the beginning of the current CS subevent directly after the last mode-0 step of that event |
| All other values | Reserved for future use |

*Mode_0_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x03 | Number of CS mode-0 steps to be included at the beginning of each CS subevent |
| All other values | Reserved for future use |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Initiator |
| 0x01 | Reflector |
| All other values | Reserved for future use |

*RTT_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA Only |
| 0x01 | RTT with 32-bit sounding sequence |
| 0x02 | RTT with 96-bit sounding sequence |
| 0x03 | RTT with 32-bit random sequence |
| 0x04 | RTT with 64-bit random sequence |
| 0x05 | RTT with 96-bit random sequence |
| 0x06 | RTT with 128-bit random sequence |
| All other values | Reserved for future use |

*CS_SYNC_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE 2M 2BT PHY |
| All other values | Reserved for future use |

*Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXX | This parameter contains 80 1-bit fields.<br>The nth such field (in the range 0 to 78) contains the value for the CS channel index n.<br>Channel n is enabled for CS procedure = 1<br>Channel n is disabled for CS procedure = 0<br>Channels n = 0, 1, 23, 24, 25, 77, and 78 shall be ignored and shall be set to zero. At least 15 channels shall be enabled.<br>The most significant bit (bit 79) is reserved for future use. |

*Channel_Map_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | The number of times the Channel_Map field will be cycled through for non-mode-0 steps within a CS procedure |
| All other values | Reserved for future use |

*Channel_Selection_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use Channel Selection Algorithm #3b for non-mode-0 CS steps |
| 0x01 | Use Channel Selection Algorithm #3c for non-mode-0 CS steps |
| All other values | Reserved for future use |

*Ch3c_Shape:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use Hat shape for user-specified channel sequence |
| 0x01 | Use X shape for user-specified channel sequence |
| All other values | Reserved for future use |

*Ch3c_Jump:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x08 | Number of channels skipped in each rising and falling sequence |
| All other values | Reserved for future use |

*CS_Enhancements:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | IPT is enabled in the CS reflector |
| All other bits | Reserved for future use |

*T_IP1_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0A, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, or 0x91 | Interlude time in microseconds between the CS_SYNC packets used in mode-0 and mode-1 steps |
| All other values | Reserved for future use |

*T_IP2_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0A, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, or 0x91 | Interlude time in microseconds between the CS tones |
| All other values | Reserved for future use |

*T_FCS_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0F, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, 0x64, 0x78, or 0x96 | Time in microseconds for frequency changes |
| All other values | Reserved for future use |

*T_PM_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0A, 0x14, or 0x28 | Time in microseconds for the phase measurement period of the CS tones |
| All other values | Reserved for future use |

#### 7.7.65.43 LE CS Procedure Enable Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Procedure_Enable_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Config_ID,<br>State,<br>Tone_Antenna_Config_Selection,<br>Selected_TX_Power,<br>Subevent_Len,<br>Subevents_Per_Event,<br>Subevent_Interval,<br>Event_Interval,<br>Procedure_Interval,<br>Procedure_Count,<br>Max_Procedure_Len |

**Description:**

This event shall be generated when the local or remote Controller has scheduled a new CS procedure or disabled an ongoing CS procedure as a result of an HCI_LE_-CS_Procedure_Enable command. When a new CS procedure is enabled, the HCI_LE_-CS_Procedure_Enable_Complete event shall be sent to the Host after the LL_CS_IND is transmitted or received and before any CS subevent results are available.

When the State parameter is set to 0x00, all the remaining event parameters are ignored.

The Tone_Antenna_Config_Selection parameter indicates the Antenna Configuration Index used in the CS procedure.

The Controller shall set the Selected_TX_Power parameter to the transmit power level that it will use for the CS procedures as described in [Vol 6] Part B, Section 5.1.26. If the radiated power level will vary between packets (e.g., because of frequency-dependent properties of the transmitter), then the value should be the best estimate of the transmit power level that will be used.

The Subevent_Len parameter indicates the selected maximum duration of each CS subevent during the CS procedure. The Subevents_Per_Event parameter indicates the number of CS subevents that are anchored off the same associated LE ACL anchor point. The Subevent_Interval parameter indicates the gap between the start

of two consecutive CS subevents that are anchored off the same associated LE ACL anchor point. The Event_Interval parameter indicates the number of connection intervals between consecutive LE ACL anchor points from which CS subevents are anchored.

The Procedure_Interval parameter indicates the selected interval between consecutive CS procedures, in units of ACL connection events. The Procedure_Count parameter indicates the selected number of consecutive CS procedures to be scheduled as part of this measurement. The Max_Procedure_Len parameter indicates the selected maximum duration of each CS procedure.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x30 | Subevent code for the HCI_LE_CS_Procedure_Enable event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE_CS_Procedure_Enable command successful |
| 0x01 to 0xFF | LE_CS_Procedure_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*State:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CS procedures are disabled |
| 0x01 | CS procedures are enabled |
| All other values | Reserved for future use |

*Tone_Antenna_Config_Selection:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x07 | Antenna Configuration Index as described in [Vol 6] Part A, Section 5.3 |
| All other values | Reserved for future use |

*Selected_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Transmit power level used for CS procedure<br>Range: -127 to 20<br>Units: dBm |
| 0x7F | Transmit power level is unavailable |
| All other values | Reserved for future use |

*Subevent_Len:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Duration for each CS subevent in microseconds<br>Range: 1250 µs to 4 s |
| All other values | Reserved for future use |

*Subevents_Per_Event:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x20 | Number of CS subevents anchored off the same ACL connection event |
| All other values | Reserved for future use |

*Subevent_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Time between consecutive CS subevents anchored off the same ACL connection event.<br>Units: 0.625 ms |

*Event_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of ACL connection events between consecutive CS event anchor points |

*Procedure_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of ACL connection events between consecutive CS procedure anchor points |

*Procedure_Count:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | CS procedures to continue until disabled |
| 0xXXXX | Number of CS procedures to be scheduled<br>Range: 0x0001 to 0xFFFF |

*Max_Procedure_Len:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum duration for each CS procedure<br>Range: 0x0001 to 0xFFFF<br>Time = N × 0.625 ms<br>Time range: 0.625 ms to 40.959375 s |
| All other values | Reserved for future use |

#### 7.7.65.44 LE CS Subevent Result event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Subevent_Result | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Config_ID,<br>Start_ACL_Conn_Event_Counter,<br>Procedure_Counter,<br>Frequency_Compensation,<br>Reference_Power_Level,<br>Procedure_Done_Status,<br>Subevent_Done_Status,<br>Abort_Reason,<br>Num_Antenna_Paths,<br>Num_Steps_Reported,<br>Step_Mode[i],<br>Step_Channel[i],<br>Step_Data_Length[i],<br>Step_Data[i] |

**Description:**

This event shall be generated when the local Controller has results to report for a CS subevent during the CS procedure. Depending on the number of CS steps in the CS subevent, the Controller may choose to report complete or partial results. When the number of CS steps exceeds the maximum HCI event size, the Controller may report further results for the CS subevent using the HCI_LE_CS_Subevent_Result_Continue event.

When Connection_Handle is set to 0x0FFF, the Config_ID and Start_ACL_Conn_-Event_Counter parameters shall be ignored.

The Start_ACL_Conn_Event_Counter parameter indicates the starting ACL connection event count from which the CS event results reported in this HCI event are anchored. A CS procedure may have CS subevents in multiple ACL connection events anchored to the same ACL connection event count as defined by the Event_Interval return parameter in the HCI_LE_CS_Procedure_Enable_Complete event.

The Procedure_Counter parameter indicates the associated CS procedure count for the results reported in this HCI event (see [Vol 6] Part B, Section 5.1.26).

The Frequency_Compensation parameter indicates the value of the fractional frequency offset (FFO) used by the initiator device to align the timing of CS steps and transmit frequencies during non-mode-0 CS steps (see $FFO_{RX^*}$ in [Vol 6] Part A, Section 3.5.1).

The Reference_Power_Level parameter is expressed in dBm and is described in [Vol 6] Part H, Section 4.6 and [Vol 6] Part H, Section 3.3.1. If the reference power level value is not available during a subevent, then this value shall be set to 0x7F.

When bits 0 to 3 of Subevent_Done_Status are set to 0x1, the Controller shall send one or more LE CS Subevent Result Continue events for the current CS subevent. Otherwise, the Controller sends no further events for the current CS subevent.

When bits 0 to 3 of Procedure_Done_Status are set to 0x1, the Controller shall send one or more LE CS Subevent Result or LE CS Subevent Result Continue events for the current or subsequent CS procedure. Otherwise, the Controller sends no further events for the current CS procedure. When bits 0 to 3 of Procedure_Done_Status are set to 0xF or when results of all CS procedures are sent to the Host, the Controller sends no further events until a new CS procedure is enabled by the Host.

The allowed combinations of Procedure_Done_Status and Subevent_Done_Status are shown in Table 7.2.

| **Procedure_Done_Status (Bits 0 to 3)** | **Allowed values of Subevent_Done_Status (Bits 0 to 3)** |
|---|---|
| 0x0 | 0x0, 0xF |
| 0x1 | 0x0, 0x1, 0xF |
| 0xF | 0x0, 0xF |

*Table 7.2: Allowed combinations of done status*

The Num_Antenna_Paths parameter indicates the number of antenna paths used for CS tone exchanges.

The Num_Steps_Reported parameter indicates the number of CS steps for which results are reported. The Step_Mode[i] parameter indicates the CS mode for each CS step, and the Step_Channel[i] parameter indicates the channel used for each CS step. A Controller may return a value 0 when a subevent is aborted.

The Step_Data_Length[i] and Step_Data[i] parameters indicate the reported data that varies based on the CS mode and the CS role of the local device. When Step_Data_Length[i] is set to zero, it indicates that the Step may have been aborted for unspecified reasons. Otherwise, the Step_Data[i] indicates the mode- and role-specific information being reported.

For each CS procedure in a repeat sequence, the Controller shall report the results for each CS step in each CS subevent of the CS procedure even if the CS step has

been aborted for any reason. In order to terminate reporting of pending CS steps in a CS subevent, the Controller shall set the Subevent_Done_Status parameter to 0xF in the HCI_LE_CS_Subevent_Result or the HCI_LE_CS_Subevent_Result_Continue event and shall not report any additional HCI_LE_CS_Subevent_Result_Continue events for that CS subevent. The Controller may terminate reporting of pending CS procedures by setting the Procedure_Done_Status parameter to 0xF in the HCI_LE_CS_Subevent_Result or the HCI_LE_CS_Subevent_Result_Continue event and shall not report any additional HCI_LE_CS_Subevent_Result_Continue events for that CS procedure.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x31 | Subevent code for the HCI_LE_CS_Subevent_Result event |

*Connection_Handle:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to x0EFF | Connection_Handle |
| 0x0FFF | CS test |
| All other values | Reserved for future use |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*Start_ACL_Conn_Event_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Starting ACL connection event counter for the results reported in the event |

*Procedure_Counter:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | CS procedure count since completion of the Channel Sounding Security Start procedure<br>Range: 0x0000 to 0xFFFF |

*Frequency_Compensation:* *Size: 2 octets (15 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Frequency compensation value in units of 0.01 ppm (15-bit signed integer)<br>Range: -100 ppm (0x58F0) to +100 ppm (0x2710)<br>Units: 0.01 ppm |
| 0xC000 | Frequency compensation value is not available, or the role is not initiator |
| All other values | Reserved for future use |

*Reference_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reference power level<br>Range: -127 to 20<br>Units: dBm |
| 0x7F | Reference power level is not applicable |
| All other values | Reserved for future use |

*Procedure_Done_Status:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = All results complete for the CS procedure<br>0x1 = Partial results with more to follow for the CS procedure<br>0xF = All subsequent CS procedures aborted<br>All other values = Reserved for future use |
| 4 to 7 | Reserved for future use |

*Subevent_Done_Status:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = All results complete for the CS subevent<br>0x1 = Partial results with more to follow for the CS subevent<br>0xF = Current CS subevent aborted<br>All other values = Reserved for future use |
| 4 to 7 | Reserved for future use |

*Abort_Reason:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | Indicates the abort reason when Procedure_Done_Status is set to 0xF, otherwise the default value is set to zero.<br>0x0 = Report with no abort<br>0x1 = Abort because of local Host or remote request<br>0x2 = Abort because filtered channel map has less than 15 channels<br>0x3 = Abort because the channel map update instant has passed<br>0xF = Abort because of unspecified reasons<br>All other values = Reserved for future use |
| 4 to 7 | Indicates the abort reason when Subevent_Done_Status is set to 0xF, otherwise the default value is set to zero.<br>0x0 = Report with no abort<br>0x1 = Abort because of local Host or remote request<br>0x2 = Abort because no CS_SYNC (mode-0) received<br>0x3 = Abort because of scheduling conflicts or limited resources<br>0xF = Abort because of unspecified reasons<br>All other values = Reserved for future use |

*Num_Antenna_Paths:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Ignored because phase measurement does not occur during the CS step |
| 0x01 to 0x04 | Number of antenna paths used during the phase measurement stage of the CS step |
| All other values | Reserved for future use |

*Num_Steps_Reported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xA0 | Number of steps in the CS subevent for which results are reported |
| All other values | Reserved for future use |

*Step_Mode[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x03 | Mode type |
| All other values | Reserved for future use |

*Step_Channel[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x4E | CS channel index. Refer to [Vol 6] Part A, Section 2 for valid CS channels. |
| All other values | Reserved for future use |

*Step_Data_Length[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | Length of mode- and role-specific information being reported |
| All other values | Reserved for future use |

*Step_Data[i]:* *Size: $\sum_i$(Step_Data_Length[i]) octets*

| Parameter Description |
|---|
| Mode- and role-specific information being reported as Mode_Role_Specific_Info object<br>Note: Each element of this array has a variable length. |

When the mode type is 0 and the role is initiator, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_RSSI
- Packet_Antenna
- Measured_Freq_Offset

When the mode type is 0 and the role is reflector, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_RSSI
- Packet_Antenna

When the mode type is 1 and the role is initiator, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToA_ToD_Initiator
- Packet_Antenna

When the mode type is 1, the role is initiator, sounding phase-based ranging is supported, and the RTT type contains a sounding sequence, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToA_ToD_Initiator
- Packet_Antenna
- Packet_PCT1
- Packet_PCT2

When the mode type is 1 and the role is reflector, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToD_ToA_Reflector
- Packet_Antenna

When the mode type is 1, the role is reflector, sounding phase-based ranging is supported, and the RTT type contains a sounding sequence, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToD_ToA_Reflector
- Packet_Antenna
- Packet_PCT1
- Packet_PCT2

When the mode type is 2 and the role is either initiator or reflector, the parameters of the Mode_Role_Specific_Info object are as follows:

- Antenna_Path_Permutation_Index
- Tone_PCT[k]
- Tone_Quality_Indicator[k]

When the mode type is 3 and the role is initiator, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToA_ToD_Initiator
- Packet_Antenna
- Antenna_Path_Permutation_Index
- Tone_PCT[k]
- Tone_Quality_Indicator[k]

When the mode type is 3, the role is initiator, sounding phase-based ranging is supported, and the RTT type contains a sounding sequence, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM

- Packet_RSSI
- ToA_ToD_Initiator
- Packet_Antenna
- Packet_PCT1
- Packet_PCT2
- Antenna_Path_Permutation_Index
- Tone_PCT[k]
- Tone_Quality_Indicator[k]

When the mode type is 3 and the role is reflector, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToD_ToA_Reflector
- Packet_Antenna
- Antenna_Path_Permutation_Index
- Tone_PCT[k]
- Tone_Quality_Indicator[k]

When the mode type is 3, the role is reflector, sounding phase-based ranging is supported, and the RTT type contains a sounding sequence, the parameters of the Mode_Role_Specific_Info object are as follows:

- Packet_Quality
- Packet_NADM
- Packet_RSSI
- ToD_ToA_Reflector
- Packet_Antenna
- Packet_PCT1
- Packet_PCT2
- Antenna_Path_Permutation_Index
- Tone_PCT[k]

- Tone_Quality_Indicator[k]

The bits 0 to 3 of the Tone_Quality_Indicator[k] parameter indicate the CS tone quality values as described in [Vol 6] Part H, Section 4.6. The bits 4 to 7 of the Tone_Quality_Indicator[k] parameter indicate whether the tone slot is a tone extension slot, and if so, whether it was expected to carry a transmission, as described in [Vol 6] Part H, Section 4.3.3 and [Vol 6] Part H, Section 4.3.4.

When the RTT type contains a random or sounding sequence, the Packet_Quality parameter indicates the number of bit errors detected. Otherwise, the parameter is ignored by the Host. A Controller may calculate the number of bit errors in an implementation-dependent manner.

The Packet_NADM parameter indicates the estimated chance of an attack on the received packet based on the normalized attack detector metric (NADM) as described in [Vol 6] Part H, Section 3.5.1.

The Measured_Freq_Offset parameter indicates the value of the FFO measured by the initiator device during the mode-0 CS step (see *FFO*[*k*] in [Vol 6] Part A, Section 3.5.1).

The parameters of the Mode_Role_Specific_Info object are described below.

*Packet_Quality:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = CS Access Address check is successful, and all bits match the expected sequence<br>0x1 = CS Access Address check contains one or more bit errors<br>0x2 = CS Access Address not found<br>All other values = Reserved for future use |
| 4 to 7 | Number of bit errors being reported on the payload with a random or sounding sequence. Value 0 may indicate zero bit errors or no report available. Value 15 may indicate 15 or more bit errors. |

*Packet_NADM:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Attack is extremely unlikely |
| 0x01 | Attack is very unlikely |
| 0x02 | Attack is unlikely |
| 0x03 | Attack is possible |
| 0x04 | Attack is likely |

| Value | Parameter Description |
|---|---|
| 0x05 | Attack is very likely |
| 0x06 | Attack is extremely likely |
| 0xFF | Unknown NADM.<br>Default value for RTT types that do not have a random or sounding sequence. |
| All other values | Reserved for future use |

*Packet_RSSI:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | RSSI is not available |
| All other values | Reserved for future use |

*Packet_Antenna:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Antenna identifier used for the CS_SYNC packets |
| All other values | Reserved for future use |

*Packet_PCT1:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00XXXXXX | Phase Correction Term (bits 0 to 11 are the I sample with type *sint12*, bits 12 to 23 are the Q sample with type *sint12*, and bits 24 to 31 are reserved for future use). The unitless I and Q values may be combined with Reference_Power_Level to convert to power in dBm, as described in [Vol 6] Part H, Section 4.6. |
| 0xFFFFFFFF | Phase Correction Term is not available |
| All other values | Reserved for future use |

*Packet_PCT2:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00XXXXXX | Phase Correction Term (bits 0 to 11 are the I sample with type *sint12*, bits 12 to 23 are the Q sample with type *sint12*, and bits 24 to 31 are reserved for future use). The unitless I and Q values may be combined with Reference_Power_Level to convert to power in dBm, as described in [Vol 6] Part H, Section 4.6. |

| Value | Parameter Description |
|---|---|
| 0xFFFFFFFF | Phase Correction Term is not available |
| All other values | Reserved for future use |

*Measured_Freq_Offset:* *Size: 2 octets (15 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Measured frequency offset in units of 0.01 ppm (15-bit signed integer)<br>Range: -100 ppm (0x58F0) to +100 ppm (0x2710)<br>Units: 0.01 ppm |
| 0xC000 | Frequency offset is not available |
| All other values | Reserved for future use |

*ToA_ToD_Initiator:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Time difference in units of 0.5 nanoseconds between the time of arrival and the time of departure of the CS packets at the initiator during a CS step (16-bit signed integer), where the known nominal offsets are omitted. The known offsets (i.e., the interlude time between packets and the length of the packet itself) are described in [Vol 6] Part H, Section 4.3.2 and [Vol 6] Part H, Section 4.3.4 for Mode-1 and Mode-3, respectively. |
| 0x8000 | Time difference is not available |
| All other values | Reserved for future use |

*ToD_ToA_Reflector:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Time difference in units of 0.5 nanoseconds between the time of departure and the time of arrival of the CS packets at the reflector during a CS step (16-bit signed integer), where the known nominal offsets are omitted. The known offsets (i.e., the interlude time between packets and the lengths of the packets itself) are described in [Vol 6] Part H, Section 4.3.2 and [Vol 6] Part H, Section 4.3.4 for Mode-1 and Mode-3, respectively. |
| 0x8000 | Time difference is not available |
| All other values | Reserved for future use |

*Antenna_Path_Permutation_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x17 | Antenna path permutation index for the chosen Num_Antenna_Paths parameter used during the phase measurement stage of the CS step |
| All other values | Reserved for future use |

*Tone_PCT[k]:* *Size: (Num_Antenna_Paths + 1) × 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Phase Correction Term (bits 0 to 11 are the I sample with type *sint12* and bits 12 to 23 are the Q sample with type *sint12*). The unitless I and Q values may be combined with Reference_Power_Level to convert to power in dBm, as described in [Vol 6] Part H, Section 4.6.<br>If IPT is enabled in the reflector, then the Q sample shall be set to 0 on the reflector. The I sample conveys the measured signal amplitude as described in [Vol 6] Part H, Section 4.6. On the initiator side, the Tone_PCT format remains unchanged to when IPT is not enabled in the reflector. |

*Tone_Quality_Indicator[k]:* *Size: (Num_Antenna_Paths + 1) × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = Tone quality is high<br>0x1 = Tone quality is medium<br>0x2 = Tone quality is low<br>0x3 = Tone quality indication is not available<br>All other values = Reserved for future use |
| 4 to 7 | 0x0 = Not tone extension slot<br>0x1 = Tone extension slot; tone not expected to be present<br>0x2 = Tone extension slot; tone expected to be present<br>All other values = Reserved for future use |

#### 7.7.65.45 LE CS Subevent Result Continue event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Subevent_Result_Continue | 0x3E | Subevent_Code,<br>Connection_Handle,<br>Config_ID,<br>Procedure_Done_Status,<br>Subevent_Done_Status,<br>Abort_Reason,<br>Num_Antenna_Paths,<br>Num_Steps_Reported,<br>Step_Mode[i],<br>Step_Channel[i],<br>Step_Data_Length[i],<br>Step_Data[i] |

**Description:**

This event shall be generated after the local Controller has completed a new CS subevent measurement and has already sent an HCI_LE_CS_Subevent_Result event for the specified CS subevent.

When Connection_Handle is set to 0x0FFF, the Config_ID parameter shall be ignored.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x32 | Subevent code for the HCI_LE_CS_Subevent_Result_Continue event |

*Connection_Handle:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to x0EFF | Connection_Handle |
| 0x0FFF | CS test |
| All other values | Reserved for future use |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*Procedure_Done_Status:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = All results complete for the CS procedure<br>0x1 = Partial results with more to follow for the CS procedure<br>0xF = All subsequent CS procedures aborted<br>All other values = Reserved for future use |
| 4 to 7 | Reserved for future use |

*Subevent_Done_Status:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | 0x0 = All results complete for the CS subevent<br>0x1 = Partial results with more to follow for the CS subevent<br>0xF = Current CS subevent aborted<br>All other values = Reserved for future use |
| 4 to 7 | Reserved for future use |

*Abort_Reason:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 to 3 | Indicates the abort reason when Procedure_Done_Status is set to 0xF, otherwise the default value is set to zero.<br>0x0 = Report with no abort<br>0x1 = Abort because of local Host or remote request<br>0x2 = Abort because filtered channel map has less than 15 channels<br>0x3 = Abort because the channel map update instant has passed<br>0xF = Abort because of unspecified reasons<br>All other values = Reserved for future use |

| Bit Number | Parameter Description |
|---|---|
| 4 to 7 | Indicates the abort reason when Subevent_Done_Status is set to 0xF, otherwise the default value is set to zero.<br>0x0 = Report with no abort<br>0x1 = Abort because of local Host or remote request<br>0xF = Abort because of unspecified reasons<br>All other values = Reserved for future use |

*Num_Antenna_Paths:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Ignored because phase measurement does not occur during the CS step |
| 0x01 to 0x04 | Number of antenna paths used during the phase measurement stage of the CS step |
| All other values | Reserved for future use |

*Num_Steps_Reported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xA0 | Number of steps in the CS subevent for which results are reported |
| All other values | Reserved for future use |

*Step_Mode[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x03 | Mode type |
| All other values | Reserved for future use |

*Step_Channel[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x4E | CS channel index. Refer to [Vol 6] Part A, Section 2 for valid CS channels. |
| All other values | Reserved for future use |

*Step_Data_Length[i]:* *Size: Num_Steps_Reported × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | Length of mode- and role-specific information being reported |
| All other values | Reserved for future use |

*Step_Data[i]:* *Size:* $\sum_i$ (*Step_Data_Length[i]*) *octets*

| Parameter Description |
| --- |
| Mode- and role-specific information being reported as Mode_Role_Specific_Info object. Refer to Section 7.7.65.44 for details on Mode_Role_Specific_Info object.<br>Note: Each element of this array has a variable length. |

#### 7.7.65.46 LE CS Test End Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_CS_Test_End_Complete | 0x3E | Subevent_Code,<br>Status |

**Description:**

This event shall be generated when the local Controller has stopped an ongoing CS test as a result of the HCI_LE_CS_Test_End command. When the HCI_LE_CS_Test_End command is issued by the Host, the HCI_LE_CS_Test_End_Complete event shall be sent to the Host after any pending CS test result events are sent to the Host.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x33 | Subevent code for the HCI_LE_CS_Test_End_Complete event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | LE_CS_Test_End command successful. |
| 0x01 to 0xFF | LE_CS_Test_End command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### 7.7.65.47 LE Monitored Advertisers Report event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Monitored_Advertisers_Report | 0x3E | Subevent_Code,<br>Address_Type,<br>Address,<br>Condition |

**Description:**

This event indicates that an advertiser on the Monitored Advertisers List has met an RSSI threshold condition established by the HCI_LE_Add_Device_To_Monitored_Advertisers_List command.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x34 | Subevent code for the HCI_LE_Monitored_Advertisers_Report event |

*Address_Type:* *Size: 1 octet*

| Value | Parameters Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (corresponds to the Resolved Private Address) |
| 0x03 | Random (static) Identity Address (corresponds to the Resolved Private Address) |
| All other values | Reserved for future use |

*Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device being monitored. |

*Condition:* *Size: 1 octet*

| Value | Parameters Description |
|---|---|
| 0x00 | Received RSSI value below the RSSI low threshold for longer than the timeout period or no RSSI value received |
| 0x01 | Received RSSI value greater than or equal to the RSSI high threshold |
| All other values | Reserved for future use |

#### 7.7.65.48 LE Frame Space Update Complete event

| Event | OCF | Event Parameters |
|---|---|---|
| HCI_LE_Frame_Space_Update_Complete | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Initiator,<br>Frame_Space,<br>PHYS,<br>Spacing_Types |

**Description:**

This event is used to indicate that the Frame Space Update procedure has completed (see [Vol 6] Part B, Section 5.1.30) and, if initiated autonomously by the local Controller or the peer device, that at least one frame space value has changed.

The Initiator parameter indicates who initiated the Frame Space Update procedure.

The Frame_Space parameter indicates the new frame space value that the Controller is now using.

The PHYS and Spacing_Types parameters indicate which PHYs and spacing types are using the new frame space value.

**Event parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x35 | Subevent code for the HCI_LE_Frame_Space_Update event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Frame_Space_Update command succeeded or the Controller updated the frame space value autonomously. |
| 0x01 to 0xFF | HCI_LE_Frame_Space_Update command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Used to identify a connection handle. Range: 0x0000 to 0x0EFF |

*Initiator:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Local Host initiated |
| 0x01 | Local Controller initiated |
| 0x02 | Peer initiated |
| All other values | Reserved for future use |

*Frame_Space:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The new frame space value being used, in microseconds<br>Range: 0x0000 to 0x2710 |

*PHYS:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | LE 1M |
| 1 | LE 2M |
| 2 | LE Coded PHY |
| All other bits | Reserved for future use |

*Spacing_Types:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | T_IFS_ACL_CP |
| 1 | T_IFS_ACL_PC |
| 2 | T_MCES |
| 3 | T_IFS_CIS |
| 4 | T_MSS_CIS |
| All other bits | Reserved for future use |

#### 7.7.65.49 LE UTP Receive event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_UTP_Receive | 0x3E | Subevent_Code,<br>UTP_Data_Length,<br>UTP_Data |

**Description:**

This event is used by the IUT to transmit UTP messages to the Upper Tester.

**Event Parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x36 | Subevent code for the HCI_LE_UTP_Receive event |

*UTP_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Length of the Data field |

*UTP_Data:* *Size: UTP_Data_Length octets*

| Parameter Description |
|---|
| Unified Test Protocol data (see [Vol 6] Part F, Section 5)<br>Note: This parameter has a variable length |

### 7.7.65.50 LE Connection Rate Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_LE_Connection_Rate_Change | 0x3E | Subevent_Code,<br>Status,<br>Connection_Handle,<br>Connection_Interval,<br>Subrate_Factor,<br>Peripheral_Latency,<br>Continuation_Number,<br>Supervision_Timeout |

**Description:**

The HCI_LE_Connection_Rate_Change event is used to indicate that the Connection Rate Update procedure has completed.

The event shall be issued if the HCI_LE_Connection_Rate_Request command was issued by the Host or if the connection parameters are updated following a request from the peer device. If no parameters are updated following a request from the peer device or the parameters were changed using the Connection Update procedure or the Connection Subrate Update procedure, then this event shall not be issued.

Note: The parameter values returned in this event can be different from the parameter values provided by the Host through the HCI_LE_Connection_Rate_Request command (Section 7.8.154).

**Event Parameters:**

*Subevent_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x37 | Subevent code for the HCI_LE_Connection_Rate_Change event |

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Connection_Rate_Request command succeeded or this event was generated following a request from the peer device. |
| 0x01 to 0xFF | The HCI_LE_Connection_Rate_Request command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Connection_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | New connection interval used on this connection<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s |

*Subrate_Factor:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | New subrate factor applied to the specified underlying connection interval<br>Range 0x0001 to 0x01F4 |

*Peripheral_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | New peripheral latency for the connection in number of subrated connection intervals<br>Range: 0x0000 to 0x01F3 |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | New number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Supervision timeout for this connection<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

### 7.7.66 Triggered Clock Capture event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Triggered_Clock_Capture | 0x4E | Connection_Handle,<br>Which_Clock,<br>Clock,<br>Slot_Offset |

**Description:**

This event is sent to indicate that a triggering event has occurred at the specified clock and offset value. The Which_Clock parameter indicates whether the clock is local or a piconet clock. The Connection_Handle parameter is used when the clock is a piconet clock to indicate which piconet's clock was reported.

The Clock parameter indicates the value of the selected clock at the instant of the triggering event, with bits 1 and 0 set to 0b00.

The Slot_Offset parameter indicates the number of microseconds (from 0 to 1249) from the instant at which the selected clock took the value Clock until the triggering event.

Note: What constitutes a triggering event is defined by the Controller implementation. For example, it could be an interrupt signal received by the Controller hardware.

**Event parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Which_Clock:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | 0x00 = Local Clock (Connection_Handle does not have to be valid)<br>0x01 = Piconet Clock (Connection_Handle shall be valid)<br>0x02 to 0xFF = Reserved for future use |

*Clock:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Bluetooth clock of the device requested with bits 1 and 0 set to 0b00. |

*Slot_Offset:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of microseconds from the selected clock attaining the value Clock until the triggering event.<br>Range: 0 to 1249. |

### 7.7.67 Synchronization Train Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Synchronization_Train_Complete | 0x4F | Status |

**Description:**

This event indicates that the HCI_Start_Synchronization_Train command has completed.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Start_Synchronization_Train command completed successfully. |
| 0x01 to 0xFF | HCI_Start_Synchronization_Train command failed. See [Vol 1] Part F, Controller Error Codes, for error codes and descriptions. |

### 7.7.68 Synchronization Train Received event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Synchronization_Train_Received | 0x50 | Status,<br>BD_ADDR,<br>Clock_Offset,<br>AFH_Channel_Map,<br>LT_ADDR,<br>Next_Broadcast_Instant,<br>Connectionless_Peripheral_Broadcast_Interval,<br>Service_Data |

**Description:**

This event provides information received from a synchronization train packet transmitted by a Connectionless Peripheral Broadcast transmitter with the given BD_ADDR.
If synchronization was successful, it provides the clock offset, AFH channel map, LT_ADDR, next broadcast instant, broadcast interval, and service data as received from the synchronization train payload. If the command returns a status of 0x01 to 0xFF, then all other parameters are undefined and shall be ignored.

A packet with the Connectionless Peripheral Broadcast LT_ADDR field in the payload set to zero shall be ignored for the purposes of this event.

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Receive_Synchronization_Train command completed successfully. |
| 0x01 to 0xFF | HCI_Receive_Synchronization_Train command failed. See [Vol 1] Part F, for error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Connectionless Peripheral Broadcast transmitter. |

*Clock_Offset:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | (CLKNreceiver - CLKNtransmitter) *mod* $2^{28}$ |

*AFH_Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX<br>XXXXXXXXXX<br>XX | This parameter contains 80 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>0: channel n is unused<br>1: channel n is used<br>The most significant bit (bit 79) is reserved for future use |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR of Connectionless Peripheral Broadcast channel. |
| All other values | Reserved for future use |

*Next_Broadcast_Instant:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | CLK of a future broadcast on this channel |

*Connectionless_Peripheral_Broadcast_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Interval between Connectionless Peripheral Broadcast instants in slots.<br>Range: 0x0002 to 0xFFFE; only even values are valid |

*Service_Data:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Value from octet 27 of the Synchronization Train packet; see [Vol 2] Part B, Table 8.11. |

### 7.7.69 Connectionless Peripheral Broadcast Receive event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connectionless_Peripheral_Broadcast_Receive | 0x51 | BD_ADDR,<br>LT_ADDR,<br>Clock,<br>Offset,<br>RX_Status,<br>Fragment,<br>Data_Length,<br>Data |

**Description:**

This event shall be sent by the BR/EDR Controller every Connectionless Peripheral Broadcast Instant on which the BR/EDR Controller is scheduled to receive a Connectionless Peripheral Broadcast packet. If the packet is not received successfully, the event returns a RX_Status of 0x01. Otherwise, the event returns the payload Data along with the Piconet Clock and the offset from the local CLKN when the packet was received.

The BR/EDR Controller shall send multiple HCI_Connectionless_Peripheral_Broadcast_Receive events if the length of the received data exceeds the capacity of a single HCI_Connectionless_Peripheral_Broadcast_Receive event. The fragments shall be marked as starting, continuation, or ending to allow the Host to reassemble the received packet. Only a single event shall be generated for a Connectionless Peripheral Broadcast instant on which a Connectionless Peripheral Broadcast packet was scheduled for reception but the BR/EDR Controller failed to successfully receive it.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the broadcasting (transmitter) device |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR of the Connectionless Peripheral Broadcast |
| All other values | Reserved for future use |

*Clock:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | CLK when Connectionless Peripheral Broadcast data was received |

*Offset:* *Size: 4 octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | (CLKNreceiver - CLKNtransmitter) *mod* $2^{28}$ |

*RX_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Packet received successfully |
| 0x01 | Packet not received successfully (Fragment, Data_Length, and Data fields invalid) |
| All other values | Reserved for future use |

*Fragment:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Continuation fragment |
| 0x01 | Starting fragment |
| 0x02 | Ending fragment |
| 0x03 | No fragmentation (single fragment) |
| All other values | Reserved for future use |

*Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of Data field |

*Data:* *Size: Data_Length octets*

| Parameter Description |
|---|
| Data received from a Connectionless Peripheral Broadcast packet.<br>Note: This parameter has a variable length. |

### 7.7.70 Connectionless Peripheral Broadcast Timeout event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connectionless_Peripheral_Broadcast_Timeout | 0x52 | BD_ADDR,<br>LT_ADDR |

**Description:**

On the Connectionless Peripheral Broadcast Receiver, this event indicates to the Host that the BR/EDR Controller has lost synchronization with the Connectionless Peripheral Broadcast because no Connectionless Peripheral Broadcast packets have been received for the timeout interval, *CPB_supervisionTO*, specified in the HCI_Set_Connectionless_Peripheral_Broadcast_Receive command.

On the Connectionless Peripheral Broadcast Transmitter, this event indicates to the Host that the BR/EDR Controller has been unable to transmit a Connectionless Peripheral Broadcast packet for the timeout interval, *CPB_supervisionTO*, specified in the HCI_Set_Connectionless_Peripheral_Broadcast command.

**Event parameters:**

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the broadcasting (transmitter) device |

*LT_ADDR:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | LT_ADDR of the Connectionless Peripheral Broadcast |
| All other values | Reserved for future use |

### 7.7.71 Truncated Page Complete event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Truncated_Page_Complete | 0x53 | Status,<br>BD_ADDR |

**Description:**

This event indicates to the Host that an HCI_Truncated_Page command completed. Truncated Paging is considered to be successful when a Peripheral page response ID packet has been received by the local BR/EDR Controller. See [Vol 2] Part B, Section 8.3.3 for more information.

An HCI_Truncated_Page_Complete event shall always be sent for each HCI_Truncated_Page command. If the Host issues an HCI_Truncated_Page_Cancel command before the Controller returns the HCI_Truncated_Page_Complete event, then the HCI_Truncated_Page_Complete event shall be sent after the HCI_Command_Complete event for the HCI_Truncated_Page_Cancel command. If the cancellation was successful, the HCI_Truncated_Page_Complete event shall be generated with the error code *Unknown Connection Identifier* (0x02).

**Event parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Truncated_Page command completed successfully. |
| 0x01 to 0xFF | HCITruncated_Page command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the paged (Peripheral) device |

### 7.7.72 Peripheral Page Response Timeout event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Peripheral_Page_Response_Timeout | 0x54 | *none* |

**Description:**

This event indicates to the Host that a Peripheral page response timeout has occurred in the BR/EDR Controller.

Note: This event will be generated if the Peripheral BR/EDR Controller responds to a page but does not receive the Central FHS packet (see [Vol 2] Part B, Section 8.3.3) within *pagerespTO*.

**Event parameters:**

None

### 7.7.73 Connectionless Peripheral Broadcast Channel Map Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Connectionless_Peripheral_Broadcast_Channel_Map_Change | 0x55 | Channel_Map |

**Description:**

This event is sent by the Transmitter's BR/EDR Controller to the Transmitter's Host to indicate that the Transmitter's BR/EDR Controller has moved to a new AFH channel map for the PBD logical link.

After an AFH channel map change takes effect for the PBD logical link, the Connectionless Peripheral Broadcast Transmitter BR/EDR Controller shall send this event to the Host. Upon reception of this event, the Host may restart the synchronization train to allow receivers to obtain the updated AFH channel map.

This event shall also be sent if the Host issues an HCI_Set_AFH_Host_Channel_Classification command which causes the Connectionless Peripheral Broadcast Channel Map to change.

**Event parameters:**

*Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX<br>XXXXXXXXXX<br>XX | This parameter contains 80 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>0: channel *n* is unused<br>1: channel *n* is used<br>The most significant bit (bit 79) is reserved for future use |

### 7.7.74 Inquiry Response Notification event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Inquiry_Response_Notification | 0x56 | LAP,<br>RSSI |

**Description:**

This event indicates to the Host that the BR/EDR Controller responded to an Inquiry message. The LAP parameter in the event indicates the LAP used to create the access code received. The parameter may be used by the Host to determine which access code was used in cases where the BR/EDR Controller is performing inquiry scan on multiple inquiry access codes using parallel scanning or sequential scanning. See [Vol 3] Part C, Section 4.1.2.1 for details on sequential and parallel scanning.

The LAP parameter returned by the BR/EDR Controller shall be one of the LAPs currently enabled. LAPs are enabled via the HCI_Write_Current_IAC_LAP command.

The RSSI parameter indicates the signal strength of the received ID packet.

**Event parameters:**

*LAP:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | The LAP from which the IAC was derived;<br>see Assigned Numbers.<br>Range: 0x9E8B00 to 0x9E8B3F |

*RSSI:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -100 to 20, +127 indicates unknown RSSI<br>Units: dBm |

### 7.7.75 Authenticated Payload Timeout Expired event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_Authenticated_Payload_Timeout_Expired | 0x57 | Connection_Handle |

**Description:**

This event is used to indicate that a packet containing a valid MIC on the Connection_Handle was not received within the *authenticatedPayloadTO* (see [Vol 2] Part B, Appendix B for the BR/EDR and [Vol 6] Part B, Section 5.4 for the LE connection). The Host may disconnect the link when this occurs.

**Event parameters:**

*Connection_Handle:* *Size: 2 octet (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### 7.7.76 SAM Status Change event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI_SAM_Status_Change | 0x58 | Connection_Handle,<br>Local_SAM_Index,<br>Local_SAM_TX_Availability,<br>Local_SAM_RX_Availability,<br>Remote_SAM_Index,<br>Remote_SAM_TX_Availability,<br>Remote_SAM_RX_Availability |

**Description:**

This event indicates that the Controller has changed the SAM status for the connection identified by the Connection_Handle; i.e., a new SAM slot map has been enabled or the existing one disabled.

Note: A change from one SAM slot map to another only generates one event, not two.

**Event parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Local_SAM_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The index of the current SAM slot map used by the local device.0xFF means SAM is disabled (i.e. all slots are available) |

*Local_SAM_TX_Availability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The proportion of slots available for the local device to transmit. 0 represents "less than 1 in 255" and 255 represents "all"; other proportions shall be linearly scaled. That is, Local_SAM_-TX_Availability = (total_available_TX_slots ÷ local_$T_{SAM}$) × 255, rounded to the integer below, where total_available_TX_slots is the total number of slots available for transmission in the current local SAM slot map and local_TSAM is $T_{SAM}$ for the current local SAM slot map. |

*Local_SAM_RX_Availability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The proportion of slots available for the local device to receive. 0 represents "less than 1 in 255" and 255 represents "all"; other proportions shall be linearly scaled. That is, Local_SAM_-RX_Availability = (total_available_RX_slots ÷ local_$T_{SAM}$) × 255, rounded to the integer below, where total_available_RX_slots is the total number of slots available for reception in the current local SAM slot map and local_$T_{SAM}$ is $T_{SAM}$ for the current local SAM slot map. |

*Remote_SAM_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The index of the current SAM slot map used by the remote device. 0xFF means SAM is disabled (i.e. all slots are available). |

*Remote_SAM_TX_Availability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The proportion of slots available for the remote device to transmit. 0 represents "less than 1 in 255" and 255 represents "all"; other proportions shall be linearly scaled. That is, Remote_SAM_TX_Availability = (total_available_TX_slots ÷ remote_$T_{SAM}$) × 255, rounded to the integer below, where total_available_TX_slots is the total number of slots available for transmission in the current remote SAM slot map and remote_$T_{SAM}$ is $T_{SAM}$ for the current remote SAM slot map. |

*Remote_SAM_RX_Availability:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The proportion of slots available for the remote device to receive. 0 represents "less than 1 in 255" and 255 represents "all"; other proportions shall be linearly scaled. That is, Remote_SAM_RX_Availability = (total_available_RX_slots ÷ remote_$T_{SAM}$) × 255, rounded to the integer below, where total_available_RX_slots is the total number of slots available for reception in the current remote SAM slot map and remote_$T_{SAM}$ is $T_{SAM}$ for the current remote SAM slot map. |

## 7.8 LE Controller commands

The LE Controller commands provide access and control to various capabilities of the Bluetooth hardware, as well as methods for the Host to affect how the Link Layer manages the piconet and controls connections.

For the LE Controller commands, the OGF code is defined as 0x08.

### 7.8.1 LE Set Event Mask command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Event_Mask [v2] | 0x00A4 | LE_Event_Mask | Status |
| HCI_LE_Set_Event_Mask [v1] | 0x0001 | LE_Event_Mask | Status |

**Description:**

This command is used to control which LE events are generated by the HCI for the Host. If the bit in the LE_Event_Mask is set to a one, then the event associated with that bit will be enabled. The event mask allows the Host to control which events will interrupt it.

The Controller shall ignore those bits which are reserved for future use or represent events which it does not support. If the Host sets any of these bits to 1, the Controller shall act as if they were set to 0.

For LE events to be generated, the LE Meta event bit in the Event_Mask shall also be set. If that bit is not set, then LE events shall not be generated, regardless of how the LE_Event_Mask is set.

**Command parameters:**

*LE_Event_Mask [v1]:* *Size: 8 octets*
*LE_Event_Mask [v2]:* *Size: 255 octets*

| Bit | LE Subevent Types |
|---|---|
| 0 | HCI_LE_Connection_Complete event |
| 1 | HCI_LE_Advertising_Report event |
| 2 | HCI_LE_Connection_Update_Complete event |
| 3 | HCI_LE_Read_Remote_Features_Page_0_Complete event |
| 4 | HCI_LE_Long_Term_Key_Request event |
| 5 | HCI_LE_Remote_Connection_Parameter_Request event |
| 6 | HCI_LE_Data_Length_Change event |

| Bit | LE Subevent Types |
|---|---|
| 7 | HCI_LE_Read_Local_P-256_Public_Key_Complete event |
| 8 | HCI_LE_Generate_DHKey_Complete event |
| 9 | HCI_LE_Enhanced_Connection_Complete event [v1] |
| 10 | HCI_LE_Directed_Advertising_Report event |
| 11 | HCI_LE_PHY_Update_Complete event |
| 12 | HCI_LE_Extended_Advertising_Report event |
| 13 | HCI_LE_Periodic_Advertising_Sync_Established event [v1] |
| 14 | HCI_LE_Periodic_Advertising_Report event [v1] |
| 15 | HCI_LE_Periodic_Advertising_Sync_Lost event |
| 16 | HCI_LE_Scan_Timeout event |
| 17 | HCI_LE_Advertising_Set_Terminated event |
| 18 | HCI_LE_Scan_Request_Received event |
| 19 | HCI_LE_Channel_Selection_Algorithm event |
| 20 | HCI_LE_Connectionless_IQ_Report event |
| 21 | HCI_LE_Connection_IQ_Report event |
| 22 | HCI_LE_CTE_Request_Failed event |
| 23 | HCI_LE_Periodic_Advertising_Sync_Transfer_Received event [v1] |
| 24 | HCI_LE_CIS_Established event [v1] |
| 25 | HCI_LE_CIS_Request event |
| 26 | HCI_LE_Create_BIG_Complete event |
| 27 | HCI_LE_Terminate_BIG_Complete event |
| 28 | HCI_LE_BIG_Sync_Established event |
| 29 | HCI_LE_BIG_Sync_Lost event |
| 30 | HCI_LE_Request_Peer_SCA_Complete event |
| 31 | HCI_LE_Path_Loss_Threshold event |
| 32 | HCI_LE_Transmit_Power_Reporting event |
| 33 | HCI_LE_BIGInfo_Advertising_Report event |
| 34 | HCI_LE_Subrate_Change event |
| 35 | HCI_LE_Periodic_Advertising_Sync_Established event [v2] |
| 36 | HCI_LE_Periodic_Advertising_Report event [v2] |
| 37 | HCI_LE_Periodic_Advertising_Sync_Transfer_Received event [v2] |
| 38 | HCI_LE_Periodic_Advertising_Subevent_Data_Request event |
| 39 | HCI_LE_Periodic_Advertising_Response_Report event |

| Bit | LE Subevent Types |
|---|---|
| 40 | HCI_LE_Enhanced_Connection_Complete event [v2] |
| 41 | HCI_LE_CIS_Established event [v2] |
| 42 | HCI_LE_Read_All_Remote_Features_Complete event |
| 43 | HCI_LE_CS_Read_Remote_Supported_Capabilities_Complete event [v1] |
| 44 | HCI_LE_CS_Read_Remote_FAE_Table_Complete event |
| 45 | HCI_LE_CS_Security_Enable_Complete event |
| 46 | HCI_LE_CS_Config_Complete event |
| 47 | HCI_LE_CS_Procedure_Enable_Complete event |
| 48 | HCI_LE_CS_Subevent_Result event |
| 49 | HCI_LE_CS_Subevent_Result_Continue event |
| 50 | HCI_LE_CS_Test_End_Complete event |
| 51 | HCI_LE_Monitored_Advertisers_Report event |
| 52 | HCI_LE_Frame_Space_Update_Complete event |
| 53 | HCI_LE_UTP_Receive event |
| 54 | HCI_LE_Connection_Rate_Change event |
| 55 | HCI_LE_CS_Read_Remote_Supported_Capabilities_Complete event [v2] |
| 60 to 63 | Reserved for specification development purposes |

The value with all bits set to 0 indicates that no events are specified. The default is for just bits 0 to 4 to be set.

All bits not listed in this table are reserved for future use.

The [v1] command sets bits 0 to 63 of the event mask and leaves bits 64 and above unchanged. The [v2] command sets all the bits of the event mask.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Event_Mask command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Event_Mask command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Event_Mask command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.2 LE Read Buffer Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Buffer_Size [v2] | 0x0060 | *none* | Status,<br>LE_ACL_Data_Packet_Length,<br>Total_Num_LE_ACL_Data_Packets,<br>ISO_Data_Packet_Length,<br>Total_Num_ISO_Data_Packets |
| HCI_LE_Read_Buffer_Size [v1] | 0x0002 | *none* | Status,<br>LE_ACL_Data_Packet_Length,<br>Total_Num_LE_ACL_Data_Packets |

**Description:**

This command is used to read the maximum size of the data portion of ACL data packets and isochronous data packets sent from the Host to the Controller. The Host shall fragment the data transmitted to the Controller according to these values so that the HCI ACL Data packets and HCI ISO Data packets will contain data up to this size ("data" includes optional fields in the HCI ISO Data packet, such as ISO_SDU_Length). The HCI_LE_Read_Buffer_Size command also returns the total number of HCI LE ACL Data packets and isochronous data packets that can be stored in the data buffers of the Controller. The HCI_LE_Read_Buffer_Size command shall be issued by the Host before it sends any data to an LE Controller (see Section 4.1.1). If the Controller supports HCI ISO Data packets, it shall return non-zero values for the ISO_Data_Packet_Length and Total_Num_ISO_Data_Packets parameters.

If the Controller returns a length value of zero for ACL data packets, the Host shall use the HCI_Read_Buffer_Size command to determine the size of the data buffers (shared between BR/EDR and LE transports).

Both the HCI_Read_Buffer_Size command and the HCI_LE_Read_Buffer_Size command may return buffer length and number of packets parameter values that are nonzero. This allows a Controller to offer different buffers and number of buffers for BR/EDR data packets and LE data packets.

The LE_ACL_Data_Packet_Length parameter shall be used to determine the maximum size of the L2CAP PDU fragments that are contained in ACL data packets, and which are transferred from the Host to the Controller to be broken up into packets by the Link Layer. The Total_Num_LE_ACL_Data_Packets parameter contains the total number of HCI ACL Data packets that can be stored in the data buffers of the Controller. The Host determines how to divide the buffers between different connection handles.

The ISO_Data_Packet_Length parameter shall be used to determine the maximum size of the SDU fragments that are contained in isochronous data packets, and which are transferred from the Host to the Controller. The Total_Num_ISO_Data_Packets parameter contains the total number of isochronous data packets that can be stored in the data buffers of the Controller. The Host determines how to divide the buffers between different connection handle(s).

Note: The LE_ACL_Data_Packet_Length and ISO_Data_Packet_Length return parameters do not include the length of the HCI ACL Data packet header or the HCI ISO Data packet header respectively.

**Command parameters:**

None

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Read_Buffer_Size command succeeded. |
| 0x01 to 0xFF | The HCI_LE_Read_Buffer_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*LE_ACL_Data_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. |
| 0x0001 to 0x001A | Reserved for future use. |
| 0x001B to 0xFFFF | Maximum length (in octets) of the data portion of each HCI ACL data packet. |

*Total_Num_LE_ACL_Data_Packets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No dedicated LE Buffer exists. Use the HCI_Read_Buffer_Size command. |
| 0x01 to 0xFF | The total number of HCI ACL data packets that can be stored in the data buffers of the Controller. |

*ISO_Data_Packet_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No dedicated ISO Buffer exists. |
| 0x0001 to 0xFFFF | The maximum length (in octets) of the data portion of each HCI ISO data packet. |

*Total_Num_ISO_Data_Packets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No dedicated ISO Buffer exists. |
| 0x01 to 0xFF | The total number of HCI ISO data packets that can be stored in the ISO buffers of the Controller. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Buffer_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.3 LE Read Local Supported Features Page 0 command[1]

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Local_Supported_-Features_Page_0 | 0x0003 | *none* | Status,<br>LE_Features |

**Description:**

This command requests page 0 of the list of the supported LE features for the Controller.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Local_Supported_Features_Page_0 command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Local_Supported_Features_Page_0 command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*LE_Features:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Bit Mask List of page 0 of the supported LE features. See [Vol 6] Part B, Section 4.6. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Local_Supported_Features_Page_0 command has completed, an HCI_Command_Complete event shall be generated.

[1]This command was formerly called “LE Read Local Supported Features”.

### 7.8.4 LE Set Random Address command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Random_Address | 0x0005 | Random_Address | Status |

**Description:**

This command is used by the Host to set the LE Random Device Address in the Controller (see [Vol 6] Part B, Section 1.3).

If this command is used to change the address, the new random address shall take effect for advertising no later than the next successful HCI_LE_Set_Advertising_Enable command, for scanning no later than the next successful HCI_LE_Set_Scan_Enable command or HCI_LE_Set_Extended_Scan_Enable command, and for initiating or creating a connection from a PAwR train no later than the next successful HCI_LE_Create_Connection command or HCI_LE_Extended_Create_Connection command.

Note: If the extended advertising commands are in use, this command only affects the address used for scanning and initiating. The addresses used for advertising are set by the HCI_LE_Set_Advertising_Set_Random_Address command (see Section 7.8.52).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Any of advertising (created using legacy advertising commands), scanning, or creating a connection are enabled in the Controller. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Random_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Random Device Address as defined by [Vol 6] Part B, Section 1.3. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Random_Address command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Random_Address command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Random_Address command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.5 LE Set Advertising Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Advertising_Parameters | 0x0006 | Advertising_Interval_Min,<br>Advertising_Interval_Max,<br>Advertising_Type,<br>Own_Address_Type,<br>Peer_Address_Type,<br>Peer_Address,<br>Advertising_Channel_Map,<br>Advertising_Filter_Policy | Status |

**Description:**

This command is used by the Host to set the advertising parameters.

The Advertising_Interval_Min shall be less than or equal to the Advertising_Interval_Max. The Advertising_Interval_Min and Advertising_Interval_Max should not be the same value to enable the Controller to determine the best advertising interval given other activities.

For high duty cycle directed advertising, i.e. when Advertising_Type is 0x01 (ADV_DIRECT_IND, high duty cycle), the Advertising_Interval_Min and Advertising_Interval_Max parameters are not used and shall be ignored.

The Advertising_Type is used to determine the packet type that is used for advertising when advertising is enabled.

Own_Address_Type parameter indicates the type of address being used in the advertising packets.

If Own_Address_Type equals 0x02 or 0x03, the Peer_Address parameter contains the peer's Identity Address and the Peer_Address_Type parameter contains the Peer's Identity Type (i.e. 0x00 or 0x01). These parameters are used to locate the corresponding local IRK in the resolving list; this IRK is used to generate the own address used in the advertisement.

If directed advertising is performed, i.e. when Advertising_Type is set to 0x01 (ADV_DIRECT_IND, high duty cycle) or 0x04 (ADV_DIRECT_IND, low duty cycle mode), then the Peer_Address_Type and Peer_Address shall be valid.

If Own_Address_Type equals 0x02 or 0x03, the Controller generates the peer's Resolvable Private Address using the peer's IRK corresponding to the peer's Identity

Address contained in the Peer_Address parameter and peer's Identity Address Type (i.e. 0x00 or 0x01) contained in the Peer_Address_Type parameter.

The Advertising_Channel_Map is a bit field that indicates the advertising channel indices that shall be used when transmitting advertising packets. At least one channel bit shall be set in the Advertising_Channel_Map parameter.

The Advertising_Filter_Policy parameter shall be ignored when directed advertising is enabled.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising is enabled in the Controller. | *Command Disallowed* (0x0C) |
| MC | The advertising interval range (Advertising_Interval_Min, Advertising_Interval_Max) does not overlap with the advertising interval range supported by the Controller. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Advertising_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum advertising interval for undirected and low duty cycle directed advertising.<br>Range: 0x0020 to 0x4000<br>Default: 0x0800 (1.28 s)<br>Time = N × 0.625 ms<br>Time Range: 20 ms to 10.24 s |

*Advertising_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum advertising interval for undirected and low duty cycle directed advertising.<br>Range: 0x0020 to 0x4000<br>Default: 0x0800 (1.28 s)<br>Time = N × 0.625 ms<br>Time Range: 20 ms to 10.24 s |

*Advertising_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connectable and scannable undirected advertising (ADV_IND) (default) |
| 0x01 | Connectable high duty cycle directed advertising (ADV_DIRECT_IND, high duty cycle) |
| 0x02 | Scannable undirected advertising (ADV_SCAN_IND) |
| 0x03 | Non connectable undirected advertising (ADV_NONCONN_IND) |
| 0x04 | Connectable low duty cycle directed advertising (ADV_DIRECT_IND, low duty cycle) |
| All other values | Reserved for future use |

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address (default) |
| 0x01 | Random Device Address |
| 0x02 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. |
| 0x03 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Random_Address. |
| All other values | Reserved for future use |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address (default) or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected. |

*Advertising_Channel_Map:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Channel 37 shall be used |
| 1 | Channel 38 shall be used |
| 2 | Channel 39 shall be used |
| All other bits | Reserved for future use |

The default is 0x07 (all three channels enabled).

*Advertising_Filter_Policy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Process scan and connection requests from all devices (i.e., the Filter Accept List is not in use) (default). |
| 0x01 | Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List. |
| 0x02 | Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List. |
| 0x03 | Process scan and connection requests only from devices in the Filter Accept List. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Advertising_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Advertising_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Advertising_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.6 LE Read Advertising Physical Channel Tx Power command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Advertising_Channel_Tx_-Power | 0x0007 | *none* | Status,<br>TX_Power_Level |

**Description:**

This command is used by the Host to read the transmit power level used for LE advertising physical channel packets.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Advertising_Physical_Channel_Tx_Power command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Advertising_Physical_Channel_Tx_Power failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to 20<br>Units: dBm<br>Accuracy: ±4 dB |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Advertising_Physical_Channel_Tx_Power command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.7 LE Set Advertising Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Advertising_Data | 0x0008 | Advertising_Data_Length, Advertising_Data | Status |

**Description:**

This command is used to set the data used in advertising packets that have a data field.

Only the significant part of the Advertising_Data should be transmitted in the advertising packets, as defined in [Vol 3] Part C, Section 11.

If advertising is currently enabled, the Controller shall use the new data in subsequent advertising events. If an advertising event is in progress when this command is issued, the Controller may use the old or new data for that event. If advertising is currently disabled, the data shall be kept by the Controller and used once advertising is enabled.

The default Advertising_Data_Length shall be zero and the default Advertising_Data shall be 31 all-zero octets.

**Command parameters:**

*Advertising_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | The number of significant octets in the Advertising_Data. |

*Advertising_Data:* *Size: 31 octets*

| Parameter Description |
|---|
| 31 octets of advertising data formatted as defined in [Vol 3] Part C, Section 11. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Advertising_Data command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Advertising_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Advertising_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.8 LE Set Scan Response Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Scan_Response_-Data | 0x0009 | Scan_Response_Data_Length, Scan_Response_Data | Status |

**Description:**

This command is used to provide data used in Scanning Packets that have a data field.

Only the significant part of the Scan_Response_Data should be transmitted in the Scanning Packets, as defined in [Vol 3] Part C, Section 11.

If advertising is currently enabled, the Controller shall use the new data in subsequent advertising events. If an advertising event is in progress when this command is issued, the Controller may use the old or new data for that event. If advertising is currently disabled, the data shall be kept by the Controller and used once advertising is enabled.

The default Scan_Response_Data_Length shall be zero and the default Scan_Response_Data shall be 31 all-zero octets.

**Command parameters:**

*Scan_Response_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | The number of significant octets in the Scan_Response_Data. |

*Scan_Response_Data:* *Size: 31 octets*

| Parameter Description |
|---|
| 31 octets of Scan_Response_Data formatted as defined in [Vol 3] Part C, Section 11. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Scan_Response_Data command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Scan_Response_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Scan_Response_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.9 LE Set Advertising Enable command[1]

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Advertising_Enable | 0x000A | Advertising_Enable | Status |

**Description:**

This command is used to request the Controller to start or stop advertising. The Controller manages the timing of advertisements as per the advertising parameters given in the HCI_LE_Set_Advertising_Parameters command.

The Controller shall continue advertising until the Host issues an HCI_LE_Set_Advertising_Enable command with Advertising_Enable set to 0x00 (Advertising is disabled), a connection is created using the advertising, or the Advertising is timed out due to high duty cycle Directed Advertising. In these cases, advertising is then disabled.

Enabling advertising when it is already enabled can cause the random address to change. Disabling advertising when it is already disabled has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| R | Advertising_Enable is set to 0x01, the advertising parameters' Own_Address_Type parameter is set to 0x00, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Advertising_Enable is set to 0x01, the advertising parameters' Own_Address_Type parameter is set to 0x01, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |
| R | Advertising_Enable is set to 0x01, the advertising parameters' Own_Address_Type parameter is set to 0x02, the Controller's resolving list does not contain a matching entry, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Advertising_Enable is set to 0x01, the advertising parameters' Own_Address_Type parameter is set to 0x03, the Controller's resolving list does not contain a matching entry, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |

[1]This command was formerly called “LE Set Advertise Enable”.

**Command parameters:**

*Advertising_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Advertising is disabled (default) |
| 0x01 | Advertising is enabled. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Advertising_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Advertising_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Advertising_Enable command has completed, an HCI_Command_Complete event shall be generated.

If the Advertising_Type parameter is 0x01 (ADV_DIRECT_IND, high duty cycle) and the directed advertising fails to create a connection, an HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event shall be generated with the Status code set to Advertising Timeout (0x3C).

If the Advertising_Type parameter is 0x00 (ADV_IND), 0x01 (ADV_DIRECT_IND, high duty cycle), or 0x04 (ADV_DIRECT_IND, low duty cycle) and a connection is created, an HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event shall be generated.

Note: There is a possible race condition if the Advertising_Enable parameter is set to 0x00 (Disable) and the Advertising_Type parameter is 0x00, 0x01, or 0x04. The advertisements might not be stopped before a connection is created, and therefore both the HCI_Command_Complete event and either an HCI_LE_Connection_Complete event or an HCI_LE_Enhanced_Connection_Complete event could be generated. This can also occur when high duty cycle directed advertising is timed out and this command disables advertising.

### 7.8.10 LE Set Scan Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Scan_Parameters | 0x000B | LE_Scan_Type,<br>LE_Scan_Interval,<br>LE_Scan_Window,<br>Own_Address_Type,<br>Scanning_Filter_Policy | Status |

**Description:**

This command is used to set the scan parameters.

The LE_Scan_Type parameter controls the type of scan to perform.

The LE_Scan_Interval and LE_Scan_Window parameters are recommendations from the Host on how long (LE_Scan_Window) and how frequently (LE_Scan_Interval) the Controller should scan (See [Vol 6] Part B, Section 4.4.3). The LE_Scan_Window parameter shall always be set to a value smaller or equal to the value set for the LE_Scan_Interval parameter. If they are set to the same value scanning should be run continuously.

Own_Address_Type parameter indicates the type of address being used in the scan request packets.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Scanning is enabled in the Controller. | *Command Disallowed* (0x0C) |

**Command parameters:**

*LE_Scan_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Passive Scanning. No scanning PDUs shall be sent (default) |
| 0x01 | Active scanning. Scanning PDUs may be sent. |
| All other values | Reserved for future use |

*LE_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | This is defined as the time interval from when the Controller started its last LE scan until it begins the subsequent LE scan.<br>Range: 0x0004 to 0x4000<br>Default: 0x0010 (10 ms)<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 10.24 s |

*LE_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The duration of the LE scan. LE_Scan_Window shall be less than or equal to LE_Scan_Interval<br>Range: 0x0004 to 0x4000<br>Default: 0x0010 (10 ms)<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 10.24 s |

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address (default) |
| 0x01 | Random Device Address |
| 0x02 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. |
| 0x03 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Random_Address. |
| All other values | Reserved for future use. |

*Scanning_Filter_Policy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Basic unfiltered scanning filter policy |
| 0x01 | Basic filtered scanning filter policy |
| 0x02 | Extended unfiltered scanning filter policy |

| Value | Parameter Description |
|---|---|
| 0x03 | Extended filtered scanning filter policy |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Scan_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Scan_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Scan_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.11 LE Set Scan Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Scan_Enable | 0x000C | LE_Scan_Enable,<br>Filter_Duplicates | Status |

**Description:**

This command is used to start and stop scanning for legacy PDUs (but not extended PDUs, even if the device supports extended advertising). Scanning is used to discover advertising devices nearby.

The Filter_Duplicates parameter controls whether the Link Layer should filter out duplicate advertising reports (Filtering_Enabled) to the Host, or if the Link Layer should generate advertising reports for each packet received (Filtering_Disabled). See [Vol 6] Part B, Section 4.4.3.5. If LE_Scan_Enable is set to 0x00 then Filter_Duplicates shall be ignored.

If the LE_Scan_Enable parameter is set to 0x01 and scanning is already enabled, any change to the Filter_Duplicates setting shall take effect.

Disabling scanning when it is disabled has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| R | LE_Scan_Enable is set to 0x01, the scanning parameters' Own_Address_Type parameter is set to 0x00 or 0x02, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | LE_Scan_Enable is set to 0x01, the scanning parameters' Own_Address_Type parameter is set to 0x01 or 0x03, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*LE_Scan_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Scanning disabled. |
| 0x01 | Scanning enabled. |
| All other values | Reserved for future use. |

*Filter_Duplicates:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Duplicate filtering disabled. |
| 0x01 | Duplicate filtering enabled. |
| All other values | Reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Scan_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Scan_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Scan_Enable command has completed, an HCI_Command_Complete event shall be generated.

Zero or more HCI_LE_Advertising_Report events are generated by the Controller based on legacy advertising packets received and the duplicate filtering. More than one advertising packet may be reported in each HCI_LE_Advertising_Report event. No report shall be issued for extended advertising PDUs.

When the Scanning_Filter_Policy is set to 0x02 or 0x03 (see Section 7.8.10) and a directed advertisement was received where the advertiser used a resolvable private address which the Controller is unable to resolve, an HCI_LE_Directed_Advertising_Report event shall be generated instead of an HCI_LE_Advertising_Report event.

### 7.8.12 LE Create Connection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Create_Connection | 0x000D | LE_Scan_Interval,<br>LE_Scan_Window,<br>Initiator_Filter_Policy,<br>Peer_Address_Type,<br>Peer_Address,<br>Own_Address_Type,<br>Connection_Interval_Min,<br>Connection_Interval_Max,<br>Max_Latency,<br>Supervision_Timeout,<br>Min_CE_Length,<br>Max_CE_Length | *none* |

**Description:**

This command is used to create a single ACL connection, with the local device in the Central role, to a connectable advertiser.

If a connection is created with the local device in the Peripheral role while this command is pending, then this command remains pending.

The LE_Scan_Interval and LE_Scan_Window parameters are recommendations from the Host on how long (LE_Scan_Window) and how frequently (LE_Scan_Interval) the Controller should scan. The LE_Scan_Window parameter shall be set to a value smaller or equal to the value set for the LE_Scan_Interval parameter. If both are set to the same value, scanning should run continuously.

The Initiator_Filter_Policy is used to determine whether the Filter Accept List is used. If the Filter Accept List is used, then the device to connect to is the first one that sends an advertisement received successfully by the local device and that is on the list. If the Filter Accept List is not used, then the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the advertising device to connect to.

Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer. The Host shall not set Peer_Address_Type to either 0x02 or 0x03 if both the Host and the Controller support the HCI_LE_Set_Privacy_Mode command. If a Controller that supports the HCI_LE_Set_Privacy_Mode command receives the HCI_LE_Create_Connection

command with Peer_Address_Type set to either 0x02 or 0x03, it may use either device privacy mode or network privacy mode for that peer device.

Peer_Address parameter indicates the Peer's Public Device Address, Random (static) Device Address, Non-Resolvable Private Address or Resolvable Private Address depending on the Peer_Address_Type parameter.

Own_Address_Type parameter indicates the type of address being used in the connection request packets.

The Connection_Interval_Min and Connection_Interval_Max parameters define the minimum and maximum allowed connection interval. The Connection_Interval_Min parameter shall not be greater than the Connection_Interval_Max parameter.

The Max_Latency parameter defines the maximum allowed Peripheral latency (see [Vol 6] Part B, Section 4.5.1).

The Supervision_Timeout parameter defines the link supervision timeout for the connection. The Supervision_Timeout in milliseconds shall be larger than (1 + Max_Latency) × Connection_Interval_Max × 2, where Connection_Interval_Max is given in milliseconds. (See [Vol 6] Part B, Section 4.5.2).

The Min_CE_Length and Max_CE_Length parameters provide the Controller with the expected minimum and maximum length of the connection events. The Controller is not required to use these values.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Another HCI_LE_Create_Connection command is pending in the Controller. | *Command Disallowed* (0x0C) |
| M | The local device is already connected to the same device address as the advertiser (including two different Resolvable Private Addresses that resolve to the same IRK). | *Connection Already Exists* (0x0B) |
| R | Own_Address_Type is set to 0x00 and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is set to 0x01 and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |
| R | Own_Address_Type is set to 0x02, Initiator_Filter_Policy is set to 0x00, the Controller's resolving list does not contain a matching entry, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| R | Own_Address_Type is set to 0x02, Initiator_Filter_Policy is set to 0x01, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is set to 0x03, Initiator_Filter_Policy is set to 0x00, the Controller's resolving list does not contain a matching entry, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is set to 0x03, Initiator_Filter_Policy is set to 0x01, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |
| R | Max_CE_Length is less than Min_CE_Length | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*LE_Scan_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | This is defined as the time interval from when the Controller started its last LE scan until it begins the subsequent LE scan.<br>Range: 0x0004 to 0x4000<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 10.24 s |

*LE_Scan_Window:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Amount of time for the duration of the LE scan. LE_Scan_Window shall be less than or equal to LE_Scan_Interval<br>Range: 0x0004 to 0x4000<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 10.24 s |

*Initiator_Filter_Policy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Filter Accept List is not used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be used. |
| 0x01 | Filter Accept List is used to determine which advertiser to connect to. Peer_Address_Type and Peer_Address shall be ignored. |
| All other values | Reserved for future use. |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Public Identity Address (Corresponds to peer's Resolvable Private Address). This value shall only be used by the Host if either the Host or the Controller does not support the HCI_LE_Set_Privacy_Mode command. |
| 0x03 | Random (static) Identity Address (Corresponds to peer's Resolvable Private Address). This value shall only be used by a Host if either the Host or the Controller does not support the HCI_LE_Set_Privacy_Mode command. |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected |

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. |
| 0x03 | Controller generates Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from the most recent successful HCI_LE_Set_Random_Address command. |
| All other values | Reserved for future use |

*Connection_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum value for the connection interval. This shall be less than or equal to Connection_Interval_Max.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s. |

*Connection_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum value for the connection interval. This shall be greater than or equal to Connection_Interval_Min.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s. |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency for the connection in number of connection events.<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for the LE Link. (See [Vol 6] Part B, Section 4.5.2)<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Min_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The minimum length of connection event recommended for this LE connection.<br>Time = N × 0.625 ms. |

*Max_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The maximum length of connection event recommended for this LE connection.<br>Time = N × 0.625 ms. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Create_Connection command, the Controller sends the HCI_Command_Status event to the Host. An HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event shall be generated when a connection is created because of this command or the connection creation procedure is cancelled; until one of these events is generated, the command is considered pending. If a connection is created and the Controller supports the LE Channel Selection Algorithm #2 feature, this event shall be immediately followed by an HCI_LE_Channel_Selection_Algorithm event.

### 7.8.13 LE Create Connection Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Create_Connection_Cancel | 0x000E | *none* | Status |

**Description:**

This command is used to cancel the HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection commands.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | No HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command is pending. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Create_Connection_Cancel command succeeded. |
| 0x01 to 0xFF | HCI_LE_Create_Connection_Cancel command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Create_Connection_Cancel command has completed, an HCI_Command_Complete event shall be generated.

If the cancellation was successful then, after the HCI_Command_Complete event for the HCI_LE_Create_Connection_Cancel command, either an HCI_LE_Connection_Complete or an HCI_LE_Enhanced_Connection_Complete event shall be generated. In either case, the event shall be sent with the error code *Unknown Connection Identifier* (0x02).

### 7.8.14 LE Read Filter Accept List Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Filter_Accept_List_Size | 0x000F | *none* | Status,<br>Filter_Accept_List_Size |

**Description:**

This command is used to read the number of Filter Accept List entries (including those already stored there) that the Controller can store at the present time.

Note: The number of entries that can be stored is not fixed and the Controller can change it at any time (e.g., because the memory used to store the list can also be used for other purposes).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Filter_Accept_List_Size command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Filter_Accept_List_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Filter_Accept_List_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Total number of Filter Accept List entries that can be stored in the Controller. |
| 0x00 | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Filter_Accept_List_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.15 LE Clear Filter Accept List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Clear_Filter_Accept_List | 0x0010 | *none* | Status |

**Description:**

This command is used to clear the Filter Accept List stored in the Controller.

This command shall not be used when:

- any advertising filter policy uses the Filter Accept List and advertising is enabled,
- the scanning filter policy uses the Filter Accept List and scanning is enabled, or
- the initiator filter policy uses the Filter Accept List and an HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command is pending.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Clear_Filter_Accept_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Clear_Filter_Accept_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Clear_Filter_Accept_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.16 LE Add Device To Filter Accept List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Add_Device_To_Filter_Accept_-List | 0x0011 | Address_Type,<br>Address | Status |

**Description:**

This command is used to add a single device to the Filter Accept List stored in the Controller.

This command shall not be used when:

- any advertising filter policy uses the Filter Accept List and advertising is enabled,
- the scanning filter policy uses the Filter Accept List and scanning is enabled, or
- the initiator filter policy uses the Filter Accept List and an HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command is pending.

If the device is already in the Filter Accept List, the Controller should not add the device to the Filter Accept List again and should return success.

Address shall be ignored when Address_Type is set to 0xFF.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller cannot add a device to the Filter Accept List because there is no space available. | *Memory Capacity Exceeded* (0x07) |

**Command parameters:**

*Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0xFF | Devices sending anonymous advertisements |
| All other values | Reserved for future use. |

*Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address or Random Device Address of the device to be added to the Filter Accept List. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Add_Device_To_Filter_Accept_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Add_Device_To_Filter_Accept_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Add_Device_To_Filter_Accept_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.17 LE Remove Device From Filter Accept List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_Device_From_Filter_-Accept_List | 0x0012 | Address_Type,<br>Address | Status |

**Description:**

This command is used to remove a single device from the Filter Accept List stored in the Controller.

This command shall not be used when:

- any advertising filter policy uses the Filter Accept List and advertising is enabled,
- the scanning filter policy uses the Filter Accept List and scanning is enabled, or
- the initiator filter policy uses the Filter Accept List and an HCI_LE_Create_Connection or HCI_LE_Extended_Create_Connection command is pending.

Address shall be ignored when Address_Type is set to 0xFF.

**Command parameters:**

*Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0xFF | Devices sending anonymous advertisements |
| All other values | Reserved for future use. |

*Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address or Random Device Address of the device to be removed from the Filter Accept List. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remove_Device_From_Filter_Accept_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Remove_Device_From_Filter_Accept_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_Device_From_Filter_Accept_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.18 LE Connection Update command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Connection_Update | 0x0013 | Connection_Handle,<br>Connection_Interval_Min,<br>Connection_Interval_Max,<br>Max_Latency,<br>Supervision_Timeout,<br>Min_CE_Length,<br>Max_CE_Length | *none* |

**Description:**

This command is used to change the ACL connection parameters. This command may be issued on both the Central and Peripheral.

The Connection_Interval_Min and Connection_Interval_Max parameters are used to define the minimum and maximum allowed connection interval. The Connection_Interval_Min parameter shall not be greater than the Connection_Interval_Max parameter.

The Max_Latency parameter shall define the maximum allowed Peripheral latency.

The Supervision_Timeout parameter shall define the link supervision timeout for the LE link. The Supervision_Timeout in milliseconds shall be larger than (1 + Max_Latency) × Subrate_Factor × Connection_Interval_Max × 2, where Connection_Interval_Max is given in milliseconds and Subrate_Factor is the current subrate factor of the connection.

The Min_CE_Length and Max_CE_Length are information parameters providing the Controller with a hint about the expected minimum and maximum length of the connection events.

The actual parameter values selected by the Link Layer may be different from the parameter values provided by the Host through this command.

If this command completes successfully and the connection interval has changed, then the subrating factor shall be set to 1 and the continuation number to 0. In this case, Max_Latency must be interpreted in underlying connection events. Otherwise the subrating factor and continuation number shall be unchanged and Max_Latency must be interpreted in subrated events.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | One or more CS procedures are enabled. | *Command Disallowed* (0x0C) |
| M | The Controller is the Peripheral and the local Controller does not support the Connection Parameters Request procedure (see [Vol 6] Part B, Section 5.1.7). | *Unsupported Feature or Parameter Value* (0x11) |
| M | The Controller is the Peripheral and the local Controller supports the Connection Parameters Request procedure but the peer Controller does not. | *Unsupported Remote Feature* (0x1A) |
| R | Max_CE_Length is less than Min_CE_Length | *Invalid HCI Command Parameters* (0x12) |
| MC | A Connection Rate Request procedure or a Connection Rate Update procedure is in progress. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Connection_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum value for the connection interval. This shall be less than or equal to Connection_Interval_Max.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s. |

*Connection_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum value for the connection interval. This shall be greater than or equal to Connection_Interval_Min.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s. |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency for the connection in number of subrated connection events.<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for the LE Link.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Min_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Information parameter about the minimum length of connection event needed for this LE connection. How this value is used is outside the scope of this specification.<br>Time = N × 0.625 ms. |

*Max_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Information parameter about the maximum length of connection event needed for this LE connection. How this value is used is outside the scope of this specification.<br>Time = N × 0.625 ms. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Connection_Update command, the Controller sends the HCI_Command_Status event to the Host. The HCI_LE_Connection_Update_Complete event shall be generated after the connection parameters have been applied by the Controller or if the command subsequently fails.

### 7.8.19 LE Set Host Channel Classification command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Host_Channel_Classification | 0x0014 | Channel_Map | Status |

**Description:**

This command allows the Host to specify a channel classification for the data, secondary advertising, periodic, and isochronous physical channels based on its "local information". This classification persists until overwritten with a subsequent HCI_LE_Set_Host_Channel_Classification command or until the Controller is reset using the HCI_Reset command (see [Vol 6] Part B, Section 4.5.8.1).

If this command is used, the Host should send it within 10 seconds of knowing that the channel classification has changed. The interval between two successive commands sent shall be at least one second.

**Command parameters:**

*Channel_Map:* *Size: 5 octets (37 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX | This parameter contains 37 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 36) contains the value for the Link Layer channel index n.<br>Channel n is bad = 0.<br>Channel n is unknown = 1.<br>The most significant bits are reserved for future use.<br>At least one channel shall be marked as unknown. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Host_Channel_Classification command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Host_Channel_Classification command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Host_Channel_Classification command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.20 LE Read Channel Map command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Channel_Map | 0x0015 | Connection_Handle | Status,<br>Connection_Handle,<br>Channel_Map |

**Description:**

This command returns the current Channel_Map for the specified Connection_Handle. The returned value indicates the state of the Channel_Map specified by the last transmitted or received Channel_Map (in a CONNECT_IND or LL_CHANNEL_MAP_IND message) for the specified Connection_Handle, regardless of whether the Central has received an acknowledgment.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| M | Connection_Handle does not identify an ACL connection. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Channel_Map command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Channel_Map command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Channel_Map:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX | This parameter contains 37 1-bit fields.<br>The $n^{th}$ such field (in the range 0 to 36) contains the value for the Link Layer channel index n.<br>Channel n is unused = 0.<br>Channel n is used = 1.<br>The most significant bits are reserved for future use. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Channel_Map command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.21 LE Read Remote Features Page 0 command[1]

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Remote_Features_Page_0 | 0x0016 | Connection_Handle | *none* |

**Description:**

This command requests, from the remote device identified by the Connection_Handle, page 0 of the features used on the connection and the features supported by the remote device. For details see [Vol 6] Part B, Section 4.6.

This command may be issued on both the Central and Peripheral.

If a connection already exists between the two devices and page 0 of the features have already been fetched on that connection, the Controller may use a cached copy of page 0 of the features.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Read_Remote_Features_Page_0 command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the procedure to determine the remote features or has determined that it will be using a cached copy, the Controller shall send an HCI_LE_Read_Remote_Features_Page_0_Complete event to the Host.

The HCI_LE_Read_Remote_Features_Page_0_Complete event contains the status of this command and the parameter describing page 0 of the features used on the connection and the features supported by the remote device.

[1]This command was formerly called “LE Read Remote Features”.

### 7.8.22 LE Encrypt command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Encrypt | 0x0017 | Key,<br>Plaintext_Data | Status,<br>Encrypted_Data |

**Description:**

This command is used to request the Controller to encrypt the Plaintext_Data in the command using the Key given in the command and returns the Encrypted_Data to the Host. The AES-128 bit block cypher is defined in NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf).

**Command parameters:**

*Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128 bit key for the encryption of the data given in the command.<br>The most significant octet of the key corresponds to key[0] using the notation specified in FIPS 197. |

*Plaintext_Data:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128 bit data block that is requested to be encrypted.<br>The most significant octet of the PlainText_Data corresponds to in[0] using the notation specified in FIPS 197. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Encrypt command succeeded. |
| 0x01 to 0xFF | HCI_LE_Encrypt command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Encrypted_Data:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128 bit encrypted data block.<br>The most significant octet of the Encrypted_Data corresponds to out[0] using the notation specified in FIPS 197. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Encrypt command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.23 LE Rand command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Rand | 0x0018 | *none* | Status,<br>Random_Number |

**Description:**

This command is used to request the Controller to generate 8 octets of random data to be sent to the Host. The Random_Number shall be generated according to [Vol 2] Part H, Section 2.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Rand command succeeded. |
| 0x01 to 0xFF | HCI_LE_Rand command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Random_Number:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | Random Number |

**Event(s) generated (unless masked away):**

When the HCI_LE_Rand command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.24 LE Enable Encryption command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Enable_Encryption | 0x0019 | Connection_Handle,<br>Random_Number,<br>Encrypted_Diversifier,<br>Long_Term_Key | *none* |

**Description:**

This command is used to authenticate the given encryption key associated with the remote device specified by the Connection_Handle, and once authenticated will encrypt the connection. The parameters are as defined in [Vol 3] Part H, Section 2.4.4.

If the connection is already encrypted then the Controller shall pause connection encryption before attempting to authenticate the given encryption key, and then re-encrypt the connection. While encryption is paused no user data shall be transmitted.

On an authentication failure, the connection shall be automatically disconnected by the Link Layer. If this command succeeds, then the connection shall be encrypted.

This command shall only be used when the local device's role is Central.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle identifies an ACL with a created CIS. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Random_Number:* *Size: 8 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX | 64 bit random number. |

*Encrypted_Diversifier:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | 16 bit encrypted diversifier. |

*Long_Term_Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128 bit long term key. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Enable_Encryption command it shall send the HCI_Command_Status event to the Host. If the connection is not encrypted when this command is issued, an HCI_Encryption_Change event shall occur when encryption has been started for the connection. If the connection is encrypted when this command is issued, an HCI_Encryption_Key_Refresh_Complete event shall occur when encryption has been resumed.

### 7.8.25 LE Long Term Key Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Long_Term_Key_Request_Reply | 0x001A | Connection_Handle,<br>Long_Term_Key | Status,<br>Connection_Handle |

**Description:**

This command is used to reply to an HCI_LE_Long_Term_Key_Request event from the Controller, and specifies the Long_Term_Key parameter that shall be used for this Connection_Handle. The Long_Term_Key is used as defined in [Vol 6] Part B, Section 5.1.3.

This command shall only be used when the local device's role is Peripheral.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Long_Term_Key:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128 bit long term key for the given connection. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Long_Term_Key_Request_Reply command succeeded. |
| 0x01 to 0xFF | HCI_LE_Long_Term_Key_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Long_Term_Key_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.26 LE Long Term Key Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Long_Term_Key_Request_-Negative_Reply | 0x001B | Connection_Handle | Status,<br>Connection_Handle |

**Description:**

This command is used to reply to an HCI_LE_Long_Term_Key_Request event from the Controller if the Host cannot provide a Long Term Key for this Connection_Handle.

This command shall only be used when the local device's role is Peripheral.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Long_Term_Key_Request_Negative_Reply command succeeded. |
| 0x01 to 0xFF | HCI_LE_Long_Term_Key_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Long_Term_Key_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.27 LE Read Supported States command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Supported_States | 0x001C | *none* | Status,<br>LE_States |

**Description:**

This command reads the states, roles, and combinations of states and roles that the Link Layer supports. See [Vol 6] Part B, Section 1.1.1.

The LE_States parameter is a bit field. If a bit is set to 1 then this state, role, or combination is supported by the Controller; each combination consists of the states indicated by 'X' in the associated row of the table and omits those states with an empty cell in that row. Multiple bits in LE_States may be set to 1 to indicate support for multiple combinations.

Note: This command only provides information about the supported states and roles that can be used with legacy advertising. It does not provide information about those that can only be used with the extended advertising commands (see Section 3.1.1).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Supported_States command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Supported_States command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*LE_States:* *Size: 8 octets*

| | SUPPORTED STATES & ROLES | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Bit** | **Advertising State Scannable Undirected** | **Advertising State Scannable Undirected Connectable and** | **Advertising State Non-Scannable Undirected** | **Advertising State High Duty Cycle Connectable Directed** | **Advertising State Low Duty Cycle Connectable Directed** | **Scanning State Active** | **Scanning State Passive** | **Initiating State** | **Connection State (Central Role)** | **Connection State (Peripheral Role)** |
| 0 | | | X | | | | | | | |
| 1 | X | | | | | | | | | |
| 2 | | X | | | | | | | | |
| 3 | | | | X | | | | | | |
| 4 | | | | | | | X | | | |
| 5 | | | | | | X | | | | |
| 6 | | | | | | | | X | | |
| 7 | | | | | | | | | | X |
| 8 | | | X | | | | X | | | |
| 9 | X | | | | | | X | | | |
| 10 | | X | | | | | X | | | |
| 11 | | | | X | | | X | | | |
| 12 | | | X | | | X | | | | |
| 13 | X | | | | | X | | | | |
| 14 | | X | | | | X | | | | |
| 15 | | | | X | | X | | | | |
| 16 | | | X | | | | | X | | |
| 17 | X | | | | | | | X | | |
| 18 | | | X | | | | | | X | |
| 19 | X | | | | | | | | X | |
| 20 | | | X | | | | | | | X |
| 21 | X | | | | | | | | | X |

| | SUPPORTED STATES & ROLES | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Bit** | **Scannable Undirected Advertising State** | **Connectable and Scannable Undirected Advertising State** | **Non-connectable and Non-Scannable Undirected Advertising State** | **High Duty Cycle Connectable Directed Advertising State** | **Low Duty Cycle Connectable Directed Advertising State** | **Active Scanning State** | **Passive Scanning State** | **Initiating State** | **Connection State (Central Role)** | **Connection State (Peripheral Role)** |
| 22 | | | | | | | X | X | | |
| 23 | | | | | | X | | X | | |
| 24 | | | | | | | X | | X | |
| 25 | | | | | | X | | | X | |
| 26 | | | | | | | X | | | X |
| 27 | | | | | | X | | | | X |
| 28 | | | | | | | | X | X | |
| 29 | | | | | X | | | | | |
| 30 | | | | | X | | X | | | |
| 31 | | | | | X | X | | | | |
| 32 | | X | | | | | | X | | |
| 33 | | | | X | | | | X | | |
| 34 | | | | | X | | | X | | |
| 35 | | X | | | | | | | X | |
| 36 | | | | X | | | | | X | |
| 37 | | | | | X | | | | X | |
| 38 | | X | | | | | | | | X |
| 39 | | | | X | | | | | | X |
| 40 | | | | | X | | | | | X |
| 41 | | | | | | | | X | | X |

All bits not listed in this table, and the value with all bits set to 0, are reserved for future use.

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Supported_States command has completed, an HCI_Command_Complete event will be generated.

### 7.8.28 LE Receiver Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Receiver_Test [v3] | 0x004F | RX_Channel,<br>PHY,<br>Modulation_Index,<br>Expected_CTE_Length,<br>Expected_CTE_Type,<br>Slot_Durations,<br>Switching_Pattern_Length,<br>Antenna_IDs[i] | Status |
| HCI_LE_Receiver_Test [v2] | 0x0033 | RX_Channel,<br>PHY,<br>Modulation_Index | Status |
| HCI_LE_Receiver_Test [v1] | 0x001D | RX_Channel | Status |

**Description:**

This command is used to start a test where the IUT receives test reference packets at a fixed interval. The Lower Tester generates the test reference packets.

The RX_Channel and PHY parameters specify the RF channel and PHY to be used by the receiver.

The Modulation_Index parameter specifies whether or not the Controller should assume the receiver has a stable modulation index.

The Expected_CTE_Length and Expected_CTE_Type parameters specify the expected length and type of the Constant Tone Extensions in received test reference packets. When receiving on a PHY that allows Constant Tone Extensions, if the Constant Tone Extension in a received test reference packet does not match both of these, the IUT shall discard that packet.

Slot_Durations, Switching_Pattern_Length, and Antenna_IDs[i] are only used when expecting an AoA Constant Tone Extension and shall be ignored when expecting an AoD Constant Tone Extension.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| PHY | 0x01 |
| Modulation_Index | 0x00 |
| Expected_CTE_Length | 0x00 |
| Expected_CTE_Type | *any valid value* |
| Slot_Durations | *any valid value* |
| Switching_Pattern_Length | *any valid value* |
| Antenna_IDs[i] | *any valid value* |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | PHY contains a bit for a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Expected_CTE_Length is non-zero and PHY contains a bit for a PHY that does not support CTEs. | *Command Disallowed* (0x0C) |
| MC | Slot_Durations is set to 0x01 and the Controller does not support 1 µs switching and sampling. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Antenna_IDs[i] contains a bit for an antenna that the Controller can determine is not in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*RX_Channel:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | N = (F-2402) ÷ 2<br>Range: 0x00 to 0x27.<br>Frequency Range: 2402 MHz to 2480 MHz |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Receiver set to use the LE 1M PHY |
| 0x02 | Receiver set to use the LE 2M PHY |
| 0x03 | Receiver set to use the LE Coded PHY |
| All other values | Reserved for future use |

*Modulation_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Assume transmitter will have a standard modulation index |
| 0x01 | Assume transmitter will have a stable modulation index |
| All other values | Reserved for future use |

*Expected_CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Constant Tone Extension expected (default) |
| 0x02 to 0x14 | Expected length of the Constant Tone Extension in 8 µs units |
| All other values | Reserved for future use |

*Expected_CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Expect AoA Constant Tone Extension |
| 0x01 | Expect AoD Constant Tone Extension with 1 µs slots |
| 0x02 | Expect AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Switching and sampling slots are 1 µs each |
| 0x02 | Switching and sampling slots are 2 µs each |
| All other values | Reserved for future use |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Receiver_Test command succeeded. |
| 0x01 to 0xFF | HCI_LE_Receiver_Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Receiver_Test command has completed, an HCI_Command_-Complete event shall be generated.

If the Expected_CTE_Length parameter is not set to zero, then HCI_LE_-Connectionless_IQ_Report events may be generated by the Controller.

### 7.8.29 LE Transmitter Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Transmitter_Test [v4] | 0x007B | TX_Channel,<br>Test_Data_Length,<br>Packet_Payload,<br>PHY,<br>CTE_Length,<br>CTE_Type,<br>Switching_Pattern_Length,<br>Antenna_IDs[i],<br>TX_Power_Level | Status |
| HCI_LE_Transmitter_Test [v3] | 0x0050 | TX_Channel,<br>Test_Data_Length,<br>Packet_Payload,<br>PHY,<br>CTE_Length,<br>CTE_Type,<br>Switching_Pattern_Length,<br>Antenna_IDs[i] | Status |
| HCI_LE_Transmitter_Test [v2] | 0x0034 | TX_Channel,<br>Test_Data_Length,<br>Packet_Payload,<br>PHY | Status |
| HCI_LE_Transmitter_Test [v1] | 0x001E | TX_Channel,<br>Test_Data_Length,<br>Packet_Payload | Status |

The order of the command parameters in an HCI command packet is:

TX_Channel
Test_Data_Length
Packet_Payload
PHY
CTE_Length
CTE_Type
Switching_Pattern_Length
Antenna_IDs[0]
…

Antenna_IDs[n]
TX_Power_Level

**Description:**

This command is used to start a test where the IUT generates test reference packets at a fixed interval. The Controller shall transmit at the power level indicated by the TX_Power_Level parameter.

The TX_Channel and PHY parameters specify the RF channel and PHY to be used by the transmitter.

The Test_Data_Length and Packet_Payload parameters specify the length and contents of the Payload of the test reference packets. An LE Controller supporting the HCI_LE_Transmitter_Test command shall support Packet_Payload values 0x00, 0x01 and 0x02. An LE Controller supporting the LE Coded PHY shall also support Packet_Payload value 0x04. An LE Controller may support other values of Packet_Payload.

The CTE_Length and CTE_Type parameters specify the length and type of the Constant Tone Extension in the test reference packets.

The Switching_Pattern_Length and Antenna_IDs[i] parameters specify the antenna switching pattern. They are only used when transmitting an AoD Constant Tone Extension and shall be ignored when transmitting an AoA Constant Tone Extension.

The TX_Power_Level parameter specifies the transmit power level to be used by the transmitter. If the parameter is set to a value other than 0x7E or 0x7F, then the Controller shall make the requested change or shall make the nearest change that it is capable of doing.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| PHY | 0x01 |
| CTE_Length | 0x00 |
| CTE_Type | *any valid value* |
| Switching_Pattern_Length | *any valid value* |
| Antenna_IDs[i] | *any valid value* |
| TX_Power_Level | 0x7F |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | PHY contains a bit for a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | CTE_Length is non-zero and PHY contains a bit for a PHY that does not support CTEs. | *Command Disallowed* (0x0C) |
| MC | CTE_Type is set to 0x01 and the Controller does not support 1 µs switching and sampling. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Antenna_IDs[i] contains a bit for an antenna that the Controller can determine is not in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*TX_Channel:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | N = (F-2402) ÷ 2<br>Range: 0x00 to 0x27<br>Frequency Range: 2402 MHz to 2480 MHz |

*Test_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xFF | Length in bytes of payload data in each packet |

*Packet_Payload:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | PRBS9 sequence '11111111100000111101...' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5 |
| 0x01 | Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x02 | Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x03 | PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x04 | Repeated '11111111' (in transmission order) sequence |

| Value | Parameter Description |
|---|---|
| 0x05 | Repeated '00000000' (in transmission order) sequence |
| 0x06 | Repeated '00001111' (in transmission order) sequence |
| 0x07 | Repeated '01010101' (in transmission order) sequence |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Transmitter set to use the LE 1M PHY |
| 0x02 | Transmitter set to use the LE 2M PHY |
| 0x03 | Transmitter set to use the LE Coded PHY with S=8 data coding |
| 0x04 | Transmitter set to use the LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

*CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Do not transmit a Constant Tone Extension |
| 0x02 to 0x14 | Length of the Constant Tone Extension in 8 µs units |
| All other values | Reserved for future use |

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern |

*TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Set transmitter to the specified or the nearest transmit power level.<br>Range: -127 to +20<br>Units: dBm |
| 0x7E | Set transmitter to minimum transmit power level |
| 0x7F | Set transmitter to maximum transmit power level |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_-Transmitter_-Test command succeeded. |
| 0x01 to 0xFF | HCI_LE_-Transmitter_-Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Transmitter_Test command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.30 LE Test End command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Test_End | 0x001F | *none* | Status,<br>Num_Packets |

**Description:**

This command is used to stop any test which is in progress. The Num_Packets for a transmitter test shall be reported as 0x0000. The Num_Packets is an unsigned number and contains the number of received packets.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Test_End command succeeded. |
| 0x01 to 0xFF | HCI_LE_Test_End command failed See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Packets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of packets received |

**Event(s) generated (unless masked away):**

When the HCI_LE_Test_End command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.31 LE Remote Connection Parameter Request Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remote_Connection_-Parameter_Request_Reply | 0x0020 | Connection_Handle,<br>Interval_Min,<br>Interval_Max,<br>Max_Latency,<br>Timeout,<br>Min_CE_Length,<br>Max_CE_Length | Status,<br>Connection_Handle |

**Description:**

Both the Central's Host and the Peripheral's Host use this command to reply to the HCI_LE_Remote_Connection_Parameter_Request event. This indicates that the Host has accepted the remote device's request to change connection parameters.

The Interval_Min and Interval_Max parameters define the minimum and maximum allowed connection interval.

The Max_Latency parameter shall define the maximum allowed Peripheral latency for the connection.

The Timeout parameter shall define the link supervision timeout for the LE link.

The Min_CE_Length and Max_CE_Length are information parameters providing the Controller with a hint about the expected minimum and maximum length of the connection events.

The actual parameter values selected by the Link Layer may be different from the parameter values provided by the Host through this command.

If this command completes successfully and the connection interval has changed, then the subrating factor shall be set to 1 and the continuation number to 0. In this case, Max_Latency must be interpreted in underlying connection events. Otherwise the subrating factor and continuation number shall be unchanged and Max_Latency must be interpreted in subrated events.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| R | Interval_Min is greater than Interval_Max. | *Invalid HCI Command Parameters* (0x12) |
| R | Max_CE_Length is less than Min_CE_Length. | *Invalid HCI Command Parameters* (0x12) |
| R | The Timeout in milliseconds is less than or equal to (1 + Max_Latency) × Subrate_Factor × Interval_Max × 2, where Interval_Max is given in milliseconds and Subrate_Factor is the current subrate factor. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum value of the connection interval.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |

*Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum value of the connection interval.<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum allowed Peripheral latency for the connection specified as the number of subrated connection events.<br>Range: 0x0000 to 0x01F3 (499) |

*Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for the connection.<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Min_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Information parameter about the minimum length of connection event needed for this LE connection.<br>Time = N × 0.625 ms<br>Time Range: 0 ms to 40.9 s |

*Max_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Information parameter about the maximum length of connection event needed for this LE connection.<br>Time = N × 0.625 ms<br>Time Range: 0 ms to 40.9 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remote_Connection_Parameter_Request_Reply command succeeded. |
| 0x01 to 0xFF | HCI_LE_Remote_Connection_Parameter_Request_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remote_Connection_Parameter_Request_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.32 LE Remote Connection Parameter Request Negative Reply command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remote_Connection_-Parameter_Request_Negative_Reply | 0x0021 | Connection_Handle,<br>Reason | Status,<br>Connection_Handle |

**Description:**

Both the Central's Host and the Peripheral's Host use this command to reply to the HCI_LE_Remote_Connection_Parameter_Request event. This indicates that the Host has rejected the remote device's request to change connection parameters. The reason for the rejection is given in the Reason parameter.

Instead of issuing this command, the Host should try to provide alternative connection parameters to the Link Layer via the HCI_LE_Remote_Connection_Parameter_Request_Reply command (Section 7.8.31).

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x3B | Reason that the connection parameter request was rejected: *Unacceptable Connection Parameters*. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply command succeeded. |
| 0x01 to 0xFF | HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.33 LE Set Data Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Data_Length | 0x0022 | Connection_Handle,<br>TX_Octets,<br>TX_Time | Status,<br>Connection_Handle |

**Description:**

This command allows the Host to suggest the maximum transmission payload size and maximum packet transmission time (*connMaxTxOctets* and *connMaxTxTime* - see [Vol 6] Part B, Section 4.5.10) to be used for LL Data PDUs on a given connection. The Controller may use smaller or larger values based on local information.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*TX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Preferred maximum number of payload octets that the local Controller should include in a single LL Data PDU on this connection.<br>Range 0x001B to 0x00FB |

*TX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Preferred maximum number of microseconds that the local Controller should use to transmit a single Link Layer packet containing an LL Data PDU on this connection.<br>Range 0x0148 to 0x4290 |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Data_Length command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Data_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Data_Length command has completed, an HCI_Command_Complete event shall be generated.

If the command causes the maximum transmission packet size or maximum packet transmission time to change, an HCI_LE_Data_Length_Change event shall be generated.

### 7.8.34 LE Read Suggested Default Data Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Suggested_-Default_Data_Length | 0x0023 | *none* | Status,<br>Suggested_Max_TX_Octets,<br>Suggested_Max_TX_Time |

**Description:**

This command allows the Host to read the Host's suggested values (Suggested_Max_TX_Octets and Suggested_Max_TX_Time) for the Controller's maximum transmitted number of payload octets and maximum packet transmission time for packets containing LL Data PDUs to be used for new connections (see [Vol 6] Part B, Section 4.5.10).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Suggested_Default_Data_Length command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Suggested_Default_Data_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Suggested_Max_TX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Host's suggested value for the Controller's maximum transmitted number of payload octets in LL Data PDUs to be used for new connections.<br>Range 0x001B to 0x00FB<br>Default: 0x001B |

*Suggested_Max_TX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Host's suggested value for the Controller's maximum packet transmission time for packets containing LL Data PDUs to be used for new connections.<br>Range 0x0148 to 0x4290<br>Default: 0x0148 |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Suggested_Default_Data_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.35 LE Write Suggested Default Data Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Write_Suggested_Default_-Data_Length | 0x0024 | Suggested_Max_TX_Octets,<br>Suggested_Max_TX_Time | Status |

**Description:**

This command allows the Host to specify its suggested values for the Controller's maximum transmission number of payload octets and maximum packet transmission time for packets containing LL Data PDUs to be used for new connections.
The Controller may use smaller or larger values for *connInitialMaxTxOctets* and *connInitialMaxTxTime* based on local information. (See [Vol 6] Part B, Section 4.5.10).

**Command parameters:**

*Suggested_Max_TX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Host's suggested value for the Controller's maximum transmitted number of payload octets in LL Data PDUs to be used for new connections.<br>Range 0x001B to 0x00FB |

*Suggested_Max_TX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Host's suggested value for the Controller's maximum packet transmission time for packets containing LL Data PDUs to be used for new connections.<br>Range 0x0148 to 0x4290 |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Write_Suggested_Default_Data_Length command succeeded. |
| 0x01 to 0xFF | HCI_LE_Write_Suggested_Default_Data_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Write_Suggested_Default_Data_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.36 LE Read Local P-256 Public Key command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Local_P-256_Public_Key | 0x0025 | *none* | *none* |

**Description:**

This command is used to return the local P-256 public key from the Controller. The Controller shall generate a new P-256 public/private key pair upon receipt of this command.

The keys returned via this command shall not be used when Secure Connections is used over the BR/EDR transport.

**Command parameters:**

None.

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Read_Local_P-256_Public_Key command, the Controller shall send the HCI_Command_Status event to the Host. When the local P-256 public key generation finishes, an HCI_LE_Read_Local_P-256_Public_Key_Complete event shall be generated.

### 7.8.37 LE Generate DHKey command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Generate_DHKey [v2] | 0x005E | Key_X_Coordinate,<br>Key_Y_Coordinate,<br>Key_Type | *none* |
| HCI_LE_Generate_DHKey [v1] | 0x0026 | Key_X_Coordinate,<br>Key_Y_Coordinate | *none* |

**Description:**

This command is used to initiate generation of a Diffie-Hellman key in the Controller for use over the LE transport. This command takes the remote P-256 public key specified in the Key_X_Coordinate and Key_Y_Coordinate parameters as input. The Diffie-Hellman key generation uses the private key generated by the HCI_LE_Read_Local_P-256_Public_Key command or the private debug key (see [Vol 3] Part H, Section 2.3.5.6.1).

The Diffie-Hellman key returned via this command shall not be generated using any keys used for Secure Connections over the BR/EDR transport.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| Key_Type | 0x00 |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| M | The remote P-256 public key is invalid (see [Vol 3] Part H, Section 2.3.5.6.1). | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Key_X_Coordinate:* *Size: 32 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX | The remote P-256 public key X coordinate |

*Key_Y_Coordinate:* *Size: 32 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX<br>XXXXXXXXXXXXXXXX | The remote P-256 public key Y coordinate |

*Key_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use the generated private key |
| 0x01 | Use the debug private key |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Generate_DHKey command, the Controller shall send the HCI_Command_Status event to the Host. When the DHKey generation finishes, an HCI_LE_Generate_DHKey_Complete event shall be generated.

### 7.8.38 LE Add Device To Resolving List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Add_Device_To_-Resolving_List | 0x0027 | Peer_Identity_Address_Type,<br>Peer_Identity_Address,<br>Peer_IRK,<br>Local_IRK | Status |

**Description:**

This command is used to add one device to the resolving list used to generate and resolve Resolvable Private Addresses in the Controller.

This command shall not be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- an HCI_LE_Create_Connection, HCI_LE_Extended_Create_Connection, or HCI_LE_Periodic_Advertising_Create_Sync command is pending.

This command may be used at any time when address resolution is disabled in the Controller.

The added device shall be set to Network Privacy mode.

If an entry already exists in the resolving list with the same four parameter values, the Controller shall not add the device to the resolving list again.

Note: In this situation the command can either succeed or fail; the only difference is the returned status code.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| RC | Peer_Identity_Address_Type is 0x01 and Peer_Identity_Address is a non-static address. | *Invalid HCI Command Parameters* (0x12) |
| MC | There is no space available to add a device to the list. | *Memory Capacity Exceeded* (0x07) |

| Type | Condition | Error code |
|---|---|---|
| RC | An entry with the same Peer_Identity_Address and Peer_Identity_Address_Type, or with the same non-zero Peer_IRK, already exists in the resolving list. | *Invalid HCI Command Parameters* (0x12) |
| O | An entry with the same four parameter values already exists in the resolving list. | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*Peer_Identity_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Identity Address |
| 0x01 | Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Identity_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public or Random (static) Identity Address of the peer device |

*Peer_IRK:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX | IRK of the peer device |

*Local_IRK:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX | IRK of the local device |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Add_Device_To_Resolving_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Add_Device_To_Resolving_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Add_Device_To_Resolving_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.39 LE Remove Device From Resolving List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_Device_From_-Resolving_List | 0x0028 | Peer_Identity_Address_Type,<br>Peer_Identity_Address | Status |

**Description:**

This command is used to remove one device from the resolving list used to resolve Resolvable Private Addresses in the Controller.

This command shall not be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- an HCI_LE_Create_Connection, HCI_LE_Extended_Create_Connection, or HCI_LE_Periodic_Advertising_Create_Sync command is pending.

This command may be used at any time when address resolution is disabled in the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The device is not in the resolving list. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Peer_Identity_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Identity Address |
| 0x01 | Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Identity_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public or Random (static) Identity Address of the peer device |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remove_Device_From_Resolving_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Remove_Device_From_Resolving_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_Device_From_Resolving_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.40 LE Clear Resolving List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Clear_Resolving_List | 0x0029 | *none* | Status |

**Description:**

This command is used to remove all devices from the resolving list used to resolve Resolvable Private Addresses in the Controller.

This command shall not be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- an HCI_LE_Create_Connection, HCI_LE_Extended_Create_Connection, or HCI_LE_Periodic_Advertising_Create_Sync command is pending.

This command may be used at any time when address resolution is disabled in the Controller.

**Command parameters:**

None

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Clear_Resolving_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Clear_Resolving_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Clear_Resolving_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.41 LE Read Resolving List Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Resolving_List_Size | 0x002A | *none* | Status,<br>Resolving_List_Size |

**Description:**

This command is used to read the total number of entries in the resolving list that can be stored in the Controller.

Note: The number of entries that can be stored is not fixed and the Controller can change it at any time (e.g. because the memory used to store the list can also be used for other purposes).

**Command parameters:**

None

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Resolving_List_Size command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Resolving_List_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Resolving_List_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of entries in the resolving list |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Resolving_List_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.42 LE Read Peer Resolvable Address command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Peer_-Resolvable_Address | 0x002B | Peer_Identity_Address_Type,<br>Peer_Identity_Address | Status,<br>Peer_Resolvable_Address |

**Description:**

This command is used to get the current peer Resolvable Private Address being used for the corresponding peer Public and Random (static) Identity Address. The peer's resolvable address being used may change after the command is called.

This command may be used at any time.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The resolving list does not contain the Peer Identity Address or a Resolvable Private Address associated with the Peer Identity Address. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Peer_Identity_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Identity Address |
| 0x01 | Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Identity_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public or Random (static) Identity Address of the peer device |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Peer_Resolvable_Address command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Peer_Resolvable_Address command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Peer_Resolvable_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Resolvable Private Address being used by the peer device |

### Event(s) generated (unless masked away):

When the HCI_LE_Read_Peer_Resolvable_Address command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.43 LE Read Local Resolvable Address command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Local_-Resolvable_Address | 0x002C | Peer_Identity_Address_Type,<br>Peer_Identity_Address | Status,<br>Local_Resolvable_Address |

**Description:**

This command is used to get the current local Resolvable Private Address being used for the corresponding peer Identity Address. The local resolvable address being used may change after the command is called.

This command may be used at any time.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The resolving list does not contain the Peer Identity Address or a Resolvable Private Address associated with the Peer Identity Address. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Peer_Identity_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Identity Address |
| 0x01 | Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Identity_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Identity Address or Random (static) Identity Address of the peer device, 48 bit value. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Local_Resolvable_Address command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Local_Resolvable_Address command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Local_Resolvable_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Resolvable Private Address being used by the local device |

### Event(s) generated (unless masked away):

When the HCI_LE_Read_Local_Resolvable_Address command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.44 LE Set Address Resolution Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Address_Resolution_-Enable | 0x002D | Address_Resolution_Enable | Status |

**Description:**

This command is used to enable resolution of Resolvable Private Addresses in the Controller. This causes the Controller to use the resolving list whenever the Controller receives a local or peer Resolvable Private Address.

This command shall not be used when:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- an HCI_LE_Create_Connection, HCI_LE_Extended_Create_Connection, or HCI_LE_Periodic_Advertising_Create_Sync command is pending.

Enabling address resolution when it is already enabled, or disabling it when it is already disabled, has no effect.

The requirements in [Vol 6] Part B, Section 6 related to the generation of Resolvable Private Addresses and the privacy of the device are independent of this command.

**Command parameters:**

*Address_Resolution_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Address Resolution in Controller disabled (default) |
| 0x01 | Address Resolution in Controller enabled |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Address_Resolution_Enable command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Address_Resolution_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Address_Resolution_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.45 LE Set Resolvable Private Address Timeout command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Resolvable_Private_Address_Timeout [v2] | 0x009E | RPA_Timeout_Min, RPA_Timeout_Max | Status |
| HCI_LE_Set_Resolvable_Private_Address_Timeout [v1] | 0x002E | RPA_Timeout | Status |

**Description:**

Version [v1] of this command sets the length of time the Controller uses a Resolvable Private Address before a new Resolvable Private Address is generated and starts being used.

Version [v2] of this command sets the range of time the Controller uses a Resolvable Private Address before a new Resolvable Private Address is generated and starts being used.

This timeout applies to all resolvable private addresses generated by the Controller.

The RPA_Timeout parameter specifies the time after which a new Resolvable Private Address shall start being used.

The RPA_Timeout_Min parameter specifies the minimum time after which a new Resolvable Private Address shall start being used.

The RPA_Timeout_Max parameter specifies the maximum time after which a new Resolvable Private Address shall start being used.

When the Controller supports the HCI_LE_Set_Resolvable_Private_Address_Timeout [v2] command and needs to set a new timeout (e.g., when the RPA is set for the first time, or when the current timeout expires), the new timeout shall be a random value between RPA_Timeout_Min and RPA_Timeout_Max generated so as to meet the requirements for random number generation defined in [Vol 2] Part H, Section 2.

If RPA_Timeout_Min is greater than RPA_Timeout_Max, then the Controller shall return the error code *Invalid HCI Command Parameters* (0x12).

If the Controller supports version [v2] of this command, then the default behavior shall be to use the range 8 to 15 minutes. Otherwise the default timeout shall be 15 minutes.

### Command parameters:

*RPA_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | RPA timeout, in seconds<br>Range: 0x0001 to 0x0E10<br>Time range: 1 s to 1 hour |

*RPA_Timeout_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum RPA timeout, in seconds<br>Range: 0x0001 to 0x0E10<br>Time range: 1 s to 1 hour |

*RPA_Timeout_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum RPA timeout, in seconds<br>Range: 0x0001 to 0x0E10<br>Time range: 1 s to 1 hour |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Resolvable_Private_Address_Timeout command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Resolvable_Private_Address_Timeout command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Resolvable_Private_Address_Timeout command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.46 LE Read Maximum Data Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Maximum_Data_-Length | 0x002F | *none* | Status,<br>Supported_Max_TX_Octets,<br>Supported_Max_TX_Time,<br>Supported_Max_RX_Octets,<br>Supported_Max_RX_Time |

**Description:**

This command allows the Host to read the Controller's maximum supported payload octets and packet duration times for transmission and reception (Supported_Max_TX_Octets, Supported_Max_TX_Time, Supported_Max_RX_Octets, and Supported_Max_RX_Time, see [Vol 6] Part B, Section 4.5.10).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Maximum_Data_Length command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Maximum_Data_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Supported_Max_TX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum number of payload octets that the local Controller supports for transmission of a single Link Layer packet on an ACL connection.<br>Range 0x001B to 0x00FB |

*Supported_Max_TX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum time, in microseconds, that the local Controller supports for transmission of a single Link Layer packet on an ACL connection.<br>Range 0x0148 to 0x4290 |

*Supported_Max_RX_Octets:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum number of payload octets that the local Controller supports for reception of a single Link Layer packet on an ACL connection.<br>Range 0x001B to 0x00FB |

*Supported_Max_RX_Time:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum time, in microseconds, that the local Controller supports for reception of a single Link Layer packet on an ACL connection.<br>Range 0x0148 to 0x4290 |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Maximum_Data_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.47 LE Read PHY command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_PHY | 0x0030 | Connection_Handle | Status,<br>Connection_Handle,<br>TX_PHY,<br>RX_PHY |

**Description:**

This command is used to read the current transmitter PHY and receiver PHY on the connection identified by the Connection_Handle.

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range:0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_PHY command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_PHY command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range:0x0000 to 0x0EFF |

*TX_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The transmitter PHY for the connection is LE 1M |
| 0x02 | The transmitter PHY for the connection is LE 2M |

| Value | Parameter Description |
|---|---|
| 0x03 | The transmitter PHY for the connection is LE Coded |
| All other values | Reserved for future use |

*RX_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | The receiver PHY for the connection is LE 1M |
| 0x02 | The receiver PHY for the connection is LE 2M |
| 0x03 | The receiver PHY for the connection is LE Coded |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_PHY command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.48 LE Set Default PHY command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Default_PHY | 0x0031 | All_PHYs,<br>TX_PHYs,<br>RX_PHYs | Status |

**Description:**

This command allows the Host to specify its preferred values for the transmitter PHY and receiver PHY to be used for all subsequent connections over the LE transport.

The All_PHYs parameter is a bit field that allows the Host to specify, for each direction, whether it has no preference among the PHYs that the Controller supports in a given direction or whether it has specified particular PHYs that it prefers in the TX_PHYs or RX_PHYs parameter.

The TX_PHYs parameter is a bit field that indicates the transmitter PHYs that the Host prefers the Controller to use. If the All_PHYs parameter specifies that the Host has no preference, the TX_PHYs parameter shall be ignored; otherwise at least one bit shall be set to 1.

The RX_PHYs parameter is a bit field that indicates the receiver PHYs that the Host prefers the Controller to use. If the All_PHYs parameter specifies that the Host has no preference, the RX_PHYs parameter shall be ignored; otherwise at least one bit shall be set to 1.

If the Host sets, in the TX_PHYs or RX_PHYs parameter, a bit for a PHY that the Controller does not support, including a bit that is reserved for future use, the Controller shall return the error code *Unsupported Feature or Parameter Value* (0x11).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller does not support asymmetric connections (see [Vol 6] Part B, Section 4.6.9.1), All_PHYs is 0x00, and TX_PHYs is not the same as RX_PHYs. | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*All_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host has no preference among the transmitter PHYs supported by the Controller |
| 1 | The Host has no preference among the receiver PHYs supported by the Controller |
| All other bits | Reserved for future use |

*TX_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host prefers to use the LE 1M transmitter PHY (possibly among others) |
| 1 | The Host prefers to use the LE 2M transmitter PHY (possibly among others) |
| 2 | The Host prefers to use the LE Coded transmitter PHY (possibly among others) |
| All other bits | Reserved for future use |

*RX_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host prefers to use the LE 1M receiver PHY (possibly among others) |
| 1 | The Host prefers to use the LE 2M receiver PHY (possibly among others) |
| 2 | The Host prefers to use the LE Coded receiver PHY (possibly among others) |
| All other bits | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Default_PHY command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Default_PHY command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Default_PHY command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.49 LE Set PHY command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_PHY | 0x0032 | Connection_Handle,<br>All_PHYs,<br>TX_PHYs,<br>RX_PHYs,<br>PHY_Options | *none* |

**Description:**

This command is used to set the PHY preferences for the connection identified by the Connection_Handle. The Controller might not be able to make the change (e.g. because the peer does not support the requested PHY) or may decide that the current choice of PHY is preferable (e.g., because it could not schedule other activities if the requested PHY was used or because it requires a PHY that supports Constant Tone Extensions).

The All_PHYs parameter is a bit field that allows the Host to specify, for each direction, whether it has no preference among the PHYs that the Controller supports in a given direction or whether it has specified particular PHYs that it prefers in the TX_PHYs or RX_PHYs parameter.

The TX_PHYs parameter is a bit field that indicates the transmitter PHYs that the Host prefers the Controller to use. If the All_PHYs parameter specifies that the Host has no preference, the TX_PHYs parameter shall be ignored; otherwise at least one bit shall be set to 1.

The RX_PHYs parameter is a bit field that indicates the receiver PHYs that the Host prefers the Controller to use. If the All_PHYs parameter specifies that the Host has no preference, the RX_PHYs parameter shall be ignored; otherwise at least one bit shall be set to 1.

The Controller shall request a change unless it determines that this is unnecessary or that the current PHY in both directions is preferable, in which case it may, but need not, request a change.

The PHY preferences provided by the HCI_LE_Set_PHY command override those provided via the HCI_LE_Set_Default_PHY command (Section 7.8.48) or any preferences previously set using the HCI_LE_Set_PHY command on the same connection.

The PHY_Options parameter is a bit field that allows the Host to specify options for PHYs. The default value for a new connection shall be all zero bits. The Controller may override any preferred coding for transmitting on the LE Coded PHY.

The Host may specify a preferred coding even if it prefers not to use the LE Coded transmitter PHY since the Controller may override the PHY preference.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | TX_PHYs or RX_PHYs contains a bit for a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller does not support asymmetric connections (see [Vol 6] Part B, Section 4.6.9.1), All_PHYs is 0x00, and TX_PHYs is not the same as RX_PHYs. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*All_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host has no preference among the transmitter PHYs supported by the Controller |
| 1 | The Host has no preference among the receiver PHYs supported by the Controller |
| All other bits | Reserved for future use |

*TX_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host prefers to use the LE 1M transmitter PHY (possibly among others) |
| 1 | The Host prefers to use the LE 2M transmitter PHY (possibly among others) |
| 2 | The Host prefers to use the LE Coded transmitter PHY (possibly among others) |
| All other bits | Reserved for future use |

*RX_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The Host prefers to use the LE 1M receiver PHY (possibly among others) |
| 1 | The Host prefers to use the LE 2M receiver PHY (possibly among others) |
| 2 | The Host prefers to use the LE Coded receiver PHY (possibly among others) |
| All other bits | Reserved for future use |

*PHY_Options:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 to 1 | 0 = the Host has no preferred coding when transmitting on the LE Coded PHY<br>1 = the Host prefers that S=2 coding be used when transmitting on the LE Coded PHY<br>2 = the Host prefers that S=8 coding be used when transmitting on the LE Coded PHY<br>3 = Reserved for future use |
| All other bits | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Set_PHY command, the Controller shall send the HCI_Command_Status event to the Host. The HCI_LE_PHY_Update_Complete event shall be generated either when one or both PHY changes or when the Controller determines that neither PHY will change immediately.

Note: If the peer negotiation resulted in no change to either PHY, this is not an error and the HCI_LE_PHY_Update_Complete event will contain a status indicating success.

### 7.8.50 [This section is no longer used]

See Section 7.8.28 for the LE Receiver Test command.

### 7.8.51 [This section is no longer used]

See Section 7.8.29 for the LE Transmitter Test command.

### 7.8.52 LE Set Advertising Set Random Address command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Advertising_Set_Random_-Address | 0x0035 | Advertising_Handle,<br>Random_Address | Status |

**Description:**

This command is used by the Host to set the random device address specified by the Random_Address parameter. This address is used in the Controller (see [Vol 6] Part B, Section 1.3.2 ) for the advertiser's address contained in the advertising PDUs for the advertising set specified by the Advertising_Handle parameter.

This command may be issued at any time after an advertising set identified by the Advertising_Handle parameter has been created using the HCI_LE_Set_Extended_Advertising_Parameters command (see Section 7.8.53).

If this command is used to change the address, the new random address shall take effect for advertising no later than the next successful HCI_LE_Set_Extended_Advertising_Enable command and for periodic advertising no later than the next successful HCI_LE_Periodic_Advertising_Enable command.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle identifies an advertising set that is enabled and is using connectable advertising. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Random_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Random Device Address as defined by [Vol 6] Part B, Section 1.3.2 |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Advertising_Set_Random_Address command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Advertising_Set_Random_Address command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Advertising_Set_Random_Address command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.53 LE Set Extended Advertising Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_-Advertising_Parameters [v2] | 0x007F | Advertising_Handle,<br>Advertising_Event_Properties,<br>Primary_Advertising_Interval_Min,<br>Primary_Advertising_Interval_Max,<br>Primary_Advertising_Channel_Map,<br>Own_Address_Type,<br>Peer_Address_Type,<br>Peer_Address,<br>Advertising_Filter_Policy,<br>Advertising_TX_Power,<br>Primary_Advertising_PHY,<br>Secondary_Advertising_Max_Skip,<br>Secondary_Advertising_PHY,<br>Advertising_SID,<br>Scan_Request_Notification_Enable<br>Primary_Advertising_PHY_Options,<br>Secondary_Advertising_PHY_Options | Status,<br>Selected_TX_Power |
| HCI_LE_Set_Extended_-Advertising_Parameters [v1] | 0x0036 | Advertising_Handle,<br>Advertising_Event_Properties,<br>Primary_Advertising_Interval_Min,<br>Primary_Advertising_Interval_Max,<br>Primary_Advertising_Channel_Map,<br>Own_Address_Type,<br>Peer_Address_Type,<br>Peer_Address,<br>Advertising_Filter_Policy,<br>Advertising_TX_Power,<br>Primary_Advertising_PHY,<br>Secondary_Advertising_Max_Skip,<br>Secondary_Advertising_PHY,<br>Advertising_SID,<br>Scan_Request_Notification_Enable | Status,<br>Selected_TX_Power |

**Description:**

This command is used by the Host to set the advertising parameters.

The Advertising_Handle parameter identifies the advertising set whose parameters are being configured.

The Advertising_Event_Properties parameter describes the type of advertising event that is being configured and its basic properties. The type shall be one supported by the Controller. In particular, the following restrictions apply to this parameter:

- If legacy advertising PDU types are being used, then the parameter value shall be one of those specified in Table 7.3. If the advertising set already contains data, the type shall be one that supports advertising data and the amount of data shall not exceed 31 octets.

| Event Type | PDU Type | Advertising Event Properties | Advertising Data |
|---|---|---|---|
| Connectable and scannable undirected | ADV_IND | 0b00000000_00010011 | Supported |
| Connectable directed (low duty cycle) | ADV_DIRECT_IND | 0b00000000_00010101 | Not allowed |
| Connectable directed (high duty cycle) | ADV_DIRECT_IND | 0b00000000_00011101 | Not allowed |
| Scannable undirected | ADV_SCAN_IND | 0b00000000_00010010 | Supported |
| Non-connectable and non-scannable undirected | ADV_NON-CONN_IND | 0b00000000_00010000 | Supported |

*Table 7.3: Advertising_Event_Properties values for legacy PDUs*

- If extended advertising PDU types are being used (bit 4 = 0), then the advertisement shall not be both connectable and scannable (bits 0 and 1 must not both be set to 1) and high duty cycle directed connectable advertising (≤ 3.75 ms advertising interval) shall not be used (bit 3 = 0).

The parameters beginning with “Secondary” are only valid when extended advertising PDU types are being used (bit 4 = 0).

The Own_Address_Type parameter shall be ignored for undirected anonymous advertising (bit 2 = 0 and bit 5 = 1).

If Directed advertising is selected, the Peer_Address_Type and Peer_Address shall be valid and the Advertising_Filter_Policy parameter shall be ignored.

The Primary_Advertising_Interval_Min parameter shall be less than or equal to the Primary_Advertising_Interval_Max parameter. The Primary_Advertising_Interval_Min

and Primary_Advertising_Interval_Max parameters should not be the same value so that the Controller can choose the best advertising interval given other activities.

For high duty cycle connectable directed advertising event type (ADV_DIRECT_IND), the Primary_Advertising_Interval_Min and Primary_Advertising_Interval_Max parameters are not used and shall be ignored.

The Primary_Advertising_Channel_Map is a bit field that indicates the advertising channel indices that shall be used when transmitting advertising packets. At least one channel bit shall be set in the Primary_Advertising_Channel_Map parameter.

The Own_Address_Type parameter specifies the type of address being used in the advertising packets. For random addresses, the address is specified by the HCI_LE_Set_Advertising_Set_Random_Address command.

If Own_Address_Type equals 0x02 or 0x03, the Peer_Address parameter contains the peer's Identity Address and the Peer_Address_Type parameter contains the peer's Identity Type (i.e., 0x00 or 0x01). These parameters are used to locate the corresponding local IRK in the resolving list; this IRK is used to generate their own address used in the advertisement.

The Advertising_TX_Power parameter indicates the maximum power level at which the advertising packets are to be transmitted on the advertising physical channels. The Controller shall choose a power level lower than or equal to the one specified by the Host.

The Primary_Advertising_PHY parameter indicates the PHY on which the advertising packets are transmitted on the primary advertising physical channel. If legacy advertising PDUs are being used, the Primary_Advertising_PHY shall indicate the LE 1M PHY. The Secondary_Advertising_PHY parameter indicates the PHY on which the advertising packets are be transmitted on the secondary advertising physical channel.

If the Primary_Advertising_PHY indicates the LE Coded PHY, then the Primary_Advertising_PHY_Options shall indicate the Host's preference or requirement concerning coding scheme. Otherwise, Primary_Advertising_PHY_Options shall be ignored. If the Secondary_Advertising_PHY indicates the LE Coded PHY, then the Secondary_Advertising_PHY_Options shall indicate the Host's preference or requirement concerning coding scheme (including for periodic advertising). Otherwise, Secondary_Advertising_PHY_Options shall be ignored.

The Secondary_Advertising_Max_Skip parameter is the maximum number of advertising events that can be skipped before the AUX_ADV_IND can be sent.

The Advertising_SID parameter specifies the value to be transmitted in the Advertising SID subfield of the ADI field of the Extended Header of those advertising physical

channel PDUs that have an ADI field. If the advertising set only uses PDUs that do not contain an ADI field, Advertising_SID shall be ignored.

The Scan_Request_Notification_Enable parameter indicates whether the Controller shall send notifications upon the receipt of a scan request PDU that is in response to an advertisement from the specified advertising set that contains its device address and is from a scanner that is allowed by the advertising filter policy.

The Controller shall set the Selected_TX_Power parameter to the transmit power that it will use for transmitting the advertising packets for the specified advertising set. The Controller shall only change this value if requested by the Host. If the radiated power level will vary between packets (e.g., because of frequency-dependent properties of the transmitter) then the value should be the best estimate of the maximum power used.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| Primary_Advertising_PHY_Options | 0x00 |
| Secondary_Advertising_PHY_Options | 0x00 |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Event_Properties does not describe an event type supported by the Controller or contains an invalid bit combination. | *Invalid HCI Command Parameters* (0x12) |
| MC | Advertising_Event_Properties specifies an event type that does not support advertising data and the advertising set already contains data. | *Invalid HCI Command Parameters* (0x12) |
| MC | The primary advertising interval range defined by Primary_Advertising_Interval_Min and Primary_Advertising_Interval_Max does not overlap the advertising interval range supported by the Controller. | *Unsupported Feature or Parameter Value* (0x11) |
| RC | Primary_Advertising_PHY specifies a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The advertising set enables Constant Tone Extensions but Secondary_Advertising_PHY specifies a PHY that does not allow Constant Tone Extensions. | *Command Disallowed* (0x0C) |

| Type | Condition | Error code |
|---|---|---|
| MC | Primary_Advertising_PHY_Options or Secondary_Advertising_PHY_Options is set to 0x03 or 0x04 and the controller supports the Advertising Coding Selection feature but is unable to provide all the required settings. | *Command Disallowed* (0x0C) |
| MC | Advertising is enabled for the specified advertising set. | *Command Disallowed* (0x0C) |
| MC | Periodic advertising is enabled for an advertising set that is connectable, scannable, legacy, or anonymous advertising. | *Invalid HCI Command Parameters* (0x12) |
| MC | Periodic advertising is enabled for the advertising set and Secondary_Advertising_PHY does not specify the PHY being used for that periodic advertising. | *Command Disallowed* (0x0C) |
| MC | Advertising_Handle is not an existing advertising set and the controller is not able to support a new advertising set. | *Memory Capacity Exceeded* (0x07) |
| MC | The advertising set already contains advertising data or scan response data, extended advertising is being used, and the length of the data is greater than the maximum that the Controller can transmit within the longest possible auxiliary advertising segment consistent with the parameters. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |
| MC | The Controller does not support the Advertising Coding Selection feature and Primary_Advertising_PHY_Options or Secondary_Advertising_PHY_Options are not zero. | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Advertising_Event_Properties:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Connectable advertising |
| 1 | Scannable advertising |
| 2 | Directed advertising |
| 3 | High Duty Cycle Directed Connectable advertising (≤ 3.75 ms Advertising Interval) |
| 4 | Use legacy advertising PDUs |

| Bit Number | Parameter Description |
|---|---|
| 5 | Omit advertiser's address from all PDUs ("anonymous advertising") |
| 6 | Include TxPower in the extended header of at least one advertising PDU |
| 7 | Use decision PDUs when advertising |
| 8 | Include AdvA in the extended header of all decision PDUs |
| 9 | Include ADI in the extended header of all decision PDUs |
| All other bits | Reserved for future use |

*Primary_Advertising_Interval_Min:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXXXX | Minimum advertising interval for undirected and low duty cycle directed advertising.<br>Range: 0x000020 to 0xFFFFFF<br>Time = N × 0.625 ms<br>Time Range: 20 ms to 10,485.759375 s |

*Primary_Advertising_Interval_Max:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXXXX | Maximum advertising interval for undirected and low duty cycle directed advertising.<br>Range: 0x000020 to 0xFFFFFF<br>Time = N × 0.625 ms<br>Time Range: 20 ms to 10,485.759375 s |

*Primary_Advertising_Channel_Map:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Channel 37 shall be used |
| 1 | Channel 38 shall be used |
| 2 | Channel 39 shall be used |
| All other bits | Reserved for future use |

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |

| Value | Parameter Description |
|---|---|
| 0x02 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the public address. |
| 0x03 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, use the random address from LE_Set_Advertising_Set_Random_Address. |
| All other values | Reserved for future use |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected. |

*Advertising_Filter_Policy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Process scan and connection requests from all devices (i.e., the Filter Accept List is not in use) |
| 0x01 | Process connection requests from all devices and scan requests only from devices that are in the Filter Accept List. |
| 0x02 | Process scan requests from all devices and connection requests only from devices that are in the Filter Accept List. |
| 0x03 | Process scan and connection requests only from devices in the Filter Accept List. |
| All other values | Reserved for future use |

*Advertising_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |
| 0x7F | Host has no preference |

*Primary_Advertising_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Primary advertisement PHY is LE 1M |
| 0x03 | Primary advertisement PHY is LE Coded |
| All other values | Reserved for future use |

*Secondary_Advertising_Max_Skip:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AUX_ADV_IND shall be sent prior to the next advertising event |
| 0x01 to 0xFF | Maximum advertising events the Controller can skip before sending the AUX_ADV_IND packets on the secondary advertising physical channel |

*Secondary_Advertising_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Secondary advertisement PHY is LE 1M |
| 0x02 | Secondary advertisement PHY is LE 2M |
| 0x03 | Secondary advertisement PHY is LE Coded |
| All other values | Reserved for future use |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Value of the Advertising SID subfield in the ADI field of the PDU |
| All other values | Reserved for future use |

*Scan_Request_Notification_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Scan request notifications disabled |
| 0x01 | Scan request notifications enabled |
| All other values | Reserved for future use |

*Primary_Advertising_PHY_Options:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Host has no preferred or required coding when transmitting on the LE Coded PHY |
| 0x01 | The Host prefers that S=2 coding be used when transmitting on the LE Coded PHY |

| Value | Parameter Description |
|---|---|
| 0x02 | The Host prefers that S=8 coding be used when transmitting on the LE Coded PHY |
| 0x03 | The Host requires that S=2 coding be used when transmitting on the LE Coded PHY |
| 0x04 | The Host requires that S=8 coding be used when transmitting on the LE Coded PHY |
| All other values | Reserved for future use |

*Secondary_Advertising_PHY_Options:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Host has no preferred or required coding when transmitting on the LE Coded PHY |
| 0x01 | The Host prefers that S=2 coding be used when transmitting on the LE Coded PHY |
| 0x02 | The Host prefers that S=8 coding be used when transmitting on the LE Coded PHY |
| 0x03 | The Host requires that S=2 coding be used when transmitting on the LE Coded PHY |
| 0x04 | The Host requires that S=8 coding be used when transmitting on the LE Coded PHY |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Advertising_Parameters command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Advertising_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Selected_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Extended_Advertising_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.54 LE Set Extended Advertising Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_Advertising_-Data | 0x0037 | Advertising_Handle,<br>Operation,<br>Fragment_Preference,<br>Advertising_Data_Length,<br>Advertising_Data | Status |

**Description:**

This command is used to set the data used in advertising PDUs that have a data field. This command may be issued at any time after an advertising set identified by the Advertising_Handle parameter has been created using the HCI_LE_Set_Extended_Advertising_Parameters command (see Section 7.8.53), regardless of whether advertising in that set is enabled or disabled.

If advertising is currently enabled for the specified advertising set, the Controller shall use the new data in subsequent extended advertising events for this advertising set. If an extended advertising event is in progress when this command is issued, the Controller may use the old or new data for that event.

If advertising is currently disabled for the specified advertising set, the data shall be kept by the Controller and used once advertising is enabled for that set. The data shall be discarded when the advertising set is removed.

Only the significant part of the advertising data should be transmitted in the advertising packets as defined in [Vol 3] Part C, Section 11.

The Host may set the advertising data in one or more operations using the Operation parameter in the command.

If Operation indicates the start of new data (values 0x01 or 0x03), then any existing partial or complete advertising data shall be discarded.

If the advertising data is discarded by the command or the combined length of the data after the command completes is zero, the advertising set will have no advertising data.

If Operation is 0x04, the behavior is the same as if the current advertising data had been sent again; this can be used to cause the Advertising DID value to be updated (see [Vol 6] Part B, Section 4.4.2.11).

The Fragment_Preference parameter provides a hint to the Controller as to whether advertising data should be fragmented.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The combined length of the data exceeds the capacity of the advertising set identified by Advertising_Handle (see Section 7.8.57) or the amount of memory available. All the data shall be discarded. | *Memory Capacity Exceeded* (0x07) |
| MC | The advertising set uses extended advertising and the combined length of the data is greater than the maximum that the Controller can transmit within the longest possible auxiliary advertising segment consistent with the current parameters of the advertising set (using the current advertising interval if advertising is enabled). All the data shall be discarded. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |
| MC | The advertising set has a type that does not support advertising data. | *Invalid HCI Command Parameters* (0x12) |
| MC | The advertising set uses legacy advertising PDUs that support advertising data and either Operation is not 0x03 or Advertising_Data_Length exceeds 31 octets. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is 0x04 and:<br>• advertising is disabled for the advertising set;<br>• the advertising set contains no data;<br>• the advertising set uses legacy PDUs; or<br>• Advertising_Data_Length is not zero. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 or 0x04 and Advertising_Data_Length is zero. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 or 0x04 and the advertising set is enabled. | *Command Disallowed* (0x0C) |
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Operation:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Intermediate fragment of fragmented extended advertising data |
| 0x01 | First fragment of fragmented extended advertising data |
| 0x02 | Last fragment of fragmented extended advertising data |
| 0x03 | Complete extended advertising data |
| 0x04 | Unchanged data (just update the Advertising DID) |
| All other values | Reserved for future use |

*Fragment_Preference:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Controller may fragment all Host advertising data |
| 0x01 | The Controller should not fragment or should minimize fragmentation of Host advertising data |
| All other values | Reserved for future use |

*Advertising_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 251 | The number of octets in the Advertising Data parameter |
| All other values | Reserved for future use |

*Advertising_Data:* *Size: Advertising_Data_Length octets*

| Parameter Description |
|---|
| Advertising data formatted as defined in [Vol 3] Part C, Section 11<br>Note: This parameter has a variable length. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Advertising_Data command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Advertising_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Extended_Advertising_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.55 LE Set Extended Scan Response Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_Scan_-Response_Data | 0x0038 | Advertising_Handle,<br>Operation,<br>Fragment_Preference,<br>Scan_Response_Data_Length,<br>Scan_Response_Data | Status |

**Description:**

This command is used to provide scan response data used in scanning response PDUs. This command may be issued at any time after the advertising set identified by the Advertising_Handle parameter has been created using the HCI_LE_Set_Extended_Advertising_Parameters command (see Section 7.8.53) regardless of whether advertising in that set is enabled or disabled.

If advertising is currently enabled for the specified advertising set, the Controller shall use the new data in subsequent extended advertising events for this advertising set. If an extended advertising event is in progress when this command is issued, the Controller may use the old or new data for that event.

If advertising is currently disabled for the specified advertising set, the data shall be kept by the Controller and used once advertising is enabled for that set. The data shall be discarded when the advertising set is removed.

Only the significant part of the scan response data should be transmitted in the advertising packets as defined in [Vol 3] Part C, Section 11.

The Host may set the scan response data in one or more operations using the Operation parameter in the command.

If Operation indicates the start of new data (values 0x01 or 0x03), then any existing partial or complete scan response data shall be discarded.

If the scan response data is discarded by the command or the combined length of the data after the command completes is zero, the advertising set will have no scan response data.

The Fragment_Preference parameter provides a hint to the Controller as to whether advertising data should be fragmented.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The combined length of the data exceeds the capacity of the advertising set identified by Advertising_Handle (see Section 7.8.57) or the amount of memory available. All the data shall be discarded. | *Memory Capacity Exceeded* (0x07) |
| MC | The advertising set uses extended advertising and the combined length of the data is greater than the maximum that the Controller can transmit within the longest possible auxiliary advertising segment consistent with the current parameters of the advertising set (using the current advertising interval if advertising is enabled). All the data shall be discarded. | *Packet Too Long* (0x45) |
| MC | The advertising set is non-scannable and the command is used other than to discard existing data. | *Invalid HCI Command Parameters* (0x12) |
| MC | The advertising set uses scannable legacy advertising PDUs and either Operation is not 0x03 or Advertising_Data_Length exceeds 31 octets. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 and Scan_Response_Data_Length is zero. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 and the advertising set is enabled. | *Command Disallowed* (0x0C) |
| MC | The advertising set uses scannable extended advertising PDUs, advertising is enabled for the advertising set, and Scan_Response_Data_Length is zero. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Command Disallowed* (0x0C) |
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Operation:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Intermediate fragment of fragmented scan response data |
| 0x01 | First fragment of fragmented scan response data |

| Value | Parameter Description |
|---|---|
| 0x02 | Last fragment of fragmented scan response data |
| 0x03 | Complete scan response data |
| All other values | Reserved for future use |

*Fragment_Preference:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Controller may fragment all scan response data |
| 0x01 | The Controller should not fragment or should minimize fragmentation of scan response data |
| All other values | Reserved for future use |

*Scan_Response_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 251 | The number of octets in the Scan_Response Data parameter |
| All other values | Reserved for future use |

*Scan_Response_Data:* *Size: Scan_Response_Data_Length octets*

| Parameter Description |
|---|
| Scan response data formatted as defined in [Vol 3] Part C, Section 11<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Scan_Response_Data command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Scan_Response_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Extended_Scan_Response_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.56 LE Set Extended Advertising Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_-Advertising_Enable | 0x0039 | Enable,<br>Num_Sets,<br>Advertising_Handle[i],<br>Duration[i],<br>Max_Extended_Advertising_Events[i] | Status |

**Description:**

This command is used to request the Controller to enable or disable one or more advertising sets using the advertising sets identified by the Advertising_Handle[i] parameter. The Controller manages the timing of advertisements in accordance with the advertising parameters given in the HCI_LE_Set_Extended_Advertising_Parameters command. The Num_Sets parameter is the number of advertising sets contained in the parameter arrays. If Enable and Num_Sets are both set to 0x00, then all advertising sets are disabled.

The Controller shall only start an advertising event when the corresponding advertising set is enabled. The Controller shall continue advertising until all advertising sets have been disabled. An advertising set shall be disabled when the Host issues an HCI_LE_Set_Extended_Advertising_Enable command with the Enable parameter set to 0x00 (Advertising is disabled), a connection is created using that advertising set, the duration specified in the Duration[i] parameter expires, or the number of extended advertising events transmitted for the set exceeds the Max_Extended_Advertising_Events[i] parameter.

The Duration[i] parameter indicates the duration for which that advertising set is enabled. The duration begins at the start of the first advertising event of this advertising set. The Controller should not start an extended advertising event that it cannot complete within the duration.

If the advertising is high duty cycle connectable directed advertising, then Duration[i] shall be less than or equal to 1.28 seconds and shall not be equal to 0.

The Max_Extended_Advertising_Events[i] parameter, if non-zero, indicates the maximum number of extended advertising events that shall be sent prior to disabling the extended advertising set even if the Duration[i] parameter has not expired.

Duration[i] and Max_Extended_Advertising_Events[i] shall be ignored when Enable is set to 0x00.

If the HCI_LE_Set_Extended_Advertising_Enable command is sent again for an advertising set while that set is enabled, the timer used for the duration and the number of events counter are reset and any change to the random address shall take effect.

Disabling the advertising set identified by the Advertising_Handle[i] parameter does not disable any periodic advertising associated with that set.

Disabling an advertising set that is already disabled has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The same advertising set is identified by more than one entry in Advertising_Handle[i]. | *Invalid HCI Command Parameters* (0x12) |
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |

The following error conditions only apply if Enable is set to 0x01.

| Type | Condition | Error code |
|---|---|---|
| R | Insufficient resources to schedule all the advertising sets. | *Rejected due to Limited Resources* (0x0D) |
| MC | Num_Sets is set to 0x00. | *Invalid HCI Command Parameters* (0x12) |
| MC | The advertising set contains partial advertising data or partial scan response data. | *Command Disallowed* (0x0C) |
| MC | The advertising set uses scannable extended advertising PDUs and there is no scan response data. | *Command Disallowed* (0x0C) |
| MC | The advertising set uses connectable extended advertising and the advertising data in the advertising set does not fit into the AUX_ADV_IND PDU.<br>(The maximum amount of data that will fit in the PDU depends on which options are selected and on the maximum length of PDU that the Controller is able to transmit.) | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| MC | The advertising set uses extended advertising and the combined length of the advertising data or of any scan response data is greater than the maximum that the Controller can transmit within the longest possible auxiliary advertising segment consistent with the chosen advertising interval. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long*(0x45) |
| R | The advertising set's Own_Address_Type is set to 0x00 and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | The advertising set's Own_Address_Type is set to 0x01 and the device's random address has not been set using the HCI_LE_Set_Advertising_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |
| R | The advertising set's Own_Address_Type is set to 0x02, the Controller's resolving list does not contain a matching entry, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | The advertising set's Own_Address_Type is set to 0x03, the Controller's resolving list does not contain a matching entry, and the device's random address has not been set using the HCI_LE_Set_Advertising_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Advertising is disabled |
| 0x01 | Advertising is enabled |
| All other values | Reserved for future use |

*Num_Sets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable all advertising sets |
| 0x01 to 0x3F | Number of advertising sets to enable or disable |
| All other values | Reserved for future use |

*Advertising_Handle[i]:* *Size: Num_Sets × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify an advertising set |
| All other values | Reserved for future use |

*Duration[i]:* *Size: Num_Sets × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No advertising duration. Advertising to continue until the Host disables it. |
| N = 0xXXXX | Advertising duration<br>Range: 0x0001 to 0xFFFF<br>Time = N × 10 ms<br>Time Range: 10 ms to 655,350 ms |

*Max_Extended_Advertising_Events[i]:* *Size: Num_Sets × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Maximum number of extended advertising events the Controller shall attempt to send prior to terminating the extended advertising |
| 0x00 | No maximum number of advertising events. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Advertising_Enable command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Advertising_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Extended_Advertising_Enable command has completed, an HCI_Command_Complete event shall be generated.

If the Duration[i] parameter is set to a value other than 0x0000, an HCI_LE_Advertising_Set_Terminated event shall be generated when the duration specified in the Duration[i] parameter expires. However, if the advertising set is for high duty cycle connectable directed advertising and no connection is created before the duration expires, an HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event with the Status parameter set to the

error code *Advertising Timeout* (0x3C) may be generated instead of or in addition to the HCI_LE_Advertising_Set_Terminated event. If the Controller generates both events, they may be in either order.

If the Max_Extended_Advertising_Events[i] parameter is set to a value other than 0x00, an HCI_LE_Advertising_Set_Terminated event shall be generated when the maximum number of extended advertising events has been transmitted by the Controller.

If the advertising set is connectable and a connection gets created, an HCI_LE_Connection_Complete or HCI_LE_Enhanced_Connection_Complete event shall be generated followed by an HCI_LE_Advertising_Set_Terminated event with the Status parameter set to 0x00. The Controller should not send any other events in between these two events. If the Controller supports the LE Channel Selection Algorithm #2 feature, then the HCI_LE_Advertising_Set_Terminated event may be immediately preceded or followed by an HCI_LE_Channel_Selection_Algorithm event.

Note: If this command is used to disable advertising at about the same time that a connection is established or the advertising duration expires, there is a possible race condition in that it is possible to receive both an HCI_LE_Connection_Complete, HCI_LE_Enhanced_Connection_Complete, or HCI_LE_Advertising_Set_Terminated event and the HCI_Command_Complete event for this command.

### 7.8.57 LE Read Maximum Advertising Data Length command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Maximum_-Advertising_Data_Length | 0x003A | *none* | Status,<br>Max_Advertising_Data_Length |

**Description:**

This command is used to read the maximum length of data supported by the Controller for use as advertisement data or scan response data in an advertising event or as periodic advertisement data.

Note: The maximum amount might require fragmenting across multiple PDUs (see [Vol 6] Part B, Section 2.3.4.9).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Maximum_Advertising_Data_Length command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Maximum_Advertising_Data_Length command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Max_Advertising_Data_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x001F to 0x0672 | Maximum supported advertising data length |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Maximum_Advertising_Data_Length command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.58 LE Read Number of Supported Advertising Sets command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Number_of_-Supported_Advertising_Sets | 0x003B | *none* | Status,<br>Num_Supported_Advertising_Sets |

**Description:**

This command is used to read the number of advertising sets (including those already created) that the advertising Controller can support at the present time.

Note: The number of advertising sets that can be supported is not fixed and the Controller can change it at any time (e.g., because the memory used to store advertising sets can also be used for other purposes).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Number_of_Supported_Advertising_Sets command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Number_of_Supported_Advertising_Sets command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Supported_Advertising_Sets:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xF0 | Number of advertising sets supported at the same time |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Number_of_Supported_Advertising_Sets command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.59 LE Remove Advertising Set command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_Advertising_Set | 0x003C | Advertising_Handle | Status |

**Description:**

This command is used to remove an advertising set from the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |
| MC | Advertising or periodic advertising is enabled on the advertising set. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remove_Advertising_Set command succeeded |
| 0x01 to 0xFF | HCI_LE_Remove_Advertising_Set command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_Advertising_Set command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.60 LE Clear Advertising Sets command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Clear_Advertising_Sets | 0x003D | *none* | Status |

**Description:**

This command is used to remove all existing advertising sets from the Controller.

Note: All advertising sets are cleared on HCI reset.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising or periodic advertising is enabled on the advertising set. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Clear_Advertising_Sets command succeeded |
| 0x01 to 0xFF | HCI_LE_Clear_Advertising_Sets command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Clear_Advertising_Sets command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.61 LE Set Periodic Advertising Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_-Advertising_Parameters [v2] | 0x0086 | Advertising_Handle, Periodic_Advertising_Interval_Min, Periodic_Advertising_Interval_Max, Periodic_Advertising_Properties, Num_Subevents, Subevent_Interval, Response_Slot_Delay, Response_Slot_Spacing, Num_Response_Slots | Status, Advertising_Handle |
| HCI_LE_Set_Periodic_-Advertising_Parameters [v1] | 0x003E | Advertising_Handle, Periodic_Advertising_Interval_Min, Periodic_Advertising_Interval_Max, Periodic_Advertising_Properties | Status |

**Description:**

This command is used by the Host to set the parameters for periodic advertising.

The Advertising_Handle parameter identifies the advertising set whose periodic advertising parameters are being configured.

The Periodic_Advertising_Interval_Min parameter shall be less than or equal to the Periodic_Advertising_Interval_Max parameter. The Periodic_Advertising_Interval_Min and Periodic_Advertising_Interval_Max parameters should not be the same value to enable the Controller to determine the best advertising interval given other activities.

The Periodic_Advertising_Properties parameter indicates which fields should be included in the advertising packet.

The Num_Subevents parameter identifies the number of subevents that shall be transmitted for each periodic advertising event. If the Num_Subevents parameter value is 0x00, then the Periodic Advertising does not have responses and the Controller shall ignore the Subevent_Interval, Response_Slot_Delay, Response_Slot_Spacing, and Num_Response_Slots parameters. If Num_Subevents is greater than 0, then the Periodic Advertising is PAwR.

The Subevent_Interval parameter identifies the time between the subevents of PAwR. The Subevent_Interval shall be less than or equal to the Periodic_Advertising_Interval_Min divided by the Num_Subevents of the advertising

set. If Num_Subevents is set to 1, then the Controller shall ignore Subevent_Interval and uses of Subevent_Interval in the next two paragraphs shall be replaced by Periodic_Advertising_Interval_Max.

The Response_Slot_Delay parameter identifies the time between the start of the advertising packet at the start of a subevent and the start of the first response slot. The Response_Slot_Delay shall be less than the Subevent_Interval.

The Response_Slot_Spacing parameter identifies the time between the start of two consecutive response slots. The Response_Slot_Spacing shall be less than or equal to 10 × (Subevent_Interval − Response_Slot_Delay) ÷ Num_Response_Slots. If the Num_Response_Slots parameter is set to 1, then the Controller shall ignore the Response_Slot_Spacing parameter.

The Num_Response_Slots parameter identifies the number of response slots in a subevent.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| Num_Subevents | 0 |
| Subevent_Interval | *ignored* |
| Response_Slot_Delay | *ignored* |
| Response_Slot_Spacing | *ignored* |
| Num_Response_Slots | *ignored* |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |
| M | The periodic advertising interval range defined by Periodic_Advertising_Interval_Min and Periodic_Advertising_Interval_Max does not overlap the periodic advertising interval range supported by the Controller. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Advertising_Handle identifies an advertising set with scannable, connectable, legacy, or anonymous advertising. | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| MC | Periodic advertising is enabled for the advertising set. | *Command Disallowed* (0x0C) |
| MC | The Advertising_Handle does not identify an advertising set that is already configured for periodic advertising and the Controller is unable to support more periodic advertising at present. | *Memory Capacity Exceeded* (0x07) |
| MC | The advertising set already contains periodic advertising data and the length of the data is greater than the maximum that the Controller can transmit within a periodic advertising interval of Periodic_Advertising_Max. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

*Periodic_Advertising_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N =0xXXXX | Minimum advertising interval for periodic advertising.<br>Range: 0x0006 to 0xFFFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 81.91875 s |

*Periodic_Advertising_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum advertising interval for periodic advertising.<br>Range: 0x0006 to 0xFFFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 81.91875 s |

*Periodic_Advertising_Properties:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 6 | Include TxPower in the advertising PDU |
| All other bits | Reserved for future use |

*Num_Subevents:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic Advertising without responses |
| 0xXX | Number of subevents in the PAwR.<br>Range: 0x01 to 0x80 |

*Subevent_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Interval between subevents.<br>Range: 0x06 to 0xFF<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 318.75 ms |

*Response_Slot_Delay:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Time between the advertising packet in a subevent and the first response slot.<br>Range: 0x01 to 0xFE<br>Time = N × 1.25 ms<br>Time Range: 1.25 ms to 317.5 ms |

*Response_Slot_Spacing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Time between response slots.<br>Range: 0x02 to 0xFF<br>Time = N × 0.125 ms<br>Time Range: 0.25 ms to 31.875 ms |

*Num_Response_Slots:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of subevent response slots.<br>Range: 0x01 to 0xFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Parameters command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Periodic_Advertising_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.62 LE Set Periodic Advertising Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_Data | 0x003F | Advertising_Handle,<br>Operation,<br>Advertising_Data_Length,<br>Advertising_Data | Status |

**Description:**

This command is used to set the data used in periodic advertising PDUs. This command may be issued at any time after the advertising set identified by the Advertising_Handle parameter has been configured for periodic advertising using the HCI_LE_Set_Periodic_Advertising_Parameters command (see Section 7.8.61), regardless of whether periodic advertising in that set is enabled or disabled.

If periodic advertising is currently enabled for the specified advertising set, the Controller shall use the new data in subsequent periodic advertising events for this advertising set. If a periodic advertising event is in progress when this command is issued, the Controller may use the old or new data for that event.

If periodic advertising is currently disabled for the specified advertising set, the data shall be kept by the Controller and used once periodic advertising is enabled for that set. The data shall be discarded when the advertising set is removed.

Only the significant part of the periodic advertising data should be transmitted in the advertising packets as defined in [Vol 3] Part C, Section 11.

The Host may set the periodic advertising data in one or more operations using the Operation parameter in the command.

If Operation indicates the start of new data (values 0x01 or 0x03), then any existing partial or complete data shall be discarded.

If Operation is 0x04, then the behavior is the same as if the current periodic advertising data had been sent again; this can be used to cause the Advertising DID value to be updated (see [Vol 6] Part B, Section 4.4.2.11).

If the periodic advertising data is discarded by the command or the combined length of the data after the command completes is zero, the advertising set will have no periodic advertising data.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The advertising set for the Advertising_Handle is not configured for Periodic Advertising or Periodic Advertising with Responses. | *Command Disallowed* (0x0C) |
| MC | The combined length of the data exceeds the capacity of the advertising set identified by the Advertising_Handle parameter (see Section 7.8.57) or the amount of memory available. All the data shall be discarded. | *Memory Capacity Exceeded* (0x07) |
| MC | The combined length of data is greater than the maximum that the Controller can transmit within the periodic advertising interval, if periodic advertising is enabled, or the Periodic_Advertising_Interval_Max for the advertising set, if disabled. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |
| MC | Operation is set to 0x04 and periodic advertising is disabled for the advertising set, the periodic advertising set contains no data, or Advertising_Data_Length is not zero. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 or 0x04 and Advertising_Data_Length is zero. | *Invalid HCI Command Parameters* (0x12) |
| MC | Operation is not 0x03 or 0x04 and Periodic advertising is enabled for the advertising set. | *Command Disallowed* (0x0C) |
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Operation:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Intermediate fragment of fragmented periodic advertising data |
| 0x01 | First fragment of fragmented periodic advertising data |
| 0x02 | Last fragment of fragmented periodic advertising data |
| 0x03 | Complete periodic advertising data |
| 0x04 | Unchanged data (just update the Advertising DID of the periodic advertising) |

| Value | Parameter Description |
|---|---|
| All other values | Reserved for future use |

*Advertising_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 252 | The number of octets in the Advertising Data parameter |
| All other values | Reserved for future use |

*Advertising_Data:* *Size: Advertising_Data_Length octets*

| Parameter Description |
|---|
| Periodic advertising data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Data command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Periodic_Advertising_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.63 LE Set Periodic Advertising Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_Enable | 0x0040 | Enable,<br>Advertising_Handle | Status |

**Description:**

This command is used to request the Controller to enable or disable the periodic advertising for the advertising set specified by the Advertising_Handle parameter (ordinary advertising is not affected).

If the advertising set is not currently enabled (see the HCI_LE_Set_Extended_Advertising_Enable command), the periodic advertising is not started until the advertising set is enabled. Once the advertising set has been enabled, the Controller shall continue periodic advertising until the Host issues an HCI_LE_Set_Periodic_Advertising_Enable command with bit 0 of Enable set to 0 (periodic advertising is disabled). Disabling the advertising set has no effect on the periodic advertising once the advertising set has been enabled.

The Controller manages the timing of advertisements in accordance with the advertising parameters given in the HCI_LE_Set_Periodic_Advertising_Parameters command.

If bit 0 of Enable is set to 0 and the Controller supports the Periodic Advertising ADI Support feature, then the Controller shall ignore bit 1.

Enabling periodic advertising when it is already enabled can cause the random address to change. Disabling periodic advertising when it is already disabled has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |
| MC | Bit 0 of Enable is set to 1 and the advertising set contains partial periodic advertising data. | *Command Disallowed* (0x0C) |
| MC | Bit 0 of Enable is set to 1 and the Host has not issued the HCI_LE_Set_Periodic_Advertising_Parameters command for the advertising set.<br>The Controller may use vendor-specific periodic advertising parameters instead of reporting an error. | *Command Disallowed* (0x0C) |

| Type | Condition | Error code |
|---|---|---|
| MC | Bit 0 of Enable is set to 1 and the length of the periodic advertising data is greater than the maximum that the Controller can transmit within the specified periodic advertising interval. If advertising on the LE Coded PHY, the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |
| MC | Bit 0 of Enable is set to 1 and Advertising_Handle identifies an advertising set with scannable, connectable, legacy, or anonymous advertising. | *Command Disallowed* (0x0C) |
| M | Bit 1 of Enable is set to 1 and the Controller does not support the Periodic Advertising ADI Support feature. | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*Enable:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Enable periodic advertising |
| 1 | Include the ADI field in AUX_SYNC_IND and AUX_SYNC_SUBEVENT_IND PDUs |
| All other bits | Reserved for future use |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Enable command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Periodic_Advertising_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.64 LE Set Extended Scan Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_Scan_Parameters | 0x0041 | Own_Address_Type,<br>Scanning_Filter_Policy,<br>Scanning_PHYs,<br>Scan_Type[i],<br>Scan_Interval[i],<br>Scan_Window[i] | Status |

**Description:**

This command is used to set the extended scan parameters to be used on the advertising physical channels.

The Scanning_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel. The Host may enable one or more scanning PHYs. If the Host specifies a PHY that is not supported by the Controller, including a bit that is reserved for future use, it should return the error code *Unsupported Feature or Parameter Value* (0x11). The Scan_Type[i], Scan_Interval[i], and Scan_Window[i] parameters array elements are ordered in the same order as the set bits in the Scanning_PHY parameter, starting from bit 0. The number of array elements is determined by the number of bits set in the Scanning_PHY parameter.

The Scan_Type[i] parameter specifies the type of scan to perform. For each advertisement received, the type of scan performed shall be determined by the Scan_Type[i] for the PHY the primary advertising PDU was received on.

The Scan_Interval[i] and Scan_Window[i] parameters are recommendations from the Host on how long (Scan_Window[i]) and how frequently (Scan_Interval[i]) the Controller should scan (see [Vol 6] Part B, Section 4.4.3); however the frequency and length of the scan is implementation specific.

The Own_Address_Type parameter indicates the type of address being used in the scan request packets.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The requested scan is not supported. | *Invalid HCI Command Parameters* (0x12) |
| MC | Scanning is enabled. | *Command Disallowed* (0x0C) |
| M | Decision-Based Advertising Filtering is not supported and either bit 2 or bit 3 of Scanning_Filter_Policy is set. | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address. |
| 0x03 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from LE_Set_Random_Address. |
| All other values | Reserved for future use |

*Scanning_Filter_Policy:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 0 = unfiltered scanning policy<br>1 = filtered scanning policy |
| 1 | 0 = Basic filter policy<br>1 = Extended filter policy |
| 2 and 3 | Decision scanning filter policy mode:<br>0b00 = No-decisions mode<br>0b01 = All-PDUs mode<br>0b11 = Decisions-only mode |
| All other bits | Reserved for future use |

*Scanning_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Scan advertisements on the LE 1M PHY |
| 2 | Scan advertisements on the LE Coded PHY |
| All other bits | Reserved for future use |

*Scan_Type[i]:* *Size: Bits set in Scanning_PHYs × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Passive Scanning. No scan request PDUs shall be sent. |
| 0x01 | Active Scanning. Scan request PDUs may be sent. |
| All other values | Reserved for future use |

*Scan_Interval[i]:* *Size: Bits set in Scanning_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel.<br>Range: 0x0004 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 40.959375 s |

*Scan_Window[i]:* *Size: Bits set in Scanning_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Duration of the scan on the primary advertising physical channel.<br>Range: 0x0004 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 40.959375 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Scan_Parameters command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Scan_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Extended_Scan_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.65 LE Set Extended Scan Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Extended_Scan_Enable | 0x0042 | Enable,<br>Filter_Duplicates,<br>Duration,<br>Period | Status |

**Description:**

This command is used to enable or disable scanning for both legacy and extended advertising PDUs.

The Enable parameter determines whether scanning is enabled or disabled. If it is set to 0x00, the remaining parameters shall be ignored.

The Filter_Duplicates parameter controls whether the Link Layer should filter out duplicate advertising reports (filtering duplicates enabled) to the Host or if the Link Layer should generate advertising reports for each packet received (filtering duplicates disabled). See [Vol 6] Part B, Section 4.4.3.5.

If the Filter_Duplicates parameter is set to 0x00, all advertisements received from advertisers shall be sent to the Host in advertising report events.

If the Filter_Duplicates parameter is set to 0x01, duplicate advertisements should not be sent to the Host in advertising report events until scanning is disabled.

If the Filter_Duplicates parameter is set to 0x02, duplicate advertisements in a single scan period should not be sent to the Host in advertising report events; this setting shall only be used if both Period and Duration are non-zero.

If the Duration parameter is zero or both the Duration parameter and Period parameter are non-zero, the Controller shall continue scanning until scanning is disabled by the Host issuing an HCI_LE_Set_Extended_Scan_Enable command with the Enable parameter set to 0x00 (Scanning is disabled). The Period parameter shall be ignored when the Duration parameter is zero.

If the Duration parameter is non-zero and the Period parameter is zero, the Controller shall continue scanning until the duration specified in the Duration parameter has expired.

When the Duration and Period parameters are non-zero, the Controller shall scan for the duration of the Duration parameter within a scan period specified by the Period parameter. After the scan period has expired, a new scan period shall begin and

scanning shall begin again for the duration specified. The scan periods continue until the Host disables scanning.

If the HCI_LE_Set_Extended_Scan_Enable command with Enable set to 0x01 is sent while scanning is already enabled, the timers used for duration and period are reset to the new parameter values and a new scan period is started. Any change to the Filter_Duplicates setting or the random address shall take effect.

Disabling scanning when it is disabled has no effect.

Note: The duration of a scan period refers to the time spent scanning on both the primary and secondary advertising physical channels. However, expiry of the duration does not prevent the Link Layer from scanning for and receiving auxiliary packets of received advertisements.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Enable is set to 0x01 and the Host has not issued the HCI_LE_Set_Extended_Scan_Parameters command for the advertising set. The Controller may use vendor-specific scanning parameters instead of reporting an error. | *Command Disallowed* (0x0C) |
| MC | Filter_Duplicates is set to 0x02 and either Period or Duration is set to 0. | *Invalid HCI Command Parameters* (0x12) |
| MC | Both Duration and Period are non-zero and Duration is greater than or equal to Period. Note: this compares the times, not how they are represented. | *Invalid HCI Command Parameters* (0x12) |
| R | Enable is set to 0x01, the scanning parameters' Own_Address_Type is set to 0x00 or 0x02, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Enable is set to 0x01, the scanning parameters' Own_Address_Type is set to 0x01 or 0x03, and the random address for the device has not been initialized using the HCI_LE_Set_Random_Address command. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Scanning disabled |
| 0x01 | Scanning enabled |
| All other values | Reserved for future use |

*Filter_Duplicates:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Duplicate filtering disabled |
| 0x01 | Duplicate filtering enabled |
| 0x02 | Duplicate filtering enabled, reset for each scan period |
| All other values | Reserved for future use |

*Duration:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Scan continuously until explicitly disable |
| N = 0xXXXX | Scan duration<br>Range: 0x0001 to 0xFFFF<br>Time = N × 10 ms<br>Time Range: 10 ms to 655.35 s |

*Period:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Scan continuously |
| N = 0xXXXX | Time interval from when the Controller started its last Scan_Duration until it begins the subsequent Scan_Duration.<br>Range: 0x0001 to 0xFFFF<br>Time = N × 1.28 sec<br>Time Range: 1.28 s to 83,884.8 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Extended_Scan_Enable command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Extended_Scan_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Extended_Scan_Enable command has completed, an HCI_Command_Complete event shall be generated.

Zero or more LE Extended Advertising Reports are generated by the Controller based on any advertising packets received and the duplicate filtering in effect. More than one advertising packet may be reported in each HCI_LE_Extended_Advertising_Report event.

At the end of a single scan (Duration non-zero but Period zero), an HCI_LE_Scan_Timeout event shall be generated.

### 7.8.66 LE Extended Create Connection command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Extended_Create_Connection [v2] | 0x0085 | Advertising_Handle,<br>Subevent,<br>Initiator_Filter_Policy,<br>Own_Address_Type,<br>Peer_Address_Type,<br>Peer_Address,<br>Initiating_PHYs,<br>Scan_Interval[i],<br>Scan_Window[i],<br>Connection_Interval_Min[i],<br>Connection_Interval_Max[i],<br>Max_Latency[i],<br>Supervision_Timeout[i],<br>Min_CE_Length[i],<br>Max_CE_Length[i] | *none* |
| HCI_LE_Extended_Create_Connection [v1] | 0x0043 | Initiator_Filter_Policy,<br>Own_Address_Type,<br>Peer_Address_Type,<br>Peer_Address,<br>Initiating_PHYs,<br>Scan_Interval[i],<br>Scan_Window[i],<br>Connection_Interval_Min[i],<br>Connection_Interval_Max[i],<br>Max_Latency[i],<br>Supervision_Timeout[i],<br>Min_CE_Length[i],<br>Max_CE_Length[i] | *none* |

**Description:**

This command is used to create a single ACL connection, with the local device in the Central role, to a connectable advertiser. The command is also used to create a single ACL connection between a periodic advertiser and a synchronized device.

If a connection is created with the local device in the Peripheral role while this command is pending, then this command remains pending.

The Advertising_Handle parameter is used to identify the periodic advertising train.

The Subevent parameter is used to identify the subevent where a connection request shall be initiated from a periodic advertising train. The Host may use this subevent whether or not the Controller has requested data for it using the HCI_LE_Periodic_Advertising_Subevent_Data_Request event.

If the Advertising_Handle and Subevent parameters are set to values other than 0xFF, then the Initiator_Filter_Policy, Scan_Interval[i], and Scan_Window[i] parameters shall be ignored.

The Initiator_Filter_Policy parameter is used to determine whether the Filter Accept List is used and whether to process decision PDUs and other advertising PDUs. If the Filter Accept List is used or the policy allows decision PDUs to be processed (or both), then the device to connect to is the first one that sends an advertisement received successfully by the local device and that is on the list or that is accepted by the decision instructions (see Section 7.8.145), as specified by the policy. If the Filter Accept List is not used, then the Peer_Address_Type and the Peer_Address parameters specify the address type and address of the advertising device to connect to for advertisements not using decision PDUs. If Initiator_Filter_Policy is set to 0x03, then devices on the Filter Accept List shall still be processed using the decision instructions.

The Own_Address_Type parameter indicates the type of address being used in the connection request packets.

The Peer_Address_Type parameter indicates the type of address used in the connectable advertisement sent by the peer.

The Peer_Address parameter indicates the Peer's Public Device Address, Random (static) Device Address, Non-Resolvable Private Address, or Resolvable Private Address depending on the Peer_Address_Type parameter.

The Initiating_PHYs parameter indicates the PHY(s) on which the advertising packets should be received on the primary advertising physical channel and the PHYs for which connection parameters have been specified. The Host may enable one or more initiating PHYs. The array elements of the arrayed parameters are ordered in the same order as the set bits in the Initiating_PHYs parameter, starting from bit 0. The number of array elements is determined by the number of bits set in the Initiating_PHYs parameter. When a connectable advertisement is received and a connection request is sent on one PHY, scanning on any other PHYs is terminated.

The Scan_Interval[i] and Scan_Window[i] parameters are recommendations from the Host on how long (Scan_Window[i]) and how frequently (Scan_Interval[i]) the Controller

should scan (see [Vol 6] Part B, Section 4.5.3); however the frequency and length of the scan is implementation specific. If bit 1 is set in Initiating_PHYs, the values for the LE 2M PHY shall be ignored.

The Connection_Interval_Min[i] and Connection_Interval_Max[i] parameters define the minimum and maximum allowed connection interval. The Connection_Interval_Min[i] parameter shall not be greater than the Connection_Interval_Max[i] parameter.

The Max_Latency[i] parameter defines the maximum allowed Peripheral latency (see [Vol 6] Part B, Section 4.5.1).

The Supervision_Timeout[i] parameter defines the link supervision timeout for the connection. The Supervision_Timeout[i] in milliseconds shall be larger than (1 + Max_Latency[i]) × Connection_Interval_Max[i] × 2, where Connection_Interval_Max[i] is given in milliseconds (see [Vol 6] Part B, Section 4.5.2).

The Min_CE_Length[i] and Max_CE_Length[i] parameters provide the Controller with the expected minimum and maximum length of the connection events. The Controller is not required to use these values.

Where the connection is made on a PHY whose bit is not set in the Initiating_PHYs parameter, the Controller shall use the Connection_Interval_Min[i], Connection_Interval_Max[i], Max_Latency[i], Supervision_Timeout[i], Min_CE_Length[i], and Max_CE_Length[i] parameters for an implementation-chosen PHY whose bit is set in the Initiating_PHYs parameter.

**Missing parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| Advertising_Handle | 0xFF |
| Subevent | 0xFF |

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| M | Advertising_Handle or Subevent, but not both, is set to 0xFF. | *Invalid HCI Command Parameters* (0x12) |
| RC | Initiating_PHYs has a bit set which is not supported by the Controller or is RFU. | *Unsupported Feature or Parameter Value* (0x11) |
| M | Advertising_Handle is not 0xFF and Initiating_PHYs does not have the bit set for the PHY used for the specified periodic advertising train. | *Invalid HCI Command Parameters* (0x12) |
| MC | The requested scan is not supported. | *Invalid HCI Command Parameters* (0x12) |
| MC | Another HCI_LE_Extended_Create_Connection command is pending. | *Command Disallowed* (0x0C) |
| M | The local device is already connected to the same device address as the advertiser (including two different Resolvable Private Addresses that resolve to the same IRK). | *Connection Already Exists* (0x0B) |
| R | Own_Address_Type is 0x00 and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is 0x01 and the random address for the device has not been initialized using HCI_LE_Set_Random_Address. | *Invalid HCI Command Parameters* (0x12) |
| R | Own_Address_Type is 0x02, Initiator_Filter_Policy is 0x00, the Controller's resolving list does not contain a matching entry, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| R | Own_Address_Type is 0x02, Initiator_Filter_Policy is not 0x00, and the device does not have a public address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is 0x03, Initiator_Filter_Policy is 0x00, the Controller's resolving list does not contain a matching entry, and the device's random address has not been initialized using HCI_LE_Set_Random_Address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Own_Address_Type is 0x03, Initiator_Filter_Policy is not 0x00, and the device's random address has not been initialized using HCI_LE_Set_Random_Address. | *Invalid HCI Command Parameters* (0x12) |
| MC | Initiating_PHYs does not have at least one bit set for a PHY allowed for scanning on the primary advertising physical channel. | *Invalid HCI Command Parameters* (0x12) |
| MC | The Controller has insufficient resources to handle any more connections. | *Rejected due to Limited Resources* (0x0D) |

| Type | Condition | Error code |
|---|---|---|
| M | Decision-Based Advertising Filtering is not supported and Initiator_Filter_Policy is not 0x00 or 0x01. | *Unsupported Feature or Parameter Value* (0x11) |
| R | Max_CE_Length is less than Min_CE_Length. | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Advertising_Handle identifying the periodic advertising train<br>Range: 0x00 to 0xEF or 0xFF |

*Subevent:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Subevent where the connection request is to be sent.<br>Range: 0x00 to 0x7F or 0xFF |

*Initiator_Filter_Policy:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Filter Accept List is not used to determine which advertiser to connect to. Decision PDUs shall be ignored. Peer_Address_Type and Peer_Address shall be used. |
| 0x01 | Filter Accept List is used to determine which advertiser to connect to. Decision PDUs shall be ignored. Peer_Address_Type and Peer_Address shall be ignored. |
| 0x02 | Filter Accept List is not used to determine which advertiser to connect to. Only Decision PDUs shall be processed. Peer_Address_Type and Peer_Address shall be ignored. |
| 0x03 | Filter Accept List is used to determine which advertiser to connect to. All PDUs shall be processed. Peer_Address_Type and Peer_Address shall be ignored. |
| 0x04 | All decision PDUs shall be processed. Filter Accept List is used to determine which other PDUs to process. Peer_Address_Type and Peer_Address shall be ignored. |
| All other values | Reserved for future use |

*Own_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| 0x02 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the public address. |
| 0x03 | Controller generates the Resolvable Private Address based on the local IRK from the resolving list. If the resolving list contains no matching entry, then use the random address from the most recent successful HCI_LE_Set_Random_Address command. |
| All other values | Reserved for future use |

*Peer_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the device to be connected. |

*Initiating_PHYs:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Scan connectable advertisements on the LE 1M PHY. Connection parameters for the LE 1M PHY are provided. |
| 1 | Connection parameters for the LE 2M PHY are provided. |
| 2 | Scan connectable advertisements on the LE Coded PHY. Connection parameters for the LE Coded PHY are provided. |
| All other bits | Reserved for future use |

*Scan_Interval[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Time interval from when the Controller started its last scan until it begins the subsequent scan on the primary advertising physical channel.<br>Range: 0x0004 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 40.959375 s |

*Scan_Window[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Duration of the scan on the primary advertising physical channel.<br>Range: 0x0004 to 0xFFFF<br>Time = N × 0.625 ms<br>Time Range: 2.5 ms to 40.959375 s |

*Connection_Interval_Min[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum value for the connection interval. This shall be less than or equal to Connection_Interval_Max[i].<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |
| All other values | Reserved for future use |

*Connection_Interval_Max[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum value for the connection interval. This shall be greater than or equal to Connection_Interval_Min[i].<br>Range: 0x0006 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 7.5 ms to 4 s |
| All other values | Reserved for future use |

*Max_Latency[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency for the connection in number of connection events.<br>Range: 0x0000 to 0x01F3 |
| All other values | Reserved for future use |

*Supervision_Timeout[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for the LE Link. (See [Vol 6] Part B, Section 4.5.2)<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |
| All other values | Reserved for future use |

*Min_CE_Length[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The minimum length of connection event recommended for this LE connection.<br>Time = N × 0.625 ms |

*Max_CE_Length[i]:* *Size: Bits set in Initiating_PHYs × 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The maximum length of connection event recommended for this LE connection.<br>Time = N × 0.625 ms |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Extended_Create_Connection command, the Controller sends the HCI_Command_Status event to the Host. An HCI_LE_Enhanced_Connection_Complete event shall be generated when a connection is created because of this command or the connection creation procedure is cancelled; until the event is generated, the command is considered pending. If a connection creation is discarded, then the error code *Connection Failed to be Established / Synchronization Timeout* (0x3E) shall be used. If a connection is created,

this event shall be immediately followed by an HCI_LE_Channel_Selection_Algorithm event.

### 7.8.67 LE Periodic Advertising Create Sync command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Periodic_Advertising_Create_-Sync | 0x0044 | Options,<br>Advertising_SID,<br>Advertiser_Address_Type,<br>Advertiser_Address,<br>Skip,<br>Sync_Timeout,<br>Sync_CTE_Type | *none* |

**Description:**

This command is used to synchronize with a single periodic advertising train from an advertiser and begin receiving periodic advertising packets.

This command may be issued whether or not scanning is enabled and scanning may be enabled and disabled (see Section 7.8.65) while this command is pending. However, synchronization can only occur when scanning is enabled. While scanning is disabled, no attempt to synchronize will take place.

The Options parameter is used to determine whether the Periodic Advertiser List is used, whether HCI_LE_Periodic_Advertising_Report events for this periodic advertising train are initially enabled or disabled, and whether duplicate reports are filtered or not. If the Periodic Advertising List is used, then the periodic advertising device to listen to is the first one that sends an AUX_SYNC_IND PDU received successfully by the local device, that is on the list, and that does not have the wrong type of Constant Tone Extension. If the Periodic Advertiser List is not used, then the Advertising_SID, Advertiser Address_Type, and Advertiser Address parameters specify the periodic advertising device to listen to; otherwise they shall be ignored.

The Advertising_SID parameter, if used, specifies the value that shall match the Advertising SID subfield in the ADI field of the received advertisement for it to be used to synchronize.

The Skip parameter specifies the maximum number of consecutive periodic advertising events that the receiver may skip after successfully receiving a periodic advertising packet.

The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.

The Sync_CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension (a value of 0 indicates that the presence or absence of a Constant Tone Extension is irrelevant).

If the Controller does not support the Connectionless CTE Receiver feature, then the Host should set Sync_CTE_Type to 0.

If the periodic advertiser changes the type of Constant Tone Extension after the scanner has synchronized with the periodic advertising, the scanner's Link Layer shall remain synchronized.

Irrespective of the value of the Skip parameter, the Controller should stop skipping packets before the Sync_Timeout would be exceeded.

If bit 1 of Options is set to 1 and the Controller supports the Periodic Advertising ADI Support feature, then the Controller shall ignore bit 2.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | All the non-reserved bits of Sync_CTE_Type are set to 1. | *Command Disallowed* (0x0C) |
| MC | Another HCI_LE_Periodic_Advertising_Create_Sync is pending. | *Command Disallowed* (0x0C) |
| MC | Options bit 0 is set to 0 and Advertising_SID, Advertiser_Address_Type, and Advertiser_Address identify a periodic advertising train that the Controller is already synchronized to. | *Connection Already Exists* (0x0B) |
| MC | The Controller has insufficient resources to handle any more periodic advertising trains. | *Memory Capacity Exceeded* (0x07) |
| M | Options bit 1 is set to 0, bit 2 is set to 1, and the Controller does not support the Periodic Advertising ADI Support feature. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Options bit 1 is set to 1 and the Controller does not support the HCI_LE_Set_Periodic_Advertising_Receive_Enable command. | *Connection Failed to be Established / Synchronization Timeout* (0x3E) |
| R | Sync_CTE_Type is non-zero and the Controller does not support the Connectionless CTE Receiver feature.<br>If the Controller does not return this error, then it should behave as if Sync_CTE_Type is set to zero. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Options:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 0: Use the Advertising_SID, Advertiser_Address_Type, and Advertiser_Address parameters to determine which advertiser to listen to<br>1: Use the Periodic Advertiser List to determine which advertiser to listen to. |
| 1 | 0: Reporting initially enabled<br>1: Reporting initially disabled |
| 2 | 0: Duplicate filtering initially disabled<br>1: Duplicate filtering initially enabled |
| All other bits | Reserved for future use |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Advertising SID subfield in the ADI field used to identify the Periodic Advertising |
| All other values | Reserved for future use |

*Advertiser_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Advertiser_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser |

*Skip:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum number of periodic advertising events that can be skipped after a successful receive<br>Range: 0x0000 to 0x01F3 |

*Sync_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Synchronization timeout for the periodic advertising train<br>Range: 0x000A to 0x4000<br>Time = N × 10 ms<br>Time Range: 100 ms to 163.84 s |

*Sync_CTE_Type:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Do not sync to packets with an AoA Constant Tone Extension |
| 1 | Do not sync to packets with an AoD Constant Tone Extension with 1 µs slots |
| 2 | Do not sync to packets with an AoD Constant Tone Extension with 2 µs slots |
| 3 | Do not sync to packets with a type 3 Constant Tone Extension (currently reserved for future use) |
| 4 | Do not sync to packets without a Constant Tone Extension |
| All other bits | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the HCI_LE_Periodic_Advertising_Create_Sync command has been received, the Controller sends the HCI_Command_Status event to the Host. An HCI_LE_Periodic_Advertising_Sync_Established event shall be generated when the Controller starts receiving periodic advertising packets.

If Sync_CTE_Type is non-zero and the Controller receives periodic advertising that has the wrong type of Constant Tone Extension then:

- If bit 0 of Options is set, then the Controller shall ignore this address and SID and continue to search for other periodic advertisements.
- Otherwise, the Controller shall generate an HCI_LE_Periodic_Advertising_Sync_Established event with Status set to Unsupported Remote Feature (0x1A).

When the Controller receives periodic advertising packets then, if reporting is enabled, it sends HCI_LE_Periodic_Advertising_Report events to the Host.

If the Controller does not receive a periodic advertising packet within 6 periodic advertising events of first listening, then it shall generate an HCI_LE_Periodic_Advertising_Sync_Established event that should have Status set to *Connection Failed to be Established / Synchronization Timeout* (0x3E).

Note: The HCI_LE_Periodic_Advertising_Sync_Established event can be sent as a result of synchronization being canceled by an HCI_LE_Periodic_Advertising_Create_Sync_Cancel command.

### 7.8.68 LE Periodic Advertising Create Sync Cancel command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Periodic_Advertising_Create_-Sync_Cancel | 0x0045 | *none* | Status |

**Description:**

This command is used to cancel the HCI_LE_Periodic_Advertising_Create_Sync command while it is pending.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | No HCI_LE_Periodic_Advertising_Create_Sync command is pending. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Periodic_Advertising_Create_Sync_Cancel command succeeded |
| 0x01 to 0xFF | HCI_LE_Periodic_Advertising_Create_Sync_Cancel command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Periodic_Advertising_Create_Sync_Cancel command has completed, the Controller sends an HCI_Command_Complete event to the Host.

After the HCI_Command_Complete is sent and if the cancellation was successful, the Controller sends an HCI_LE_Periodic_Advertising_Sync_Established event to the Host with the error code *Operation Cancelled by Host* (0x44).

### 7.8.69 LE Periodic Advertising Terminate Sync command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Periodic_Advertising_Terminate_-Sync | 0x0046 | Sync_Handle | Status |

**Description:**

This command is used to stop reception of the periodic advertising train identified by the Sync_Handle parameter.

Following successful completion of this command the Sync_Handle is destroyed.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Sync_Handle does not identify an existing periodic advertising train. | *Unknown Advertising Identifier* (0x42) |

**Command parameters:**

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Periodic_Advertising_Terminate_Sync command succeeded |
| 0x01 to 0xFF | HCI_LE_Periodic_Advertising_Terminate_Sync command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Periodic_Advertising_Terminate_Sync command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.70 LE Add Device To Periodic Advertiser List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Add_Device_To_Periodic_-Advertiser_List | 0x0047 | Advertiser_Address_Type, Advertiser_Address, Advertising_SID | Status |

**Description:**

This command is used to add an entry, consisting of a single device address and SID, to the Periodic Advertiser list stored in the Controller. Any additions to the Periodic Advertiser list take effect immediately.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The entry is already on the list. | *Invalid HCI Command Parameters* (0x12) |
| MC | An HCI_LE_Periodic_Advertising_Create_Sync is pending. | *Command Disallowed* (0x0C) |
| MC | The entry cannot be added because the list is full. | *Memory Capacity Exceeded* (0x07) |

**Command parameters:**

*Advertiser_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Advertiser_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Advertising SID subfield in the ADI field used to identify the Periodic Advertising |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Add_Device_To_Periodic_Advertiser_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Add_Device_To_Periodic_Advertiser_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Add_Device_To_Periodic_Advertiser_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.71 LE Remove Device From Periodic Advertiser List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_Device_From_-Periodic_Advertiser_List | 0x0048 | Advertiser_Address_Type, Advertiser_Address, Advertising_SID | Status |

**Description:**

This command is used to remove one entry from the list of Periodic Advertisers stored in the Controller. Removals from the Periodic Advertisers List take effect immediately.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | An HCI_LE_Periodic_Advertising_Create_Sync is pending | *Command Disallowed* (0x0C) |
| MC | The entry is not on the list. | *Unknown Advertising Identifier* (0x42) |

**Command parameters:**

*Advertiser_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address or Public Identity Address |
| 0x01 | Random Device Address or Random (static) Identity Address |
| All other values | Reserved for future use |

*Advertiser_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address, Random Device Address, Public Identity Address, or Random (static) Identity Address of the advertiser |

*Advertising_SID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Advertising SID subfield in the ADI field used to identify the Periodic Advertising |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remove_Device_From_Periodic_Advertiser_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Remove_Device_From_Periodic_Advertiser_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_Device_From_Periodic_Advertiser_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.72 LE Clear Periodic Advertiser List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Clear_Periodic_Advertiser_List | 0x0049 | *none* | Status |

**Description:**

This command is used to remove all entries from the list of Periodic Advertisers in the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | An HCI_LE_Periodic_Advertising_Create_Sync is pending. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Clear_Periodic_Advertiser_List command succeeded |
| 0x01 to 0xFF | HCI_LE_Clear_Periodic_Advertiser_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Clear_Periodic_Advertiser_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.73 LE Read Periodic Advertiser List Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Periodic_-Advertiser_List_Size | 0x004A | *none* | Status,<br>Periodic_Advertiser_List_Size |

**Description:**

This command is used to read the number of Periodic Advertiser list entries (including those already stored there) that the Controller can store at the present time.

Note: The number of entries that can be stored is not fixed and the Controller can change it at any time (e.g., because the memory used to store the list can also be used for other purposes).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Periodic_Advertiser_List_Size command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Periodic_Advertiser_List_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Periodic_Advertiser_List_Size:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Total number of Periodic Advertiser list entries that can be stored in the Controller |
| 0x00 | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Periodic_Advertiser_List_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.74 LE Read Transmit Power command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Transmit_Power | 0x004B | *none* | Status,<br>Min_TX_Power,<br>Max_TX_Power |

**Description:**

This command is used to read the minimum and maximum transmit powers supported by the Controller across all supported PHYs.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Transmit_Power command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_Transmit_Power command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Min_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

*Max_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read _Transmit_Power command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.75 LE Read RF Path Compensation command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_RF_Path_-Compensation | 0x004C | *none* | Status,<br>RF_TX_Path_Compensation_Value,<br>RF_RX_Path_Compensation_Value |

**Description:**

This command is used to read the RF path compensation value parameters used in the Tx power level and RSSI calculation.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_RF_Path_Compensation command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_RF_Path_Compensation command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*RF_TX_Path_Compensation_Value:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)<br>Units: 0.1 dB |

*RF_RX_Path_Compensation_Value:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)<br>Units: 0.1 dB |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_RF_Path_Compensation command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.76 LE Write RF Path Compensation command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Write_RF_Path_-Compensation | 0x004D | RF_TX_Path_Compensation_Value, RF_RX_Path_Compensation_Value | Status |

**Description:**

This command is used to indicate the RF path gain or loss between the RF transceiver and the antenna contributed by intermediate components. A positive value means a net RF path gain and a negative value means a net RF path loss. The RF_TX_Path_Compensation_Value parameter shall be used by the Controller to calculate the radiative Tx power level used in HCI commands, HCI events, Advertising physical channel PDUs, and Link Layer Control PDUs using the following equation:

Radiative Tx power level = Tx power level at RF transceiver output + RF_TX_Path_Compensation_Value

For example, if the Tx power level is +4 (dBm) at RF transceiver output and the RF_TX_Path_Compensation_Value is -1.5 (dB), the radiative Tx power level is +4+ (-1.5) = 2.5 (dBm).

The RF_RX_Path_Compensation_Value parameter shall be used by the Controller to calculate the RSSI value reported to the Host using the following equation:

Rx power level at RF transceiver input = Rx power level at antenna + RF_RX_Path_Compensation_Value

For example, if the Rx power level is -45 (dBm) at RF transceiver input and the RF_RX_Path_Compensation_Value is -3.2 (dB), the Rx power level at antenna is -41.8 (dBm).

The default values for the RF path compensation are vendor-specific.

This command can be issued at any time. If this command is issued during an ongoing over-the-air RF activity, the Controller may apply the Tx path compensation immediately or after a vendor-specific delay.

The Controller shall apply a change to the Tx path compensation value either by leaving the power at the transceiver output unchanged and altering the radiative Tx power level or by altering the power at the transceiver output to maintain any previously chosen radiative Tx power level.

If the Host needs to maintain a specific radiative transmit power level for an advertising set, it should disable that set before issuing this command then, after the command

completes, reissue the HCI_LE_Set_Extended_Advertising_Parameters command for that set and then re-enable it.

**Command parameters:**

*RF_TX_Path_Compensation_Value:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)<br>Units: 0.1 dB |

*RF_RX_Path_Compensation_Value:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Range: -128.0 dB (0xFB00) to 128.0 dB (0x0500)<br>Units: 0.1 dB |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Write_RF_Path_Compensation command succeeded |
| 0x01 to 0xFF | HCI_LE_Write_RF_Path_Compensation command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Write_RF_Path_Compensation command has completed, an HCI_Command_Complete event shall be generated.

If the command leads to a change in the local radiative transmit power level for an LE ACL connection, then the Controller shall generate an HCI_LE_Transmit_Power_Reporting event if local reporting is enabled and shall initiate a Link Layer Power Change Indication procedure (see [Vol 6] Part B, Section 5.1.18) if remote reporting is enabled.

### 7.8.77 LE Set Privacy Mode command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Privacy_Mode | 0x004E | Peer_Identity_Address_Type,<br>Peer_Identity_Address,<br>Privacy_Mode | Status |

**Description:**

This command is used to allow the Host to specify the privacy mode to be used for a given entry on the resolving list. The effect of this setting is specified in [Vol 6] Part B, Section 4.7.

When an entry on the resolving list is removed, the mode associated with that entry shall also be removed.

This command shall not be used when address resolution is enabled in the Controller and:

- Advertising (other than periodic advertising) is enabled,
- Scanning is enabled, or
- an HCI_LE_Create_Connection, HCI_LE_Extended_Create_Connection, or HCI_LE_Periodic_Advertising_Create_Sync command is pending.

This command may be used at any time when address resolution is disabled in the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The device is not on the resolving list. | *Unknown Connection Identifier* (0x02) |

### Command parameters:

*Peer_Identity_Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Identity Address |
| 0x01 | Random (static) Identity Address |
| All other values | Reserved for future use |

*Peer_Identity_Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Identity Address or Random (static) Identity Address of the advertiser |

*Privacy_Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use Network Privacy Mode for this peer device (default) |
| 0x01 | Use Device Privacy Mode for this peer device |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Privacy_Mode command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Privacy_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Privacy_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.78 [This section is no longer used]

See Section 7.8.28 for the LE Receiver Test command.

### 7.8.79 [This section is no longer used]

See Section 7.8.29 for the LE Transmitter Test command.

### 7.8.80 LE Set Connectionless CTE Transmit Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Connectionless_CTE_-Transmit_Parameters | 0x0051 | Advertising_Handle,<br>CTE_Length,<br>CTE_Type,<br>CTE_Count,<br>Switching_Pattern_Length ,<br>Antenna_IDs[i] | Status |

**Description:**

This command is used to set the type, length, and antenna switching pattern for the transmission of Constant Tone Extensions in any periodic advertising on the advertising set identified by the Advertising_Handle parameter.

The CTE_Count parameter specifies how many packets with a Constant Tone Extension are to be transmitted in each periodic advertising event. If the number of packets that would otherwise be transmitted is less than this, the Controller shall transmit sufficient AUX_CHAIN_IND PDUs with no AdvData to make up the number. However, if a change in circumstances since this command was issued means that the Controller can no longer schedule all of these packets, it should transmit as many as possible.

The Switching_Pattern_Length and Antenna_IDs[i] parameters are only used when transmitting an AoD Constant Tone Extension and shall be ignored if CTE_Type specifies an AoA Constant Tone Extension.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Constant Tone Extensions have been enabled in the advertising set. | *Command Disallowed* (0x0C) |
| MC | CTE_Length is greater than the maximum length of Constant Tone Extension supported. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | CTE_Type is not supported. | *Unsupported Feature or Parameter Value* (0x11) |

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller is unable to schedule CTE_Count packets in each event. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Advertising_Handle does not identify an existing handle. | *Unknown Advertising Identifier* (0x42) |
| MC | Switching_Pattern_Length is greater than the maximum length supported. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Any of the Antenna_IDs[i] values does not identify an antenna in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x14 | Constant Tone Extension length in 8 µs units |
| All other values | Reserved for future use |

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

*CTE_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Constant Tone Extensions to transmit in each periodic advertising interval<br>Range: 0x01 to 0x10 |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.81 LE Set Connectionless CTE Transmit Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Connectionless_CTE_-Transmit_Enable | 0x0052 | Advertising_Handle, CTE_Enable | Status |

**Description:**

This command is used to request that the Controller enables or disables the use of Constant Tone Extensions in any periodic advertising on the advertising set identified by Advertising_Handle.

In order to start sending periodic advertisements containing a Constant Tone Extension, the Host must also enable periodic advertising using the HCI_LE_Set_Periodic_Advertising_Enable command (see Section 7.8.63).

Note: Periodic advertising can only be enabled when advertising is enabled on the same advertising set, but can continue after advertising has been disabled.

Once enabled, the Controller shall continue advertising with Constant Tone Extensions until either one of the following occurs:

- The Host issues an HCI_LE_Set_Connectionless_CTE_Transmit_Enable command with CTE_Enable set to 0x00 (disabling Constant Tone Extensions but allowing periodic advertising to continue).
- The Host issues an HCI_LE_Set_Periodic_Advertising_Enable command (see Section 7.8.63) with Enable set to 0x00 (disabling periodic advertising). If periodic advertising is re-enabled then it shall continue to contain Constant Tone Extensions.

The Host may issue this command when advertising or periodic advertising is enabled in the advertising set.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The HCI_LE_Set_Periodic_Advertising_Parameters command has not been issued for the advertising set. | *Command Disallowed* (0x0C) |
| MC | The HCI_LE_Set_Connectionless_CTE_Transmit_Parameters command has not been issued for the advertising set. | *Command Disallowed* (0x0C) |
| MC | The periodic advertising is on a PHY that does not allow Constant Tone Extensions. | *Command Disallowed* (0x0C) |

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an existing handle. | *Unknown Advertising Identifier* (0x42) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Identifier for the advertising set in which Constant Tone Extension is being enabled or disabled<br>Range: 0x00 to 0xEF |

*CTE_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Advertising with Constant Tone Extension is disabled (default) |
| 0x01 | Advertising with Constant Tone Extension is enabled |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Connectionless_CTE_Transmit_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Connectionless_CTE_Transmit_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Connectionless_CTE_Transmit_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.82 LE Set Connectionless IQ Sampling Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Connectionless_IQ_-Sampling_Enable | 0x0053 | Sync_Handle,<br>Sampling_Enable,<br>Slot_Durations,<br>Max_Sampled_CTEs,<br>Switching_Pattern_Length,<br>Antenna_IDs[i] | Status,<br>Sync_Handle |

**Description:**

This command is used to request that the Controller enables or disables capturing IQ samples from the Constant Tone Extension of periodic advertising packets in the periodic advertising train identified by the Sync_Handle parameter.

The Max_Sampled_CTEs parameter specifies the maximum number of Constant Tone Extensions in each periodic advertising event that the Controller should collect and report IQ samples from. The Controller should sample all Constant Tone Extensions up to this number.

If the Sampling_Enable parameter is set to 0x01 (sampling is enabled), the Controller starts attempting to capture IQ samples from the periodic advertisements.

Once sampling has been enabled, the Controller shall continue taking IQ samples until the Host issues an HCI_LE_Set_Connectionless_IQ_Enable command with Sampling_Enable set to 0x00 (sampling is disabled) or synchronization with the periodic advertising train is lost.

If Sampling_Enable is set to 0x00, Slot_Durations, Max_Sampled_CTEs, Switching_Pattern_Length, and Antenna_IDs shall be ignored.

The command is also used to set the antenna switching pattern and switching and sampling slot durations to be used while receiving the Constant Tone Extension.

The Slot_Durations, Switching_Pattern_Length, and Antenna_IDs parameters are only used when receiving an AoA Constant Tone Extension and do not affect the reception of an AoD Constant Tone Extension.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Sync_Handle does not identify an existing periodic advertising train. | *Unknown Advertising Identifier* (0x42) |
| MC | Slot_Durations is 0x01 and the Controller does not support 1 µs switching and sampling. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Switching_Pattern_Length is greater than the maximum length of switching pattern supported. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Any of the Antenna_IDs[i] values does not identify an antenna in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Sampling_Enable is 0x01 and the periodic advertising is on a PHY that does not allow Constant Tone Extensions. | *Command Disallowed* (0x0C) |

### Command parameters:

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

*Sampling_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connectionless IQ sampling is disabled (default) |
| 0x01 | Connectionless IQ sampling is enabled |
| All other values | Reserved for future use |

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Switching and sampling slots are 1 µs each |
| 0x02 | Switching and sampling slots are 2 µs each |
| All other values | Reserved for future use |

*Max_Sampled_CTEs:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Sample and report all available Constant Tone Extensions |
| 0xXX | The maximum number of Constant Tone Extensions to sample and report in each periodic advertising interval<br>Range: 0x01 to 0x10 |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Connectionless_IQ_Sampling_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Connectionless_IQ_Sampling_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising.<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Connectionless_IQ_Sampling_Enable command has completed, an HCI_Command_Complete event shall be generated.

HCI_LE_Connectionless_IQ_Report events are generated by the Controller based on the advertising packets received.

### 7.8.83 LE Set Connection CTE Receive Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Connection_CTE_-Receive_Parameters | 0x0054 | Connection_Handle,<br>Sampling_Enable,<br>Slot_Durations,<br>Switching_Pattern_Length,<br>Antenna_IDs[i] | Status,<br>Connection_Handle |

**Description:**

This command is used to enable or disable sampling received Constant Tone Extension fields on the connection identified by the Connection_Handle parameter and to set the antenna switching pattern and switching and sampling slot durations to be used.

If the Sampling_Enable parameter is set to 0x01, the Controller shall sample Constant Tone Extensions on the specified connection and report the samples to the Host. If it is set to 0x00, the Controller shall cease sampling on the specified connection; the remaining parameters shall be ignored.

The Slot_Durations, Switching_Pattern_Length, and Antenna_IDs parameters are only used when receiving an AoA Constant Tone Extension and do not affect the reception of an AoD Constant Tone Extension.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Slot_Durations is 0x01 and the Controller does not support 1 µs switching and sampling. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Switching_Pattern_Length is greater than the maximum length of switching pattern supported. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Any of the Antenna_IDs[i] values does not identify an antenna in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Sampling_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection IQ sampling is disabled (default) |
| 0x01 | Connection IQ sampling is enabled |
| All other values | Reserved for future use |

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Switching and sampling slots are 1 µs each |
| 0x02 | Switching and sampling slots are 2 µs each |
| All other values | Reserved for future use |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Connection_CTE_Receive_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Connection_CTE_Receive_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Connection_CTE_Receive_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.84 LE Set Connection CTE Transmit Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Connection_CTE_-Transmit_Parameters | 0x0055 | Connection_Handle,<br>CTE_Types,<br>Switching_Pattern_Length,<br>Antenna_IDs[i] | Status,<br>Connection_Handle |

**Description:**

This command is used to set the antenna switching pattern and permitted Constant Tone Extension types used for transmitting Constant Tone Extensions requested by the peer device on the connection identified by the Connection_Handle parameter.

The Switching_Pattern_Length and Antenna_IDs[i] parameters are only used when transmitting an AoD Constant Tone Extension and shall be ignored when CTE_Types does not have a bit set for an AoD Constant Tone Extension; they do not affect the transmission of an AoA Constant Tone Extension.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Constant Tone Extension responses have been enabled on the connection. | *Command Disallowed* (0x0C) |
| MC | CTE_Types has a bit set for a type of Constant Tone Extension that the Controller does not support. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Switching_Pattern_Length is greater than the maximum length of switching pattern supported. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Any of the Antenna_IDs[i] values does not identify an antenna in the device's antenna array (some Controllers might be unable to determine which values do or do not identify an antenna). | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*CTE_Types:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Allow AoA Constant Tone Extension Response |
| 1 | Allow AoD Constant Tone Extension Response with 1 µs slots |
| 2 | Allow AoD Constant Tone Extension Response with 2 µs slots |
| All other values | Reserved for future use |

*Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | The number of Antenna IDs in the pattern |
| All other values | Reserved for future use |

*Antenna_IDs[i]:* *Size: Switching_Pattern_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID in the pattern. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Connection_CTE_Transmit_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Connection_CTE_Transmit_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Connection_CTE_Transmit_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.85 LE Connection CTE Request Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Connection_CTE_Request_-Enable | 0x0056 | Connection_Handle,<br>Enable,<br>CTE_Request_Interval,<br>Requested_CTE_Length,<br>Requested_CTE_Type | Status,<br>Connection_Handle |

**Description:**

This command is used to request the Controller to start or stop initiating the Constant Tone Extension Request procedure (see [Vol 6] Part B, Section 5.1.12) on a connection identified by the Connection_Handle parameter.

If Enable is set to 0x00, the remaining parameters shall be ignored. Any Constant Tone Extension Request procedures that have already been initiated are not affected.

The CTE_Request_Interval parameter defines whether the Constant Tone Extension Request procedure is initiated only once or periodically. In the case of periodic operation, the procedure is initiated every CTE_Request_Interval. However, the Controller may delay initiating the procedure beyond the requested interval (e.g., in order to prioritize other activities).

The Requested_CTE_Length parameter indicates the minimum length of the Constant Tone Extension and the Requested_CTE_Type parameter indicates the type of Constant Tone Extension that the Controller shall request from the remote device.

A request is active on a connection from when the Host issues a successful command with Enable set to 0x01 until a command with Enable set to 0x00 has succeeded or, if CTE_Request_Interval was set to zero, until the single Link Layer procedure has been performed, whichever happens first.

If Enable is set to 0x01 and the receiver PHY for the connection changes to a PHY that does not allow Constant Tone Extensions, then the Controller shall automatically disable Constant Tone Extension requests as if the Host had issued this command with Enable set to 0x00.

Note: If the PHY changes back to a PHY that allows Constant Tone Extensions, then the Controller will not automatically re-enable Constant Tone Extension requests.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Controller is aware (e.g. through a previous feature exchange) that the peer's device Link Layer does not support the Connection CTE Response feature. | *Unsupported Remote Feature* (0x1A) |
| RC | The Controller is aware that the peer's device Link Layer does not support the requested CTE type. | *Unsupported Remote Feature* (0x1A) |
| MC | Enable is 0x01 and a request is active for the specified connection. This does not affect the behavior of the Link Layer in respect of the -active request. | *Command Disallowed* (0x0C) |
| MC | The HCI_LE_Set_Connection_CTE_Receive_Parameters command has not been issued for the connection. | *Command Disallowed* (0x0C) |
| MC | The receiver PHY for the connection is not a PHY that allows Constant Tone Extensions. | *Command Disallowed* (0x0C) |
| MC | CTE_Request_Interval has a non-zero value less than *connSubrateFactor* × (*connPeripheralLatency* +1 ). | *Command Disallowed* (0x0C) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable Constant Tone Extension Request for the connection (default) |
| 0x01 | Enable Constant Tone Extension Request for the connection |
| All other values | Reserved for future use |

*CTE_Request_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Initiate the Constant Tone Extension Request procedure once, at the earliest practical opportunity |
| 0x0001 to 0xFFFF | Requested interval for initiating the Constant Tone Extension Request procedure in number of underlying connection events. |

*Requested_CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x14 | Minimum length of the Constant Tone Extension being requested in 8 µs units |
| All other values | Reserved for future use |

*Requested_CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Connection_CTE_Request_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Connection_CTE_Request_Enable command failed. See [Vol 1] Part F for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Connection_CTE_Request_Enable command has completed, an HCI_Command_Complete event shall be generated.

HCI_LE_Connection_IQ_Report events are generated by the Controller based on Constant Tone Extensions received, whether in packets containing an LL_CTE_RSP PDU or otherwise. If a packet is received containing an LL_CTE_RSP PDU but no Constant Tone Extension, or if the peer device rejects the request, an HCI_LE_CTE_Request_Failed event shall be generated.

### 7.8.86 LE Connection CTE Response Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Connection_CTE_Response_-Enable | 0x0057 | Connection_Handle,<br>Enable | Status,<br>Connection_Handle |

**Description:**

This command is used to request the Controller to respond to LL_CTE_REQ PDUs with LL_CTE_RSP PDUs on the specified connection.

If the transmitter PHY for the connection changes to a PHY that does not allow Constant Tone Extensions, then the Controller shall automatically disable Constant Tone Extension responses.

Note: If the PHY changes back to a PHY that allows Constant Tone Extensions, then the Controller will not automatically re-enable Constant Tone Extension responses.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The HCI_LE_Set_Connection_CTE_Transmit_Parameters command has not been issued for the connection. | *Command Disallowed* (0x0C) |
| MC | The transmitter PHY for the connection is not a PHY that allows Constant Tone Extensions. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable Constant Tone Extension Response for the connection (default) |
| 0x01 | Enable Constant Tone Extension Response for the connection |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Connection_CTE_Response_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Connection_CTE_Response_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_LE_Connection_CTE_Response_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.87 LE Read Antenna Information command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Antenna_-Information | 0x0058 | *none* | Status,<br>Supported_Switching_Sampling_Rates,<br>Num_Antennae,<br>Max_Switching_Pattern_Length,<br>Max_CTE_Length |

**Description:**

This command allows the Host to read the switching rates, the sampling rates, the number of antennae, and the maximum length of a transmitted Constant Tone Extension supported by the Controller.

If the Controller does not support antenna switching, the value of Max_Switching_Pattern_Length shall still be valid but will not be used by the Host.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Antenna_Information command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Antenna_Information command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Supported_Switching_Sampling_Rates:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 1 µs switching supported for AoD transmission |
| 1 | 1 µs sampling supported for AoD reception |
| 2 | 1 µs switching and sampling supported for AoA reception |
| 3 to 7 | Reserved for future use |

*Num_Antennae:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x4B | The number of antennae supported by the Controller |
| All other values | Reserved for future use |

*Max_Switching_Pattern_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x4B | Maximum length of antenna switching pattern supported by the Controller |
| All other values | Reserved for future use |

*Max_CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x14 | Maximum length of a transmitted Constant Tone Extension supported in 8 µs units |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Antenna_Information command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.88 LE Set Periodic Advertising Receive Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_-Receive_Enable | 0x0059 | Sync_Handle,<br>Enable | Status |

**Description:**

This command will enable or disable reports for the periodic advertising train identified by the Sync_Handle parameter.

The Enable parameter determines whether reporting and duplicate filtering are enabled or disabled. If the value is the same as the current state, the command has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Sync_Handle does not identify an existing periodic advertising train. | *Unknown Advertising Identifier* (0x42) |
| M | Enable bits 0 and 1 are both set to 1 and the Controller does not support the Periodic Advertising ADI Support feature. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

*Enable:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Reporting enabled |
| 1 | Duplicate filtering enabled |
| All other bits | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Receive_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Receive_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Periodic_Advertising_Receive_Enable command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.89 LE Periodic Advertising Sync Transfer command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Periodic_Advertising_Sync_-Transfer | 0x005A | Connection_Handle,<br>Service_Data,<br>Sync_Handle | Status,<br>Connection_Handle |

**Description:**

This command is used to instruct the Controller to send synchronization information about the periodic advertising train identified by the Sync_Handle parameter to a connected device.

The Service_Data parameter is a value provided by the Host for use by the Host of the peer device. It is not used by the Controller.

The connected device is identified by the Connection_Handle parameter.

This command may successfully complete after the periodic advertising synchronization information is queued for transmission but before it is actually sent. No indication is given as to how the recipient handled the information.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Sync_Handle does not identify an existing periodic advertising train. | *Unknown Advertising Identifier* (0x42) |
| MC | Connection_Handle does not identify a current connection. | *Unknown Connection Identifier* (0x02) |
| MC | The remote device has not indicated support for the Periodic Advertising Sync Transfer – Recipient feature. | *Unsupported Remote Feature* (0x1A) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Service_Data:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | A value provided by the Host |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Periodic_Advertising_Sync_Transfer command succeeded. |
| 0x01 to 0xFF | HCI_LE_Periodic_Advertising_Sync_Transfer command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Periodic_Advertising_Sync_Transfer command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.90 LE Periodic Advertising Set Info Transfer command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Periodic_Advertising_Set_Info_-Transfer | 0x005B | Connection_Handle, Service_Data, Advertising_Handle | Status, Connection_Handle |

**Description:**

This command is used to instruct the Controller to send synchronization information about the periodic advertising in an advertising set to a connected device.

The Advertising_Handle parameter identifies the advertising set. If the parameters in the advertising set have changed since the periodic advertising was first enabled, the current parameters – not the original ones – are sent.

The Service_Data parameter is a value provided by the Host to identify the periodic advertising train to the peer device. It is not used by the Controller.

The connected device is identified by the Connection_Handle parameter.

This command may successfully complete after the periodic advertising synchronization information is queued for transmission but before it is actually sent. No indication is given as to how the recipient handled the information.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Sync_Handle does not identify an existing periodic advertising train. | *Unknown Advertising Identifier* (0x42) |
| MC | Periodic advertising is not in progress for the advertising set. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify a current connection. | *Unknown Connection Identifier* (0x02) |
| MC | The remote device has not indicated support for the Periodic Advertising Sync Transfer – Recipient feature. | *Unsupported Remote Feature* (0x1A) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Service_Data:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | A value provided by the Host |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify an advertising set |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Periodic_Advertising_Set_Info_Transfer command succeeded. |
| 0x01 to 0xFF | HCI_LE_Periodic_Advertising_Set_Info_Transfer command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Periodic_Advertising_Set_Info_Transfer command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.91 LE Set Periodic Advertising Sync Transfer Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_-Sync_Transfer_Parameters | 0x005C | Connection_Handle,<br>Mode,<br>Skip,<br>Sync_Timeout,<br>CTE_Type | Status,<br>Connection_Handle |

**Description:**

This command is used to specify how the Controller will process periodic advertising synchronization information received from the device identified by the Connection_Handle parameter (the "transfer mode").

The Mode parameter specifies the action to be taken when periodic advertising synchronization information is received. If Mode is 0x00, the Controller will ignore the information. Otherwise it will notify the Host and synchronize to the periodic advertising. Mode also specifies whether periodic advertising reports are initially enabled or disabled and whether duplicates are filtered.

The Skip parameter specifies the number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet.

The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.

Irrespective of the value of the Skip parameter, the Controller should stop skipping packets before the Sync_Timeout would be exceeded.

The CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If the periodic advertiser changes the type of the Constant Tone Extension after the Controller has synchronized with the periodic advertising, it shall remain synchronized.

Note: A value of 0 (i.e. all bits clear) indicates that the presence or absence of a Constant Tone Extension is irrelevant.

This command does not affect any processing of any periodic advertising synchronization information already received from the peer device, whether or not the Controller has yet synchronized to the periodic advertising train it describes.

The parameter values provided by this command override those provided via the HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters

command (Section 7.8.92) or any preferences previously set using the HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command on the same connection.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a current connection. | *Unknown Connection Identifier* (0x02) |
| M | Mode is 0x03 and the Controller does not support the Periodic Advertising ADI Support feature. | *Unsupported Feature or Parameter Value* (0x11) |
| RC | All non-reserved bits of CTE_Type are set to 1. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No attempt is made to synchronize to the periodic advertising and no HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. |
| 0x01 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be disabled. |
| 0x02 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be enabled with duplicate filtering disabled. |
| 0x03 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be enabled with duplicate filtering enabled. |
| All other values | Reserved for future use |

*Skip:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of periodic advertising packets that can be skipped after a successful receive<br>Range: 0x0000 to 0x01F3 |

*Sync_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Synchronization timeout for the periodic advertising train<br>Range: 0x000A to 0x4000<br>Time = N × 10 ms<br>Time Range: 100 ms to 163.84 s |

*CTE_Type:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Do not sync to packets with an AoA Constant Tone Extension |
| 1 | Do not sync to packets with an AoD Constant Tone Extension with 1 µs slots |
| 2 | Do not sync to packets with an AoD Constant Tone Extension with 2 µs slots |
| 4 | Do not sync to packets without a Constant Tone Extension |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.92 LE Set Default Periodic Advertising Sync Transfer Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Default_Periodic_-Advertising_Sync_Transfer_Parameters | 0x005D | Mode,<br>Skip,<br>Sync_Timeout,<br>CTE_Type | Status |

**Description:**

This command is used to specify the initial value for the mode, skip, timeout, and Constant Tone Extension type (set by the HCI_LE_Set_Periodic_Advertising_Sync_Transfer_Parameters command; see Section 7.8.91) to be used for all subsequent connections over the LE transport.

The Mode parameter specifies the initial action to be taken. If Mode is 0x00, the Controller will ignore the information. Otherwise it will notify the Host and synchronize to the periodic advertising. Mode also specifies whether periodic advertising reports are initially enabled or disabled and whether duplicates are filtered.

The Skip parameter specifies the number of consecutive periodic advertising packets that the receiver may skip after successfully receiving a periodic advertising packet.

The Sync_Timeout parameter specifies the maximum permitted time between successful receives. If this time is exceeded, synchronization is lost.

The CTE_Type parameter specifies whether to only synchronize to periodic advertising with certain types of Constant Tone Extension. If the periodic advertiser changes the type of the Constant Tone Extension after the Controller has synchronized with the periodic advertising, it shall remain synchronized.

Note: A value of 0 (i.e. all bits clear) indicates that the presence or absence of a Constant Tone Extension is irrelevant.

This command does not affect any existing connection.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| M | Mode is 0x03 and the Controller does not support the Periodic Advertising ADI Support feature. | *Unsupported Feature or Parameter Value* (0x11) |

| Type | Condition | Error code |
|---|---|---|
| RC | All non-reserved bits of CTE_Type are set to 1. | *Command Disallowed* (0x0C) |

### Command parameters:

*Mode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No attempt is made to synchronize to the periodic advertising and no HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host (default). |
| 0x01 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be disabled. |
| 0x02 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be enabled with duplicate filtering disabled. |
| 0x03 | An HCI_LE_Periodic_Advertising_Sync_Transfer_Received event is sent to the Host. HCI_LE_Periodic_Advertising_Report events will be enabled with duplicate filtering enabled. |
| All other values | Reserved for future use |

*Skip:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The number of periodic advertising packets that can be skipped after a successful receive<br>Range: 0x0000 to 0x01F3 |

*Sync_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Synchronization timeout for the periodic advertising train<br>Range: 0x000A to 0x4000<br>Time = N × 10 ms<br>Time Range: 100 ms to 163.84 s |

*CTE_Type:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Do not sync to packets with an AoA Constant Tone Extension |
| 1 | Do not sync to packets with an AoD Constant Tone Extension with 1 µs slots |

| Bit Number | Parameter Description |
|---|---|
| 2 | Do not sync to packets with an AoD Constant Tone Extension with 2 µs slots |
| 4 | Do not sync to packets without a Constant Tone Extension |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Default_Periodic_Advertising_Sync_Transfer_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.93 [This section is no longer used]

See Section 7.8.37 for the LE Generate DHKey command.

### 7.8.94 LE Modify Sleep Clock Accuracy command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Modify_Sleep_Clock_Accuracy | 0x005F | Action | Status |

**Description:**

This command is used to request that the Controller changes its sleep clock accuracy for testing purposes. It should not be used under other circumstances.

The Action parameter specifies whether the sleep clock should be changed to one that is more accurate or one that is less accurate.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Action is 0x00 and the Controller is already using the most accurate clock. | *Limit Reached* (0x43) |
| MC | Action is 0x01 and the Controller is already using the least accurate clock. | *Limit Reached* (0x43) |
| MC | The Controller is unable to switch to a different sleep clock accuracy because some other activity requires the current accuracy. | *Controller Busy* (0x3A) |
| MC | The Controller is unable to switch to a different sleep clock accuracy for any other reason. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Action:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Switch to a more accurate clock |
| 0x01 | Switch to a less accurate clock |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Modify_Sleep_Clock_Accuracy command succeeded |
| 0x01 to 0xFF | HCI_LE_Modify_Sleep_Clock_Accuracy command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Modify_Sleep_Clock_Accuracy command has completed, an HCI_Command_Complete event shall be generated.

The Controller may return the HCI_Command_Complete event indicating success before it has changed the clock accuracy, for example because it still needs to notify peers of the pending change.

### 7.8.95 [This section is no longer used]

### 7.8.96 LE Read ISO TX Sync command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_ISO_TX_Sync | 0x0061 | Connection_Handle | Status,<br>Connection_Handle,<br>Packet_Sequence_Number,<br>TX_Time_Stamp,<br>Time_Offset |

**Description:**

This command is used to read the TX_Time_Stamp and Time_Offset of a transmitted SDU identified by the Packet_Sequence_Number on a CIS or BIS identified by the Connection_Handle parameter on the Central or Peripheral.

The Packet_Sequence_Number parameter contains the sequence number of a transmitted SDU.

The TX_Time_Stamp and Time_Offset parameters are described in [Vol 6] Part G, Section 3.3 and [Vol 6] Part G, Section 3.1 respectively. When the Connection_Handle identifies a CIS or BIS that is transmitting unframed PDUs, the value of Time_Offset returned shall be zero.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The connection handle does not exist or is not associated with a CIS or BIS. | *Unknown Connection Identifier* (0x02) |
| MC | The connection handle is for a CIS or BIS that is not configured for transmitting SDUs. | *Command Disallowed* (0x0C) |
| MC | An SDU has not been transmitted by the Controller. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection handle of the CIS or BIS<br>Range: 0x0000 to 0x0EFF |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Read_ISO_TX_Sync command succeeded. |
| 0x01 to 0xFF | The HCI_LE_Read_ISO_TX_Sync command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection handle of the CIS or BIS<br>Range: 0x0000 to 0x0EFF |

*Packet_Sequence_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0xFFFF | The packet sequence number of an SDU |

*TX_Time_Stamp:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 to 0xFFFFFFFF | The CIG reference point or BIG anchor point of a transmitted SDU derived using the Controller's free running reference clock (in microseconds). |

*Time_Offset:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x000000 to 0xFFFFFF | The time offset, in microseconds, that is associated with a transmitted SDU. |

### Event(s) generated (unless masked away):

When the HCI_LE_Read_ISO_TX_Sync command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.97 LE Set CIG Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_CIG_Parameters | 0x0062 | CIG_ID,<br>SDU_Interval_C_To_P,<br>SDU_Interval_P_To_C,<br>Worst_Case_SCA,<br>Packing,<br>Framing,<br>Max_Transport_Latency_C_To_P,<br>Max_Transport_Latency_P_To_C,<br>CIS_Count,<br>CIS_ID[i],<br>Max_SDU_C_To_P[i],<br>Max_SDU_P_To_C[i],<br>PHY_C_To_P[i],<br>PHY_P_To_C[i],<br>RTN_C_To_P[i]<br>RTN_P_To_C[i] | Status,<br>CIG_ID,<br>CIS_Count,<br>Connection_Handle[i] |

**Description:**

This command is used by a Central's Host to create a CIG and to set the parameters of one or more CISes that are associated with a CIG in the Controller.

The CIG_ID parameter identifies a CIG. This parameter is allocated by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of creating a CIS. If the CIG_ID does not exist, then the Controller shall first create a new CIG. Once the CIG is created (whether through this command or previously), the Controller shall modify or add CIS configurations in the CIG that is identified by the CIG_ID and update all the parameters that apply to the CIG.

The SDU_Interval_C_To_P parameter specifies the time interval between the start of consecutive SDUs from the Central's Host for all the CISes in the CIG. This parameter shall be ignored for all CISes that are unidirectional from Peripheral to Central.

The SDU_Interval_P_To_C parameter specifies the time interval between the start of consecutive SDUs from the Peripheral's Host for all the CISes in the CIG. This parameter shall be ignored for all CISes that are unidirectional from Central to Peripheral.

The Worst_Case_SCA parameter shall be the worst-case sleep clock accuracy of all the Peripherals that will participate in the CIG. The Host should get the sleep clock accuracy from all the Peripherals before issuing this command. If the Host cannot get the sleep clock accuracy from all the Peripherals, it shall set the Worst_Case_SCA parameter to zero.

Note: The Worst_Case_SCA parameter can be used by the Link Layer to better allow for clock drift when scheduling the CISes in the CIG. For example, if a CIS has more than two subevents, the Link Layer of the Central can set the timing of the subevents such that the worst case drift in the Peripheral's clock will not exceed 2 × Sub_Interval. This prevents the Peripheral from synchronizing its timing to the wrong subevent (adjacent subevents cannot be on the same channel).

The Packing parameter indicates the preferred method of arranging subevents of multiple CISes. The subevents can be arranged in Sequential or Interleaved arrangement (see [Vol 6] Part B, Section 4.5.14.2). This is a recommendation to the Controller which the Controller may ignore. This parameter shall be ignored when there is only one CIS in the CIG.

The Framing parameter indicates the format of the CIS Data PDUs of the specified CISes' framing mode (see [Vol 6] Part G, Section 2) that the Host is requesting for the CIG. The Controller may use any framing mode permitted by [Vol 6] Part G, Table 2.1 but shall set the framing mode of all the CISes in the CIG to the same mode. This overrides any framing mode previously set for the CIG.

The Max_Transport_Latency_C_To_P parameter contains the maximum transport latency from the Central to the Peripheral, in milliseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This parameter shall be ignored for all CISes that are unidirectional from Peripheral to Central.

The Max_Transport_Latency_P_To_C parameter contains the maximum transport latency from the Peripheral to the Central, in milliseconds, as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This parameter shall be ignored for all CISes that are unidirectional from Central to Peripheral.

The CIS_Count parameter indicates the number of CIS configurations being modified or added by this command. The Controller shall set the CIS_Count return parameter equal to this.

The CIS_ID[i] parameter identifies a CIS and is set by the Central’s Host and passed to the Peripheral’s Host through the Link Layers during the process of establishing a CIS.

The Max_SDU_C_To_P[i] parameter identifies the maximum size of an SDU from the Central’s Host. If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to 0.

The Max_SDU_P_To_C[i] parameter identifies the maximum size of an SDU from the Peripheral's Host. If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to 0.

The PHY_C_To_P[i] parameter identifies which PHY to use for transmission from the Central to the Peripheral. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits that are set.

The PHY_P_To_C[i] parameter identifies which PHY to use for transmission from the Peripheral to the Central. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits that are set.

The RTN_C_To_P[i] (Retransmission Number) parameter contains the number of times that a CIS Data PDU should be retransmitted from the Central to Peripheral before being acknowledged or flushed (irrespective of which CIS events the retransmission opportunities occur in). If the CIS is unidirectional from Peripheral to Central, this parameter shall be ignored. Otherwise, this parameter is a recommendation to the Controller which the Controller may ignore.

The RTN_P_To_C[i] parameter contains the number of times that a CIS Data PDU should be retransmitted from the Peripheral to Central before being acknowledged or flushed (irrespective of which CIS events the retransmission opportunities occur in). If the CIS is unidirectional from Central to Peripheral, this parameter shall be ignored. Otherwise, this parameter is a recommendation to the Controller which the Controller may ignore.

In each direction, if the Controller satisfies the recommendation, then every PDU will have at least RTN+1 opportunities for transmission (assuming that the initial transmission of that PDU happens at the earliest allowed subevent). The RTN value indicates that the Host is recommending that the Controller selects a combination of CIS parameters that satisfy the inequality:

$$\mathrm{NSE} \times \mathrm{FT} - \lfloor \mathrm{NSE} \div \mathrm{BN} \rfloor \times (\mathrm{BN}-1) \geq \mathrm{RTN} + 1$$

If the Status parameter is non-zero, then the state of the CIG and its CIS configurations shall not be changed by the command. If the CIG did not already exist, it shall not be created.

If Status is zero, then the Controller shall set each Connection_Handle[i] to the connection handle corresponding to the CIS configuration specified in CIS_ID[i]. If the same CIS_ID is being reconfigured, the same connection handle shall be returned.

The connection handle of a CIS shall refer to the CIS when it exists and to the configuration of the CIS stored in a CIG when the CIG exists but the CIS with that CIS_ID does not.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | A CIS configuration that is being modified has a data path set in the Central to Peripheral direction and Max_SDU_C_To_P[i] is zero. | *Command Disallowed* (0x0C) |
| MC | A CIS configuration that is being modified has a data path set in the Peripheral to Central direction and Max_SDU_P_To_C[i] is zero. | *Command Disallowed* (0x0C) |
| MC | The CIG is not in the configurable state. | *Command Disallowed* (0x0C) |
| MC | The resulting configuration would exceed the maximum supported resources in the Controller. | *Memory Capacity Exceeded* (0x07) |
| MC | The resulting configuration would exceed the maximum supported connections in the Controller. | *Connection Limit Exceeded* (0x09) |
| MC | In PHY_C_To_P[i] or PHY_P_To_C[i], a bit is set for a PHY that the Controller does not support, including a bit that is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller does not support asymmetric PHYs and PHY_C_To_P[i] is set to a different value than PHY_P_To_C[i]. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Host specifies an invalid combination of CIS parameters. | *Invalid HCI Command Parameters* (0x12) (recommended) OR *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the CIG. |
| All other values | Reserved for future use |

*SDU_Interval_C_To_P:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use |

*SDU_Interval_P_To_C:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use |

*Worst_Case_SCA:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 251 ppm to 500 ppm |
| 0x01 | 151 ppm to 250 ppm |
| 0x02 | 101 ppm to 150 ppm |
| 0x03 | 76 ppm to 100 ppm |
| 0x04 | 51 ppm to 75 ppm |
| 0x05 | 31 ppm to 50 ppm |
| 0x06 | 21 ppm to 30 ppm |
| 0x07 | 0 ppm to 20 ppm |
| All other values | Reserved for future use |

*Packing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Sequential |
| 0x01 | Interleaved |
| All other values | Reserved for future use |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

*Max_Transport_Latency_C_To_P:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0005 to 0x0FA0 | Maximum transport latency, in milliseconds, from the Central's Controller to the Peripheral's Controller. |
| All other values | Reserved for future use |

*Max_Transport_Latency_P_To_C:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0005 to 0x0FA0 | Maximum transport latency, in milliseconds, from the Peripheral's Controller to the Central's Controller. |
| All other values | Reserved for future use |

*CIS_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | Total number of CIS configurations in the CIG being added or modified. |
| All other values | Reserved for future use |

*CIS_ID[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify a CIS |
| All other values | Reserved for future use |

*Max_SDU_C_To_P[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0FFF | Maximum size, in octets, of the payload from the Central's Host |

*Max_SDU_P_To_C[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0FFF | Maximum size, in octets, of the payload from the Peripheral's Host |

*PHY_C_To_P[i]:* *Size: CIS_Count × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets from the Central is LE 1M |
| 1 | The transmitter PHY of packets from the Central is LE 2M |

| Bit Number | Parameter Description |
|---|---|
| 2 | The transmitter PHY of packets from the Central is LE Coded |
| All other bits | Reserved for future use |

*PHY_P_To_C[i]:* *Size: CIS_Count × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets from the Peripheral is LE 1M |
| 1 | The transmitter PHY of packets from the Peripheral is LE 2M |
| 2 | The transmitter PHY of packets from the Peripheral is LE Coded |
| All other bits | Reserved for future use |

*RTN_C_To_P[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of times every CIS Data PDU should be retransmitted from the Central to the Peripheral |

*RTN_P_To_C[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of times every CIS Data PDU should be retransmitted from the Peripheral to the Central |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Set_CIG_Parameters.command succeeded |
| 0x01 to 0xFF | The HCI_LE_Set_CIG_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify a CIG |
| All other values | Reserved for future use |

*CIS_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | Total number of CIS configurations added or modified by this command |
| All other values | Reserved for future use |

*Connection_Handle[i]:* *Size: CIS_Count × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS in the CIG.<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_CIG_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.98 LE Set CIG Parameters Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_CIG_Parameters_Test | 0x0063 | CIG_ID,<br>SDU_Interval_C_To_P,<br>SDU_Interval_P_To_C,<br>FT_C_To_P,<br>FT_P_To_C,<br>ISO_Interval,<br>Worst_Case_SCA,<br>Packing,<br>Framing,<br>CIS_Count,<br>CIS_ID[i],<br>NSE[i],<br>Max_SDU_C_To_P,[i],<br>Max_SDU_P_To_C[i],<br>Max_PDU_C_To_P,[i],<br>Max_PDU_P_To_C[i],<br>PHY_C_To_P,[i],<br>PHY_P_To_C[i],<br>BN_C_To_P,[i],<br>BN_P_To_C[i] | Status,<br>CIG_ID,<br>CIS_Count,<br>Connection_Handle[i] |

**Description:**

This command should only be used for testing purposes.

The command is used by a Central's Host to create a CIG and to set the parameters of one or more CISes that are associated with a CIG in the Controller.

The CIG_ID parameter identifies a CIG. This parameter is allocated by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of creating a CIS. If the CIG_ID does not exist, then the Controller shall first create a new CIG. Once the CIG is created (whether through this command or previously), the Controller shall modify or add CIS configurations in the CIG that is identified by the CIG_ID and update all the parameters that apply to the CIG.

The SDU_Interval_C_To_P parameter specifies the time interval of periodic SDUs from the Central's Host.

The SDU_Interval_P_To_C parameter specifies the time interval of periodic SDUs from the Peripheral's Host.

The FT_C_To_P parameter identifies the maximum time for a payload from the Central to Peripheral to be transmitted and re-transmitted, after which it is flushed (see [Vol 6] Part B, Section 4.5.13.5). This parameter is expressed in multiples of ISO_Interval.

The FT_P_To_C parameter identifies the maximum time for a payload from the Peripheral to Central to be transmitted and re-transmitted, after which it is flushed (see [Vol 6] Part B, Section 4.5.13.5). This parameter is expressed in multiples of ISO_Interval.

The ISO_Interval parameter specifies the time between two consecutive CIS anchor points.

The CIS_Count parameter contains the number of CIS configurations being added or modified by this command. The Controller shall set the CIS_Count return parameter equal to this.

The CIS_ID[i] parameter identifies the CIS and is set by the Central's Host and passed to the Peripheral's Host through the Link Layers during the process of establishing a CIS.

The Worst_Case_SCA parameter is the worst-case sleep clock accuracy of all the Peripherals that will participate in the CIG. The Host should get the sleep clock accuracy from all the Peripherals before issuing this command. In case the Host cannot get the sleep clock accuracy from all the Peripherals, it shall set the Worst_Case_SCA parameter to zero.

Note: The Worst_Case_SCA parameter can be used by the Link Layer to better allow for clock drift when scheduling the CISes in the CIG. For example, if a CIS has more than two subevents, the Link Layer of the Central can set the timing of the subevents such that the worst case drift in the Peripheral's clock will not exceed 2 × Sub_Interval. This prevents the Peripheral from synchronizing its timing to the wrong subevent (adjacent subevents cannot be on the same channel).

The Packing parameter is used to indicate the method of arranging subevents of multiple CISes. The subevents can be arranged in Sequential or Interleaved arrangement. This parameter shall be ignored when there is only one CIS in the CIG.

The Framing parameter specifies the framing mode (see [Vol 6] Part G, Section 2) that the Controller shall use for all the CISes in the CIG. This overrides any framing mode previously set for the CIG.

The NSE[i] parameter identifies the maximum number of subevents for each CIS in a CIG event.

The Max_SDU_C_To_P[i] parameter identifies the maximum size of SDU from the Central's Host. If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to 0. If a CIS configuration that is being modified has a data path set in the Central to Peripheral direction and the Host has specified that Max_SDU_C_To_P[i] shall be set to zero, the Controller shall return the error code *Command Disallowed* (0x0C). The minimum value of the Max_SDU_Size parameter in the ISO Transmit Test mode when the Payload_Type = 1 or 2 shall be 4 octets.

The Max_SDU_P_To_C[i] parameter identifies the maximum size of SDU from the Peripheral's Host. If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to 0. If a CIS configuration that is being modified has a data path set in the Peripheral to Central direction and the Host has specified that Max_SDU_P_To_C[i] shall be set to zero, the Controller shall return the error code *Command Disallowed* (0x0C).The minimum value of the Max_SDU parameter in the ISO Transmit Test mode when the Payload_Type = 1 or 2 shall be 4 octets.

The Max_PDU_C_To_P[i] parameter identifies the maximum size PDU from the Central to Peripheral.

The Max_PDU_P_To_C[i] parameter identifies the maximum size PDU from the Peripheral to Central.

The PHY_C_To_P[i] parameter identifies the PHY to be used for transmission of packets from the Central to the Peripheral. The Host shall set only one bit in this parameter and the Controller shall use the PHY set by the Host.

The PHY_P_To_C[i] parameter identifies the PHY to be used for transmission of packets from the Peripheral to the Central. The Host shall set only one bit in this parameter and the Controller shall use the PHY set by the Host.

The BN_C_To_P[i] parameter identifies the burst number for Central to Peripheral (see [Vol 6] Part B, Section 4.5.13). If the CIS is unidirectional from Peripheral to Central, this parameter shall be set to zero.

The BN_P_To_C[i] parameter identifies the burst number for Peripheral to Central (see [Vol 6] Part B, Section 4.5.13). If the CIS is unidirectional from Central to Peripheral, this parameter shall be set to zero.

If the Status parameter is non-zero, then the state of the CIG and its CIS configurations shall not be changed by the command. If the CIG did not already exist, it shall not be created.

If Status is zero, then the Controller shall set Connection_Handle[i] to the connection handle corresponding to the CIS configuration specified in CIS_ID[i]. If the same CIS_ID is being reconfigured, the same connection handle shall be returned.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The CIG is not in the configurable state. | *Command Disallowed* (0x0C) |
| MC | The resulting configuration would exceed the maximum supported resources in the Controller. | *Memory Capacity Exceeded* (0x07) |
| MC | The resulting configuration would exceed the maximum supported connections in the Controller. | *Connection Limit Exceeded* (0x09) |
| MC | In PHY_C_To_P[i] or PHY_P_To_C[i], a bit is set for a PHY that the Controller does not support, including a bit that is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller does not support asymmetric PHYs and PHY_C_To_P[i] is set to a different value than PHY_P_To_C[i]. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Host specifies an invalid combination of CIS parameters. | *Invalid HCI Command Parameters* (0x12) (recommended) OR *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the CIG. |
| All other values | Reserved for future use. |

*SDU_Interval_C_To_P:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use. |

*SDU_Interval_P_To_C:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use. |

*FT_C_To_P:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The flush timeout in multiples of ISO_Interval for each payload sent from the Central to Peripheral.<br>Range: 0x01 to 0xFF |

*FT_P_To_C:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The flush timeout in multiples of ISO_Interval for each payload sent from the Peripheral to Central.<br>Range: 0x01 to 0xFF |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Time between consecutive CIS anchor points.<br>Range: 0x0004 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 5 ms to 4 s |

*Worst_Case_SCA:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | 251 ppm to 500 ppm |
| 0x01 | 151 ppm to 250 ppm |
| 0x02 | 101 ppm to 150 ppm |
| 0x03 | 76 ppm to 100 ppm |
| 0x04 | 51 ppm to 75 ppm |
| 0x05 | 31 ppm to 50 ppm |
| 0x06 | 21 ppm to 30 ppm |

| Value | Parameter Description |
|---|---|
| 0x07 | 0 ppm to 20 ppm |
| All other values | Reserved for future use |

*Packing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Sequential |
| 0x01 | Interleaved |
| All other values | Reserved for future use |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

*CIS_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | Total number of CIS configurations in the CIG being added or modified. |
| All other values | Reserved for future use |

*CIS_ID[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify a CIS |
| All other values | Reserved for future use |

*NSE[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Maximum number of subevents in each CIS event |
| All other values | Reserved for future use |

*Max_SDU_C_To_P[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0FFF | Maximum size, in octets, of the payload from the Central's Host |
| All other values | Reserved for future use |

*Max_SDU_P_To_C[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0FFF | Maximum size, in octets, of the payload from the Peripheral's Host |
| All other values | Reserved for future use |

*Max_PDU_C_To_P[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x00FB | Maximum size, in octets, of the payload from the Central's Link Layer to the Peripheral's Link Layer. |
| All other values | Reserved for future use |

*Max_PDU_P_To_C[i]:* *Size: CIS_Count × 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x00FB | Maximum size, in octets, of the payload from the Peripheral's Link Layer to the Central's Link Layer. |
| All other values | Reserved for future use |

*PHY_C_To_P[i]:* *Size: CIS_Count × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets from the Central is LE 1M |
| 1 | The transmitter PHY of packets from the Central is LE 2M |
| 2 | The transmitter PHY of packets from the Central is LE Coded |
| All other bits | Reserved for future use |

*PHY_P_To_C[i]:* *Size: CIS_Count × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets from the Peripheral is LE 1M |
| 1 | The transmitter PHY of packets from the Peripheral is LE 2M |

| Bit Number | Parameter Description |
|---|---|
| 2 | The transmitter PHY of packets from the Peripheral is LE Coded |
| All other bits | Reserved for future use |

*BN_C_To_P[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No isochronous data from the Central to the Peripheral |
| 0x01 to 0x0F | The burst number for Central to Peripheral transmission |
| All other values | Reserved for future use |

*BN_P_To_C[i]:* *Size: CIS_Count × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No isochronous data from the Peripheral to the Central |
| 0x01 to 0x0F | The burst number for Peripheral to Central transmission |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Set_CIG_Parameters_Test command succeeded |
| 0x01 to 0xFF | The HCI_LE_Set_CIG_Parameters_Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify a CIG |
| All other values | Reserved for future use |

*CIS_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1F | Total number of CIS configurations in the CIG being added or modified. |
| All other values | Reserved for future use |

*Connection_Handle[i]:* *Size: CIS_Count × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS in the CIG.<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_CIG_Parameters_Test command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.99 LE Create CIS command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Create_CIS | 0x0064 | CIS_Count,<br>CIS_Connection_Handle[i],<br>ACL_Connection_Handle[i] | *none* |

**Description:**

This command is used by the Central's Host to create one or more CISes using the connections identified by the ACL_Connection_Handle arrayed parameter.

The CIS_Count parameter is the total number of CISes created by this command.

The CIS_Connection_Handle[i] parameter specifies the connection handle corresponding to the configuration of the CIS to be created and whose configuration is already stored in a CIG.

The ACL_Connection_Handle[i] parameter specifies the connection handle of the ACL connection associated with each CIS to be created. The list of the ACL_Connection_Handles shall be in the same order as the list of the CIS_Connection_Handles e.g., CIS_Connection_Handle[1] will connect to the Peripheral associated with the ACL_Connection_Handle[1].

If this command is issued on the Central before the devices have performed the Feature Exchange procedure, then the Controller shall complete that procedure before initiating the Connected Isochronous Stream Creation procedure (see [Vol 6] Part B, Section 5.1.15).

Note: The order of the CIS connection handles in this command does not relate to the order of connection handles in the return parameters of the HCI_LE_Set_CIG_Parameters command or the HCI_LE_Set_CIG_Parameters_Test command.

Note: If an error is reported in the HCI_Command_Status event, it means that no CIS is created and the Host cannot determine which CIS had the error. Therefore, in the case of an error that only affects one CIS of several, reporting the error in the HCI_LE_CIS_Established event for that CIS means that the remaining CISes are still created and the Host can determine which CIS had the error.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | An ACL_Connection_Handle[i] does not identify an existing ACL connection. | *Unknown Connection Identifier* (0x02) |
| MC | A CIS_Connection_Handle[i] does not identify a CIS or CIS configuration. | *Unknown Connection Identifier* (0x02) |
| MC | A CIS has already been created. | *Connection Already Exists* (0x0B) |
| MC | Two different CIS_Connection_Handle[i] identify the same CIS. | *Invalid HCI Command Parameters* (0x12) |
| MC | Not all of the HCI_LE_CIS_Established events from the previous use of the command have been generated. | *Command Disallowed* (0x0C) |
| MC | An ACL_Connection_Handle[i] identifies a connection where the Controller is the Peripheral. | *Command Disallowed* (0x0C) |
| MC | The Connected Isochronous Stream (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.27) is not set. | *Command Disallowed* (0x0C) |
| M | An invalid combination of parameters was specified in the HCI_LE_Set_CIG_Parameters or HCI_LE_Set_CIG_Parameters_Test command that created the CIS configuration, but the Controller could not detect the problem without knowing the properties of the ACL connection associated with the CIS. | *Unsupported Feature or Parameter Value* (0x11) |

### Command parameters:

*CIS_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of CISes to be created. |
| All other values | Reserved for future use |

*CIS_Connection_Handle[i]:* *Size: CIS_Count × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS<br>Range: 0x0000 to 0x0EFF |

*ACL_Connection_Handle[i]:* *Size: CIS_Count × 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of an ACL connection<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Create_CIS command, the Controller sends the HCI_Command_Status event to the Host. An HCI_LE_CIS_Established event will be generated for each CIS when it is established or if it is disconnected or considered lost before being established; until all the events are generated, the command remains pending.

### 7.8.100 LE Remove CIG command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_CIG | 0x0065 | CIG_ID | Status,<br>CIG_ID |

**Description:**

This command is used by the Central's Host to remove the CIG identified by CIG_ID.

The CIG_ID parameter contains the identifier of the CIG.

This command shall delete the CIG_ID and also delete the Connection_Handles of the CIS configurations stored in the CIG.

This command shall also remove the isochronous data paths that are associated with the Connection_Handles of the CIS configurations, which is equivalent to issuing the HCI_LE_Remove_ISO_Data_Path command (see Section 7.8.110).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The CIG is in the active state. | *Command Disallowed* (0x0C) |
| MC | The CIG_ID does not exist. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifier of a CIG |
| All other values | Reserved for future use |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Remove_CIG command succeeded |
| 0x01 to 0xFF | The HCI LE Remove_CIG command failed. See [Vol 1] Part F, Controller Error Codes for a complete list of error codes and descriptions. |

*CIG_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifier of a CIG |
| All other values | Reserved for future use |

### Event(s) generated (unless masked away):

When the HCI_LE_Remove_CIG command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.101 LE Accept CIS Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Accept_CIS_Request | 0x0066 | Connection_Handle | *none* |

**Description:**

This command is used by the Peripheral's Host to inform the Controller to accept the request for the CIS that is identified by the Connection_Handle.

The command shall only be issued after an HCI_LE_CIS_Request event has occurred. The event contains the Connection_Handle of the CIS.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a CIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a connection that has already been established or already has an HCI_LE_Accept_CIS_Request or HCI_LE_Reject_CIS_Request command in progress. | *Command Disallowed* (0x0C) |
| MC | The Host is a Central. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS<br>Range: 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Accept_CIS_Request command, the Controller sends the HCI_Command_Status event to the Host. An

HCI_LE_CIS_Established event will be generated when the CIS is established or is considered lost before being established.

### 7.8.102 LE Reject CIS Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Reject_CIS_Request | 0x0067 | Connection_Handle,<br>Reason | Status,<br>Connection_Handle |

**Description:**

This command is used by the Peripheral's Host to inform the Controller to reject the request for the CIS that is identified by the Connection_Handle.

The command shall only be issued after an HCI_LE_CIS_Request event has occurred. The event contains the Connection_Handle of the CIS.

When this command succeeds, the Controller shall delete the Connection_Handle of the requested CIS.

The Reason parameter indicates the reason for rejecting the CIS request.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a CIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a connection that has already been established or already has an HCI_LE_Accept_CIS_Request or HCI_LE_Reject_CIS_Request command in progress. | *Command Disallowed* (0x0C) |
| MC | The Host is a Central. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS to be rejected<br>Range: 0x0000 to 0x0EFF |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason the CIS request was rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Reject_CIS_Request command succeeded. |
| 0x01 to 0xFF | The HCI_LE_Reject_CIS_Request command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The connection handle of the CIS to be rejected<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Reject_CIS_Request command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.103 LE Create BIG command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Create_BIG | 0x0068 | BIG_Handle,<br>Advertising_Handle,<br>Num_BIS,<br>SDU_Interval,<br>Max_SDU,<br>Max_Transport_Latency,<br>RTN,<br>PHY,<br>Packing,<br>Framing,<br>Encryption,<br>Broadcast_Code | *none* |

**Description:**

This command is used to create a BIG with one or more BISes (see [Vol 6] Part B, Section 4.4.6). All BISes in a BIG have the same value for all parameters.

The BIG_Handle contains the identifier of the BIG. This parameter is allocated by the Host and used by the Controller and the Host to identify a BIG.

The Advertising_Handle identifies the associated periodic advertising train of the BIG (see [Vol 6] Part B, Section 4.4.5.1).

The Num_BIS parameter contains the total number of BISes in the BIG.

The SDU_Interval parameter contains the time interval of the periodic SDUs.

The Max_SDU parameter contains the maximum size of an SDU.

The Max_Transport_Latency parameter is the maximum transport latency (in milliseconds) as described in [Vol 6] Part G, Section 3.2.1 and [Vol 6] Part G, Section 3.2.2. This includes pre-transmissions.

The RTN (Retransmission Number) parameter contains the number of times every PDU should be retransmitted, irrespective of which BIG events the retransmissions occur in. This is a recommendation to the Controller which the Controller may ignore.

The PHY parameter is a bit field that indicates the PHY used for transmission of PDUs of BISes in the BIG. The Host shall set at least one bit in this parameter and the Controller shall pick a PHY from the bits set.

The Packing parameter is used to indicate the preferred method of arranging subevents of multiple BISes. The subevents can be arranged in Sequential or Interleaved arrangement. This is a recommendation to the Controller which it may ignore. This parameter shall be ignored when there is only one BIS in the BIG.

The Framing parameter indicates the format of the BIS Data PDUs and the mode of framed PDUs that the Host is requesting the Controller to use on the specified BIG. The Controller may use any combination of format and mode permitted by [Vol 6] Part G, Table 2.1

The Encryption parameter identifies the encryption mode of the BISes. If the Encryption parameter is set to 1 (encrypted), then the Broadcast_Code is used in the encryption of payloads (see [Vol 6] Part B, Section 4.4.6.10).

The Broadcast_Code parameter is used to generate the encryption key for encrypting payloads of all BISes. When the Encryption parameter is set to 0 (unencrypted), the Broadcast_Code parameter shall be set to zero by the Host and ignored by the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The PHY parameter contains a bit for a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller cannot create all BISes of the BIG or Num_BIS exceeds the maximum value supported by the Controller. | *Rejected due to Limited Resources* (0x0D) |
| MC | Advertising_Handle does not identify a periodic advertising train, identifies a periodic advertising train associated with another BIG, or identifies a periodic advertising train that has responses and the Controller does not support PAwR. | *Unknown Advertising Identifier* (0x42) |
| MC | BIG_Handle identifies a BIG that is already created. | *Command Disallowed* (0x0C) |
| M | The BIG parameters are incompatible. | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| M | The length of the associated periodic advertising, with the BIGInfo added to the ACAD, is greater than the maximum that the Controller can transmit within the periodic advertising interval (if periodic advertising is enabled) or the Periodic_Advertising_Interval_Max for the advertising set (if disabled). If advertising on the LE Coded PHY, then the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |

**Command parameters:**

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the BIG. |
| All other values | Reserved for future use |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the periodic advertising train. |
| All other values | Reserved for future use |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of BISes in the BIG. |
| All other values | Reserved for future use |

*SDU_Interval:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use |

*Max_SDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x0FFF | Maximum size of an SDU, in octets. |
| All other values | Reserved for future use |

*Max_Transport_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0005 to 0x0FA0 | Maximum transport latency, in milliseconds. |
| All other values | Reserved for future use |

*RTN:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x1E | The number of times that every BIS Data PDU should be retransmitted. |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets is LE 1M. |
| 1 | The transmitter PHY of packets is LE 2M. |
| 2 | The transmitter PHY of packets is LE Coded. |
| All other bits | Reserved for future use |

*Packing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Sequential |
| 0x01 | Interleaved |
| All other values | Reserved for future use |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

*Encryption:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unencrypted |
| 0x01 | Encrypted |
| All other values | Reserved for future use |

*Broadcast_Code:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | The code used to derive the session key that is used to encrypt and decrypt BIS payloads. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Create_BIG command, the Controller sends the HCI_Command_Status event to the Host. When the HCI_LE_Create_BIG command has completed, the HCI_LE_Create_BIG_Complete event is generated.

### 7.8.104 LE Create BIG Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Create_BIG_Test | 0x0069 | BIG_Handle,<br>Advertising_Handle,<br>Num_BIS,<br>SDU_Interval,<br>ISO_Interval,<br>NSE,<br>Max_SDU,<br>Max_PDU,<br>PHY,<br>Packing,<br>Framing,<br>BN,<br>IRC,<br>PTO,<br>Encryption,<br>Broadcast_Code | *none* |

**Description:**

This command should only be used for testing purposes.

The command is used to create one or more BISes of a BIG (see [Vol 6] Part B, Section 4.4.6). All BISes in the BIG have the same values for all parameters.

The BIG_Handle contains the identifier of the BIG. This parameter is allocated by the Host and used by the Controller and the Host to identify a BIG.

The Advertising_Handle identifies the associated periodic advertising train of the BIG.

The Num_BIS parameter contains the total number of BISes in the BIG.

The SDU_Interval parameter specifies the time interval of the periodic SDUs.

The ISO_Interval parameter contains the time duration between two consecutive BIG anchor points.

The NSE (Number of SubEvents) parameter contains the total number of subevents that are used to transmit BIS Data PDUs for each BIS in a BIG event. The NSE parameter shall be greater than or equal to IRC × BN.

The Max_SDU parameter contains the maximum size, in octets, of an SDU. The minimum value of the Max_SDU parameter in the ISO Transmit Test mode when the Payload_Type = 1 or 2 shall be 4.

The Max_PDU parameter contains the maximum size of every BIS Data PDU for every BIS in the BIG.

The PHY parameter is a bit field that indicates the PHY used for transmission of PDUs of BISes in the BIG. The Host shall set only one bit in this parameter and the Controller shall use the PHY set by the Host.

The Packing parameter indicates the method of arranging subevents of multiple BISes. The subevents can be arranged in Sequential or Interleaved arrangement. This parameter shall be ignored when there is only one BIS in the BIG.

The Framing parameter indicates the format of the BIS Data PDUs and the mode of framed PDUs (see [Vol 6] Part G, Section 2) that the Controller shall use on the specified BIG.

The BN (Burst Number) parameter contains the number of new payloads for each BIS in a BIS event.

The IRC (Immediate Repetition Count) parameter contains the number of times the scheduled data packet is transmitted (see [Vol 6] Part B, Section 4.4.6). The IRC parameter shall be an integer in the range 1 to (NSE ÷ BN).

The PTO (Pre_Transmission_Offset) parameter contains the offset in number of ISO_Intervals for pre transmissions of data packets (see [Vol 6] Part B, Section 4.4.6).

The Encryption parameter identifies the encryption mode of the BISes in the BIG. If the Encryption parameter is set to 1 (encrypted), the Broadcast_Code is used in the encryption of payloads (see [Vol 6] Part B, Section 4.4.6).

The Broadcast_Code parameter is a 16-octet field that is used to generate the session key to encrypt payloads of all BISes in the BIG. When the Encryption parameter is set to 0 (unencrypted), all 16 octets of the Broadcast_Code parameter shall be set to zero by the Host and ignored by the Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The PHY parameter contains a bit for a PHY that the Controller does not support or is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller cannot create all BISes of the BIG or Num_BIS exceeds the maximum value supported by the Controller. | *Rejected due to Limited Resources* (0x0D) |
| MC | Advertising_Handle does not identify a periodic advertising train, identifies a periodic advertising train associated with another BIG, or identifies a periodic advertising train that has responses and the Controller does not support PAwR. | *Unknown Advertising Identifier* (0x42) |
| MC | BIG_Handle identifies a BIG that is already created. | *Command Disallowed* (0x0C) |
| MC | Max_PDU, NSE, BN, IRC, or PTO exceeds the value supported by the Controller. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | NSE is not an integer multiple of BN or is less than IRC × BN. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Any parameter is not in the specified range. | *Unsupported Feature or Parameter Value* (0x11) |
| M | The BIG parameters are incompatible in a way not covered by one of the previous two items. | *Invalid HCI Command Parameters* (0x12) |
| M | The length of the associated periodic advertising, with the BIGInfo added to the ACAD, is greater than the maximum that the Controller can transmit within the periodic advertising interval. If advertising on the LE Coded PHY, then the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |

### Command parameters:

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the BIG |
| All other values | Reserved for future use |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the periodic advertising train |
| All other values | Reserved for future use |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of BISes in the BIG |
| All other values | Reserved for future use |

*SDU_Interval:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0x0000FF to 0x0FFFFF | The interval, in microseconds, of periodic SDUs. |
| All other values | Reserved for future use |

*ISO_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The time between consecutive BIG anchor points.<br>Range: 0x0004 to 0x0C80<br>Time = N × 1.25 ms<br>Time Range: 5 ms to 4 s |

*NSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | The total number of subevents in each interval of each BIS in the BIG. |
| All other values | Reserved for future use |

*Max_SDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x0FFF | Maximum size, in octets, of an SDU |
| All other values | Reserved for future use |

*Max_PDU:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 to 0x00FB | Maximum size, in octets, of payload |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | The transmitter PHY of packets is LE 1M. |
| 1 | The transmitter PHY of packets is LE 2M. |
| 2 | The transmitter PHY of packets is LE Coded. |
| All other bits | Reserved for future use |

*Packing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Sequential |
| 0x01 | Interleaved |
| All other values | Reserved for future use |

*Framing:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unframed PDUs |
| 0x01 | Framed PDUs, Segmentable mode |
| 0x02 | Framed PDUs, Unsegmented mode |
| All other values | Reserved for future use |

*BN:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x07 | The number of new payloads in each interval for each BIS. |
| All other values | Reserved for future use |

*IRC:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0F | The number of times the scheduled payloads are transmitted in a given event. |
| All other values | Reserved for future use |

*PTO:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x0F | Offset used for pre-transmissions |
| All other values | Reserved for future use |

*Encryption:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unencrypted |
| 0x01 | Encrypted |
| All other values | Reserved for future use |

*Broadcast_Code:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | The code used to derive the session key that is used to encrypt and decrypt BIS payloads. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Create_BIG_Test command, the Controller sends the HCI_Command_Status event to the Host. When the HCI_LE_Create_BIG_Test command has completed, the HCI_LE_Create_BIG_Complete event is generated.

### 7.8.105 LE Terminate BIG command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Terminate_BIG | 0x006A | BIG_Handle,<br>Reason | *none* |

**Description:**

This command is used to terminate a BIG identified by the BIG_Handle parameter. The command also terminates the transmission of all BISes of the BIG, destroys the associated connection handles of the BISes in the BIG and removes the data paths for all BISes in the BIG.

The Reason parameter is used to indicate the reason why the BIG is to be terminated.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | BIG_Handle does not identify a BIG. | *Unknown Advertising Identifier* (0x42) |
| MC | The Controller is not the Isochronous Broadcaster for the BIG identified by BIG_Handle. | *Command Disallowed* (0x0C) |

**Command parameters:**

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the BIG. |
| All other values | Reserved for future use |

*Reason:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason the BIG is terminated. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Terminate_BIG command, the Controller sends the HCI_Command_Status event to the Host.

When the HCI_LE_Terminate_BIG command has completed, the HCI_LE_Terminate_BIG_Complete event will be generated.

If the Host attempts to terminate a BIG while the process of establishment of the BIG is in progress (i.e. HCI_LE_Create_BIG_Complete event has not been generated) the process of establishment shall stop and the Controller shall generate the HCI_LE_-Create_BIG_Complete event to the Host with the error code *Operation Cancelled by Host* (0x44).

### 7.8.106 LE BIG Create Sync command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_BIG_Create_Sync | 0x006B | BIG_Handle,<br>Sync_Handle,<br>Encryption,<br>Broadcast_Code,<br>MSE,<br>BIG_Sync_Timeout,<br>Num_BIS,<br>BIS[i] | *none* |

**Description:**

This command is used to synchronize to a BIG described in the periodic advertising train specified by the Sync_Handle parameter.

The BIG_Handle parameter is assigned by the Host to identify the synchronized BIG.

The Encryption parameter indicates whether the Broadcast_Code parameter is valid.

The Broadcast_Code parameter is a 16-octet field that is used to generate the session key to encrypt or decrypt payloads of an encrypted BIS. Broadcast_Code shall be ignored by the Controller if Encryption is set to 0x00.

The MSE (Maximum Subevents) parameter is the maximum number of subevents that a Controller should listen to for data payloads in each interval for a BIS. The Controller may select any of the subevents to listen to.

The BIG_Sync_Timeout parameter specifies the maximum permitted time between successful receptions of BIS PDUs. If this time is exceeded, synchronization is lost. When the Controller establishes synchronization and if the BIG_Sync_Timeout set by the Host is less than 6 × ISO_Interval, the Controller shall set the timeout to 6 × ISO_Interval.

The Num_BIS parameter contains the number of BISes specified in the BIS arrayed parameter. The number of BISes requested may be less than the number of BISes in the BIG.

The BIS arrayed parameter is a list of BIS_Numbers corresponding to BIS(es) in the synchronized BIG. The list of BIS_Numbers shall be in ascending order and shall not contain any duplicates.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Encryption is set to 0x00 for an encrypted BIG or is set to 0x01 for an unencrypted BIG. | *Encryption Mode Not Acceptable* (0x25) |
| MC | Sync_Handle does not exist. | *Unknown Advertising Identifier* (0x42) |
| MC | BIG_Handle identifies a BIG. | *Command Disallowed* (0x0C) |
| MC | The information describing the BIG does not specify a PHY supported by the Controller or does not specify exactly one PHY. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Num_BIS is greater than the total number of BISes in the BIG. | *Unsupported Feature or Parameter Value* (0x11) |
| RC | MSE is less than the BN value for the BIS. | *Invalid HCI Command Parameters* (0x12) |
| MC | The Controller is in the process of synchronizing to any BIG and the HCI_LE_BIG_Sync_Established event has not been generated. | *Command Disallowed* (0x0C) |
| MC | The Controller is unable to receive PDUs from the specified number of BISes in the synchronized BIG. | *Rejected Due To Limited Resources* (0x0D). |
| M | The Controller is already synchronized to the BIG specified by Sync_Handle. | *Command Disallowed* (0x0C) |

**Command parameters:**

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Used to identify the BIG |
| All other values | Reserved for future use |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Identifier of the periodic advertising train<br>Range: 0x0000 to 0x0EFF |

*Encryption:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Broadcast_Code invalid |
| 0x01 | Broadcast_Code valid |
| All other values | Reserved for future use |

*Broadcast_Code:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 128-bit code used for deriving the session key for decrypting payloads of BISes in the BIG. |

*MSE:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Controller can schedule reception of any number of subevents up to NSE. |
| 0x01 to 0x1F | Maximum number of subevents that should be used to receive data payloads in each BIS event |
| All other values | Reserved for future use |

*BIG_Sync_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Synchronization timeout for the BIG<br>Range: 0x000A to 0x4000<br>Time = N × 10 ms<br>Time Range: 100 ms to 163.84 s |

*Num_BIS:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | Total number of BISes to synchronize |
| All other values | Reserved for future use |

*BIS[i]:* *Size: Num_BIS × 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x1F | BIS_Number for a BIS in the BIG |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_BIG_Create_Sync command, the Controller sends the HCI_Command_Status event to the Host. When the HCI_LE_BIG_Create_Sync command has completed, the HCI_LE_BIG_Sync_Established event will be generated.

If the Controller does not receive a BIS PDU within 6 BIS events of first listening, then it shall generate an HCI_LE_BIG_Sync_Established event that should have Status set to *Connection Failed to be Established / Synchronization Timeout* (0x3E).

### 7.8.107 LE BIG Terminate Sync command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_BIG_Terminate_Sync | 0x006C | BIG_Handle | Status,<br>BIG_Handle |

**Description:**

This command is used to stop synchronizing or cancel the process of synchronizing to the BIG identified by the BIG_Handle parameter. The command also terminates the reception of BISes in the BIG specified in the HCI_LE_BIG_Create_Sync command, destroys the associated connection handles of the BISes in the BIG and removes the data paths for all BISes in the BIG.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | BIG_Handle does not identify a BIG. | *Unknown Advertising Identifier* (0x42) |
| MC | BIG_Handle identifies a BIG which the Controller is neither synchronized to nor in the process of synchronizing to. | *Command Disallowed* (0x0C) |

**Command parameters:**

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifer of the BIG |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_BIG_Terminate_Sync command succeeded |
| 0x01 to 0xFF | The HCI_LE_BIG_Terminate_Sync command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*BIG_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0xEF | Identifer of the BIG |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_BIG_Terminate_Sync command has completed, an HCI_Command_Complete event shall be generated.

If the Host attempts to terminate synchronization with a BIG while the process of synchronization with that BIG is in progress (i.e. HCI_LE_BIG_Sync_Established event has not been generated) the process of synchronization shall stop, and the Controller shall generate the HCI_LE_BIG_Sync_Established event to the Host with the error code *Operation Cancelled by Host* (0x44).

### 7.8.108 LE Request Peer SCA command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Request_Peer_SCA | 0x006D | Connection_Handle | *none* |

**Description:**

This command is used to read the Sleep Clock Accuracy (SCA) of the peer device.

The Connection_Handle parameter is the connection handle of the ACL connection.

If the Host sends this command and the peer device does not support the Sleep Clock Accuracy Updates feature, the Controller shall return the error code *Unsupported Feature or Parameter Value* (0x11) in the HCI_LE_Request_Peer_SCA_Complete event.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |
| MC | The Controller is aware (e.g., through a previous feature exchange) that the peer device's Link Layer does not support the Sleep Clock Accuracy Updates feature. | *Unsupported Remote Feature* (0x1A) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the ACL<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Request_Peer_SCA command, the Controller sends the HCI_Command_Status event to the Host.

When the HCI_LE_Request_Peer_SCA command has completed, the HCI_LE_Request_Peer_SCA_Complete event shall be generated.

### 7.8.109 LE Setup ISO Data Path command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Setup_ISO_Data_Path | 0x006E | Connection_Handle,<br>Data_Path_Direction,<br>Data_Path_ID,<br>Codec_ID,<br>Controller_Delay,<br>Codec_Configuration_Length,<br>Codec_Configuration | Status,<br>Connection_Handle |

**Description:**

This command is used to identify and create the isochronous data path between the Host and the Controller for a CIS, CIS configuration, or BIS identified by the Connection_Handle parameter. This command can also be used to configure a codec for each data path. When a connection is created no data paths are set up for that connection. When the command has completed successfully, data shall be allowed to flow over the specified path in the specified direction irrespective of the state of the other direction or any other path.

The input and output directions are defined from the perspective of the Controller, so "input" refers to data flowing from the Host to the Controller.

The Data_Path_Direction parameter specifies the direction for which the data path is being configured.

The Data_Path_ID parameter specifies the data transport path used. When set to 0x00, the data path shall be over the HCI transport. When set to a value in the range 0x01 to 0xFE, the data path shall use a vendor-specific transport interface (e.g., a PCM interface) with logical transport numbers. The meanings of these logical transport numbers are vendor-specific.

The Codec_ID parameter specifies the coding format used over the air.

When Data_Path_Direction is set to 0x00 (input), the Controller_Delay parameter specifies the delay at the data source from the reference time of an SDU to the CIG reference point (see [Vol 6] Part B, Section 4.5.14.1) or BIG anchor point (see [Vol 6] Part B, Section 4.4.6.4). When Data_Path_Direction is set to 0x01 (output), Controller_Delay specifies the delay from the SDU_Synchronization_Reference to the point in time at which the Controller begins to transfer the corresponding data to the data path interface. The Host should use the HCI_Read_Local_Supported_Controller_Delay command to obtain a suitable value for Controller_Delay.

Note: Controller vendors can provide additional guidance to the Host on how to select a suitable Controller_Delay value from the range of values provided by the HCI_Read_Local_Supported_Controller_Delay command for various configurations of the data path interface.

The Codec_Configuration parameter specifies codec-specific configuration information for the specified direction.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | A data path for the same Connection_Handle and direction has been set up and not removed. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle identifies a CIS on a Peripheral and the HCI_LE_Accept_CIS_Request command for that CIS has not completed successfully. | *Command Disallowed* (0x0C) |
| MC | The data transport path is vendor-specific and has not been configured. | *Command Disallowed* (0x0C) |
| MC | Connection Handle does not identify a CIS, CIS configuration, or BIS. | *Unknown Connection Identifier* (0x02) |
| MC | Data_Path_Direction is output and Connection_Handle identifies an Isochronous Broadcaster. | *Command Disallowed* (0x0C) |
| MC | Data_Path_Direction is input and Connection_Handle identifies a Synchronized Receiver. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle identifies a unidirectional CIS and Data_Path_Direction is the direction where BN is set to 0. | *Command Disallowed* (0x0C) |
| MC | Codec_Configuration_Length is non-zero and Codec_ID is set to transparent air mode. | *Invalid HCI Command Parameters* (0x12) |
| MC | The codec-related parameters exceed the bandwidth and latency allowed on the established CIS or BIS identified by Connection_Handle. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Handle identifies a CIS or BIS that is in ISO Test mode (see Section 7.8.111, Section 7.8.112, and [Vol 6] Part B, Section 7) for the specified Data_Path_Direction. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS or BIS<br>Range: 0x0000 to 0x0EFF |

*Data_Path_Direction:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Input (Host to Controller) |
| 0x01 | Output (Controller to Host) |
| All other values | Reserved for future use |

*Data_Path_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI |
| 0x01 to 0xFE | Logical_Channel_Number. The meaning of the logical channel is vendor-specific. |
| 0xFF | Reserved for future use |

*Codec_ID:* *Size: 5 octets*

| Value | Parameter Description |
|---|---|
| Octet 0 | See Assigned Numbers for Coding Format |
| Octets 1 to 2 | Company ID, see Assigned Numbers for Company Identifier.<br>Shall be ignored if octet 0 is not 0xFF. |
| Octets 3 to 4 | Vendor-defined codec ID.<br>Shall be ignored if octet 0 is not 0xFF. |

*Controller_Delay:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Controller delay in microseconds<br>Range: 0x000000 to 0x3D0900<br>Time range: 0 s to 4 s |

*Codec_Configuration_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of codec configuration |

*Codec_Configuration:* *Size: Codec_Configuration_Length octets*

| Parameter Description |
|---|
| Codec-specific configuration data<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Setup_ISO_Data_Path command succeeded |
| 0x01 to 0xFF | HCI_LE_Setup_ISO_Data_Path command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS or BIS<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_LE_Setup_ISO_Data_Path command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.110 LE Remove ISO Data Path command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_ISO_Data_Path | 0x006F | Connection_Handle,<br>Data_Path_Direction | Status,<br>Connection_Handle |

**Description:**

This command is used to remove the input and/or output data path(s) associated with a CIS, CIS configuration, or BIS identified by the Connection_Handle parameter.

The Data_Path_Direction parameter specifies which directions are to have the data path removed.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a CIS, CIS configuration, or BIS. | *Unknown Connection Identifier* (0x02) |
| MC | The data path has not been set up (using the HCI_LE_Setup_ISO_Data_Path command). | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS or BIS.<br>Range: 0x0000 to 0x0EFF |

*Data_Path_Direction:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Remove input data path |
| 1 | Remove output data path |
| All other bits | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The HCI_LE_Remove_ISO_Data_Path command succeeded |
| 0x01 to 0xFF | The HCI_LE_Remove_ISO_Data_Path command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS or BIS<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_ISO_Data_Path command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.111 LE ISO Transmit Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_ISO_Transmit_Test | 0x0070 | Connection_Handle,<br>Payload_Type | Status,<br>Connection_Handle |

**Description:**

This command should only be used in the ISO Test mode and only for testing purposes.

The command is used to configure an established CIS or BIS specified by the Connection_Handle parameter, and transmit test payloads which are generated by the Controller.

The Payload_Type parameter defines the configuration of SDUs in the payload.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a CIS or a BIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a CIS where the transmit BN is zero. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle identifies a BIS which is a Synchronized Receiver. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle identifies a CIS or BIS where the input data path is already set up. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the CIS or BIS<br>Range 0x0000 to 0x0EFF |

*Payload_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Zero length payload |
| 0x01 | Variable length payload |
| 0x02 | Maximum length payload |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_ISO_Transmit_Test command succeeded |
| 0x01 to 0xFF | HCI_LE_ISO_Transmit_Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_ISO_Transmit_Test command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.112 LE ISO Receive Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_ISO_Receive_Test | 0x0071 | Connection_Handle,<br>Payload_Type | Status,<br>Connection_Handle |

**Description:**

This command should only be used in the ISO Test mode and only for testing purposes.

The command is used to configure an established CIS or a synchronized BIG specified by the Connection_Handle parameter to receive payloads.

When using this command for a BIS, the Host shall synchronize with a BIG using the HCI_LE_BIG_Create_Sync command before invoking this command.

The Payload_Type parameter defines the configuration of SDUs in the payload.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an established CIS or a BIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a CIS where the receive BN is zero. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle identifies a BIS which is an Isochronous Broadcaster. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle identifies a CIS or BIS where the output data path is already set up. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range 0x0000 to 0x0EFF |

*Payload_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Zero length payload |
| 0x01 | Variable length payload |
| 0x02 | Maximum length payload |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_ISO_Receive_Test command succeeded |
| 0x01 to 0xFF | HCI_LE_ISO_Receive_Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_ISO_Receive_Test command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.113 LE ISO Read Test Counters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_ISO_Read_Test_Counters | 0x0072 | Connection_Handle | Status,<br>Connection_Handle,<br>Received_SDU_Count,<br>Missed_SDU_Count,<br>Failed_SDU_Count |

**Description:**

This command should only be used in the ISO Test mode and only for testing purposes.

The command is used to read the test counters (see [Vol 6] Part B, Section 7) in the Controller which is configured in ISO Receive Test mode for a CIS or BIS specified by the Connection_Handle. Reading the test counters does not reset the test counters.

The Received_SDU_Count, Missed_SDU_Count and Failed_SDU_Count parameters are set in the ISO Receive Test mode (see [Vol 6] Part B, Section 7.2).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an established CIS or a BIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a CIS or BIS that is not configured in the ISO Receive Test mode. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range 0x0000 to 0x0EFF |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_ISO_Read_Test_Counters command succeeded |
| 0x01 to 0xFF | HCI_LE_ISO_Read_Test_Counters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The connection handle of a CIS or BIS<br>Range: 0x0000 to 0x0EFF |

*Received_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Received_SDU_Count |

*Missed_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Missed_SDU_Count |

*Failed_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Failed_SDU_Count |

### Event(s) generated (unless masked away):

When the HCI_LE_ISO_Read_Test_Counters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.114 LE ISO Test End command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_ISO_Test_End | 0x0073 | Connection_Handle | Status,<br>Connection_Handle,<br>Received_SDU_Count,<br>Missed_SDU_Count,<br>Failed_SDU_Count |

**Description:**

This command should only be used in the ISO Test mode and only for testing purposes.

The command is used to terminate the ISO Transmit and/or Receive Test mode for a CIS or BIS specified by the Connection_Handle parameter but does not terminate the CIS or BIS.

When the Host terminates the ISO Test mode for a CIS or BIS that is set to ISO Transmit Test mode only, the test counters in the return parameters shall be set to zero.

When the Host terminates the ISO Test mode for a CIS or BIS that is set to the ISO Receive Test mode, the return parameters contain the values of the test counters as defined in [Vol 6] Part B, Section 7.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an established CIS or a BIS. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies a CIS or BIS that is not configured in the ISO Transmit or Receive Test mode. | *Unsupported Feature or Parameter Value* (0x11) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_ISO_Test_End command succeeded |
| 0x01 to 0xFF | HCI_LE_ISO_Test_End command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range: 0x0000 to 0x0EFF |

*Received_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Received_SDU_Count |

*Missed_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Missed_SDU_Count |

*Failed_SDU_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number in the Failed_SDU_Count |

**Event(s) generated (unless masked away):**

When the HCI_LE_ISO_Test_End command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.115 LE Set Host Feature command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Host_Feature [v2] | 0x0097 | Bit_Number,<br>Bit_Value | Status |
| HCI_LE_Set_Host_Feature [v1] | 0x0074 | Bit_Number,<br>Bit_Value | Status |

**Description:**

This command is used by the Host to set or clear a bit controlled by the Host in the Link Layer FeatureSet stored in the Controller (see [Vol 6] Part B, Section 4.6).

The Bit_Number parameter specifies the bit position in the FeatureSet.

The Bit_Value parameter specifies whether the feature is enabled or disabled.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Bit_Number specifies a feature bit that is not controlled by the Host. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Bit_Value is set to 0x01 and Bit_Number specifies a feature bit that requires support of a feature that the Controller does not support. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The Controller has a connection to another device. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Bit_Number [v1]:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Bit position in the FeatureSet |

*Bit_Number [v2]:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x07BF | Bit position in the FeatureSet |
| All other values | Reserved for future use |

*Bit_Value:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Host feature is disabled and so the bit in the FeatureSet shall be set to 0 |
| 0x01 | The Host feature is enabled and so the bit in the FeatureSet shall be set to 1 |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Host_Feature command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Host_Feature command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Host_Feature command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.116 LE Read ISO Link Quality command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_ISO_Link_Quality | 0x0075 | Connection_Handle | Status,<br>Connection_Handle,<br>TX_UnACKed_Packets,<br>TX_Flushed_Packets,<br>TX_Last_Subevent_Packets,<br>Retransmitted_Packets,<br>CRC_Error_Packets,<br>RX_Unreceived_Packets,<br>Duplicate_Packets |

**Description:**

This command returns the values of various counters related to link quality that are associated with the isochronous stream specified by the Connection_Handle parameter.

This command may be issued on both the Central and Peripheral if the connection handle identifies a CIS and on the Synchronized Receiver if the connection handle identifies a BIS.

Each of the remaining return parameters shall contain the current value of the corresponding counter; all the values shall be recorded at the same moment. Each counter shall be a 32-bit unsigned value, shall be initialized to zero when the isochronous stream is created, and shall be incremented by one as described below. If a counter is not associated with the type of isochronous stream specified, the value of the parameter shall be ignored.

| Counter | Associated Streams | When Incremented |
|---|---|---|
| TX_UnACKed_Packets | CIS | The Link Layer does not receive an acknowledgment for a CIS Data PDU that it transmitted at least once by its flush point (see [Vol 6] Part B, Section 4.5.13.5). |
| TX_Flushed_Packets | CIS | The Link Layer does not transmit a specific payload by its flush point. |
| TX_Last_Subevent_-Packets | CIS in Peripher-al role | The Link Layer transmits a CIS Data PDU in the last subevent of a CIS event. |
| Retransmitted_Packets | CIS | The Link Layer retransmits a CIS Data PDU. |
| CRC_Error_Packets | CIS and BIS | The Link Layer receives a packet with a CRC error. |

| Counter | Associated Streams | When Incremented |
|---|---|---|
| RX_Unreceived_-Packets | CIS and BIS | The Link Layer does not receive a specific payload by its flush point (on a CIS) or the end of the event it is associated with (on a BIS; see [Vol 6] Part B, Section 4.4.6.6). |
| Duplicate_Packets | CIS | The Link Layer receives a retransmission of a CIS Data PDU. |

*Table 7.4: Isochronous streams link quality counters*

### Errors:

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a current CIS connection or a BIS that the Controller is synchronized to. | *Unknown Connection Identifier* (0x02) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range 0x0000 to 0x0EFF |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_ISO_Link_Quality command succeeded |
| 0x01 to 0xFF | HCI_LE_Read_ISO_Link_Quality command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of a CIS or BIS<br>Range: 0x0000 to 0x0EFF |

*TX_UnACKed_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Tx_UnACKed_Packets counter |

*TX_Flushed_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Tx_Flushed_Packets counter |

*TX_Last_Subevent_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Tx_Last_Subevent_Packets counter |

*Retransmitted_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Retransmitted_Packets counter |

*CRC_Error_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the CRC_Error_Packets counter |

*RX_Unreceived_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Rx_Unreceived_Packets counter |

*Duplicate_Packets:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Value of the Duplicate_Packets counter |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_ISO_Link_Quality command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.117 LE Enhanced Read Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Enhanced_Read_-Transmit_Power_Level | 0x0076 | Connection_Handle,<br>PHY | Status,<br>Connection_Handle,<br>PHY,<br>Current_TX_Power_Level,<br>Max_TX_Power_Level |

**Description:**

This command is used to read the current and maximum transmit power levels of the local Controller on the ACL connection identified by the Connection_Handle parameter and the PHY indicated by the PHY parameter.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | PHY specifies a value that the Controller does not support, including a value that is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |

| Value | Parameter Description |
|---|---|
| 0x04 | LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Enhanced_Read_Transmit_Power_Level command succeeded. |
| 0x01 to 0xFF | HCI_LE_Enhanced_Read_Transmit_Power_Level command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |
| 0x04 | LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

*Current_TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Current transmit power level<br>Range: -127 to 20<br>Units: dBm |
| 0x7F | Current transmit power level is unavailable |

*Max_TX_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Maximum transmit power level<br>Range: -127 to 20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the HCI_LE_Enhanced_Read_Transmit_Power_Level command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.118 LE Read Remote Transmit Power Level command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Remote_Transmit_-Power_Level | 0x0077 | Connection_Handle,<br>PHY | *none* |

**Description:**

This command is used to read the transmit power level used by the remote Controller on the ACL connection that is identified by the Connection_Handle parameter and the PHY indicated by the PHY parameter.

The local Controller may use the remote transmit power level value obtained from a prior Power Change Indication or Power Control Request procedure (see [Vol 6] Part B, Section 5.1.17 and [Vol 6] Part B, Section 5.1.18). If the Controller chooses not to use these prior values, or if no prior value is available for one or more of the remote transmit power level, maximum transmit power level, or minimum transmit power level, the local Controller shall initiate a new Power Control Request procedure to obtain the remote transmit power level.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | PHY specifies a value that the Controller does not support, including a value that is reserved for future use. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |
| 0x04 | LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Read_Remote_Transmit_Power_Level command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has determined the remote transmit power, it shall generate an HCI_LE_Transmit_Power_Reporting event with Reason 0x02.

### 7.8.119 LE Set Path Loss Reporting Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Path_Loss_Reporting_-Parameters | 0x0078 | Connection_Handle,<br>High_Threshold,<br>High_Hysteresis,<br>Low_Threshold,<br>Low_Hysteresis,<br>Min_Time_Spent | Status,<br>Connection_Handle |

**Description:**

This command is used to set the path loss threshold reporting parameters for the ACL connection identified by the Connection_Handle parameter.

The path loss threshold-based mechanism is described in [Vol 6] Part B, Section 4.5.16. For each zone boundary, the upwards boundary shall equal the threshold plus the hysteresis and the downwards boundary shall equal the threshold minus the hysteresis.

The Min_Time_Spent parameter indicates the minimum time that the Controller shall observe the path loss has crossed the threshold before the Controller generates an event for the threshold crossing. The Host should specify a suitable value based on the connection interval, subrate factor, and Peripheral latency.

If the Host issues this command when path loss monitoring is enabled, the Controller shall override the existing path loss threshold reporting parameters with the parameters provided in this command.

The High_Threshold and the Low_Threshold parameters are common to all PHYs supported by the Controller. However, the Host can reissue this command with suitable parameters whenever a PHY switch is detected.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | High_Threshold + High_Hysteresis is greater than 0xFF. | *Invalid HCI Command Parameters* (0x12) |
| MC | Low_Threshold is less than Low_Hysteresis. | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| MC | Low_Threshold is greater than High_Threshold. | *Invalid HCI Command Parameters* (0x12) |
| MC | Low_Threshold + Low_Hysteresis is greater than High_Threshold – High_Hysteresis. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*High_Threshold:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | High threshold for the path loss<br>Units: dB |
| 0xFF | High Threshold unused |

*High_Hysteresis:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Hysteresis value for the high threshold<br>Units: dB |

*Low_Threshold:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Low threshold for the path loss<br>Units: dB |

*Low_Hysteresis:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Hysteresis value for the low threshold<br>Units: dB |

*Min_Time_Spent:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum time, in number of connection events, to be observed once the path loss crosses the threshold before an event is generated. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Path_Loss_Reporting_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Path_Loss_Reporting_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Path_Loss_Reporting_Parameters command has completed, an HCI_Command_Complete event shall be generated.

If the Host issues this command when path loss monitoring is enabled, and if the new parameters mean that the path loss is now in a different zone, an HCI_LE_Path_Loss_Threshold event shall be generated as soon as possible irrespective of the Min_Time_Spent parameter and the timer shall be reset.

If the Host issues this command with High_Threshold parameter set to 0xFF, then the Controller shall not generate an HCI_LE_Path_Loss_Threshold event with Zone_Entered set to 0x02.

### 7.8.120 LE Set Path Loss Reporting Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Path_Loss_Reporting_-Enable | 0x0079 | Connection_Handle,<br>Enable | Status,<br>Connection_Handle |

**Description:**

This command is used to enable or disable path loss reporting for the ACL connection identified by the Connection_Handle parameter.

If the Enable parameter is set to 0x01 and no prior LE Power Control Request procedure has been initiated on the ACL connection, then the Controller may need to initiate a new LE Power Control Request procedure on that ACL.

Path loss reporting is disabled when the connection is first created.

Enabling path loss monitoring when it is already enabled or disabling path loss monitoring when it is already disabled has no effect.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | HCI_LE_Set_Path_Loss_Reporting_Parameters has not been successfully issued on this connection. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reporting disabled |
| 0x01 | Reporting enabled |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Path_Loss_Reporting_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Path_Loss_Reporting_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Path_Loss_Reporting_Enable command has completed, an HCI_Command_Complete event shall be generated.

When reporting is enabled and was previously disabled, the Controller shall generate an HCI_LE_Path_Loss_Threshold event as soon as it has a reliable measurement of the path loss. If the Controller has to query the remote Controller for its transmit power level, then it shall generate this event within $T_{path_loss_enable}$ from the time it receives a response to its query. Otherwise, the Controller shall generate this event within $T_{path_loss_enable}$ from the time the command is issued. $T_{path_loss_enable}$ shall be Min_Time_Spent + 6 connection events or, if longer, 2 connection events where the Controller actually receives a packet from the peer, where Min_Time_Spent is specified by the HCI_LE_Set_Path_Loss_Reporting_Parameters command.

After the initial event on reporting being enabled, the Controller shall generate this event each time it determines that the path loss has moved to a different zone and stayed in that zone for Min_Time_Spent.As stated in [Vol 6] Part B, Section 4.5.16, two consecutive events must not indicate the same zone.

### 7.8.121 LE Set Transmit Power Reporting Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Transmit_Power_-Reporting_Enable | 0x007A | Connection_Handle,<br>Local_Enable,<br>Remote_Enable | Status,<br>Connection_Handle |

**Description:**

This command is used to enable or disable the reporting to the local Host of transmit power level changes in the local and remote Controllers for the ACL connection identified by the Connection_Handle parameter.

If the Remote_Enable parameter is set to 0x01 and no prior LE Power Control Request procedure has been initiated on the ACL connection, then the Controller shall initiate a new LE Power Control Request procedure on that ACL.

Reporting is disabled when the connection is first created.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Local_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable local transmit power reports |
| 0x01 | Enable local transmit power reports |
| All other values | Reserved for future use |

*Remote_Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable remote transmit power reports |
| 0x01 | Enable remote transmit power reports |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Transmit_Power_Reporting_Enable command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Transmit_Power_Reporting_Enable command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Transmit_Power_Reporting_Enable command has completed, an HCI_Command_Complete event shall be generated.

When local reporting is enabled, the Controller shall generate an HCI_LE_Transmit_Power_Reporting event with Reason 0x00 each time the local transmit power level is changed.

When remote reporting is enabled, the Controller shall generate an HCI_LE_Transmit_Power_Reporting event with Reason 0x01 each time it becomes aware that the remote transmit power level has changed.

### 7.8.122 LE Set Data Related Address Changes command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Data_Related_Address_-Changes | 0x007C | Advertising_Handle, Change_Reasons | Status |

**Description:**

This command specifies circumstances when the Controller shall refresh any Resolvable Private Address used by the advertising set identified by the Advertising_Handle parameter, whether or not the address timeout period has been reached. This command may be used while advertising is enabled.

The Change_Reasons parameter specifies the reason(s) for refreshing addresses. The default when an advertising set is created, or if legacy advertising commands (see Section 3.1.1) are used, is for all bits to be clear.

If legacy advertising commands are being used, the Controller shall ignore the Advertising_Handle parameter.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | No command specified in Table 3.2 has been issued. | *Unknown Advertising Identifier* (0x42) |
| MC | The extended advertising commands (see Section 3.1.1) are being used and Advertising_Handle does not identify an advertising set. | *Unknown Advertising Identifier* (0x42) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Change_Reasons:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Change the address whenever the advertising data changes. |
| 1 | Change the address whenever the scan response data changes. |
| All other bits | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Data_Related_Address_Changes command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Data_Related_Address_Changes command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Data_Related_Address_Changes command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.123 LE Set Default Subrate command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Default_Subrate | 0x007D | Subrate_Min,<br>Subrate_Max,<br>Max_Latency,<br>Continuation_Number,<br>Supervision_Timeout | Status |

**Description:**

This command is used by the Host to set the initial values for the acceptable parameters for subrating requests, as defined by the HCI_LE_Subrate_Request command (see Section 7.8.124), for all future ACL connections where the Controller is the Central. This command does not affect any existing connection.

The parameters have the same meanings as those in the HCI_LE_Subrate_Request command.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Subrate_Max × (Max_Latency + 1) is greater than 500. | *Invalid HCI Command Parameters* (0x12) |
| MC | Subrate_Max is less than Subrate_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Continuation_Number is greater than or equal to Subrate_Max. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Subrate_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum subrate factor allowed in requests by a Peripheral<br>Range: 0x0001 to 0x01F4<br>Default: 0x0001 |

*Subrate_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum subrate factor allowed in requests by a Peripheral<br>Range: 0x0001 to 0x01F4<br>Default: 0x0001 |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency allowed in requests by a Peripheral, in units of subrated connection intervals<br>Range: 0x0000 to 0x01F3<br>Default: 0x0000 |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received in requests by a Peripheral<br>Range: 0x0000 to 0x01F3<br>Default: 0x0000 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum supervision timeout allowed in requests by a Peripheral<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s<br>Default: 0x0C80 (32 s) |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Default_Subrate command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Default_Subrate command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Set_Default_Subrate command, the Controller sends the HCI_Command_Complete event to the Host.

### 7.8.124 LE Subrate Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Subrate_Request | 0x007E | Connection_Handle,<br>Subrate_Min,<br>Subrate_Max,<br>Max_Latency,<br>Continuation_Number,<br>Supervision_Timeout | *none* |

**Description:**

This command is used by a Central or a Peripheral to request a change to the subrating factor and/or other parameters (see [Vol 6] Part B, Section 4.5.1) applied to an existing connection using the Connection Subrate Update procedure.

The Subrate_Min and Subrate_Max parameters specify the range of acceptable subrating factors being requested.

The Max_Latency parameter specifies the maximum Peripheral latency in units of subrated connection events. The same maximum shall apply irrespective of the subrating factor actually chosen.

The Continuation_Number parameter specifies the number of underlying connection intervals to remain active after a packet (other than an empty packet) is transmitted or received.

The Supervision_Timeout parameter specifies the link supervision timeout for the connection. The Supervision_Timeout, in milliseconds, shall be greater than 2 × current connection interval × Subrate_Max × (Max_Latency + 1).

If this command is issued on the Central, the following rules shall apply when the Controller initiates the Connection Subrate Update procedure (see [Vol 6] Part B, Section 5.1.19):

- The Peripheral latency shall be less than or equal to Max_Latency.
- The subrate factor shall be between Subrate_Min and Subrate_Max.
- The continuation number shall be equal to the lesser of Continuation_Number and (subrate factor - 1).
- The connection supervision timeout shall be equal to Supervision_Timeout.

If this command is issued on the Central, it also sets the acceptable parameters for requests from the Peripheral (see [Vol 6] Part B, Section 5.1.20). The

acceptable parameters set by this command override those provided via the HCI_LE_Set_Default_Subrate command or any values set by previous uses of this command on the same connection.

If this command is issued on the Central before the devices have performed the Feature Exchange procedure, then the Controller shall complete that procedure before initiating the Connection Subrate Update procedure.

If this command is issued on the Peripheral, the following rules shall apply when the Controller initiates the Connection Subrate Request procedure:

- The Peripheral latency shall be less than or equal to Max_Latency.
- The minimum and maximum subrate factors shall be between Subrate_Min and Subrate_Max.
- The continuation number shall be equal to the lesser of Continuation_Number and (maximum subrate factor - 1).
- The connection supervision timeout shall be equal to Supervision_Timeout.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify a current ACL. | *Unknown Connection Identifier* (0x02) |
| MC | Subrate_Max is less than Subrate_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Continuation_Number is greater than or equal to Subrate_Max. | *Invalid HCI Command Parameters* (0x12) |
| MC | The Controller is the Central and the Connection Subrating (Host Support) bit is not set in the Peripheral's FeatureSet. | *Unsupported Remote Feature* (0x1A) |
| MC | Subrate_Max × (Max_Latency + 1) > 500 | *Invalid HCI Command Parameters* (0x12) |
| MC | Current connection interval × Subrate_Max × (Max_Latency + 1) ≥ Supervision_Timeout ÷ 2 | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection Rate Request or Connection Rate Update procedure is in progress | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection handle of the ACL<br>Range: 0x0000 to 0x0EFF |

*Subrate_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum subrate factor to be applied to the underlying connection interval<br>Range: 0x0001 to 0x01F4 |

*Subrate_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum subrate factor to be applied to the underlying connection interval<br>Range: 0x0001 to 0x01F4 |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency for the connection in units of subrated connection intervals<br>Range: 0x0000 to 0x01F3 |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received<br>Range: 0x0000 to 0x01F3 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Supervision timeout for this connection<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Subrate_Request command, the Controller sends the HCI_Command_Status event to the Host. An HCI_LE_Subrate_Change event shall be generated when the Connection Subrate Update procedure has completed.

### 7.8.125 LE Set Periodic Advertising Subevent Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_-Subevent_Data | 0x0082 | Advertising_Handle,<br>Num_Subevents_With_Data,<br>Subevent[i],<br>Response_Slot_Start[i],<br>Response_Slot_Count[i],<br>Subevent_Data_Length[i],<br>Subevent_Data[i] | Status,<br>Advertising_Handle |

**Description:**

This command is used by the Host to set the data for one or more subevents of PAwR in reply to an HCI_LE_Periodic_Advertising_Subevent_Data_Request event. The data for a subevent shall be transmitted only once.

When using more than one instance of this command to provide data, the Host may provide the data in any order. For example, if providing data for subevents 0 and 1 in separate commands, the first command can contain either subevent.

The Advertising_Handle parameter identifies the advertising set whose periodic advertising subevent data is being set.

The Num_Subevents_With_Data parameter is the number of subevent data contained in the parameter arrays.

The Subevent[i] parameter identifies the subevent of the PAwR that is being set.

The Response_Slot_Start[i] and Response_Slot_Count[i] parameters identify the starting response slot and the number of response slots that are expected to be used in this subevent.

Note: If the Host does not expect any responses in a subevent, then it can set Response_Slot_Count[i] for that subevent to 0. If the Host expects one or more responses in a future subevent, then it can set Response_Slot_Start[i] and Response_Slot_Count[i] for that subevent appropriately either in the same command or a subsequent use of this command.

The Subevent_Data_Length[i] parameter determines the length of the Subevent_Data that is significant.

The Subevent_Data[i] parameter contains the advertising data to be transmitted in the subevent of the advertising set. If the combined data length is greater than the

maximum that the Controller can transmit within the current subevent interval, then all data shall be discarded.

If the Subevent_Data cannot be transmitted because, for example, the subevent where this data would have been sent has already passed or is too early, then the Controller shall discard the data.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an existing advertising set. | *Unknown Advertising Identifier* (0x42) |
| MC | Advertising_Handle identifies an advertising set that has not been configured for Periodic Advertising with Responses. | *Command Disallowed* (0x0C) |
| MC | Subevent[i] is outside of the range of subevents requested using the HCI_LE_Periodic_Advertising_Subevent_Data_Request event. | *Command Disallowed* (0x0C) |
| M | Two elements of Subevent specify the same subevent. | *Invalid HCI Command Parameters* (0x12) |
| MC | The combined data length of Subevent is greater than the maximum that the Controller can transmit within the current subevent interval.<br>If advertising on the LE Coded PHY, then the S=8 coding shall be assumed unless the current advertising parameters require the use of S=2 for an advertising physical channel, in which case the S=2 coding shall be assumed for that advertising physical channel. | *Packet Too Long* (0x45) |
| MC | The Subevent_Data cannot be transmitted because, for example, the subevent where this data would have been sent has already passed. | *Too Late* (0x46) |
| MC | The Subevent_Data cannot be transmitted because, for example, the subevent where this data would have been sent is too early. | *Too Early* (0x47) |
| M | No HCI_LE_Periodic_Advertising_Subevent_Data_Request event has been issued. | *Command Disallowed* (0x0C) |
| M | This command is sent twice for the same subevent of the same periodic advertising event. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

*Num_Subevents_With_Data:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x0F | Number of subevent data in the command. |
| All other values | Reserved for future use |

*Subevent[i]:* *Size: Num_Subevents_With_Data × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The subevent index of the data contained in this command.<br>Range: 0x00 to 0x7F |

*Response_Slot_Start[i]:* *Size: Num_Subevents_With_Data × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The first response slots to be used in this subevent. |

*Response_Slot_Count[i]:* *Size: Num_Subevents_With_Data × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of response slots to be used. |

*Subevent_Data_Length[i]:* *Size: Num_Subevents_With_Data × 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 254 | The number of octets in the Subevent_Data parameter. |
| All other values | Reserved for future use |

*Subevent_Data[i]:* *Size: SUM (Subevent_Data_Length[i]) octets*

| Parameter Description |
|---|
| Advertising data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: Each element of this array has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Advertising_Subevent_Data command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Advertising_Subevent_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify a periodic advertising train<br>Range: 0x00 to 0xEF |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Periodic_Advertising_Subevent_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.126 LE Set Periodic Advertising Response Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Advertising_-Response_Data | 0x0083 | Sync_Handle,<br>Request_Event,<br>Request_Subevent,<br>Response_Subevent,<br>Response_Slot,<br>Response_Data_Length,<br>Response_Data | Status,<br>Sync_Handle |

**Description:**

This command is used by the Host to set the data for a response slot in a specific subevent of the PAwR identified by the Sync_Handle. The data for a response slot shall be transmitted only once.

The Request_Event parameter identifies the periodic advertising event in which the periodic advertising packet that the Host is responding to was received.

The Request_Subevent parameter identifies the subevent in which the periodic advertising packet that the Host is responding to was received.

The Response_Subevent parameter identifies the subevent that the response shall be sent in.

The Response_Slot parameter identifies the response slot in the subevent identified by the Response_Subevent parameter in which this response data is to be transmitted.

The Response_Data_Length specifies the length of the Response_Data that is significant.

The Response_Data contains the advertising data to be transmitted in the response slot. If the Response_Data_Length is greater than the maximum that the Controller can transmit within the response slot, then the Response_Data shall be discarded. If the periodic advertising train is on the LE Coded PHY, then the S=8 coding shall be assumed.

If the response slot identified by the Response_Slot parameter has passed by the time this command is received by the Controller, the Controller shall discard the Response_Data parameter.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Response_Data_Length is greater than the maximum that the Controller can transmit within the response slot. If the periodic advertising train is on the LE Coded PHY, then the S=8 coding shall be assumed. | *Packet Too Long* (0x45) |
| MC | The response slot identified by the Response_Slot parameter has already passed. | *Too Late* (0x46) |
| M | Response_Subevent exceeds numSubevents of the periodic advertising train or the Controller is not synchronized with the subevent. | *Command Disallowed* (0x0C) |

### Command parameters:

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the PAwR train<br>Range: 0x0000 to 0x0EFF |

*Request_Event:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The value of *paEventCounter* (see [Vol 6] Part B, Section 4.4.2.1) for the periodic advertising packet that the Host is responding to |

*Request_Subevent:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The subevent for the periodic advertising packet that the Host is responding to |

*Response_Subevent:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify the subevent of the PAwR train.<br>Range: 0x00 to 0x7F |

*Response_Slot:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify the response slot of the PAwR train.<br>Range: 0x00 to 0xFF |

*Response_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 254 | The number of octets in the Response_Data parameter. |
| All other values | Reserved for future use |

*Response_Data:* *Size: Response_Data_Length octets*

| Parameter Description |
|---|
| Response data formatted as defined in [Vol 3] Part C, Section 11.<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Periodic_Advertising_Response_Data command succeeded. |
| 0x01 to 0xFF | HCI_LE_Periodic_Advertising_Response_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train.<br>Range: 0x0000 to 0x0EFF |

### Event(s) generated (unless masked away):

When the HCI_LE_Periodic_Advertising_Response_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.127 LE Set Periodic Sync Subevent command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Periodic_Sync_-Subevent | 0x0084 | Sync_Handle,<br>Periodic_Advertising_Properties,<br>Num_Subevents_To_Sync,<br>Subevent[i] | Status,<br>Sync_Handle |

**Description:**

This command is used to instruct the Controller to synchronize with a subset of the subevents within a PAwR train identified by the Sync_Handle parameter, listen for packets sent by the peer device and pass any received data up to the Host. If the Controller is synchronized with any subevents that are not in the subset of subevents in this command, then the Controller shall no longer synchronize with those subevents.

The Periodic_Advertising_Properties parameter indicates which fields should be included in the AUX_SYNC_SUBEVENT_RSP PDUs.

The Num_Subevents_To_Sync parameter identifies the number of values in the sub-events parameter.

The Subevents arrayed parameter identifies the subevents that the Controller shall synchronize with.

**Command parameters:**

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the PAwR train<br>Range: 0x0000 to 0x0EFF |

*Periodic_Advertising_Properties:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 6 | Include TxPower in the advertising PDU |
| All other bits | Reserved for future use |

*Num_Subevents_To_Sync:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of subevents.<br>Range: 0x01 to 0x80 |

*Subevent[i]:* *Size: Num_Subevents_To_Sync × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The subevent to synchronize with.<br>Range 0x00 to 0x7F |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Periodic_Sync_Subevent command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Periodic_Sync_Subevent command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Sync_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Sync_Handle identifying the periodic advertising train<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Periodic_Sync_Subevent command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.128 LE Read All Local Supported Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_All_Local_Supported_-Features | 0x0087 | *none* | Status,<br>Max_Page,<br>LE_Features |

**Description:**

This command requests the supported LE features for the Controller.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_All_Local_Supported_Features command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_All_Local_Supported_Features command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Max_Page:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of the highest-numbered page of the supported LE features that contains at least one bit set to 1.<br>Range: 0x00 to 0x0A |

*LE_Features:* *Size: 248 octets*

| Value | Parameter Description |
|---|---|
| 0xXX...XX | Bit Mask List of the supported LE features. See [Vol 6] Part B, Section 4.6. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_All_Local_Supported_Features command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.129 LE Read All Remote Features command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_All_Remote_Features | 0x0088 | Connection_Handle,<br>Pages_Requested | *none* |

**Description:**

This command requests, from the remote device identified by the Connection_Handle, the features used on the connection and the features supported by the remote device. For details see [Vol 6] Part B, Section 4.6.

This command may be issued on both the Central and Peripheral.

The Pages_Requested parameter specifies the highest-numbered page of features that the Host requires. The Controller shall obtain all pages up to the lesser of Pages_Requested and the highest page number on the remote device that contains at least one bit set to 1, and may obtain some or all higher-numbered pages.

If a connection already exists between the two devices and the features have already been fetched on that connection, then the Controller may use a cached copy of the features.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Another HCI_LE_Read_All_Remote_Features command is pending. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Pages_Requested:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of the highest-numbered page of features that the Host requires and the Controller shall obtain.<br>Range: 0x00 to 0x0A |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Read_All_Remote_Features command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the procedure to determine the remote features or has determined that it will be using a cached copy, the Controller shall send an HCI_LE_Read_All_Remote_Features_Complete event to the Host.

The HCI_LE_Read_All_Remote_Features_Complete event contains the status of this command and the parameters describing the features supported by the remote device.

### 7.8.130 LE CS Read Local Supported Capabilities command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Read_Local_-Supported_Capabilities [v2] | 0x00A5 | *none* | Status,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability,<br>T_IP2_IPT_Times_Supported,<br>T_SW_IPT_Times_Supported,<br>RTT_2M_AA_Only_N,<br>RTT_2M_Sounding_N,<br>RTT_2M_Random_Sequence_N |

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Read_Local_-Supported_Capabilities [v1] | 0x0089 | *none* | Status,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability |

**Description:**

This command allows a Host to read the CS capabilities that are supported by the local Controller. This command may be used along with the local supported features to provide additional details of the supported CS capabilities.

The Num_Config_Supported parameter indicates the number of CS configurations that are supported per connection by the Controller.

The Max_Consecutive_Procedures_Supported parameter indicates the maximum number of consecutive CS procedures that are supported by the local Controller.

The Num_Antennae_Supported parameter indicates the number of antenna elements that are available for CS tone exchanges.

The Max_Antenna_Paths_Supported parameter indicates the maximum number of antenna paths that are supported by the local Controller for CS tone exchanges.

The Roles_Supported parameter indicates the CS roles that are supported by the local Controller.

The Modes_Supported parameter indicates the optional CS modes that are supported by the local Controller.

The RTT_Capability, RTT_AA_Only_N, RTT_2M_AA_Only_N, RTT_Sounding_N, RTT_2M_Sounding_N, RTT_Random_Sequence_N, and RTT_2M_Random_Sequence_N parameters indicate the time-of-flight accuracy as described in [Vol 6] Part B, Section 2.4.2.44.

The NADM_Sounding_Capability and NADM_Random_Capability parameters indicate the support by the local Controller for reporting Normalized Attack Detector Metric (NADM) when a CS_SYNC with a sounding sequence or random sequence is received.

The CS_SYNC_PHYs_Supported parameter indicates the optional transmit and receive PHYs that are supported by the local Controller for CS_SYNC exchanges as described in [Vol 6] Part H, Section 4.3.

The Subfeatures_Supported parameter indicates which of the following optional subfeatures are supported by the local Controller:

- A Frequency Actuation Error of zero for all allowed CS channels relative to mode-0 transmissions when in the reflector role as described in [Vol 6] Part A, Section 3.5.
- Channel Selection Algorithm #3c as described in [Vol 6] Part H, Section 4.1.4.2.
- Phase-based ranging from a sounding sequence as described in [Vol 6] Part H, Section 3.3.1.
- IPT in the CS reflector as described in [Vol 6] Part H, Section 4.3.3.
- RTT_PHY support denoting the setting of the RTT accuracy requirements for optional LE 2M and LE 2M 2BT PHYs separate from the RTT accuracy requirements for the mandatory LE 1M PHY as described in [Vol 6] Part B, Section 2.4.2.44. This bit shall only be set to 1 if [v2] or above of this command is being used.

The T_IP1_Times_Supported, T_IP2_Times_Supported, T_FCS_Times_Supported, T_PM_Times_Supported, and T_SW_Time_Supported parameters indicate the supported optional time durations used in CS steps as described in [Vol 6] Part H, Section 4.3.

The TX_SNR_Capability parameter indicates the supported SNR levels used for the CS_SYNC packets used in mode-1 and mode-3 steps as described in [Vol 6] Part A, Section 3.1.3.

The T_IP2_IPT_Times_Supported and T_SW_IPT_Times_Supported parameters indicate the supported optional time durations used in CS steps if the CS reflector

supports the Channel Sounding Enhancement #1 feature and the IPT capability (see [Vol 6] Part H, Section 4.3).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The local Channel Sounding (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.33.4) is not set. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Read_Local_Supported_Capabilities command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Read_Local_Supported_Capabilities command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Num_Config_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of CS configurations supported per connection |
| All other values | Reserved for future use |

*Max_Consecutive_Procedures_Supported:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Support for both a fixed number of consecutive CS procedures and for an indefinite number of CS procedures until termination. |
| 0x0001 to 0xFFFF | Maximum number of consecutive CS procedures supported. |

*Num_Antennae_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of antennae supported |
| All other values | Reserved for future use |

*Max_Antenna_Paths_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Maximum number of antenna paths supported |
| All other values | Reserved for future use |

*Roles_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Initiator |
| 1 | Reflector |
| All other bits | Reserved for future use |

*Modes_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Mode-3 |
| All other bits | Reserved for future use |

*RTT_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | f the RTT_AA_Only_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 1 | If the RTT_Sounding_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |

| Bit Number | Parameter Description |
|---|---|
| 2 | If the RTT_Random_Sequence_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 3 | If the RTT_2M_AA_Only_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement. |
| 4 | If the RTT_2M_Sounding_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement. |
| 5 | If the RTT_2M_Random_Sequence_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement. |
| All other bits | Reserved for future use |

*RTT_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA-only not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the time-of-flight accuracy requirements |

*NADM_Sounding_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| All other bits | Reserved for future use |

*NADM_Random_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| All other bits | Reserved for future use |

*CS_SYNC_PHYs_Supported:* *Size: 1 octet*

| Bit Number | *Parameter Description* |
|---|---|
| 1 | LE 2M PHY |
| 2 | LE 2M 2BT PHY |
| All other bits | Reserved for future use |

*Subfeatures_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | RFU |
| 1 | CS with a Frequency Actuation Error of zero relative to mode-0 transmissions in the reflector role |
| 2 | CS Channel Selection Algorithm #3c |
| 3 | CS phase-based ranging from an RTT sounding sequence |
| 4 | IPT in the CS reflector |
| 5 | CS RTT accuracy specified on a per PHY basis |
| All other bits | Reserved for future use |

*T_IP1_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other bits | Reserved for future use |

*T_IP2_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other bits | Reserved for future use |

*T_FCS_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 15 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| 7 | 100 µs supported |
| 8 | 120 µs supported |
| All other values | Reserved for future use |

*T_PM_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| All other values | Reserved for future use |

*T_SW_Time_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*TX_SNR_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 18 dB supported |
| 1 | 21 dB supported |
| 2 | 24 dB supported |
| 3 | 27 dB supported |
| 4 | 30 dB supported |
| All other bits | Reserved for future use |

*T_IP2_IPT_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other values | Reserved for future use |

*T_SW_IPT_Times_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*RTT_2M_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA-only not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

*RTT_2M_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

*RTT_2M_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Read_Local_Supported_Capabilities command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.131 LE CS Read Remote Supported Capabilities command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Read_Remote_Supported_-Capabilities | 0x008A | Connection_Handle | *none* |

**Description:**

This command allows a Host to query the CS capabilities that are supported by the remote Controller. If no Channel Sounding Capability Exchange procedure has been initiated on the ACL connection specified by the Connection_Handle and if no prior HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command has been issued by the Host, then the Controller shall initiate a Channel Sounding Capabilities Exchange procedure on the ACL. Otherwise, the Controller may use a cached copy of the capabilities of the remote device.

If this command is issued on the Central or Peripheral before the devices have performed the Feature Exchange procedure, then the Controller of the Central or Peripheral shall complete that procedure before initiating the Channel Sounding Capability Exchange procedure (see [Vol 6] Part B, Section 5.1.24).

The Host may store a copy of the remote device's capabilities and write the remote capabilities in the local Controller when it reconnects to the same remote device by using the HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The local or remote Channel Sounding (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.33.4) is not set. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not exist or is not for an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Read_Remote_Supported_Capabilities command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the Channel Sounding Capability Exchange procedure with the remote Controller or has a cached copy of the capabilities of the remote Controller, the Controller shall generate an HCI_LE_CS_Read_Remote_Supported_Capabilities_Complete event.

### 7.8.132 LE CS Write Cached Remote Supported Capabilities command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Write_-Cached_Remote_-Supported_Capabilities [v2] | 0x00A6 | Connection_Handle,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability,<br>T_IP2_IPT_Times_Supported,<br>T_SW_IPT_Times_Supported,<br>RTT_2M_AA_Only_N,<br>RTT_2M_Sounding_N,<br>RTT_2M_Random_Sequence_N | Status,<br>Connection_Handle |

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Write_-Cached_Remote_-Supported_Capabilities [v1] | 0x008B | Connection_Handle,<br>Num_Config_Supported,<br>Max_Consecutive_Procedures_Supported,<br>Num_Antennae_Supported,<br>Max_Antenna_Paths_Supported,<br>Roles_Supported,<br>Modes_Supported,<br>RTT_Capability,<br>RTT_AA_Only_N,<br>RTT_Sounding_N,<br>RTT_Random_Sequence_N,<br>NADM_Sounding_Capability,<br>NADM_Random_Capability,<br>CS_SYNC_PHYs_Supported,<br>Subfeatures_Supported,<br>T_IP1_Times_Supported,<br>T_IP2_Times_Supported,<br>T_FCS_Times_Supported,<br>T_PM_Times_Supported,<br>T_SW_Time_Supported,<br>TX_SNR_Capability | Status,<br>Connection_Handle |

**Description:**

This command allows a Host to write the cached copy of the CS capabilities that are supported by the remote Controller for the connection identified by the Connection_Handle parameter.

The Num_Config_Supported parameter indicates the number of CS configurations that are supported per connection by the remote Controller.

The Max_Consecutive_Procedures_Supported parameter indicates the maximum number of consecutive CS procedures that are supported by the remote Controller.

The Num_Antennae_Supported parameter indicates the number of antenna elements that are available for CS tone exchanges.

The Max_Antenna_Paths_Supported parameter indicates the maximum number of antenna paths that are supported by the local Controller for CS tone exchanges.

The Roles_Supported parameter indicates the CS roles that are supported by the remote Controller.

The Modes_Supported parameter indicates the optional CS modes that are supported by the remote Controller.

The RTT_Capability, RTT_AA_Only_N, RTT_2M_AA_Only_N, RTT_Sounding_N, RTT_2M_Sounding_N, RTT_Random_Sequence_N, and RTT_2M_Random_Sequence_N parameters indicate the time-of-flight accuracy as described in [Vol 6] Part B, Section 2.4.2.44.

The NADM_Sounding_Capability and NADM_Random_Capability parameters indicate the support by the remote Controller for reporting Normalized Attack Detector Metric (NADM) when a CS_SYNC with a sounding sequence or random sequence is received.

The CS_SYNC_PHYs_Supported parameter indicates the optional transmit and receive PHYs that are supported by the remote Controller for CS_SYNC exchanges as described in [Vol 6] Part H, Section 4.3.

The Subfeatures_Supported parameter indicates which of the following optional subfeatures are supported by the remote Controller:

- A Frequency Actuation Error of zero for all allowed CS channels relative to mode-0 transmissions when in the reflector role as described in [Vol 6] Part A, Section 3.5.
- Channel Selection Algorithm #3c as described in [Vol 6] Part H, Section 4.1.4.2.
- Phase-based ranging from a sounding sequence as described in [Vol 6] Part H, Section 3.3.1.
- IPT in the CS reflector as described in [Vol 6] Part H, Section 4.3.3.
- RTT_PHY support denoting the setting of the RTT accuracy requirements for optional LE 2M and LE 2M 2BT PHYs separate from the RTT accuracy requirements for the mandatory LE 1M PHY as described in [Vol 6] Part B, Section 2.4.2.44. This bit shall only be set to 1 if [v2] or above of this command is being used.

The T_IP1_Times_Supported, T_IP2_Times_Supported, T_FCS_Times_Supported, T_PM_Times_Supported, and T_SW_Time_Supported parameters indicate the supported optional time durations used in CS steps as described in [Vol 6] Part H, Section 4.3.

The TX_SNR_Capability parameter indicates the supported SNR levels used for the CS_SYNC packets used in mode-1 and mode-3 steps as described in [Vol 6] Part A, Section 3.1.3.

The T_IP2_IPT_Times_Supported and T_SW_IPT_Times_Supported parameters indicate the supported optional time durations used in CS steps if the CS reflector

supports the Channel Sounding Enhancement #1 feature and the IPT capability (see [Vol 6] Part H, Section 4.3).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | An LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU has been received from the remote Controller. | *Command Disallowed* (0x0C) |
| MC | A CS configuration has been created in the local Controller. | *Command Disallowed* (0x0C) |
| MC | The Channel Sounding (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.33.4) is not set. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Missing Parameters:**

When a version of this command is issued that does not include all the parameters, the following values shall be used for any missing parameters:

| Parameter | Value |
|---|---|
| T_IP2_IPT_Times_Supported | 0x0000 |
| T_SW_IPT_Times_Supported | 0x00 |
| RTT_2M_AA_Only_N | Equal to the value used for RTT_AA_Only_N |
| RTT_2M_Sounding_N | Equal to the value used for RTT_Sounding_N |
| RTT_2M_Random_Sequence_N | Equal to the value used for RTT_Random_Sequence_N |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |
| All other values | Reserved for future use |

*Num_Config_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of CS configurations supported per connection |
| All other values | Reserved for future use |

*Max_Consecutive_Procedures_Supported:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Support for both a fixed number of consecutive CS procedures and for an indefinite number of CS procedures until termination. |
| 0x0001 to 0xFFFF | Maximum number of consecutive CS procedures supported. |

*Num_Antennae_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Number of antennae supported |
| All other values | Reserved for future use |

*Max_Antenna_Paths_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Maximum number of antenna paths supported |
| All other values | Reserved for future use |

*Roles_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Initiator |
| 1 | Reflector |
| All other bits | Reserved for future use |

*Modes_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Mode-3 |
| All other bits | Reserved for future use |

*RTT_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | If the RTT_AA_Only_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 1 | If the RTT_Sounding_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY. |
| 2 | If the RTT_Random_Sequence_N field is set to 0, then the bit shall be ignored. Otherwise:<br>0 = the RTT_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement.<br>If Subfeatures_Supported bit 5 is 0, applies to all supported PHYs.<br>If Subfeatures_Supported bit 5 is 1, applies only to the LE 1M PHY |
| 3 | If the RTT_2M_AA_Only_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_AA_Only_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_AA_Only_N field refers to the 10 ns time-of-flight accuracy requirement. |
| 4 | If the RTT_2M_Sounding_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Sounding_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Sounding_N field refers to the 10 ns time-of-flight accuracy requirement. |

| Bit Number | Parameter Description |
|---|---|
| 5 | If the RTT_2M_Random_Sequence_N field or Subfeatures_Supported bit 5 is set to 0, then the bit shall be ignored. Otherwise this bit applies to the LE 2M and LE 2M 2BT PHYs, if supported.<br>0 = the RTT_2M_Random_Sequence_N field refers to the 150 ns time-of-flight accuracy requirement.<br>1 = the RTT_2M_Random_Sequence_N field refers to the 10 ns time-of-flight accuracy requirement. |
| All other bits | Reserved for future use |

*RTT_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA-only not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*RTT_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements |

*NADM_Sounding_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with sounding sequence is received |
| All other bits | Reserved for future use |

*NADM_Random_Capability:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | Support for Phase-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| 1 | Support for Amplitude-based Normalized Attack Detector Metric when a CS_SYNC with random sequence is received |
| All other bits | Reserved for future use |

*CS_SYNC_PHYs_Supported:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 1 | LE 2M PHY |
| 2 | LE 2M 2BT PHY |
| All other bits | Reserved for future use |

*Subfeatures_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | RFU |
| 1 | CS with a Frequency Actuation Error of zero relative to mode-0 transmissions in the reflector role |
| 2 | CS Channel Selection Algorithm #3c |
| 3 | CS phase-based ranging from an RTT sounding sequence |
| 4 | CS IPT in the CS reflector |
| 5 | CS RTT accuracy specified on a per PHY basis |
| All other bits | Reserved for future use |

*T_IP1_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |

| Bit Number | Parameter Description |
|---|---|
| 6 | 80 µs supported |
| All other bits | Reserved for future use |

*T_IP2_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other bits | Reserved for future use |

*T_FCS_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 15 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| 7 | 100 µs supported |
| 8 | 120 µs supported |
| All other values | Reserved for future use |

*T_PM_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| All other values | Reserved for future use |

*T_SW_Time_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*TX_SNR_Capability:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | 18 dB supported |
| 1 | 21 dB supported |
| 2 | 24 dB supported |
| 3 | 27 dB supported |
| 4 | 30 dB supported |
| All other bits | Reserved for future use |

*T_IP2_IPT_Times_Supported:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | 10 µs supported |
| 1 | 20 µs supported |
| 2 | 30 µs supported |
| 3 | 40 µs supported |
| 4 | 50 µs supported |
| 5 | 60 µs supported |
| 6 | 80 µs supported |
| All other values | Reserved for future use |

*T_SW_IPT_Times_Supported:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*RTT_2M_AA_Only_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA-only not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

*RTT_2M_Sounding_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Sounding not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

*RTT_2M_Random_Sequence_N:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT Random Sequence not supported on either the LE 2M or LE 2M 2BT PHY |
| 0x01 to 0xFF | Number of CS_SYNC exchanges needed to satisfy the accuracy requirements on the LE 2M or the LE 2M 2BT PHY |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |
| All other values | Reserved for future use |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.133 LE CS Security Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Security_Enable | 0x008C | Connection_Handle | *none* |

**Description:**

This command is used by a Host to start or restart the Channel Sounding Security Start procedure in the local Controller for the ACL connection identified by the Connection_Handle parameter.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |
| MC | Connection_Handle identifies an ACL where the Controller is the Peripheral. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle identifies an unencrypted ACL. | *Insufficient Security* (0x2F) |
| MC | The Channel Sounding (Host Support) feature bit (see [Vol 6] Part B, Section 4.6.33.4) is not set. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Security_Enable command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the Channel Sounding Security Start procedure with the remote Controller, the Controller shall generate an LE_CS_Security_Enable_Complete event.

### 7.8.134 LE CS Set Default Settings command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Set_Default_Settings | 0x008D | Connection_Handle,<br>Role_Enable,<br>CS_SYNC_Antenna_Selection,<br>Max_TX_Power | Status,<br>Connection_Handle |

**Description:**

This command is used by a Host to set default CS settings in the local Controller for the connection identified by the Connection_Handle parameter. The default settings specify that all roles are disabled in a Controller and CS_SYNC_Antenna_Selection is set to 0x01.

The Role_Enable parameter is used to enable or disable the CS roles in the local Controller.

The CS_SYNC_Antenna_Selection parameter indicates the antenna identifiers to be used for transmitting and receiving CS_SYNC packets.

The Max_TX_Power parameter is used to set the maximum output power, EIRP, to be used for all CS transmissions. If the value provided in this parameter is higher than the maximum output power supported by the Controller, then the Controller shall use the maximum output power that it supports. If the Controller is unable to use the exact output power requested by the Host, then the Controller shall use an output power that is lower but closest to the requested value.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The bit for a role which has a valid CS configuration present is set to 0 in Role_Enable. | *Invalid HCI Command Parameters* (0x12) |
| MC | CS_SYNC_Antenna_Selection indicates an unsupported antenna identifier. | *Unsupported Feature or Parameter Value* (0x11) |

| Type | Condition | Error code |
|---|---|---|
| MC | The bit for an unsupported role is set to 1 in Role_Enable. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Role_Enable:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Initiator role is enabled |
| 1 | Reflector role is enabled |
| All other values | Reserved for future use |

*CS_SYNC_Antenna_Selection:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Antenna identifier to be used for CS_SYNC packets by the local Controller |
| 0xFD | Antenna identifiers to be used, in repetitive order {0x01, 0x01, ..., Num_Antennae_Supported, Num_Antennae_Supported} for CS_SYNC packets by the local Controller, where Num_Antennae_Supported is the number of antenna elements available for CS tone exchanges |
| 0xFE | Antenna identifiers to be used, in repetitive order from 0x01 to Num_Antennae_Supported, for CS_SYNC packets by the local Controller, where Num_Antennae_Supported is the number of antenna elements available for CS tone exchanges |
| 0xFF | Host does not have a recommendation |
| All other values | Reserved for future use |

*Max_TX_Power:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The maximum transmit power level to be used for all CS transmissions<br>Range: -127 to 20<br>Units: dBm |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Set_Default_Settings command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Set_Default_Settings command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Set_Default_Settings command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.135 LE CS Read Remote FAE Table command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Read_Remote_FAE_Table | 0x008E | Connection_Handle | *none* |

**Description:**

This command is used by a Host to read the per-channel mode-0 Frequency Actuation Error table of the remote Controller.

The Host may store a copy of the remote device's Frequency Actuation Error table and write the remote Frequency Actuation Error table in the local Controller when it reconnects to the same remote device by using the HCI_LE_CS_Write_Cached_Remote_FAE_Table command.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The remote Controller supports a Frequency Actuation Error of zero relative to its mode-0 transmissions in the reflector role (No_FAE bit set as described in [Vol 6] Part B, Section 2.4.2.44). | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Read_Remote_FAE_Table command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the Channel Sounding Mode-0 FAE Table

Request procedure with the remote Controller, the Controller shall generate an LE_CS_Read_Remote_FAE_Table_Complete event.

### 7.8.136 LE CS Write Cached Remote FAE Table command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Write_Cached_Remote_-FAE_Table | 0x008F | Connection_Handle,<br>Remote_FAE_Table | Status,<br>Connection_Handle |

**Description:**

This command is used by a Host to write a cached copy of the per-channel mode-0 Frequency Actuation Error table of the remote device in the local Controller.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The remote Controller supports a Frequency Actuation Error of zero relative to its mode-0 transmissions in the reflector role (No_FAE bit set as described in [Vol 6] Part B, Section 2.4.2.44). | *Unsupported Feature or Parameter Value* (0x11) |
| MC | An LL_CS_FAE_RSP PDU has been received from the remote Controller. | *Command Disallowed* (0x0C) |
| MC | A CS configuration has been created in the local Controller. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0x0000 to 0x0EFF | Connection_Handle |
| 0x0FFF | Use for CS Test command |
| All other values | Reserved for future use |

*Remote_FAE_Table:* *Size: 72 octets*

| Value | Parameter Description |
|---|---|
| Variable | Per-channel mode-0 Frequency Actuation Error table of the local Controller as described in [Vol 6] Part B, Section 2.4.2.52. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Write_Cached_Remote_FAE_Table command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Write_Cached_Remote_FAE_Table command failed. See [Vol 1} Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Write_Cached_Remote_FAE_Table command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.137 LE CS Create Config command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Create_Config | 0x0090 | Connection_Handle,<br>Config_ID,<br>Create_Context,<br>Main_Mode_Type,<br>Sub_Mode_Type,<br>Min_Main_Mode_Steps,<br>Max_Main_Mode_Steps,<br>Main_Mode_Repetition,<br>Mode_0_Steps,<br>Role,<br>RTT_Type,<br>CS_SYNC_PHY,<br>Channel_Map,<br>Channel_Map_Repetition,<br>Channel_Selection_Type,<br>Ch3c_Shape,<br>Ch3c_Jump,<br>CS_Enhancements | *none* |

**Description:**

This command is used by a Host to create a new CS configuration or update an existing CS configuration with the identifier Config_ID on the connection identified by the Connection_Handle in the local and/or the remote Controller.

When the Create_Context parameter is set to 0x00, the CS configuration is written only in the local Controller. Otherwise when set to 0x01, the CS configuration is written in both the local and remote Controllers using the Channel Sounding Configuration procedure.

The Main_Mode_Type and the Sub_Mode_Type parameters indicate the CS modes (see [Vol 6] Part H, Table 4.11 for valid combinations) to be used during the CS procedure for the specified CS configuration.

The Min_Main_Mode_Steps and Max_Main_Mode_Steps parameters indicate the range of main mode CS steps to be executed before a submode CS step is executed during the CS procedure. When the Sub_Mode_Type parameter is set to 0xFF, the Min_Main_Mode_Steps and Max_Main_Mode_Steps parameters are reserved for future use.

The Main_Mode_Repetition parameter indicates the number of main mode CS steps repeated from the last CS subevent at the beginning of the current CS subevent.

The Mode_0_Steps parameter indicates the number of mode-0 CS steps to be included at the beginning of each CS subevent.

The Role parameter indicates the CS role for the local Controller for the specified CS configuration. The RTT_Type parameter indicates the RTT variant to be used during the CS procedure, and the CS_SYNC_PHY parameter indicates the PHY to be used for CS_SYNC exchanges during the CS procedure for the specified CS configuration.

The Channel_Map parameter indicates the channels to be used or unused during the CS procedure, and the Channel_Map_Repetition parameter indicates the number of times the channels specified by Channel_Map are to be repeated for non-mode-0 steps during the CS procedure (see [Vol 6] Part H, Section 4.1.4).

The Channel_Selection_Type parameter indicates the Channel Selection Algorithm to be used during the CS procedure for non-mode-0 steps. When the Channel_Selection_Type is set to 0x01, the Ch3c_Shape and the Ch3c_Jump parameters shall each be set to the selected shape and channels to be skipped as described in [Vol 6] Part H, Section 4.1.4.2. Otherwise, the Ch3c_Shape and the Ch3c_Jump parameters shall be ignored.

The CS_Enhancements parameter indicates which CS enhancements to be used during the CS procedure. For each enhancement that is not enabled, the corresponding bit shall be set to 0.

If the Host issues this command before a Channel Sounding Capability Exchange procedure has been completed or an HCI_LE_CS_Write_Cached_Remote_-Supported_Capabilities command has been issued for the connection identified by the Connection_Handle parameter, then the Controller may autonomously initiate the Channel Sounding Capability Exchange procedure.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Some parameters are not supported by the local or remote Controller. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | The parameters specify an invalid configuration according to [Vol 6] Part H (e.g., a combination of Main_Mode_Type and Sub_Mode_Type not specified in [Vol 6] Part H, Table 4.11). | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| M | Less than 15 channels are enabled. | *Invalid HCI Command Parameters* (0x12) |
| M | Any of bits 0, 1, 23, 24, 25, 77, or 78 are set to 1. | *Invalid HCI Command Parameters* (0x12) |
| MC | Role has not been enabled by a prior HCI_LE_CS_Set_Default_Settings command. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Handle identifies a connection where a Channel Sounding Capability Exchange procedure has been completed. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle identifies a connection where an HCI_LE_CS_Write_Cached_Remote_Supported_Capabilities command has been issued. | *Command Disallowed* (0x0C) |
| MC | Config_ID identifies a CS configuration that is already enabled. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

### Command parameters:

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*Create_Context:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write CS configuration in local Controller only |
| 0x01 | Write CS configuration in both local and remote Controller using Channel Sounding Configuration procedure |
| All other values | Reserved for future use |

*Main_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| All other values | Reserved for future use |

*Sub_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| 0xFF | Unused |
| All other values | Reserved for future use |

*Min_Main_Mode_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Minimum number of CS main mode steps to be executed before a submode step is executed |
| All other values | Reserved for future use |

*Max_Main_Mode_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Maximum number of CS main mode steps to be executed before a submode step is executed |
| All other values | Reserved for future use |

*Main_Mode_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x03 | The number of main mode steps taken from the end of the last CS subevent to be repeated at the beginning of the current CS subevent directly after the last mode-0 step of that event |
| All other values | Reserved for future use |

*Mode_0_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x03 | Number of CS mode-0 steps to be included at the beginning of each CS subevent |
| All other values | Reserved for future use |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Initiator |
| 0x01 | Reflector |
| All other values | Reserved for future use |

*RTT_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA-only |
| 0x01 | RTT with 32-bit sounding sequence |
| 0x02 | RTT with 96-bit sounding sequence |
| 0x03 | RTT with 32-bit random sequence |
| 0x04 | RTT with 64-bit random sequence |
| 0x05 | RTT with 96-bit random sequence |
| 0x06 | RTT with 128-bit random sequence |
| All other values | Reserved for future use |

*CS_SYNC_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE 2M 2BT PHY |
| All other values | Reserved for future use |

*Channel_Map:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXX | This parameter contains 80 1-bit fields.<br>The nth such field (in the range 0 to 78) contains the value for the CS channel index n.<br>Channel n is enabled for CS procedure = 1<br>Channel n is disabled for CS procedure = 0<br>The most significant bit (bit 79) is reserved for future use. |

*Channel_Map_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | The number of times the map represented by the Channel_Map field is to be cycled through for non-mode-0 steps within a CS procedure |
| All other values | Reserved for future use |

*Channel_Selection_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use Channel Selection Algorithm #3b for non-mode-0 CS steps |
| 0x01 | Use Channel Selection Algorithm #3c for non-mode-0 CS steps |
| All other values | Reserved for future use |

*Ch3c_Shape:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use Hat shape for user-specified channel sequence |
| 0x01 | Use X shape for user-specified channel sequence |
| All other values | Reserved for future use |

*Ch3c_Jump:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x08 | Number of channels skipped in each rising and falling sequence |
| All other values | Reserved for future use |

*CS_Enhancements:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | IPT enabled in the CS reflector |
| All other bits | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Create_Config command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the Channel Sounding Configuration procedure with the remote Controller or when the Create_Context parameter is set to 0x00, the Controller shall generate an LE_CS_Config_Complete event.

### 7.8.138 LE CS Remove Config command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Remove_Config | 0x0091 | Connection_Handle,<br>Config_ID | *none* |

**Description:**

This command is used to remove a CS configuration identified by Config_ID from the local Controller for the connection identified by the Connection_Handle parameter. When the Host issues this command, the local Controller shall initiate a Channel Sounding Configuration procedure to remove the CS configuration from both the local and remote device. The Controller shall also delete any CS procedure related parameters set using the HCI_LE_CS_Set_Procedure_Parameters command for the CS configuration identified by Config_ID.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Config_ID does not identify a CS configuration. | *Invalid HCI Command Parameters* (0x12) |
| MC | One or more CS procedures have been enabled. | *Command Disallowed* (0x0C) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Remove_Config command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has completed the Channel Sounding Configuration procedure to disable the configuration, the Controller shall generate an LE_CS_Config_Complete event.

### 7.8.139 LE CS Set Channel Classification command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Set_Channel_Classification | 0x0092 | Channel_Classification | Status |

**Description:**

This command is used by a Host to update the channel classification based on its local information. This channel classification persists until overwritten with a subsequent HCI_LE_CS_Set_Channel_Classification command or until the Controller is reset. The Controller may combine the channel classification information provided by the Host along with local channel classification information to send an updated CS channel map to the remote Controller.

If this command is used, then the Host should send updates within 10 seconds of knowing that the CS channel classification has changed.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | It is less than 1 second since the previous time this command was executed. | *Command Disallowed* (0x0C) |
| MC | Channel_Classification enables channels that are reserved for future use or enables fewer than 15 channels. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Channel_Classification:* *Size: 10 octets (79 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXX | This parameter contains 80 1-bit fields.<br>The nth such field (in the range 0 to 78) contains the value for the CS channel index n.<br>Channel n is enabled for CS procedure = 1<br>Channel n is disabled for CS procedure = 0<br>Channels n = 0, 1, 23, 24, 25, 77, and 78 shall be reserved for future use and shall be set to zero. At least 15 channels shall be enabled.<br>The most significant bit (bit 79) is reserved for future use. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Set_Channel_Classification command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Set_Channel_Classification command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Set_Channel_Classification command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.140 LE CS Set Procedure Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Set_Procedure_-Parameters | 0x0093 | Connection_Handle,<br>Config_ID,<br>Max_Procedure_Len,<br>Min_Procedure_Interval,<br>Max_Procedure_Interval,<br>Max_Procedure_Count,<br>Min_Subevent_Len,<br>Max_Subevent_Len,<br>Tone_Antenna_Config_Selection,<br>PHY,<br>Tx_Power_Delta,<br>Preferred_Peer_Antenna,<br>SNR_Control_Initiator,<br>SNR_Control_Reflector | Status,<br>Connection_Handle |

**Description:**

This command is used by a Host to set the parameters for the scheduling of one or more CS procedures by the local Controller, with the remote device for the CS configuration identified by Config_ID and the connection identified by the Connection_Handle parameter.

The Max_Procedure_Len parameter indicates the maximum duration of each CS procedure. The Min_Procedure_Interval and Max_Procedure_Interval parameters indicate the minimum and maximum interval period between consecutive CS procedures. The Max_Procedure_Count parameter indicates the maximum number of consecutive CS procedures to be scheduled as part of this measurement. If Max_Procedure_Count is set to 1, then Min_Procedure_Interval and Max_Procedure_Interval shall be ignored.

The Min_Subevent_Len and Max_Subevent_Len parameters indicate the minimum and maximum duration of each CS subevent during the CS procedure.

The values for Min_Procedure_Interval, Max_Procedure_Interval, Min_Subevent_Len, and Max_Subevent_Len are recommendations to the Controller which it may ignore.

The Tone_Antenna_Config_Selection parameter indicates the Antenna Configuration Index to be used in the CS procedure.

The power delta value Tx_Power_Delta indicates the recommended difference between the remote device's power level for the CS tones and CS_SYNC packets, and the power level for the PHY indicated by the PHY parameter. If the resulting power level goes below the minimum or goes above the maximum supported transmit power levels of the remote device, then the Controller may adjust the requested power delta value.

The Preferred_Peer_Antenna parameter indicates the preferred peer-ordered antenna elements to be used by the remote device for the antenna configuration denoted by the Tone_Antenna_Config_Selection parameter. The number of bits set in this field shall be greater than or equal to the number of antenna elements denoted by the Tone_Antenna_Config_Selection parameter.

The SNR_Control_Initiator parameter indicates the SNR control adjustment for the CS_SYNC transmissions of the initiator.

The SNR_Control_Reflector parameter indicates the SNR control adjustment for the CS_SYNC transmissions of the reflector.

The parameters specified by this command for the CS configuration identified by the Config_ID parameter become invalid after the Host issues the HCI_LE_CS_Remove_Config command for the given Config_ID.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The parameters exceed the CS capabilities or any coexistence constraints. | *Rejected Due To Limited Resources* (0x0D). |
| MC | Config_ID does not identify a CS configuration. | *Invalid HCI Command Parameters* (0x12) |
| MC | Config_ID identifies a CS configuration that has been removed. | *Invalid HCI Command Parameters* (0x12) |
| MC | A CS procedure measurement is already enabled for Config_ID in the Controller. | *Command Disallowed* (0x0C) |
| MC | The number of channels available for Channel Sounding before the start of a new CS procedure measurement is less than 15. | *Insufficient Channels* (0x48) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |

*Max_Procedure_Len:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum duration for each CS procedure<br>Range: 0x0001 to 0xFFFF<br>Time = N × 0.625 ms<br>Time range: 0.625 ms to 40.959375 s |

*Min_Procedure_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum number of connection events between consecutive CS procedures<br>Range: 0x0001 to 0xFFFF |

*Max_Procedure_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum number of connection events between consecutive CS procedures<br>Range: 0x0001 to 0xFFFF |

*Max_Procedure_Count:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | CS procedures to continue until disabled |
| 0xXXXX | Maximum number of CS procedures to be scheduled |

*Min_Subevent_Len:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Minimum suggested duration for each CS subevent in microseconds<br>Range: 1250 µs to 3.999999 s |

*Max_Subevent_Len:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Maximum suggested duration for each CS subevent in microseconds<br>Range: 1250 µs to 3.999999 s |

*Tone_Antenna_Config_Selection:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x07 | Antenna Configuration Index as described in [Vol 6] Part A, Section 5.3 |
| All other values | Reserved for future use |

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |
| 0x04 | LE Coded PHY with S=2 data coding |
| All other values | Reserved for future use |

*Tx_Power_Delta:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Transmit power delta, in signed dB, to indicate the recommended difference between the remote device's power level for the CS tones and CS_SYNC packets and the existing power level for the PHY indicated by the PHY parameter |
| 0x80 | Host does not have a recommendation for transmit power delta |

*Preferred_Peer_Antenna:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Use first ordered antenna element |
| 1 | Use second ordered antenna element |
| 2 | Use third ordered antenna element |
| 3 | Use fourth ordered antenna element |
| All other values | Reserved for future use |

*SNR_Control_Initiator:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SNR control adjustment of 18 dB. |
| 0x01 | SNR control adjustment of 21 dB. |
| 0x02 | SNR control adjustment of 24 dB. |
| 0x03 | SNR control adjustment of 27 dB. |
| 0x04 | SNR control adjustment of 30 dB. |
| 0xFF | SNR control is not to be applied. |
| All other values | Reserved for future use |

*SNR_Control_Reflector:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SNR control adjustment of 18 dB. |
| 0x01 | SNR control adjustment of 21 dB. |
| 0x02 | SNR control adjustment of 24 dB. |
| 0x03 | SNR control adjustment of 27 dB. |
| 0x04 | SNR control adjustment of 30 dB. |
| 0xFF | SNR control is not to be applied. |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Set_Procedure_Parameters command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Set_Procedure_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Set_Procedure_Parameters command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.141 LE CS Procedure Enable command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Procedure_Enable | 0x0094 | Connection_Handle,<br>Config_ID,<br>Enable | *none* |

**Description:**

This command is used by a Host to enable or disable the scheduling of CS procedures by the local Controller, with the remote device for the connection identified by the Connection_Handle parameter.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | A corresponding HCI_LE_CS_Set_Procedure_Parameters command has not been issued for the same Config_ID. | *Command Disallowed* (0x0C) |
| MC | Config_ID does not identify a CS configuration. | *Invalid HCI Command Parameters* (0x12) |
| MC | Config_ID identifies a CS configuration that has been removed. | *Invalid HCI Command Parameters* (0x12) |
| MC | The CS procedure parameters associated with Config_ID exceed any scheduling or coexistence constraints. | *Rejected Due To Limited Resources* (0x0D). |
| MC | The CS configuration identified by Config_ID is already enabled. | *Command Disallowed* (0x0C) |
| MC | The number of channels available for Channel Sounding before the start of a new CS procedure measurement is less than 15. | *Insufficient Channels* (0x48) |
| MC | Connection_Handle does not identify an ACL. | *Unknown Connection Identifier* (0x02) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range 0x0000 to 0x0EFF |

*Config_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | CS configuration identifier<br>Range: 0 to 3 |
| All other values | Reserved for future use |

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CS procedures are to be disabled |
| 0x01 | CS procedures are to be enabled |
| All other values | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Procedure_Enable command, the Controller shall send the HCI_Command_Status event to the Host. When the locally initiated Channel Sounding Start procedure has completed or when the Controller has received the LL_CS_IND PDU, it shall generate an LE_CS_Procedure_Enable_Complete event. When the Host has issued a command to disable an active CS procedure, the Controller shall generate an LE_CS_Procedure_Enable_Complete event after any pending CS subevent results have been sent to the Host and the LL_CS_TERMINATE_RSP PDU has been successfully sent or received.

### 7.8.142 LE CS Test command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Test | 0x0095 | Main_Mode_Type,<br>Sub_Mode_Type,<br>Main_Mode_Repetition,<br>Mode_0_Steps,<br>Role,<br>RTT_Type,<br>CS_SYNC_PHY,<br>CS_SYNC_Antenna_Selection,<br>Subevent_Len,<br>Subevent_Interval,<br>Max_Num_Subevents,<br>Transmit_Power_Level,<br>T_IP1_Time,<br>T_IP2_Time,<br>T_FCS_Time,<br>T_PM_Time,<br>T_SW_Time,<br>Tone_Antenna_Config_Selection,<br>CS_Enhancements,<br>SNR_Control_Initiator,<br>SNR_Control_Reflector,<br>DRBG_Nonce,<br>Channel_Map_Repetition,<br>Override_Config,<br>Override_Parameters_Length,<br>Override_Parameters_Data | Status |

**Description:**

This command is used to start a CS test where the Implementation Under Test (IUT) is placed in the role of either the initiator or reflector. The first mode-0 channel in the list is used as the starting channel for the test. At the beginning of any test, the IUT in the reflector role shall listen on the first mode-0 channel until it receives the first transmission from the initiator. Similarly, with the IUT in the initiator role, the Lower Tester will start by listening on the first mode-0 channel and the IUT shall transmit on

that channel for the first half of the first CS step. Thereafter, the parameters of this command describe the required transmit and receive behavior for the CS test.

This command is used to schedule a single CS procedure that consists of one or more CS subevents. After the channels contained in the Channel map or the Channel array in case of an override have been used the number of times indicated by Channel_Map_-Repetition to schedule CS steps, or the maximum number of allowed steps in a CS procedure has been reached, or the maximum number of subevents indicated by the Max_Num_Subevents parameter (if non-zero) has been reached, or the maximum number of allowed subevents in a CS procedure has been reached, the CS procedure will end.

The Main_Mode_Type and the Sub_Mode_Type parameters indicate the CS modes to be used during the CS procedure for the specified CS configuration.

The Main_Mode_Repetition parameter indicates the number of main mode CS steps repeated from the last CS subevent at the beginning of the current CS subevent.

The Mode_0_Steps parameter indicates the number of mode-0 CS steps to be included at the beginning of each CS subevent.

The Role parameter indicates the CS role for the local Controller.

The RTT_Type parameter indicates the RTT type and payload length to be used during the CS procedure.

The CS_SYNC_PHY parameter indicates the PHY to be used for CS_SYNC exchanges during the CS procedure.

The CS_SYNC_Antenna_Selection parameter indicates the antenna identifier to be used for transmitting and receiving CS_SYNC packets.

The Subevent_Len parameter indicates the maximum length of a CS subevent.

The Subevent_Interval parameter indicates the gap between the start of consecutive CS subevents. If Subevent_Interval is set to zero, then only one CS subevent is executed in the CS test. Otherwise, the value for Subevent_Interval shall be greater than or equal to the sum of the Subevent_Len selected plus T_MES.

The Max_Num_Subevents parameter indicates the maximum number of subevents that are in the procedure (0x00 indicates no limit). If Max_Num_Subevents is set to 0x00, then the Subevent_Interval parameter shall be ignored.

The Transmit_Power_Level parameter indicates the transmit power level used for the CS procedure.

The T_IP1_Time, T_IP2_Time, T_FCS_Time, T_PM_Time, and T_SW_Time parameters indicate the time durations used in CS steps as described in [Vol 6] Part H, Section 4.3.

The Tone_Antenna_Config_Selection parameter indicates the Antenna Configuration Index used during antenna switching during the tone phases of CS steps as described in [Vol 6] Part A, Section 5.3.

The CS_Enhancements parameter indicates which CS enhancements to be used during the CS procedure. For each enhancement that is not enabled, the corresponding bit shall be set to 0. If IPT is enabled, then the T_IP2_Time and T_SW_Time parameters indicate the time durations used in CS steps.

The SNR_Control_Initiator parameter indicates the SNR control adjustment for the CS_SYNC transmissions of the initiator.

The SNR_Control_Reflector parameter indicates the SNR control adjustment for of the CS_SYNC transmissions of the reflector.

The DRBG_Nonce parameter specifies octets 14 and 15 of the initial value of the DRBG nonce, $V_{DRBG}$, used in calls to the random bit generation function described in [Vol 6] Part E, Section 3.1.6. The remaining octets of the initial nonce value $V_{DRBG}$ are set to 0x00. All octets of the initial temporal key $K_{DRBG}$ are set to 0x00. The most significant bit of this parameter is stored in the most significant bit of the octet 14 of the DRBG nonce. The least significant bit of this parameter is stored in the least significant bit of octet 15 of the DRBG nonce.

The Channel_Map_Repetition field shall indicate the number of times the Channel_Map is cycled through for non-mode-0 steps within a CS procedure. The Channel_Map_Repetition field shall be greater than or equal to 1. The Channel_Map content is selected based on the setting of the Override_Config bit 0 as described below.

The Override_Config parameter indicates which CS parameters are not derived from the DRBG but determined from the Override_Parameters_Data parameter in this command.

If the Override_Config bit 2 corresponding to CS submode insertion is not set, then the number of main mode CS steps to be executed before a submode CS step during the CS procedure is determined using DRBG with following default values:

- Min_Main_Mode_Steps: 6
- Max_Main_Mode_Steps: 10

If the Override_Config bit 10 corresponding to the Stable Phase test is set, then the procedure is replaced with the Stable Phase test as described in [Vol 6] Part F, Section 2.4.

The Override_Parameters_Data is a variable sized object with a length indicated by Override_Parameters_Length parameter whose contents are determined by the bits set in the Override_Config parameter.

An ongoing CS test can be stopped using the HCI_LE_CS_Test_End command (see Section 7.8.143).

The CS test is considered complete when all the results of the CS procedure initiated by the CS test have been reported to the Host.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | A CS test is already enabled and has not completed. | *Command Disallowed* (0x0C) |

**Command parameters:**

*Main_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| All other values | Reserved for future use |

*Sub_Mode_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| 0xFF | Unused |
| All other values | Reserved for future use |

*Main_Mode_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x03 | The number of main mode steps taken from the end of the last CS subevent to be repeated at the beginning of the current CS subevent directly after the last mode-0 step of that event |
| All other values | Reserved for future use |

*Mode_0_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x03 | Number of CS mode-0 steps to be included at the beginning of the test CS subevent |
| All other values | Reserved for future use |

*Role:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Initiator |
| 0x01 | Reflector |
| All other values | Reserved for future use |

*RTT_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | RTT AA Only |
| 0x01 | RTT with 32-bit sounding sequence |
| 0x02 | RTT with 96-bit sounding sequence |
| 0x03 | RTT with 32-bit random sequence |
| 0x04 | RTT with 64-bit random sequence |
| 0x05 | RTT with 96-bit random sequence |
| 0x06 | RTT with 128-bit random sequence |
| All other values | Reserved for future use |

*CS_SYNC_PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE 2M 2BT PHY |
| All other values | Reserved for future use |

*CS_SYNC_Antenna_Selection:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x04 | Antenna identifier to be used for CS_SYNC packets, including mode-0 packets |
| All other values | Reserved for future use |

*Subevent_Len:* *Size: 3 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | CS subevent length in units of microseconds<br>Range: 1250 µs to 3.999999 s |

*Subevent_Interval:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | Single CS subevent |
| 0xXXXX | Gap between the start of two consecutive CS subevents<br>Units: 0.625 ms<br>Range: 0x0002 to 0xFFFF |

*Max_Num_Subevents:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No restriction on the number of subevents in the procedure. |
| 0x01 to 0x20 | The maximum allowed number of subevents in the procedure. |
| All other values | Reserved for future use. |

*Transmit_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Set transmitter to the specified or nearest transmit power level<br>Range: -127 to +20<br>Units: dBm |
| 0x7E | Set transmitter to minimum transmit power level |
| 0x7F | Set transmitter to maximum transmit power level |
| All other values | Reserved for future use |

*T_IP1_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0A, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, or 0x91 | Interlude time in microseconds between the CS_SYNC packets used in mode-0 and mode-1 steps |
| All other values | Reserved for future use |

*T_IP2_Time:* *Size: 1 octet*

| Value | ***Parameter Description*** |
|---|---|
| 0x0A, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, or 0x91 | Interlude time in microseconds between the CS tones |
| All other values | Reserved for future use |

*T_FCS_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0F, 0x14, 0x1E, 0x28, 0x32, 0x3C, 0x50, 0x64, 0x78, or 0x96 | Time in microseconds for frequency changes |
| All other values | Reserved for future use |

*T_PM_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x0A, 0x14, or 0x28 | Time in microseconds for the phase measurement period of the CS tones |
| All other values | Reserved for future use |

*T_SW_Time:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00, 0x01, 0x02, 0x04, or 0x0A | Time in microseconds for the antenna switch period of the CS tones |
| All other values | Reserved for future use |

*Tone_Antenna_Config_Selection:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x07 | Antenna Configuration Index used during antenna switching during the tone phases of CS steps as described in [Vol 6] Part A, Section 5.3 |
| All other values | Reserved for future use |

*CS_Enhancements:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | IPT is enabled in the CS reflector |
| All other bits | Reserved for future use |

*SNR_Control_Initiator:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SNR control adjustment of 18 dB. |
| 0x01 | SNR control adjustment of 21dB. |
| 0x02 | SNR control adjustment of 24 dB. |
| 0x03 | SNR control adjustment of 27 dB. |
| 0x04 | SNR control adjustment of 30 dB. |
| 0xFF | SNR control is not to be applied. |
| All other values | Reserved for future use |

*SNR_Control_Reflector:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SNR control adjustment of 18 dB. |
| 0x01 | SNR control adjustment of 21dB. |
| 0x02 | SNR control adjustment of 24 dB. |
| 0x03 | SNR control adjustment of 27 dB. |
| 0x04 | SNR control adjustment of 30 dB. |

| Value | Parameter Description |
|---|---|
| 0xFF | SNR control is not to be applied. |
| All other values | Reserved for future use |

*DRBG_Nonce:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The DRBG_Nonce value determines octets 14 and 15 of the initial value of the DRBG nonce. |

*Channel_Map_Repetition:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved for future use. |
| 0x01 to 0xFF | The number of times the channels indicated by the Channel_Map or the Channel field in the Override_Parameters_Data parameter is cycled through for non-mode-0 steps within a CS procedure. This value shall be greater than or equal to 1. |

*Override_Config:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | If the bit is set to 1, then the channel sequence for the subevent is determined by the values of Channel_Length and Channel[i] parameters.<br>If bit is set to 0, then the channel sequence for the subevent is determined by the values of Channel_Map, Channel_Selection_Type, Ch3c_Shape, and Ch3c_Jump. |
| 2 | The number of main mode CS steps to be executed before a submode CS step is executed during the CS procedure is determined by the value of Main_Mode_Steps parameter. This bit may only be set if Sub_Mode_Type is not set to Unused. |
| 3 | The transmission of tone extensions within each Mode-2 or Mode-3 step is determined by the value of T_PM_Tone_Ext parameter. |
| 4 | The antenna path permutation index for each Mode-2 or Mode-3 step is determined by the value of Antenna_Path_Permutation_Index parameter. |
| 5 | The CS Access Address of all packets sent by the initiator is determined by CS_SYNC_AA_Initiator parameter. The CS Access Address of all packets sent by the reflector is determined by CS_SYNC_AA_Reflector parameter. |
| 6 | The Marker positions for each CS_SYNC packet with a marker is determined by the value of SS_Marker1_Position and SS_Marker2_Position parameters. |
| 7 | The Marker value for each marker within a CS_SYNC packet is determined by SS_Marker_Value parameter. |

| Bit Number | Parameter Description |
|---|---|
| 8 | The payload of the CS_SYNC packet is determined by the value of CS_SYNC_Payload_Pattern parameter. |
| 10 | Stable Phase test |
| All other bits | Reserved for future use |

*Override_Parameters_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The Length of the Override_Parameters_Data variable object field. |

*Override_Parameters_Data:* *Size: Override_Parameters_Length octets*

| Parameter Description |
|---|
| Variable set of parameters which are present dependent on the bits set in the Override_Config parameter.<br>Note: This parameter has a variable length. |

The contents of the Override_Parameters_Data parameter is defined in sequence as follows

- When bit 0 of Override_Config is set, include the following parameters

  Channel_Length,

  Channel[i]
- When bit 0 of Override_Config is not set, include the following parameters
  - Channel_Map,
  - Channel_Selection_Type,
  - Ch3c_Shape,
  - Ch3c_Jump
- When bit 2 of Override_Config is set, include the following parameters

  Main_Mode_Steps
- When bit 3 of Override_Config is set, include the following parameters

  T_PM_Tone_Ext
- When bit 4 of Override_Config is set, include the following parameters

  Antenna_Path_Permutation_Index

- When bit 5 of Override_Config is set, include the following parameters

  CS_SYNC_AA_Initiator,

  CS_SYNC_AA_Reflector
- When bit 6 of Override_Config is set, include the following parameters

  SS_Marker1_Position,

  SS_Marker2_Position
- When bit 7 of Override_Config is set, include the following parameters

  SS_Marker_Value
- When bit 8 of Override_Config is set, include the following parameters

  CS_SYNC_Payload_Pattern,

  CS_SYNC_User_Payload

The parameters of the Override_Parameters_Data object are described below.

The Channel_Length and Channel[i] parameters indicate the channels to be used during the CS procedure for both mode-0 and non-mode-0 CS steps. The Channel[i] list is used independently for mode-0 and non-mode-0 steps.

The Channel_Map parameter shall indicate the CS channels to be used or unused in the CS procedure.

The Channel_Selection_Type parameter shall indicate which Channel Selection Algorithm to use when calculating the channel sequence for the CS procedure. The Ch3c_Shape parameter shall indicate which shape to use when calculating Channel Selection Algorithm #3c as described in [Vol 6] Part H, Section 4.1.4.2. If the Channel_Selection_Type parameter is not set to Channel Selection Algorithm #3c, then the Ch3c_Shape parameter shall be ignored.

The Ch3c_Jump parameter shall indicate which CSChannelJump value to use when calculating Channel Selection Algorithm #3c as described in [Vol 6] Part H, Section 4.1.4.2. If the Channel_Selection_Type parameter is not set to Channel Selection Algorithm #3c, then the Ch3c_Jump parameter shall be ignored.

The Main_Mode_Steps parameter indicates the number of main mode CS steps to be executed before a submode CS step is executed during the CS procedure.

The T_PM_Tone_Ext parameter indicates whether a physical transmission is present in the CS tone extension slot in the initiator and the reflector sides.

The Antenna_Path_Permutation_Index parameter indicates the antenna path permutation index used during antenna switching during the tone phases of CS steps as described in [Vol 6] Part H, Section 4.7.5.

The CS_SYNC_AA_Initiator and the CS_SYNC_AA_Reflector parameters indicate the access address used for CS_SYNC packets on the initiator and reflector sides. The least significant bit corresponds to the most significant bit of the CS Access Address.

When RTT_Type specifies a sounding sequence being used, the SS_Marker1_Position parameter indicates the bit number where the first marker of the sounding sequence starts, the SS_Marker2_Position indicates the bit number where the second marker of the sounding sequence starts, and the SS_Marker_Value parameter indicates the value of the sounding sequence marker.

When RTT_Type specifies a random sequence being used, a payload with the pattern indicated by the CS_SYNC_Payload_Pattern parameter is used. When the CS_SYNC_Payload_Pattern indicates a user payload to be used, then the CS_SYNC_User_Payload parameter indicates the payload used in CS packets. Otherwise, the CS_SYNC_User_Payload parameter is set to all zeros.

*Channel_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0x48 | Number of channels used in the pattern |
| All other values | Reserved for future use |

*Channel[i]:* *Size: Channel_Length × 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | List of channels used in the pattern |

*Channel_Map:* *Size: 10 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The channel map indicating which channels shall be used and unused within the CS procedure. Every channel is represented by a bit positioned according to the CS channel index [Vol 6] Part H, Section 1. The format of the field is identical to the ChM field in the LL_CS_CONFIG_REQ PDU (see [Vol 6] Part B, Section 2.4.2.45) |

*Channel_Selection_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Channel Selection Algorithm #3b |
| 0x01 | Channel Selection Algorithm #3c |
| All other values | Reserved for future use. |

*Ch3c_Shape:* *Size:1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Hat Shape |
| 0x01 | “X” Shape |
| All other values | Reserved for future use |

*Ch3c_Jump:* *Size:1 octet*

| Value | Parameter Description |
|---|---|
| 0x02 to 0x08 | CSChannelJump |
| All other values | Reserved for future use |

*Main_Mode_Steps:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | The number of CS main mode steps to be executed before a submode step is executed |
| All other values | Reserved for future use |

*T_PM_Tone_Ext:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Initiator and reflector tones sent without tone extension |
| 0x01 | Initiator tone sent with extension; reflector tone sent without extension |
| 0x02 | Initiator tone sent without extension; reflector tone sent with extension |
| 0x03 | Initiator and reflector tones sent with extension |
| 0x04 | Loop through values 0x00 to 0x03 above. Applicable for mode-2 and mode-3 steps only. |
| All other values | Reserved for future use |

*Antenna_Path_Permutation_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x17 | Antenna path permutation index used during antenna switching during the tone phases of CS steps as described in [Vol 6] Part H, Section 4.7.5 |
| 0xFF | Loop through all valid antenna path permutation indices starting from the lowest index |
| All other values | Reserved for future use |

*CS_SYNC_AA_Initiator:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Access Address used in CS_SYNC packets sent by the initiator |

*CS_SYNC_AA_Reflector:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Access Address used in CS_SYNC packets sent by the reflector |

*SS_Marker1_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0 to 63 | Bit number where the first marker in the sounding sequence starts.<br>The bit number shall be between 0 and 28 when RTT_Type indicates a 32-bit sounding sequence. |
| All other values | Reserved for future use |

*SS_Marker2_Position:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 67 to 92 | Bit number where the second marker in the sounding sequence starts.<br>The bit number shall be between 67 and 92 when RTT_Type indicates a 96-bit sounding sequence. |
| 0xFF | Sounding sequence or second marker is not present |
| All other values | Reserved for future use |

*SS_Marker_Value:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use pattern '0011' (in transmission order) as the sounding sequence marker |
| 0x01 | Use pattern '1100' (in transmission order) as the sounding sequence marker |

| Value | Parameter Description |
|---|---|
| 0x02 | Loop through pattern '0011' and '1100' (in transmission order) |
| All other values | Reserved for future use |

*CS_SYNC_Payload_Pattern:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | PRBS9 sequence '11111111100000111101...' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5 |
| 0x01 | Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x02 | Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x03 | PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x04 | Repeated '11111111' (in transmission order) sequence |
| 0x05 | Repeated '00000000' (in transmission order) sequence |
| 0x06 | Repeated '00001111' (in transmission order) sequence |
| 0x07 | Repeated '01010101' (in transmission order) sequence |
| 0x80 | Use CS_SYNC_User_Payload |
| All other values | Reserved for future use |

*CS_SYNC_User_Payload:* *Size: 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXX..XX | Payload for CS_SYNC packets. The least significant bit corresponds to the most significant bit of the CS Payload. When the sequence is less than 16 octets, the least significant octets shall be padded with zeros accordingly. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Test command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Test command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Test command has completed, an HCI_Command_Complete event shall be generated. The Controller shall either generate the LE_CS_Subevent_-

Result event once or may generate the LE_CS_Subevent_Result_Continue event multiple times to send results from the completed CS steps to the Host.

### 7.8.143 LE CS Test End command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Test_End | 0x0096 | *none* | *none* |

**Description:**

This command is used to stop any CS test that is in progress.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | No CS test has been enabled but not completed. | *Command Disallowed* (0x0C) |

**Command parameters:**

None.

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_CS_Test_End command, the Controller shall send the HCI_Command_Status event to the Host. When the Controller has successfully sent all the pending CS subevent results to the Host, the Controller shall generate an LE_CS_Test_End_Complete event.

### 7.8.144 LE Set Decision Data command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Decision_Data | 0x0080 | Advertising_Handle,<br>Decision_Type_Flags,<br>Decision_Data_Length,<br>Decision_Data | Status |

**Description:**

This command is used to set the decision data used by the specified advertising set in those advertising PDUs that contain decision data. This command may be issued at any time after an advertising set identified by the Advertising_Handle parameter has been created, regardless of whether advertising in that set is enabled or disabled. When an advertising set is created, Decision_Type_Flags shall be set to zero and there shall be zero octets of decision data.

If advertising is currently enabled for the specified advertising set, the Controller shall use the new decision data in subsequent extended advertising events for this advertising set. If an extended advertising event is in progress when this command is issued, the Controller may use the old or new decision data for that event.

If advertising is currently disabled for the specified advertising set, the decision data shall be kept by the Controller and used once advertising is enabled for that set. The decision data shall be discarded when the advertising set is removed.

The Decision_Type_Flags and Decision_Data parameters specify the decision data to be used with the advertising set.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Advertising_Handle does not identify an advertising set. | *Unknown Advertising Identifier* (0x42) |
| MC | Decision_Data_Length is not large enough to contain the subfields specified in Decision_Type_Flags. | *Invalid HCI Command Parameters* (0x12) |

### Command parameters:

*Advertising_Handle:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Used to identify an advertising set<br>Range: 0x00 to 0xEF |

*Decision_Type_Flags:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | A Resolvable Tag subfield is present in the decision data |
| All other bits | Reserved for future use |

*Decision_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of octets in the Decision_Data parameter<br>Range: 0 to 8 |

*Decision_Data:* *Size: Decision_Data_Length octets*

| Parameter Description |
|---|
| Decision data as defined in [Vol 6] Part B, Section 2.3.1.11<br>Note: This parameter has a variable length. |

### Return parameters:

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Decision_Data command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Decision_Data command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the HCI_LE_Set_Decision_Data command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.145 LE Set Decision Instructions command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Decision_Instructions | 0x0081 | Num_Tests,<br>Test_Flags[i],<br>Test_Field[i],<br>Test_Parameters[i] | Status |

**Description:**

This command is used to set the decision instructions used when listening for advertisements containing decision PDUs (when either scanning or initiating a new connection). The Controller shall support at least 8 tests in the decision instructions.

This command may be used while scanning or initiating is in progress. If so, the new decision instructions shall take effect before the HCI_Command_Complete event is sent to the Host and the change shall not take place while the decision instructions are being used to process a decision PDU (that is, each PDU is processed using either the old decision instructions or the new ones, but not a mixture).

The Num_Tests parameter specifies the number of individual tests in the decision instructions.

Decision procedure

The Controller shall use the following procedure to apply the decision instructions specified by this command to each received ADV_DECISION_IND PDU and either accept or reject it.

For each test in the instructions, the Controller shall apply that test to the received PDU to reach a pass or fail result as follows:

1. Use the value of Test_Field[i] to determine whether the “relevant field” is present in the PDU. This is some property, usually a specific field or subfield, of the ADV_DECISION_IND PDU (it is still called the relevant field in this description even if it is not a field).
2. If the relevant field is not present, then use the value of bit 3 of Test_Flags[i] to determine the result of the test.
3. If the relevant field is present, then carry out the specific check for that field described below, using Test_Parameters[i] to provide any additional parameters needed. Use the value of bit 1 of Test_Flags[i] (if the check passed) or the value of bit 2 of Test_Flags[i] (if the check failed) to determine the result of the test.

Note: The effect of steps 2 and 3 is that each combination of values of bits 1 to 3 of Test_Flags[i] has the overall behavior shown in Table 7.5.

| **Bit of Test_Flags[i]** | | | **The test passes if and only if:** |
|---|---|---|---|
| **1** | **2** | **3** | |
| 0 | 0 | 0 | Never |
| 0 | 0 | 1 | The decision data does not contain the relevant field |
| 0 | 1 | 0 | The decision data contains the relevant field and the check fails |
| 0 | 1 | 1 | Either the decision data contains the relevant field and the check fails, OR the decision data does not contain the relevant field |
| 1 | 0 | 0 | The decision data contains the relevant field and the check passes |
| 1 | 0 | 1 | Either the decision data contains the relevant field and the check passes, OR the decision data does not contain the relevant field |
| 1 | 1 | 0 | The decision data contains the relevant field, irrespective of whether the check passes |
| 1 | 1 | 1 | Always |

*Table 7.5: Conditions to pass a test based on the Test_Flags[i] value*

The Controller shall partition the list of tests into test groups based on bit 0 of Test_Flags[i]: if the bit is 0, then test i is in the same group as test i – 1; if the bit is 1, then test i is in a different group from all lower-numbered tests. A test group shall be classified as passing if all the tests in that group pass and failing if any of the tests in that group fail. See example 1 below.

An ADV_DECISION_IND PDU shall be accepted if any of the test groups pass when applied to the PDU.

The specific check for each relevant field is as follows:

- If the relevant field is the Resolvable Tag, then the check passes if the Resolvable Tag resolves against the 128-bit key provided in Test_Parameters[i] as shown in Figure 7.7.

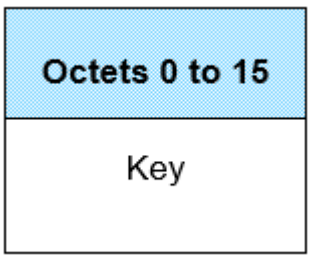



*Figure 7.7: Interpretation of Test_Parameters[i] for tests on the Resolvable Tag*

- If the relevant field is the Arbitrary Data, then Test_Parameters[i] has the format given in Figure 7.8.

| Octets 0 to 7 | Octets 8 to 15 |
|---|---|
| Mask | Target |

*Figure 7.8: Interpretation of Test_Parameters[i] for tests on the Arbitrary Data*

The Controller shall append all-zero octets to the Arbitrary Data as necessary to make the length equal to 8 octets. The Controller shall then apply a bitwise AND to that value and Mask; the check passes if the result of this equals Target.

For example, if Test_Parameters[i] is the sequence of octets [FF FF F0 F0 CC CC 55 55 01 02 00 00 00 00 00 00], then an Arbitrary Data of [01 02 03 04] will pass the check whereas [01 02 03 04 63 33] will fail because the fifth octet does not match.

- If the relevant field is the RSSI field, then Test_Parameters[i] has the format given in Figure 7.9.

| Octet 0 | Octet 1 | Octets 2 to 15 |
|---|---|---|
| MinRSSI | MaxRSSI | RFU |

*Figure 7.9: Interpretation of Test_Parameters[i] for tests on the RSSI*

The check passes if the RSSI value is between MinRSSI and MaxRSSI (both are signed values in dBm).

- If the relevant field is the AdvA field, then Test_Parameters[i] has the format given in Figure 7.10 and the check depends on the value of the Check field as shown in Table 7.6.

| Octet 0 | Octet 1 | Octets 2 to 7 | Octet 8 | Octets 9 to 14 | Octet 15 |
|---|---|---|---|---|---|
| Check | Address 1 type | Address 1 address | Address 2 type | Address 2 address | RFU |

*Figure 7.10: Interpretation of Test_Parameters[i] for tests on the AdvA field*

| Value | Meaning |
|---|---|
| 0 | The check passes if the AdvA field is for a device in the Filter Accept List. The Addresses are ignored. |
| 1 | The check passes if the AdvA field specifies the same device as Address 1. Address 2 is ignored. |
| 2 | The check passes if the AdvA field specifies the same device as Address 1 or as Address 2. |
| All other values | Reserved for future use. |

*Table 7.6: Interpretation of the Check field for tests on the AdvA field*

The value of the Address type fields shall be 0x00 for a public address and 0x01 for a random address.

To compare the AdvA field with an Address field, the AdvA is first resolved if it is a resolvable private address and is then compared with the address and type of the Address field; the check passes if they are the same. For example, if the Address Type field is 0x00 and the Address address field is the sequence of octets [22 33 44 55 66 77], then the check passes only if the AdvA field specifies the public address 0x776655443322 or a resolvable private address that resolves to that address.

- If the relevant field is the path loss, then Test_Parameters[i] has the format given in Figure 7.11.

| Octet 0 | Octet 1 | Octets 2 to 15 |
|---|---|---|
| MinLoss | MaxLoss | RFU |

*Figure 7.11: Interpretation of Test_Parameters[i] for tests on the path loss*

The check passes if the path loss, calculated as the difference between the value of the TxPower field and the RSSI value for the received PDU, is between MinLoss and MaxLoss (both are unsigned values in dB).

- If the relevant field is the AdvMode field, then Test_Parameters[i] has the format given in Figure 7.12 and the check depends on the value of the EventType field as shown in Table 7.7.

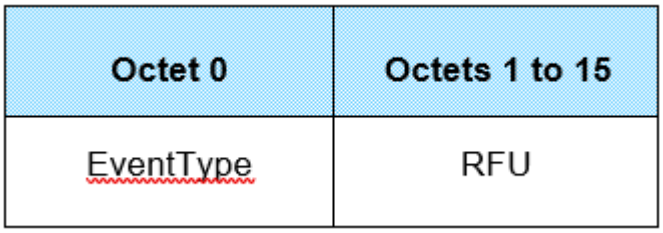



*Figure 7.12: Interpretation of Test_Parameters[i] for tests on the AdvMode field*

The check passes if the appropriate bit of the EventType field is 1.

| Bit Number | Pass Criteria |
|---|---|
| 0 | The check passes for non-connectable non-scannable undirected events |
| 1 | The check passes for connectable undirected events |
| 2 | The check passes for scannable undirected events |
| All other bits | Reserved for future use |

*Table 7.7: Interpretation of the EventType field for tests on the AdvMode field*

- If the relevant field is vendor-specific, then the check to be applied and the interpretation of Test_Parameters[i] are also vendor-specific.

Example 1

In this example, the decision instructions consist of the two tests in Table 7.8:

| Test | Test_Flags | Test_Field | Test_Parameters |
|---|---|---|---|
| 0 | 0b0011 | 0 | Key |
| 1 | 0b0010 | 6 | 0x05 0x00 0x00 0x00 0x00 0x00 0x00 0x00<br>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 |

*Table 7.8: Example 1 test instructions*

These instructions form a single test group. Test 0 passes if there is a Resolvable Tag field in the PDU that resolves against the key provided, while test 1 passes if the advertising is undirected and non-connectable.

Therefore a PDU will be accepted if it resolves against the key and is undirected but not connectable.

If the Test_Flags for test 1 were changed to 0b0011, the two tests would each form their own test group. A PDU would then be accepted if it resolved against the key or is undirected and non-connectable.

Example 2

In this example, the decision instructions consist of the three tests in Table 7.9:

| Test | Test_Flags | Test_Field | Test_Parameters |
|---|---|---|---|
| 0 | 0b0011 | 20 | 0xFF 0xFF 0xFF 0xFF 0x00 0x00 0x00 0x00<br>0x41 0x42 0x43 0x44 0x00 0x00 0x00 0x00 |
| 1 | 0b0011 | 36 | 0xFF 0xFF 0xFF 0xFF 0x00 0x00 0x00 0x00<br>0x45 0x46 0x47 0x48 0x00 0x00 0x00 0x00 |
| 2 | 0b0011 | 52 | 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00<br>0x4A 0x00 0x00 0x00 0x00 0x00 0x00 0x00 |

*Table 7.9: Example 2 test instructions*

These instructions each form their own test group, so a PDU will be accepted if any of the tests pass.

- Test 0 passes if the Arbitrary Data is exactly four octets long, consisting of “ABCD”.
- Test 1 passes if the Arbitrary Data is at least four octets long and begins with “EFGH”.
- Test 2 passes if the Arbitrary Data is at most four octets long and begins with “J”.

Example 3

In this example, the decision instructions consist of the five tests in Table 7.10:

| Test | Test_Flags | Test_Field | Test_Parameters |
|---|---|---|---|
| 0 | 0b1001 | 0 | Ignored |
| 1 | 0b1010 | 8 | 0x50 0x64 0x00 0x00 0x00 0x00 0x00 0x00<br>0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 |
| 2 | 0b0010 | 36 | 0xFF 0xFF 0xFF 0xFF 0x00 0x00 0x00 0x00<br>0x43 0x44 0x57 0x46 0x00 0x00 0x00 0x00 |
| 3 | 0b0011 | 0 | Key |
| 4 | 0b0010 | 18 | 0xDF 0xDF 0x00 0x00 0x00 0x00 0x00 0x00<br>0x42 0x54 0x00 0x00 0x00 0x00 0x00 0x00 |

*Table 7.10: Example 3 test instructions*

These instructions form two test groups, one consisting of tests 0 to 2 and the other of tests 3 and 4.

Test 0 passes if there is no Resolvable Tag field in the PDU.

Test 1 passes if the calculated path loss is between 80 dB and 100 dB or if the loss cannot be calculated (if the value of Test_Flags[1] had been 0b0010, test 1 would fail if the loss could not be calculated).

Test 2 passes if the Arbitrary Data is at least 4 octets long and begins with “CDWF”.

Test 3 passes if there is a Resolvable Tag field in the PDU and that tag resolves against the specified key.

Test 4 passes if the Arbitrary Data consists of exactly two octets, which are “bt” in either case (because the bit distinguishing case is masked out for these two octets).

Therefore:

- A PDU not containing a Resolvable Tag will be accepted if it contains Arbitrary Data beginning with “CDWF” and the path loss (if available) is within certain bounds.
- A PDU containing a Resolvable Tag will be accepted if the tag resolves against the key in Test_Parameters[3] and it is followed by one of “BT”, “Bt”, “bT”, or “bt”.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Num_Tests is 0 or bit 0 of Test_Flags[0] is 0. | *Invalid HCI Command Parameters* (0x12) |
| MC | Num_Tests is greater than the maximum that the Controller supports. | *Limit Reached* (0x43) |

**Command parameters:**

*Num_Tests:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved for future use |
| 0x01 to 0xFF | The number of tests in the decision instructions |

*Test_Flags[i]:* *Size: Num_Tests × 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Start of new test group |
| 1 | Test passes if the decision data contains the relevant field and the check passes |
| 2 | Test passes if the decision data contains the relevant field and the check fails |
| 3 | Test passes if the decision data does not contain the relevant field |
| All other bits | Reserved for future use |

*Test_Field[i]:* *Size: Num_Tests × 1 octet*

| Value | Parameter Description |
|---|---|
| 0 | The relevant field is the Resolvable Tag subfield |
| 6 | The relevant field is the AdvMode field (which is always present) |
| 7 | The relevant field is the RSSI field |
| 8 | The relevant field is the path loss computed from the TxPower field and the RSSI value for the received PDU; if this cannot be computed, the decision data is deemed not to contain the relevant field |
| 9 | The relevant field is the AdvA field |
| 17 to 24 | N = the relevant field is the Arbitrary Data, but only if it contains exactly N – 16 octets; for any other length, the decision data is treated as if it does not contain the relevant field |
| 33 to 40 | N = the relevant field is the Arbitrary Data, but only if it contains at least N – 32 octets; if it contains fewer octets, the decision data is treated as if it does not contain the relevant field |
| 49 to 56 | N = the relevant field is the Arbitrary Data, but only if it contains at least one and at most N – 48 octets; if it contains more octets, the decision data is treated as if it does not contain the relevant field |
| 240 to 255 | Vendor-specific |
| All other values | Reserved for future use |

*Test_Parameters[i]:* *Size: Num_Tests × 16 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX | Specific parameters for the test. The meaning of this parameter depends on the value of Test_Field[i]. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Decision_Instructions command succeeded |
| 0x01 to 0xFF | HCI_LE_Set_Decision_Instructions command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Set_Decision_Instructions command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.146 LE Add Device To Monitored Advertisers List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Add_Device_To_Monitored_-Advertisers_List | 0x0098 | Address_Type,<br>Address,<br>RSSI_Low_Threshold,<br>RSSI_High_Threshold,<br>Timeout | Status |

**Description:**

This command is used to add a single device to a list of devices monitored while scanning.

The Address_Type and Address parameters are used to identify an advertising device.

The RSSI_Low_Threshold parameter is set to the RSSI value below which an HCI_LE_Monitored_Advertisers_Report event shall be generated when the associated timer expires for that device.

The RSSI_High_Threshold parameter is set to the RSSI value equal to or above which an HCI_LE_Monitored_Advertisers_Report event may be triggered for that device.

The Timeout parameter is set to the timeout time in seconds for the device.

If the device is already in the Monitored Advertisers List, then the Controller shall not add the device to the Monitored Advertisers List again and shall return success.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | There is no space available for another device in the Monitored Advertisers List. | *Memory Capacity Exceeded* (0x07) |
| MC | RSSI_Threshold_High is less than RSSI_Threshold_Low. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| All other values | Reserved for future use |

*Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address or Random Device Address of the device to be added to the list of monitored devices. |

*RSSI_Low_Threshold:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range -127 to +20<br>Units: dBm |

*RSSI_High_Threshold:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range -127 to +20<br>Units: dBm |

*Timeout:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved for future use |
| 0x01 to 0xFF | Time (in seconds) that the device from which advertisements are received has an RSSI value that remains below the RSSI_Low_Threshold before an event is generated. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Add_Device_To_Monitored_Advertisers_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Add_Device_To_Monitored_Advertisers_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Add_Device_To_Monitored_Advertisers_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.147 LE Remove Device From Monitored Advertisers List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Remove_Device_From_-Monitored_Advertisers_List | 0x0099 | Address_Type,<br>Address | Status |

**Description:**

This command is used to remove a single device from the Monitored Advertisers List.

The Address_Type and Address parameters are used to identify an advertising device.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The device is not in the Monitored Advertisers List. | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Address_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Public Device Address |
| 0x01 | Random Device Address |
| All other values | Reserved for future use |

*Address:* *Size: 6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | Public Device Address or Random Device Address of the device to be removed from the Monitored Advertisers List. |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Remove_Device_From_Monitored_Advertisers_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Remove_Device_From_Monitored_Advertisers_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Remove_Device_From_Monitored_Advertisers_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.148 LE Clear Monitored Advertisers List command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Clear_Monitored_Advertisers_List | 0x009A | *none* | Status |

**Description:**

This command is used to remove all devices from the Monitored Advertisers List.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Clear_Monitored_Advertisers_List command succeeded. |
| 0x01 to 0xFF | HCI_LE_Clear_Monitored_Advertisers_List command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Clear_Monitored_Advertisers_List command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.149 LE Enable Monitoring Advertisers command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Enable_Monitoring_Advertisers | 0x009C | Enable | Status |

**Description:**

This command is used by a Host to request the Controller to enable or disable the monitoring of advertisers in the Monitored Advertisers List and generate the HCI_LE_Monitored_Advertisers_Report event when necessary.

When the Enable parameter is set to 0x01, all entries in the Monitored Advertisers List shall be marked as not awaiting an RSSI value above the RSSI high threshold and their timers shall be reset. The Enable parameter may be set to 0x01 at any time as a method of restarting monitoring. When the Enable parameter is set to 0x00 monitoring is disabled.

**Command parameters:**

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable |
| 0x01 | Enable |
| All other values | Reserved for future use |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Enable_Monitoring_Advertisers command succeeded. |
| 0x01 to 0xFF | HCI_LE_Enable_Monitoring_Advertisers command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Enable_Monitoring_Advertisers command has completed, an HCI_Command_Complete event shall be generated.

After monitoring is enabled using the HCI_LE_Enable_Monitoring_Advertisers command, HCI_LE_Monitored_Advertisers_Report events shall be generated as required.

### 7.8.150 LE Read Monitored Advertisers List Size command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Monitored_Advertisers_-List_Size | 0x009B | *none* | Status,<br>Number |

**Description:**

This command is used to read the total number of entries in the Monitored Advertisers List that the Controller can store.

Note: The number of entries that can be stored is not fixed and the Controller can change it at any time (e.g., because the memory used to store the Monitored Advertisers List can also be used for other purposes).

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Monitored_Advertisers_List_Size command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Monitored_Advertisers_List_Size command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved for future use |
| 0x01 to 0xFF | Number of entries allowed in the Monitored Advertisers List. |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read Monitored_Advertisers_List_Size command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.151 LE Frame Space Update command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Frame_Space_Update | 0x009D | Connection_Handle,<br>Frame_Space_Min,<br>Frame_Space_Max,<br>PHYS,<br>Spacing_Types | *none* |

**Description:**

This command allows the Host to request a change to one or more frame space values. This command may be issued on both the Central and the Peripheral.

The Frame_Space_Min and Frame_Space_Max parameters indicate the minimum and maximum allowed frame space values that the Controller should use, respectively. The Frame_Space_Min parameter shall not be greater than the Frame_Space_Max parameter.

The PHYS and Spacing_Types parameters indicate which frame space values are to be updated. At least one bit shall be set in the PHYS parameter and at least one bit shall be set in the Spacing_Types parameter.

The actual parameter values selected by the Link Layer may be different from the parameter values provided by the Host through this command.

If the Host issues this command with Frame_Space_Max less than the frame space value, then the Controller shall ignore the value provided by the Host and shall use the frame space value in use as the Frame_Space_Max.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not identify an active ACL connection. | *Unknown Connection Identifier* (0x02) |
| MC | Frame_Space_Max is less than Frame_Space_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | No bits are set in PHYS or in Spacing_Types. | *Invalid HCI Command Parameters* (0x12) |

| Type | Condition | Error code |
|---|---|---|
| MC | An unsupported bit is set in PHYS or Spacing_Types. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | A bit set in PHYS or Spacing_Types corresponds to a feature that the Controller already knows is not supported by the remote device (e.g. because it has already received a corresponding error response from the remote device or has checked the features supported by the remote device). | *Unsupported Remote Feature* (0x1A) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Used to identify a connection handle.<br>Range: 0x0000 to 0x0EFF |

*Frame_Space_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The minimum requested frame space value, in microseconds<br>Range: 0x0000 to 0x2710 |

*Frame_Space_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The maximum requested frame space value, in microseconds<br>Range: 0x0000 to 0x2710 |

*PHYS:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | LE 1M |
| 1 | LE 2M |
| 2 | LE Coded PHY |
| All other bits | Reserved for future use |

*Spacing_Types:* *Size: 2 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | T_IFS_ACL_CP |
| 1 | T_IFS_ACL_PC |
| 2 | T_MCES |
| 3 | T_IFS_CIS |
| 4 | T_MSS_CIS |
| All other bits | Reserved for future use |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Frame_Space_Update command, the Controller shall send the HCI_Command_Status event to the Host. When the HCI_LE_Frame_Space_Update command has completed, an HCI_LE_Frame_Space_Update_Complete event shall be generated.

### 7.8.152 LE Enable UTP OTA mode

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Enable_UTP_OTA_Mode | 0x009F | Enable | Status |

**Description:**

This command enables or disables UTP OTA mode on a device to permit it to process UTP messages over the Link Layer. Enabling UTP OTA mode when it is already enabled, or disabling it when already disabled, has no effect.

**Command Parameters:**

*Enable:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disable UTP OTA mode. |
| 0x01 | Enable UTP OTA mode. |
| All other values | Reserved for future use. |

**Return Parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Enable_UTP_Mode command succeeded. |
| 0x01 to 0xFF | HCI_LE_Enable_UTP_Mode command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) Generated (unless masked away):**

When the HCI_LE_Enable_UTP_Mode command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.153 LE UTP Send command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_UTP_Send | 0x00A0 | UTP_Data_Length,<br>UTP_Data | Status |

**Description:**

This command is used by the Upper Tester to transmit UTP messages to the IUT.

**Command Parameters:**

*UTP_Data_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 to 0xFF | Length of the UTP Data field |

*UTP_Data:* *Size: UTP_Data_Length octets*

| Parameter Description |
|---|
| UTP data (see [Vol 6] Part F, Section 5)<br>Note: This parameter has a variable length |

**Return Parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_UTP_Send command succeeded. |
| 0x01 to 0xFF | HCI_LE_UTP_Send command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) Generated (unless masked away):**

When the HCI_LE_UTP_Send command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.154 LE Connection Rate Request command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Connection_Rate_Request | 0x00A1 | Connection_Handle,<br>Connection_Interval_Min,<br>Connection_Interval_Max,<br>Subrate_Min,<br>Subrate_Max,<br>Max_Latency,<br>Continuation_Number,<br>Supervision_Timeout,<br>Min_CE_Length,<br>Max_CE_Length | *none* |

**Description:**

This command is used by a Central or a Peripheral to request a change to existing connection parameters (see Section 4.5.1) using the Connection Rate Update procedure (see [Vol 6] Part B, Section 5.1.32) or the Connection Rate Request procedure (see [Vol 6] Part B, Section 5.1.33).

The Connection_Interval_Min and Connection_Interval_Max parameters are used to define the minimum and maximum allowed connection interval.

The Subrate_Min and Subrate_Max parameters specify the range of acceptable subrating factors being requested.

The Max_Latency parameter specifies the maximum allowed Peripheral latency in units of subrated connection events. The same maximum shall apply irrespective of the subrating factor actually chosen.

The Continuation_Number parameter specifies the number of underlying connection intervals to remain active after a packet other than an empty packet is transmitted or received.

The Supervision_Timeout parameter specifies the link supervision timeout for the connection.

If this command is issued on the Central, then it also sets the acceptable parameters for requests from the Peripheral (see [Vol 6] Part B, Section 5.1.33). The acceptable parameters set by this command override those provided via the HCI_LE_Set_Default_Rate_Parameters command or any values set by previous uses of this command on the same connection.

The Min_CE_Length and Max_CE_Length parameters provide the Controller with the expected minimum and maximum length of the connection events. The Controller is not required to use these values.

If this command is issued on the Central before the devices have performed the Feature Exchange procedure, then the Controller shall complete that procedure before initiating the Connection Rate Request procedure.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error Code |
|---|---|---|
| MC | One or more CS procedures are enabled. | *Command Disallowed* (0x0C) |
| MC | A Connection Parameters Request procedure or a Connection Subrate Request procedure is in progress. | *Command Disallowed* (0x0C) |
| MC | The Connection_Handle parameter does not identify an established ACL connection. | *Unknown Connection Identifier* (0x02) |
| MC | Subrate_Max × (Max_Latency + 1) is greater than 500. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Interval_Max × Subrate_Max × (Max_Latency + 1) is greater than or equal to Supervision_Timeout × 40. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Interval_Max is less than Connection_Interval_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Interval_Min and Connection_Interval_Max specify a range of connection intervals which is not supported by the Controller. | *Unsupported LL Parameter Value* (0x20) |
| MC | Connection_Interval_Max is less than *connIntervalRequired* (see [Vol 6] Part B, Section 4.5.10). | *Unsupported Feature or Parameter Value* (0x11) |
| MC | Subrate_Max is less than Subrate_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Continuation_Number is greater than or equal to Subrate_Max. | *Invalid HCI Command Parameters* (0x12) |
| MC | The Connection Rate (Host Support) bit is not set in the local or remote device's FeatureSet. | *Unsupported Remote Feature* (0x1A) |
| MC | Max_CE_Length is less than Min_CE_Length | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*Connection_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Minimum value for the connection interval<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s |

*Connection_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Maximum value for the connection interval<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s |

*Subrate_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum value for the subrate factor<br>Range: 0x0001 to 0x01F4 (500) |

*Subrate_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum value for the subrate factor<br>Range: 0x0001 to 0x01F4 (500) |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency for the connection in units of subrated connection intervals<br>Range: 0x0000 to 0x01F3 (499) |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received<br>Range: 0x0000 to 0x01F3 (499) |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Supervision timeout for this connection<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s |

*Min_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The minimum length of connection event recommended for this LE connection.<br>Range: 0x0001 to 0x7CFF<br>Time = N × 125 µs<br>Time Range: 0.125 ms to 3.999875 s |

*Max_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The maximum length of connection event recommended for this LE connection.<br>Range: 0x0001 to 0x7CFF<br>Time = N × 125 µs<br>Time Range: 0.125 ms to 3.999875 s |

**Return parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Connection_Rate_Request command, the Controller shall send the HCI_Command_Status event to the Host. An HCI_LE_Connection_Rate_Change event shall be generated when the Connection Rate Update procedure or the Connection Rate Request procedure has completed.

### 7.8.155 LE Set Default Rate Parameters command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Set_Default_Rate_Parameters | 0x00A2 | Connection_Interval_Min,<br>Connection_Interval_Max,<br>Subrate_Min,<br>Subrate_Max,<br>Max_Latency,<br>Continuation_Number,<br>Supervision_Timeout,<br>Min_CE_Length,<br>Max_CE_Length | Status |

**Description:**

This command is used by the Host to set the initial values of the acceptable parameters for the Connection Rate Request procedure for all future ACL connections where the Controller is Central. The command does not affect any existing connection.

The parameters have the same meanings as those in the HCI_LE_Connection_Rate_Request command (see Section 7.8.154).

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error Code |
|---|---|---|
| MC | Subrate_Max × (Max_Latency + 1) is greater than 500. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Interval_Max × Subrate_Max × (Max_Latency + 1) is greater than or equal to Supervision_Timeout × 40. | *Invalid HCI Command Parameters* (0x12) |
| MC | Connection_Interval_Max is less than Connection_Interval_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Subrate_Max is less than Subrate_Min. | *Invalid HCI Command Parameters* (0x12) |
| MC | Continuation_Number is greater than or equal to Subrate_Max. | *Invalid HCI Command Parameters* (0x12) |
| MC | Max_CE_Length is less than Min_CE_Length | *Invalid HCI Command Parameters* (0x12) |

**Command parameters:**

*Connection_Interval_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Minimum value for the connection interval allowed in requests by a Peripheral<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s<br>Default: 0x003C (7.5 ms) |

*Connection_Interval_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Maximum value for the connection interval allowed in requests by a Peripheral<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s<br>Default: 0x7D00 (4 s) |

*Subrate_Min:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum value for the subrate factor allowed in requests by a Peripheral<br>Range: 0x0001 to 0x01F4 (500)<br>Default: 0x0001 |

*Subrate_Max:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum value for the subrate factor allowed in requests by a Peripheral<br>Range: 0x0001 to 0x01F4 (500)<br>Default: 0x0001 |

*Max_Latency:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Maximum Peripheral latency allowed in requests by a Peripheral, specified as the number of subrated connection intervals<br>Range: 0x0000 to 0x01F3 (499)<br>Default: 0x0000 |

*Continuation_Number:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Minimum Number of underlying connection events to remain active after a packet containing a Link Layer PDU with a non-zero Length field is sent or received in requests by a Peripheral<br>Range: 0x0000 to 0x01F3 (499)<br>Default: 0x0000 |

*Supervision_Timeout:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Maximum supervision timeout allowed in requests by a Peripheral<br>Range: 0x000A to 0x0C80<br>Time = N × 10 ms<br>Time Range: 100 ms to 32 s<br>Default: 0x0C80 (32 s) |

*Min_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The minimum length of connection event recommended in requests by a Peripheral<br>Range: 0x0001 to 0x7CFF<br>Time = N × 125 µs<br>Time Range: 0.125 ms to 3.999875 s |

*Max_CE_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The maximum length of connection event recommended in requests by a Peripheral<br>Range: 0x0001 to 0x7CFF<br>Time = N × 125 µs<br>Time Range: 0.125 ms to 3.999875 s |

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Set_Default_Rate_Parameters command succeeded. |
| 0x01 to 0xFF | HCI_LE_Set_Default_Rate_Parameters command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_LE_Set_Default_Rate_Parameters command, the Controller shall send the HCI_Command_Complete event to the Host.

### 7.8.156 LE Read Minimum Supported Connection Interval command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_Read_Minimum_-Supported_Connection_Interval | 0x00A3 | *none* | Status,<br>Minimum_Supported_-Connection_Interval,<br>Num_Groups,<br>Group_Min[i],<br>Group_Max[i],<br>Group_Stride[i] |

**Description:**

This command is used by the Host to determine the minimum supported connection interval and which other connection intervals (see [Vol 6] Part B, Section 4.5.1) the Controller supports.

Each group shall represent a set of supported connection intervals forming an arithmetic sequence starting at Group_Min[i], ending at Group_Max[i], and in steps of Group_Stride[i]. For example, if Group_Min[i] = 0x0005, Group_Max[i] = 0x003C, and Group_Strides[i] = 0x000B, then this group indicates that the Controller supports the intervals 0.625 ms, 2.000 ms, 3.375 ms, 4.750 ms, 6.125 ms, and 7.500 ms.

The returned groups should include between them every supported value that is in ECV and not in RCV but may include values in RCV. A supported value may be in more than one group. The Controller shall not include any group that only contains values from RCV or the minimum supported. If it would require too many groups to represent all supported values, then the Controller should represent as many supported values as possible.

For example, if the Controller returns Num_Groups = 2, Group_Min[0] = 0x0005, Group_Max[0] = 0x0050, Group_Strides[0] = 0x0005, Group_Min[1] = 0x0050, Group_Max[1] = 0x0320, and Group_Strides[1] = 0x0008, then the Controller supports all connection interval values in RCV, all values up to 10 ms that are multiples of 625 µs, and all values between 10 ms and 100 ms that are multiples of 1 ms. If the Controller supports any other values, it should have included more groups.

If the Controller only supports values from RCV, then it shall set Num_Groups to zero.

**Command parameters:**

None.

**Return parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_Read_Minimum_Supported_Connection_Interval command succeeded. |
| 0x01 to 0xFF | HCI_LE_Read_Minimum_Supported_Connection_Interval command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Minimum_Supported_Connection_Interval:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | Minimum supported value for the connection interval<br>Range: 0x03 to 0x3C<br>Time = N × 125 µs<br>Time Range: 375 µs to 7.5 ms |

*Num_Groups:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N=0xXX | The number of groups |

*Group_Min[i]:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Minimum supported connection interval in this group<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s |

*Group_Max[i]:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | Maximum supported connection interval in this group<br>Range: 0x0003 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 375 µs to 4 s |

*Group_Stride[i]:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| N=0xXXXX | The Connection Interval resolution of this group<br>Range: 0x0001 to 0x7D00<br>Time = N × 125 µs<br>Time Range: 125 µs to 4 s |

**Event(s) generated (unless masked away):**

When the HCI_LE_Read_Minimum_Supported_Connection_Interval command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.157 LE CS Set Security Requirements

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Set_Security_Requirements | 0x00A7 | Connection_Handle,<br>CS_Security_Requirements | Status,<br>Connection_Handle |

**Description:**

The HCI_LE_CS_Set_Security_Requirements command is used by a Host to set the Channel Sounding security requirements that the Controller shall enforce on the connection specified by the Connection_Handle parameter. Until the Host sets the Channel Sounding security requirements for a connection (using this command or the HCI_LE_Set_CS_Set_Default_Security_Requirements command (see Section 7.8.158) prior to connection creation), the Controller need not enforce any Channel Sounding security requirements on that connection.

The CS_Security_Requirements parameter indicates the Channel Sounding security requirements.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | Connection_Handle does not exist, or the Connection_Handle is not for an ACL. | *Unknown Connection Identifier* (0x02) |
| MC | The Channel Sounding (Host Support) feature bit is not set. | *Command Disallowed* (0x0C) |
| MC | The CS_Security_Requirements parameter is used to enforce a security requirement that the Controller does not support. | *Unsupported Feature or Parameter Value* (0x11) |
| MC | One or more CS procedures have been enabled. | *Command Disallowed* (0x0C) |

**Command Parameters:**

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

*CS_Security_Requirements:* *Size: 8 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | CS tone |
| 1 | 150 ns RTT accuracy |
| 2 | 10 ns RTT accuracy |
| 3 | RTT sounding sequence or random sequence |
| 4 | Normalized Attack Detector Metric |
| All other bits | Reserved for future use |

**Return Parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Set_Security_Requirements command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Set_Security_Requirements command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection_Handle<br>Range: 0x0000 to 0x0EFF |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Set_Security_Requirements command has completed, an HCI_Command_Complete event shall be generated.

### 7.8.158 CS Set Default Security Requirements

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_LE_CS_Set_Default_Security_Requirements | 0x00A8 | CS_Security_Requirements | Status |

**Description:**

The HCI_LE_CS_Set_Default_Security_Requirements command is used by a Host to set the initial Channel Sounding security requirements that the Controller shall enforce on all future ACL connections. This command does not affect any existing connection.

The CS_Security_Requirements parameter indicates the Channel Sounding security requirements.

**Errors:**

See Section 4.5.2 for a list of error types and descriptions.

| Type | Condition | Error code |
|---|---|---|
| MC | The Channel Sounding (Host Support) feature bit is not set. | *Command Disallowed* (0x0C) |
| MC | The CS_Security_Requirements parameter is used to enforce a security requirement that the Controller does not support. | *Unsupported Feature or Parameter Value* (0x11) |

**Command Parameters:**

*CS_Security_Requirements:* *Size: 8 octets*

| Bit Number | Parameter Description |
|---|---|
| 0 | CS tone |
| 1 | 150 ns RTT accuracy |
| 2 | 10 ns RTT accuracy |
| 3 | RTT sounding sequence or random sequence |
| 4 | Normalized Attack Detector Metric |
| All other bits | Reserved for future use |

**Return Parameters:**

*Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_LE_CS_Set_Default_Security_Requirements command succeeded |
| 0x01 to 0xFF | HCI_LE_CS_Set_Default_Security_Requirements command failed. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the HCI_LE_CS_Set_Default_Security_Requirements command has completed, an HCI_Command_Complete event shall be generated.

# Appendix A [This Appendix is no longer used]

# Appendix B Removed commands and events

Table B.1 lists commands that have been removed from the specification. The OGF/OCF combinations in this table are previously used (see [Vol 1] Part E, Section 2.4).

| OGF | OCF | Removed command name |
|---|---|---|
| 0x01 | 0x0007 | Add SCO Connection command |
| 0x01 | 0x0035 | Create Physical Link command |
| 0x01 | 0x0036 | Accept Physical Link command |
| 0x01 | 0x0037 | Disconnect Physical Link command |
| 0x01 | 0x0038 | Create Logical Link command |
| 0x01 | 0x0039 | Accept Logical Link command |
| 0x01 | 0x003A | Disconnect Logical Link command |
| 0x01 | 0x003B | Logical Link Cancel command |
| 0x01 | 0x003C | Flow Spec Modify command |
| 0x02 | 0x0005 | Park State command |
| 0x02 | 0x0006 | Exit Park State command |
| 0x03 | 0x000B | Create New Unit Key command |
| 0x03 | 0x0021 | Read Encryption Mode command |
| 0x03 | 0x0022 | Write Encryption Mode command |
| 0x03 | 0x003B | Read Page Scan Period Mode command |
| 0x03 | 0x003C | Write Page Scan Period Mode command |
| 0x03 | 0x003D | Read Page Scan Mode command |
| 0x03 | 0x003E | Write Page Scan Mode command |
| 0x03 | 0x0061 | Read Logical Link Accept Timeout command |
| 0x03 | 0x0062 | Write Logical Link Accept Timeout command |
| 0x03 | 0x0064 | Read Location Data command |
| 0x03 | 0x0065 | Write Location Data command |
| 0x03 | 0x0069 | Read Best Effort Flush Timeout command |
| 0x03 | 0x006A | Write Best Effort Flush Timeout command |
| 0x03 | 0x006B | Short Range Mode command |
| 0x04 | 0x0007 | Read Country Code command |
| 0x05 | 0x0009 | Read Local AMP Info command |

| OGF | OCF | Removed command name |
|---|---|---|
| 0x05 | 0x000A | Write Local AMP ASSOC command |
| 0x05 | 0x000B | Write Remote AMP ASSOC command |
| 0x06 | 0x0007 | Enable AMP Receiver Reports command |
| 0x06 | 0x0008 | AMP Test End command |
| 0x06 | 0x0009 | AMP Test command |

*Table B.1: Removed commands*

Table B.2 lists events that have been removed from the specification. The event codes in this table are previously used.

| Event code | Removed event name |
|---|---|
| 0x1F | Page Scan Mode Change event |
| 0x40 | Physical Link Complete event |
| 0x41 | Channel Selected event |
| 0x42 | Disconnection Physical Link Complete event |
| 0x43 | Physical Link Loss Early Warning event |
| 0x44 | Physical Link Recovery event |
| 0x45 | Logical Link Complete event |
| 0x46 | Disconnection Logical Link Complete event |
| 0x47 | Flow Spec Modify Complete event |
| 0x49 | AMP Start Test event |
| 0x4A | AMP Test End event |
| 0x4B | AMP Receiver Report event |
| 0x4C | Short Range Mode Change Complete event |
| 0x4D | AMP Status Change event |

*Table B.2: Removed events*

# [This Volume Is No Longer Used]

**Specification of the *Bluetooth*® System**

**Volume 5**

**Covered Core Package Version:** v6.3

**Version Date:** 2026-05-05

# Low Energy Controller

**Specification of the *Bluetooth*® System**

**Volume 6**

**Covered Core Package Version: v6.3**

**Version Date: 2026-05-05**

Bluetooth SIG Proprietary

Low Energy Controller
Part A

# RADIO PHYSICAL LAYER SPECIFICATION

*This Part describes the Bluetooth Low Energy radio physical layer.*

# CONTENTS

**1 Scope ..... 2942**

**2 Frequency bands and channel arrangement ..... 2944**

**3 Transmitter characteristics ..... 2945**
3.1 Modulation characteristics ..... 2946
3.1.1 Stable modulation index ..... 2947
3.1.2 Modulation characteristics of Channel Sounding steps ..... 2947
3.1.3 SNR control for Channel Sounding steps ..... 2947
3.2 Spurious emissions ..... 2948
3.2.1 Modulation spectrum ..... 2948
3.2.2 In-band spurious emission ..... 2948
3.2.3 Out-of-band spurious emission ..... 2949
3.3 Radio frequency tolerance ..... 2949
3.4 Stable phase ..... 2949
3.5 Frequency measurement and generation in Channel Sounding ..... 2950
3.5.1 Fractional frequency offset ..... 2951
3.5.2 Expected transmitted frequencies ..... 2952

**4 Receiver characteristics ..... 2953**
4.1 Actual sensitivity level ..... 2953
4.2 Interference performance ..... 2953
4.3 Out-of-band blocking ..... 2955
4.4 Intermodulation characteristics ..... 2956
4.5 Maximum usable level ..... 2956
4.6 Reference signal definition ..... 2956
4.7 Stable modulation index ..... 2957
4.8 Received Signal Strength Indication ..... 2957

**5 Antenna switching ..... 2958**
5.1 Antenna Switching for AoA/AoD ..... 2958
5.2 Receiver characteristics for AoA/AoD ..... 2959
5.2.1 Definitions ..... 2959
5.2.2 Requirements ..... 2960
5.2.3 Test switching pattern ..... 2960
5.3 Antenna Switching for Channel Sounding ..... 2960

**6 Phase measurements ..... 2962**

6.1 Reference receiver for phase-based ranging ..... 2962
6.2 Phase measurement accuracy ..... 2962
6.3 Frequency actuation error correction ..... 2963
6.4 Phase measurement timing ..... 2964
6.5 Inline PCT Transfer ..... 2965

**Appendix A Test Conditions ..... 2966**
A.1 Normal operating conditions ..... 2966
A.1.1 Normal temperature and air humidity ..... 2966
A.1.2 Nominal supply voltage ..... 2966

**Appendix B Example test equipment setup for Channel Sounding receiver and transmitter ..... 2967**

# 1 SCOPE

Bluetooth Low Energy (LE) devices operate in the unlicensed 2.4 GHz ISM (Industrial Scientific Medical) band. A frequency hopping transceiver is used to combat interference and fading.

Two modulation schemes are defined. The mandatory modulation scheme (“1 Msym/s modulation”) uses a shaped, binary FM to minimize transceiver complexity. The symbol rate is 1 Msym/s. An optional modulation scheme (“2 Msym/s modulation”) is similar but uses a symbol rate of 2 Msym/s.

The 1 Msym/s modulation supports two PHYs:

- LE 1M, with uncoded data at 1 Mb/s;
- LE Coded, with the Access Address, Coding Indicator, and TERM1 coded at 125 kb/s and the payload coded at either 125 kb/s or 500 kb/s.

A device shall support the LE 1M PHY. Support for the LE Coded PHY is optional.

The 2 Msym/s modulation supports two PHYs:

- LE 2M, with uncoded data at 2 Mb/s with BT=0.5.
- LE 2M 2BT, with uncoded data at 2 Mb/s with BT=2.0. This PHY may only be used with the Channel Sounding feature.

A device may support Channel Sounding (CS). The modulation requirements for LE 1M, an optional LE 2M, and an optional LE 2M 2BT also apply to any CS_SYNC packet (see [Vol 6] Part H, Section 2). The CS tone transmitter and receiver requirements are specified in Section 3.4 and Section 6, respectively.

CS uses an additional modulation scheme known as amplitude-shift keying (ASK). When using ASK, symbols are sent by transmitting a fixed-amplitude carrier wave at a fixed frequency for a specific time duration.

A Time Division Duplex (TDD) scheme is used on all PHYs. The specification defines the requirements for a Bluetooth radio for the Low Energy radio.

Requirements are defined for two reasons:

- Provide compatibility between radios used in the system
- Define the quality of the system

An LE radio shall have a transmitter or a receiver, or both.

The LE radio shall fulfill the stated requirements for the operating conditions declared by the equipment manufacturer (see Section A.1).

The Bluetooth SIG maintains regulatory content associated with Bluetooth technology in the 2.4 GHz ISM band on its web site, at https://www.bluetooth.com/regulatory-requirements/.

# 2 FREQUENCY BANDS AND CHANNEL ARRANGEMENT

The LE system operates in the 2.4 GHz ISM band at 2400 MHz to 2483.5 MHz. The LE system uses 40 RF channels with center frequencies at a 2 MHz spacing from 2402 MHz to 2480 MHz.

An LE system supporting CS uses 72 RF channels for CS exchanges. These RF channels have center frequencies at 2402 + k MHz, where k is an integer from 2 to 22 and 26 to 76.

# 3 TRANSMITTER CHARACTERISTICS

The requirements stated in this section are given as power levels at the antenna connector of the LE device; this is also referred to as the radiative transmit power level of the device. If the device does not have a connector, a reference antenna with 0 dBi gain is assumed. Power level values used in HCI commands, HCI events, Advertising physical channel PDUs, and Link Layer Control PDUs shall be assumed to be the radiative transmit power level of the device unless specified otherwise.

Due to the difficulty in making accurate radiated measurements, systems with an integral antenna should provide a temporary antenna connector during LE PHY qualification testing.

For a transmitter, the radiative transmit power level at the maximum power setting shall be between 0.01 mW (-20 dBm) and 100 mW (+20 dBm).

Using high transmit power in use cases where short ranges could be encountered can cause the receiver on the remote device to be saturated and result in link failure. The LE Power Control Request feature can be used to adjust a connected remote device's transmit power level based on the receiver's signal level. When the LE Power Control Request feature is used on a connection with long connection intervals, devices should use reliable RSSI measurements from recent connection events to determine whether or not to send power control requests. When a device is capable of adjusting its transmit power level using the LE Power Control Request feature, the difference between any two adjacent transmit power levels supported by the radio design should be no greater than 8 dB. When the LE Power Control Request feature is not supported by either the local or remote device, implementers should avoid use of high output power in such scenarios or employ a mechanism for switching between two or more transmit power levels in an attempt to establish, re-establish, or maintain connections.

The output power control of a device may be changed locally, for example to optimize the power consumption or reduce interference to other equipment.

Bluetooth devices are classified into power classes based on the radiative transmit power level at the maximum power setting the LE PHY supports ($P_{max}$), as defined in Table 3.1.

| Power Class | Requirements |
|---|---|
| 1 | 100 mW (+20 dBm) ≥ $P_{max}$ > 10 mW (+10 dBm) |
| 1.5 | 10 mW (+10 dBm) ≥ $P_{max}$ > 2.5 mW (+4 dBm) |

| Power Class | Requirements |
|---|---|
| 2 | 2.5 mW (+4 dBm) ≥ $P_{max}$ > 1 mW (0 dBm) |
| 3 | 1 mW (0 dBm) ≥ $P_{max}$ ≥ 0.01 mW (-20 dBm) |

*Table 3.1: LE PHY power classes*

## 3.1 Modulation characteristics

The modulation is Gaussian Frequency Shift Keying (GFSK) with a bandwidth-bit period product BT=0.5. The modulation index shall be between 0.45 and 0.55. A binary one shall be represented by a positive frequency deviation, and a binary zero shall be represented by a negative frequency deviation.

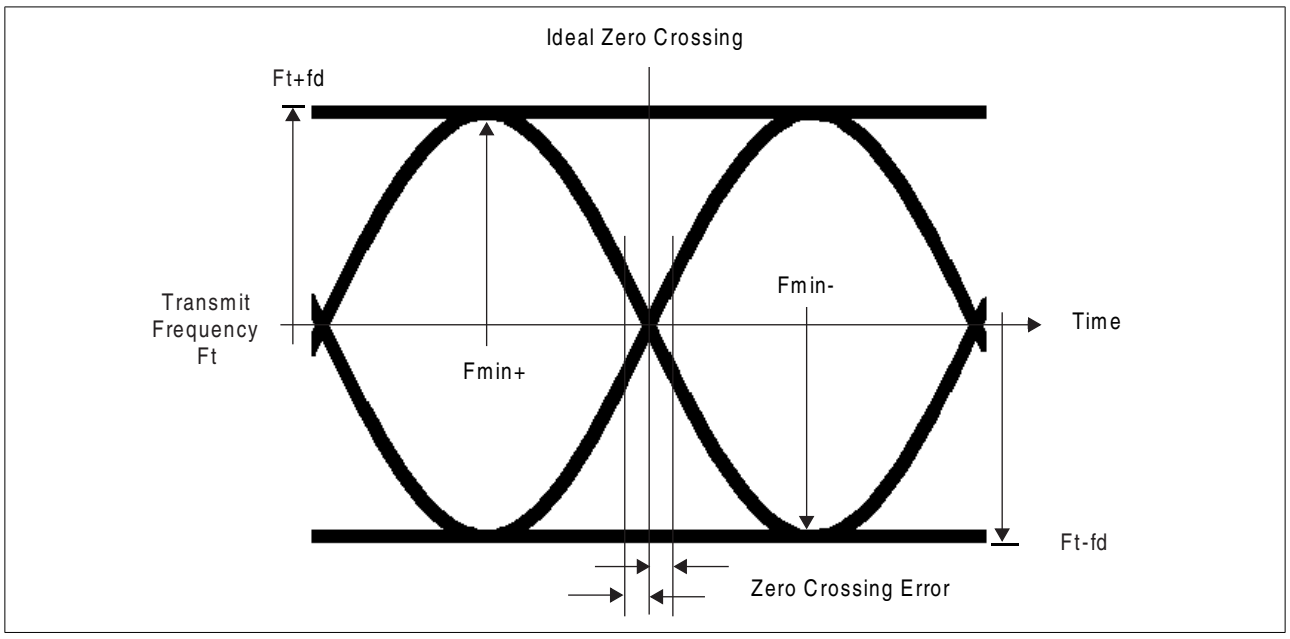



*Figure 3.1: GFSK parameters definition*

For each transmission the minimum frequency deviation,

$$F_{min} = \min \{ |F_{min+}| , F_{min-} \}$$

which corresponds to a 1010 sequence, shall be no smaller than ±80% of the frequency deviation with respect to the transmit frequency, which corresponds to a 00001111 sequence.

The minimum frequency deviation shall never be less than 185 kHz when transmitting at 1 megasymbol per second (Msym/s) symbol rate and never be less than 370 kHz when transmitting at 2 Msym/s symbol rate. The symbol timing accuracy shall be better than ±50 ppm.

The zero crossing error is the time difference between the ideal symbol period and the measured crossing time. This shall be less than $\pm\frac{1}{8}$ of a symbol period.

See Figure 3.1 for the definitions of some symbols and terms in these requirements.

### 3.1.1 Stable modulation index

An LE device with a transmitter that has a stable modulation index may inform the receiving LE device of this fact through the feature support mechanism (see [Vol 6] Part B, Section 4.6). The modulation index for these transmitters shall be between 0.495 and 0.505. A device shall only state that it has a stable modulation index if that applies to all LE transmitter PHYs it supports.

A transmitter that does not have a stable modulation index is said to have a standard modulation index.

### 3.1.2 Modulation characteristics of Channel Sounding steps

During Channel Sounding steps where a CS_SYNC packet is present (see [Vol 6] Part H, Section 2) and the LE 2M 2BT PHY is used, the modulation characteristics are identical to those described in Section 3.1 with the following exceptions:

- The 2 Msym/s transmission rate shall be used with a bandwidth-bit period product BT=2.0.
- The minimum frequency deviation at the center of the symbol shall never be less than 420 kHz.

### 3.1.3 SNR control for Channel Sounding steps

An LE device may have a transmitter that is capable of adjusting its SNR output within a given range for the transmission of CS_SYNC packets used in mode-1 and mode-3 steps. There are five different SNR levels defined in Table 3.2, each identified by an SNR Output Index (SOI).

| SNR Output Index (SOI) | SNR Output Level (dB) | Mandatory/Optional/Conditional |
|---|---|---|
| 0 | 18 | C.1 |
| 1 | 21 | C.1 |
| 2 | 24 | C.1 |
| 3 | 27 | C.1 |
| 4 | 30 | C.1 |
| C.1: A device that supports SNR output control shall support at least one output index/level listed in Table 3.2. | | |

*Table 3.2: SNR output control levels*

Let $\hat{x}(k,t)$ be the continuous version of the observed CS_SYNC packet transmitted by the IUT at step *k*, as defined in [Vol 6] Part H, Section 3.1.1. Let $\hat{\varphi}(k,t)$ be the phase of the observed $\hat{x}(k,t)$.

Let $\varphi_{ideal}(k,t)$ represent the phase trajectory generated from the modulated bit sequence at step *k*, using the modulation rules described in Section 3.1.2.

Let $SNR_{TX}^{desired}$ be the configured SNR value for the IUT.

Let T represent the integration period for the entire CS_SYNC packet transmitted at step *k*.

Let $\tau_0$, $f_0$, and $\varphi_0$ represent the best fit timing, frequency, and phase alignment respectively between the observed signal transmitted by the IUT and $\varphi_{ideal}(k,t)$.

The IUT transmitter SNR is then computed using equation:

$$SNR_{TX}(k) = \max_{\tau_0, f_0, \varphi_0}\left(-10log_{10}\left(\frac{1}{\tau}\int_0^{\tau}\left|e^{\varphi_{ideal}(k,t)} - e^{\hat{\varphi}(k,t-\tau_0)} \times e^{-j(2\pi f_0 t + \varphi_0)}\right|^2 dt\right)\right)$$

The SNR control error can be computed:

$$SNR_{TX}^{error}(k) = \left|SNR_{TX}^{desired} - SNR_{TX}(k)\right|$$

The SNR control error shall satisfy:

$$SNR_{TX}^{error}(k) \leq 3 \text{ dB}$$

and the standard deviation of the randomness of the added error shall satisfy:

$$std(SNR_{TX}^{error}(k)) \geq 0.25 \text{ dB}$$

for 95% of the steps.

## 3.2 Spurious emissions

### 3.2.1 Modulation spectrum

For products that transmit modulated packets and also follow the requirements of FCC part 15.247, the minimum 6 dB bandwidth of the transmitter spectrum shall be at least 500 kHz using a resolution bandwidth of 100 kHz.

### 3.2.2 In-band spurious emission

An adjacent channel power is specified for channels at least 2 MHz from the carrier when transmitting with1 Msym/s modulation (applies to the LE 1M and LE Coded PHYs)

or at least 4 MHz from the carrier when transmitting with 2 Msym/s modulation (applies to the LE 2M and LE 2M 2BT PHYs). This adjacent channel power is defined as the sum of the measured power in a 1 MHz bandwidth.

The spectrum measurement shall be performed with a 100 kHz resolution bandwidth and an average detector. The device shall transmit on an RF channel with the center frequency M and the adjacent channel power shall be measured on a 1 MHz RF frequency N. The transmitter shall transmit a pseudo random data pattern in the payload throughout the test.

| Frequency offset | Spurious Power |
|---|---|
| 2 MHz (\|M-N\| = 2) | -20 dBm |
| 3 MHz or greater (\|M-N\| ≥ 3) | -30 dBm |

*Table 3.3: Transmit spectrum mask when transmitting with 1 Msym/s modulation*

| Frequency offset | Spurious Power |
|---|---|
| 4 MHz (\|M-N\| = 4) | -20 dBm |
| 5 MHz (\|M-N\| = 5) | -20 dBm |
| 6 MHz or greater (\|M-N\| ≥ 6) | -30 dBm |

*Table 3.4: Transmit spectrum mask when transmitting with 2 Msym/s modulation*

Exceptions are allowed in up to three bands of 1 MHz width, centered on a frequency which is an integer multiple of 1 MHz. These exceptions shall have an absolute value of -20 dBm or less.

### 3.2.3 Out-of-band spurious emission

The equipment manufacturer is responsible for the ISM out-of-band spurious emissions requirements in the intended countries of sale.

## 3.3 Radio frequency tolerance

The deviation of the center frequency during the packet shall not exceed ±150 kHz, including both the initial frequency offset and drift. The frequency drift during any packet shall be less than 50 kHz. The drift rate shall be less than 400 Hz/µs anywhere that frequency drift occurs within a packet.

## 3.4 Stable phase

Devices supporting the CS feature shall support the generation of an RF signal with a phase that is stable during the period of T_PM_MEAS. T_PM_MEAS shall be equal to

the CS step mode-2 duration, excluding the final ramp-down period, as defined in [Vol 6] Part H, Section 4.3.3 using the maximum values allowed for the following:

- Phase measurement period
- Antenna switch period
- Ramp-down period
- Interlude periods
- Number of antenna paths

This requirement shall also be applicable to the CS tone duration defined for CS step mode-3 as described in [Vol 6] Part H, Section 4.3.4.

Let ϕ[*n*] be the unwrapped phase of the signal, where unwrap refers to the operation that corrects the radian phase of the array elements by adding multiples of ±2 pi when absolute jumps between consecutive array elements are larger than pi. Also, let ϕ[*n*] be sampled at a 1 µs interval during the period of T_PM_MEAS (in microseconds), where N is equal to (T_PM_MEAS in microseconds), and where n is an integer in the range of 1 to N.

Let Δϕ be the phase detrend factor for the absolute unwrapped phase signal, defined as:

$$\Delta\phi = \frac{6}{N^3 - N}\left(2\sum_{i=1}^{N} i \,.\, \phi[i] - \left(N+1\right)\sum_{i=1}^{N}\phi\left[i\right]\right)$$

So that the detrended phase is expressed as:

$$\phi_d[n] = \phi[n] - n\,\Delta\,\phi$$

Let $\overline{\phi_d}$ be the mean value of the detrended sequence, expressed as:

$$\overline{\phi_d} = \frac{1}{N}\sum_{i=1}^{N}\phi_d\left[n\right]$$

So, the final, zero-mean, detrended phase vector is given by:

$$\phi_{zmd}\left[n\right] = \phi_d\left[n\right] - \overline{\phi_{\mathrm{d}}}$$

Following these definitions, 95% of the absolute values of $\phi_{zmd}[n]$ shall be 20 degrees or less.

## 3.5 Frequency measurement and generation in Channel Sounding

The structure for CS subevents is defined in [Vol 6] Part H, Section 4.4. Each CS subevent contains *K* + *M* CS steps, *k* = 1, …, *K* + *M*. The first *M* steps within the

subevent are CS mode-0 steps as defined in [Vol 6] Part H, Section 4.3.1, and are followed by *K* non-mode-0 steps. Each non-mode-0 step that has CS tones contains P phase measurement periods, where 1 ≤ *P* ≤ (N_AP plus any added CS tone extensions). N_AP is defined in Section 5.3 and CS tone extensions are defined in [Vol 6] Part H, Section 4.3.

Denote the expected start and end times of the CS tone transmitted by a device for the *k*th step in phase measurement period *p*, as $t_{CT,1}^{TX}[k,p]$ and $t_{CT,2}^{TX}[k,p]$, respectively, if a transmitted CS tone exists within that step and phase measurement period. Within this phase measurement period, the time *t* = 0 corresponds to the start of the first scheduled transmission within a CS subevent.

Denote $f_0[k]$ as the nominal frequency of the CS Channel for step *k* as defined in Section 2.

Denote $f[k,p]$ as the average frequency of the transmitted CS tone during the interval $\left[t_{CT,1}^{TX}[k,p], t_{CT,2}^{TX}[k,p]\right]$.

### 3.5.1 Fractional frequency offset

The fractional frequency offset (FFO) is the frequency offset normalized by the carrier frequency. The FFO for the *k*th mode-0 step within a subevent is denoted as *FFO*[*k*], and is given by

$$FFO\left[k\right] = 10^6 \frac{f\left[k,1\right] - f_0\left[k\right]\left(1 + 10^{-6} \times FAE\left[k\right]\right)}{f_0\left[k\right]\left(1 + 10^{-6} \times FAE\left[k\right]\right)} ppm, \qquad k = 1, \ldots, M$$

where *FAE*[*k*] is the value of the FFO actuation error (FAE) for the CS channel used in step *k* taken from the mode-0 FFO actuation error table of the local Controller. The FAE is communicated by the reflector and represents a known additional fractional frequency offset error on mode-0 channels.

The value of each FAE for each CS channel shall satisfy -4 ppm ≤ *FAE* ≤ 3.96875 ppm. For all CS mode-0 steps, the absolute value of *FFO*[*k*] shall be less than 50 ppm.

For all CS mode-0 steps within a subevent, the values of |*FFO*[*k*] − *FFO*[1]| shall be less than 1 ppm for 2 ≤ *k* ≤ *M*.

The receiving device of a mode-0 CS tone should estimate and report the FFO of the transmitting device of the mode-0 CS tone. In this case, the FFO is given by

$$FFO_{RX}\left[k\right] = 10^6 \frac{f_{RX}\left[k\right] - f_0\left[k\right]\left(1 + 10^{-6} \times FAE\left[k\right]\right)}{f_0\left[k\right]\left(1 + 10^{-6} \times FAE\left[k\right]\right)} ppm, \qquad k = 1, \ldots, M$$

where $f_{RX}[k]$ is the receiving device's estimate of the average frequency of the received CS tone in step k. The receiving device should compensate for its own local frequency

actuation error (LFAE) as defined in Section 6.3, as part of its $FFO_{RX}[k]$ estimate. The term $FFO_{RX^*}$ is used to refer to the estimated FFO derived from all mode-0 CS tones within a CS subevent.

### 3.5.2 Expected transmitted frequencies

The FFO for a CS subevent, $FFO_E$, is defined as the value of $FFO[1]$ for the device transmitting the CS tone during the first mode-0 step of the same subevent.

The expected transmit frequency of each CS device for the $k$th step is given by

$$f_E[k] = f_0[k]\left(1 + 10^{-6} FFO_E\right)$$

where $k = M + 1, \ldots, K + M$.

For the non-mode-0 steps $M + 1 \leq k \leq M + K$, and for each phase measurement period $1 \leq p \leq P[k]$, the average transmitted frequency $f[k, p]$ shall satisfy

$$|f[k, \mathrm{p}] - f_E[k]| < 10\mathrm{kHz}$$

for 95% of CS tone transmissions within non-mode-0 steps.

The center frequency of the transmitted CS_SYNC packet within any non-mode-0 step shall not deviate by more than 20 kHz with respect to the expected transmit frequency $f_E[k]$, for 95% of CS_SYNC transmissions within non-mode-0 steps.

The center frequency of the transmitted CS_SYNC packet within any mode-0 step $1 \leq k \leq M$ shall not deviate by more than 20 kHz with respect to the average frequency of the transmitted CS tone $f[k, 1]$, for 95% of CS_SYNC transmissions within mode-0 steps.

# 4 RECEIVER CHARACTERISTICS

The reference sensitivity level referred to in this section is -70 dBm. The packet error rate corresponding to the defined bit error ratio (BER) shall be used in all receiver characteristic measurements.

## 4.1 Actual sensitivity level

The actual sensitivity level is defined as the receiver input level for which the BER specified in Table 4.1 is achieved.

| Maximum Supported Payload Length (bytes) | BER (%) |
|---|---|
| 1 to 37 | 0.1 |
| 38 to 63 | 0.064 |
| 64 to 127 | 0.034 |
| 128 to 255 | 0.017 |

*Table 4.1: Actual sensitivity BER by maximum payload length*

The actual sensitivity level of the receiver for a given PHY shall be as specified in Table 4.2. This shall apply when receiving signals specified in Section 3 together with any combination of the following allowed parameter variations:

- Initial frequency offset
- Frequency drift
- Symbol rate
- Frequency deviation

| PHY | Sensitivity (dBm) |
|---|---|
| LE Uncoded PHYs | ≤ -70 |
| LE Coded PHY with S=2 coding | ≤ -75 |
| LE Coded PHY with S=8 coding | ≤ -82 |

*Table 4.2: Receiver sensitivity for a given PHY*

## 4.2 Interference performance

The interference performance shall be measured with a wanted signal of -67 dBm on the LE Uncoded PHYs, -72 dBm on the LE Coded PHY with S=2 coding, or -79 dBm on the LE Coded PHY with S=8 coding. If the frequency of an interfering signal is outside of the band 2400 MHz to 2483.5 MHz, the out-of-band blocking specification

(see Section 4.3) shall apply. Both the desired and the interfering signal shall be reference signals as specified in Section 4.6. The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 4.3, Table 4.4, Table 4.5, and Table 4.6.

These measurements are made using the LE 1M and LE 2M PHYs.

| Frequency of Interference | Ratio |
|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 21 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$ | 15 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | -17 dB |
| Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$ | -27 dB |
| Image frequency interference[1 2 3], $C/I_{Image}$ | -9 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 1MHz}$ | -15 dB |

*Table 4.3: Interference performance for LE 1M PHY*

| Frequency of Interference | Ratio |
|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 21 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | 15 dB |
| Adjacent (4 MHz) interference[1], $C/I_{4\ MHz}$ | -17 dB |
| Adjacent (≥6 MHz) interference[1], $C/I_{\geq 6\ MHz}$ | -27 dB |
| Image frequency interference[1 2 4], $C/I_{Image}$ | -9 dB |
| Adjacent (2 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 2MHz}$ | -15 dB |

*Table 4.4: Interference performance for LE 2M PHY*

| Frequency of Interference | Ratio |
|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 12 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$ | 6 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | -26 dB |
| Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$ | -36 dB |
| Image frequency interference[1 2 3], $C/I_{Image}$ | -18 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 1MHz}$ | -24 dB |

*Table 4.5: Interference performance for the LE Coded PHY with S=8 coding (125 kb/s data rate)*

| Frequency of Interference | Ratio |
|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 17 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$ | 11 dB |
| Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$ | -21 dB |

| Frequency of Interference | Ratio |
|---|---|
| Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$ | -31 dB |
| Image frequency interference[1 2 3], $C/I_{Image}$ | -13 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 1MHz}$ | -19 dB |

*Table 4.6: Interference performance for the LE Coded PHY with S=2 coding (500 kb/s data rate)*

Notes:

1. If two adjacent frequency specifications from Table 4.3, Table 4.4, Table 4.5, or Table 4.6 (as appropriate) are applicable to the same frequency, the more relaxed specification applies.
2. In-band image frequency.
3. If the image frequency ≠ n×1 MHz, then the image reference frequency is defined as the closest n×1 MHz frequency for integer n.
4. If the image frequency ≠ n×2 MHz, then the image reference frequency is defined as the closest n×2 MHz frequency for integer n.

Any frequencies where the requirements are not met are called spurious response RF channels. Five spurious response RF channels are allowed with a distance of ≥2 MHz from the wanted signal when receiving with 1 Msym/s modulation and a distance of ≥4 MHz when receiving with 2 Msym/s modulation; different spurious response channels are allowed for the two modulation schemes. On these spurious response RF channels, a relaxed interference requirement C/I = -17 dB shall be met by both 1 Msym/s and 2 Msym/s modulation transmitters.

## 4.3 Out-of-band blocking

The out-of-band blocking applies to interfering signals outside the band 2400 MHz to 2483.5 MHz. The out-of-band suppression (or rejection) shall be measured with a wanted signal 3 dB over the reference sensitivity level. The interfering signal shall be a continuous wave signal. The desired signal shall be a reference signal as specified in Section 4.6, with a center frequency of 2426 MHz. The BER shall be ≤ 0.1%. The out-of-band blocking shall fulfill the following requirements:

| Interfering Signal Frequency | Interfering Signal Power Level | Measurement resolution |
|---|---|---|
| 30 MHz to 2000 MHz | -30 dBm | 10 MHz |
| 2003 MHz to 2399 MHz | -35 dBm | 3 MHz |
| 2484 MHz to 2997 MHz | -35 dBm | 3 MHz |
| 3000 MHz to 12.75 GHz | -30 dBm | 25 MHz |

*Table 4.7: Out-of-band suppression (or rejection) requirements*

Up to 10 exceptions are permitted, which are dependent upon the given RF channel and are centered at a frequency which is an integer multiple of 1 MHz:

- For at least 7 of these spurious response frequencies, a reduced interference level of at least -50 dBm is allowed in order to achieve the required BER ≤ 0.1%.
- For a maximum of 3 of the spurious response frequencies, the interference level may be lower.

## 4.4 Intermodulation characteristics

The actual sensitivity performance, BER ≤ 0.1%, shall be met under the following conditions:

- The wanted signal shall be at a frequency $f_0$ with a power level 6 dB over the reference sensitivity level. The wanted signal shall be a reference signal as specified in Section 4.6.
- A static sine wave signal shall be at a frequency $f_1$ with a power level of -50 dBm.
- An interfering signal shall be at a frequency $f_2$ with a power level of -50 dBm. The interfering signal shall be a reference signal as specified in Section 4.6.

When receiving with 1 Msym/s modulation, frequencies $f_0$, $f_1$ and $f_2$ shall be chosen such that $f_0 = 2 \times f_1 - f_2$ and

$| f_2 - f_1 |$ = n × 1 MHz, where n can be 3, 4, or 5.

When receiving with 2 Msym/s modulation, frequencies $f_0$, $f_1$ and $f_2$ shall be chosen such that

$f_0 = 2 \times f_1 - f_2$ and

$| f_2 - f_1 |$ = n × 2 MHz, where n can be 3, 4, or 5.

The system shall fulfill at least one of the three alternatives (n=3, 4, or 5); different modulation schemes can use different alternatives.

## 4.5 Maximum usable level

The maximum usable input level the receiver can operate at shall be greater than -10 dBm, and the BER shall be less than or equal to 0.1% at -10 dBm input power. The input signal shall be a reference signal as specified in Section 4.6.

## 4.6 Reference signal definition

The reference signal for LE is defined as:

Modulation = GFSK
Modulation index = 0.5 ± 1% for standard modulation index, 0.5 ± 0.5% for

stable modulation index
BT = 0.5 ± 1%
Data Bit Rate =

- 1 Mb/s ±1 ppm for the LE 1M PHY
- 2 Mb/s ±1 ppm for the LE 2M PHY
- 125 kb/s ±1 ppm for the LE Coded PHY when using S=8 coding
- 500 kb/s ±1 ppm for the LE Coded PHY when using S=2 coding

Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
Frequency accuracy better than ±1 ppm

## 4.7 Stable modulation index

An LE device may have a receiver that can take advantage of the fact that the remote device indicates support for the Stable Modulation Index - Transmitter feature (see [Vol 6] Part B, Section 4.6). Such a receiver is said to have stable modulation index support.

## 4.8 Received Signal Strength Indication

If a device supports Received Signal Strength Indication (RSSI) the accuracy should be ±6 dB. If the device is aware that the RSSI varies across frequencies, then it should update the RSSI value of a packet depending on the frequency that the packet was received on before using the value, e.g., before reporting it to the Host.

# 5 ANTENNA SWITCHING

## 5.1 Antenna Switching for AoA/AoD

A device may support an antenna array consisting of two or more antennae that are controlled by a switch. The device switches between the antennae either while receiving the Constant Tone Extension of a packet (see [Vol 6] Part B, Section 2.1.5) (Angle of Arrival method) or while transmitting the Constant Tone Extension of a packet (Angle of Departure method). The switching takes place during time periods called switch slots. The first 4 µs of the Constant Tone Extension are termed the guard period and the next 8 µs are termed the reference period. The receiving Link Layer captures IQ samples during the reference period and during time periods called sample slots.

When a Controller that supports two or more antennae transmits a packet containing an AoD Constant Tone Extension or receives a packet containing an AoA Constant Tone Extension, it shall switch the antennae according to the switching pattern configured by its Host.

The first antenna in the pattern shall be used during the reference period (see [Vol 6] Part B, Section 2.5.1 for the Constant Tone Extension format). The second antenna in the pattern shall be used during the first sample slot, the third antenna during the second sample slot, and so on. The same antenna ID may appear more than once in the pattern. The antenna in use shall only be changed during the guard period and switch slots.

If the pattern specified by the Host is exhausted before the last sample slot, it shall be restarted from the beginning (this can happen more than once); that is, the first antenna in the pattern is used in the sample slot following that used for the last antenna in the pattern. If the pattern has not been completely used by the end of the Constant Tone Extension, any remaining terms shall be ignored.

The Controller shall support the antenna switching pattern lengths specified in Table 5.1. It may support other lengths.

| Number of antennae | Mandatory supported lengths of antenna switching pattern |
|---|---|
| 2 | 1 to 4 |
| 3 | 1 to 8 |
| ≥ 4 | 1 to 12 |

*Table 5.1: Mandatory supported lengths of antenna switching pattern*

## 5.2 Receiver characteristics for AoA/AoD

A receiver shall meet the requirements of Section 5.2.2 when the test switching pattern specified in Section 5.2.3 is used. The definitions in Section 5.2.1 shall apply throughout those sections.

### 5.2.1 Definitions

$i = \sqrt{-1}$

All angles are measured in radians.

*Arg*(*x*) is the principal value of the argument, or phase angle, of the complex number *x*, in the range $(-\pi, \pi]$.

*principal*(*a*) is the principal value of the real angle *a* That is, it equals $a + 2\pi k$ where *k* is an integer chosen so that $-\pi < principal(a) \leq \pi$.

The sample slots of a Constant Tone Extension shall be numbered from 1 to S.

$A_0$, $A_1$, etc. shall be the identifiers of the different antennae in the switching pattern; $A_0$ shall be the antenna used during the reference period.

Given the IQ samples *I*(*n*) and *Q*(*n*) from sample slot *n*, the phase $\phi(n)$ equals $Arg(I(n) + iQ(n))$.

For $2 < n \leq S$, the relative phase $\theta(n)$ equals

$$principal(\phi(n) - 2\phi(n-1) + \phi(n-2))$$

For each antenna $A_m$ except $A_0$, the set of relative phase values for $A_m$ is:

$$RP(m) = \{\theta(n) | 2 < n \leq S, A_m \text{ used in sample slot n}\}$$

and the mean of the relative phase values is:

$$MRP(m) = Arg\left(\sum_{\theta \in RP(m)} e^{i\theta}\right)$$

For $2 < n < S$, the reference phase deviation $\psi(n)$ equals

$$principal(\phi(n+1) - 3\phi(n-1) + 2\phi(n-2))$$

The set of reference phase deviation values is:

$$RPD = \{\psi(n) | 2 < n < S, A_0 \text{ not used in sample slot n}\}$$

and its mean is:

$$MRPD = Arg\left(\sum_{\psi \in RPD} e^{i\psi}\right)$$

If a sample slot *n* has $I(n) = Q(n) = 0$ or has no valid sample available, then $\phi(n)$ and any value derived from it shall be considered undefined. Such undefined values shall be omitted from the sets *RP*(*m*) and *RPD*.

### 5.2.2 Requirements

For each antenna $A_m$ used in the switching pattern except $A_0$, the results of the summations in the formulae for *MRP*(*m*) and *MRPD* shall be non-zero.

For each antenna $A_m$ used in the switching pattern except $A_0$, 95% of the values *v* in the set *RP*(*m*) shall meet –0.52 ≤ *principal* (*v* – *MRP*(*m*)) ≤0.52.

The condition –1.125 ≤ *MRPD* ≤ 1.125 shall be true.

### 5.2.3 Test switching pattern

When testing the receiver characteristics, the switching patterns specified in Table 5.2 shall be used by an AoD transmitter or an AoA receiver. The antennae $A_0$, $A_1$, etc. shall be chosen by the implementation and shall all be different.

Note: $A_0$ is the antenna used during the reference period.

| Number of antennae | Test switching pattern |
|---|---|
| 2 | $A_0, A_1, A_0, A_0$ |
| 3 | $A_0, A_1, A_0, A_0, A_0, A_2, A_0, A_0$ |
| 4 or more | $A_0, A_1, A_0, A_0, A_0, A_2, A_0, A_0, A_0, A_3, A_0, A_0$ |

*Table 5.2: Test switching patterns*

## 5.3 Antenna Switching for Channel Sounding

A device supporting CS that supports an antenna array consisting of two or more antennae may switch between the antennae during the phase measurement period in each CS step that includes CS tone exchanges.

An antenna path is the combination of a given antenna selected by the initiator and a given antenna selected by the reflector, which is further described in [Vol 6] Part H, Section 4.7. The number of antenna paths is designated N_AP, whose value shall be set to a minimum of 1 and a maximum of 4.

Antenna paths are identified using indices AP1, AP2, AP3, AP4, with the maximum index dependent on the value of N_AP selected. Each antenna path is identified using

the colon separated nomenclature of "device A:device B" antenna numbering. Most antenna configurations described below are 1:X or X:1 configurations, where X is in the set of 1 to 4. In these configurations, antenna path AP1 is assigned to the 1:1 antenna combination, AP2 is assigned to the 1:2 or 2:1 combination, AP3 is assigned to the 1:3 or 3:1 combination, and AP4 is assigned to the 1:4 or 4:1 combination. The exception is the 2:2 configuration, where AP1 is assigned to 1:1, AP2 is assigned to 1:2, AP3 is assigned to 2:1 and AP4 is assigned to 2:2.

CS supports four groups of antenna configurations, which are identified by antenna configuration index (ACI) values:

- 1:1 configuration, where both devices A and B support only 1 antenna each
- 1:N_AP configuration, where device A supports 1 antenna, device B supports N_AP antennae, and N_AP is a value in the 2 to 4 range
- N_AP:1 configuration, where device A supports N_AP antennae, device B supports 1 antenna, and N_AP is a value in the 2 to 4 range
- 2:2 configuration, where device A supports 2 antennae, device B supports 2 antennae, and N_AP = 4

There are a total of eight possible combinations of these configurations, as shown in Table 5.3.

| Antenna Configuration Index (ACI) | Total Number of Paths | Number of Device A Antennae | Number of Device B Antennae | Configuration |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1:1 |
| 1 | 2 | 2 | 1 | N_AP:1 |
| 2 | 3 | 3 | 1 | N_AP:1 |
| 3 | 4 | 4 | 1 | N_AP:1 |
| 4 | 2 | 1 | 2 | 1:N_AP |
| 5 | 3 | 1 | 3 | 1:N_AP |
| 6 | 4 | 1 | 4 | 1:N_AP |
| 7 | 4 | 2 | 2 | 2:2 |

*Table 5.3: Supported antenna path configurations for CS*

# 6 PHASE MEASUREMENTS

## 6.1 Reference receiver for phase-based ranging

The reference down-conversion for the CS tone for step $k = M + 1, \ldots, M + K$, is defined at the antenna of the device as

$$\hat{x}\left(k, p, t\right) = LPF\left[x\left(p, t\right)e^{-j2\pi f_E[k]t}\right]$$

where $x(p,t)$ is the signal at the antenna of the device used in phase measurement period $p$ as defined in Appendix B, $f_E[k]$ is the expected transmit frequency as defined in Section 3.5.2, and LPF is a low-pass filter that removes the high frequency components.

Define the transmission observation window as

$$\mathrm{TxWin}\left[k, p\right] = \left[t^{TX}_{CT,1}[k, p], t^{TX}_{CT,2}[k, p]\right]$$

Denote the expected start and end times of the CS tone to be received by a device for the $k$th step in phase measurement period $p$ as $t^{RX}_{CT,1}[k, p]$ and $t^{RX}_{CT,2}[k, p]$, respectively, if a received CS tone exists within that step and phase measurement period. Within this phase measurement period, the time $t = 0$ corresponds to the start of the first transmission within a CS subevent.

The receive observation window is defined as

$$\mathrm{RxWin}\left[k, p\right] = \left[t^{RX}_{CT,1}[k, p], t^{RX}_{CT,2}[k, p]\right]$$

## 6.2 Phase measurement accuracy

The observed average transmitted phase for step $k$, phase measurement period $p$, is defined as

$$\overline{\phi_{TX}}\left[k, p\right] = \angle\frac{1}{t^{TX}_{CT,2}[k, p] - t^{TX}_{CT,1}[k, p]}\int_{\mathrm{TxWin}[k]}\hat{x}\left(k, p, t\right)\,\mathrm{dt}$$

The observed average received phase for step $k$, phase measurement period $p$, is defined as

$$\overline{\phi_{RX}}\left[k, p\right] = \angle\frac{1}{t^{RX}_{CT,2}[k, p] - t^{RX}_{CT,1}[k, p]}\int_{\mathrm{RxWin}[k]}\hat{x}\left(k, p, t\right)\,\mathrm{dt}$$

The internal phase offset for step k as described in Section 6.1 is defined as

$$\theta_C[k,p] = principal\left(\overline{\phi_{TX}}[k,p] - \overline{\phi_{RX}}[k,p]\right) + \angle PCT[k,p], \quad k = \mathrm{M}+1,\ldots,\mathrm{K}$$

where *principal*(·) is defined in Section 5.2.1 and *PCT*[*k*, *p*] is the phase correction terms for step *k*, phase measurement period *p*, provided by the device.

Denote $\theta_{C,UW}[k,p]$ as the phase-unwrapped version of $\theta_C[k,p]$.

Denote $\alpha[p]f_E[k] + \beta[p]$ as the solution to the linear regression of the set of points defined by ($f_E[k]$, $\theta_{C,UW}[k,p]$). Values of $\theta_{C,UW}[k,p]$, where the quality indication term for the phase measurement provided by the device is not the highest quality the device supports (see [Vol 6] Part H, Section 4.6), are not used in the calculation of the linear regression.

For any subevent where

- a CS tone is exchanged at least once for all CS channels,
- the transmit signal satisfies the conditions for frequency offset described in Section 3.5.2, and
- the receiver input level is -70 dBm,

the solution to the linear regression shall satisfy

$$|\alpha[p]| < 2\pi \times 1.7\text{ns}$$

for 95% of subevents, and

$$|principal(\alpha[p]f_E[k] + \beta[p] - \theta_{C,UW}[k,p])| < 0.28\text{ rad}$$

for 95% of the values of $\theta_C[k,p]$ within a subevent.

## 6.3 Frequency actuation error correction

If the initiator or reflector is aware of any local frequency actuation error (LFAE), then it shall also compensate for the phase rotation caused by the LFAE. The requirement for this compensation is applicable to both phase measurements over CS tones (see [Vol 6] Part H, Section 4.6), as well as over the sounding sequence payload of the CS_SYNC (see [Vol 6] Part H, Section 3.3.1).

For CS tones the time period over which this compensation occurs is between reception and transmission of the center of the CS tone at the antenna port, T_PM_CENTER_DELTA. For CS step mode-2, T_PM_CENTER_DELTA is specified in [Vol 6] Part H, Section 4.3.3. For CS step mode-3, T_PM_CENTER_DELTA is specified in [Vol 6] Part H, Section 4.3.4. The LFAE represents the frequency offset between the generated frequency and the expected frequency (see Section 3.5.2). In

the absence of any circuit propagation delays or synchronization errors, this correction can be nominally expressed as

$$\angle PCT' = \angle PCT - 2\pi \Delta f_k \times T_PM_CENTER_DELTA$$

where $\Delta f_k$ represents the frequency actuation error (in Hertz) for that local device at frequency k and the terms in parentheses represent the nominal time separation between the centers of the device's transmitted and received CS tones. Devices should modify this equation as necessary to account for any known circuit delays or synchronization errors impacting this correction.

Similarly, these corrections are also applied to the sounding sequence CS_SYNC field when phase information is extracted. The period over which this compensation occurs is between the reception and transmission of the center of the sounding sequence field at the antenna port, T_SY_CENTER_DELTA. For CS step mode-1, T_SY_CENTER_DELTA is specified in [Vol 6] Part H, Section 4.3.2. For CS step mode-3, T_SY_CENTER_DELTA is specified in [Vol 6] Part H, Section 4.3.4. The LFAE represents the frequency offset between the generated frequency and the expected frequency (see Section 3.5.2). In the absence of any circuit propagation delays or synchronization errors, this correction can be nominally expressed as

$$\angle SS_PCT' = \angle SS_PCT - 2\pi \Delta f_k \times T_SY_CENTER_DELTA$$

where $\Delta f_k$ represents the frequency actuation error (in Hertz) for that local device at frequency k and the terms in parentheses represent the nominal time separation between the centers of the device's transmitted and received sounding sequence fields. Devices should modify this equation as necessary to account for any known circuit delays or synchronization errors impacting this correction.

## 6.4 Phase measurement timing

The PCT value is measured during the tone's phase measurement period of length T_PM as shown in Figure 6.1. If there is no antenna switching, the phase measurement period of the tone represents the entire tone. If the measurement involves antenna switching, then the phase measurement period of the tone represents the time between antenna switch periods. In addition, a 1 µs tone measurement exclusion period is defined for the receiver (see [Vol 6] Part H, Section 4.6) to compensate for device clock drift, which might introduce some ambiguity as to the precise location of the beginning and end of the phase measurement period. The valid region of the tone is the time between the exclusion periods.

A frequency actuation error will cause the phase of this tone to increase or decrease linearly during a tone. The PCT term should represent the phase measured in the center of the valid region of the tone so that the corrected PCT term described in Section 6.3 can be used to compensate for the local frequency actuation error (LFAE).

A device should not compensate for any residual frequency error it observes during the phase measurement period except its own LFAE.

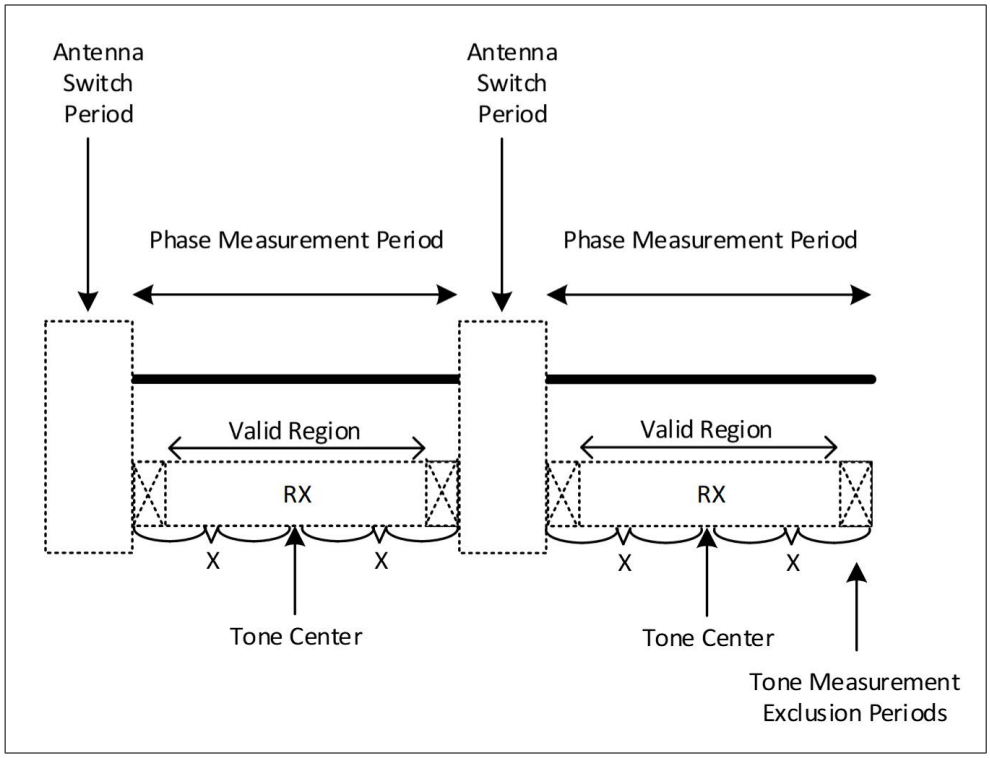



*Figure 6.1: Definition of the center of the valid region of the CS tone and CS tone center*

In addition to the phase, the amplitude of the PCT is used to convey the measured signal amplitude as described in [Vol 6] Part H, Section 4.6. Any frequency offset caused by a frequency actuation error should not impact the measured amplitude.

## 6.5 Inline PCT Transfer

This section applies when the CS reflector has inline phase transfer (IPT) enabled.

A CS reflector shall only return PCT values where the imaginary (Q) part of each PCT value is zero. For such PCT values, the real (I) part of the PCT is used to convey the measured signal amplitude as described in [Vol 6] Part H, Section 4.6.

A CS reflector shall satisfy the phase accuracy requirements (see Section 6.2). When IPT is enabled, the phase of each PCT value is zero. Because the phase of the PCT is zero, a device that supports IPT must adjust the phase of its CS Tone transmissions accordingly.

# Appendix A Test Conditions

## A.1 Normal operating conditions

### A.1.1 Normal temperature and air humidity

The normal operating temperature shall be declared by the product manufacturer. The nominal test temperature shall be within ±10°C of the normal operating temperature.

### A.1.2 Nominal supply voltage

The nominal test voltage for the equipment under normal test conditions shall be the nominal supply voltage as declared by the product manufacturer.

# Appendix B Example test equipment setup for Channel Sounding receiver and transmitter

Figure B.1 shows a block diagram of components used to verify CS phase-based and round-trip time reporting accuracy. The vector signal generator represents the reference transmitter and transmits CS tones and packets toward the IUT (Implementation Under Test). The IUT transmits CS tones and packets toward the reference receiver represented by the vector signal analyzer. The vector signal generator and the vector signal analyzer are synchronized with the timing cadence of the CS procedure as described in [Vol 6] Part H, Section 4.5. In Figure B.1, x(t) represents the capture by the vector signal analyzer of the RF signal for each CS step. This capture should be continuous and should include the TX packet from the vector signal generator and the TX packet from the IUT.

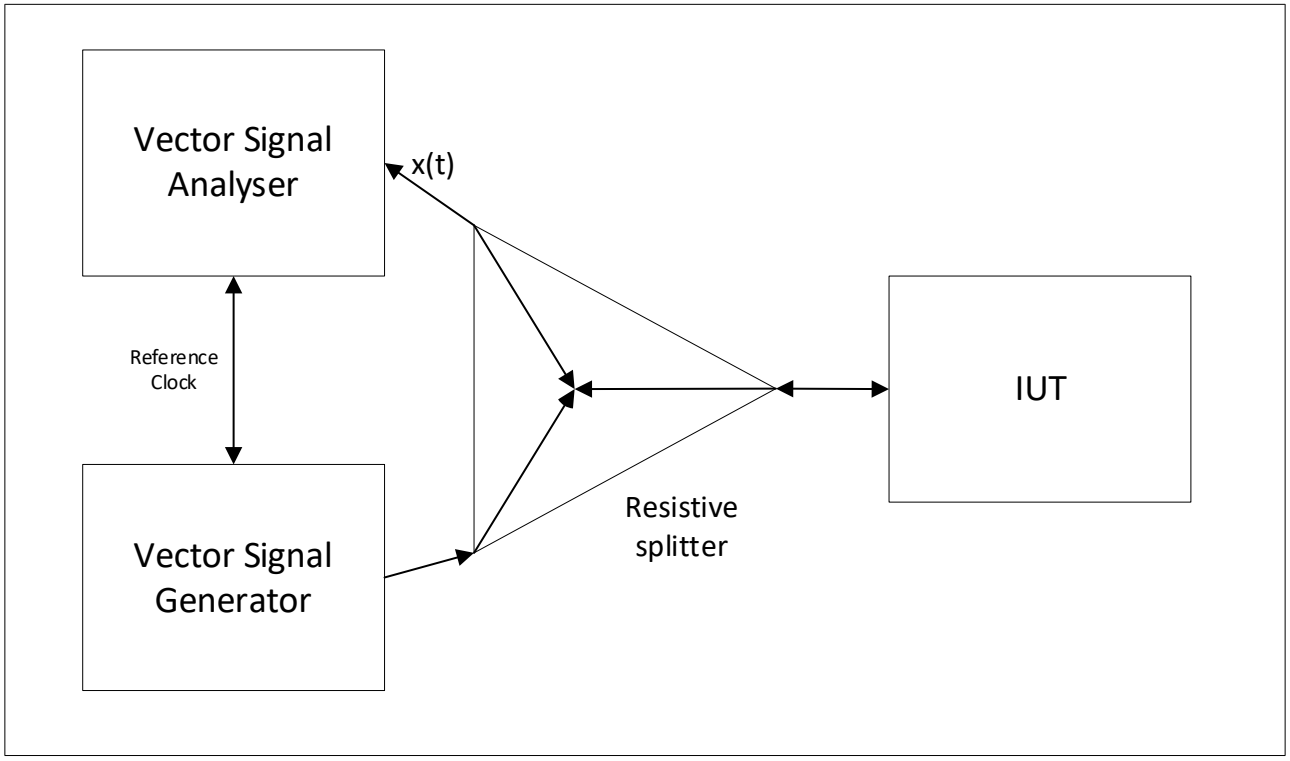



*Figure B.1: Channel Sounding reference block diagram*

Low Energy Controller
Part B

# LINK LAYER SPECIFICATION

*This Part describes the Bluetooth Low Energy Link Layer.*

# CONTENTS

**1 General description ..... 2979**
1.1 Link Layer states ..... 2979
1.1.1 Permitted state and role combinations ..... 2981
1.1.2 Devices supporting only some states ..... 2982
1.2 Bit ordering ..... 2982
1.3 Device address ..... 2983
1.3.1 Public device address ..... 2983
1.3.2 Random device address ..... 2983
1.3.2.1 Static device address ..... 2984
1.3.2.2 Private device address generation ..... 2984
1.3.2.3 Private device address resolution ..... 2986
1.4 Physical channel ..... 2987
1.4.1 Physical channel indices ..... 2987

**2 Air interface packets ..... 2989**
2.1 Packet format for the LE Uncoded PHYs ..... 2989
2.1.1 Preamble ..... 2989
2.1.2 Access Address ..... 2990
2.1.3 PDU ..... 2992
2.1.4 CRC ..... 2992
2.1.5 Constant Tone Extension ..... 2993
2.2 Packet format for the LE Coded PHY ..... 2993
2.2.1 Preamble ..... 2994
2.2.2 Access Address ..... 2994
2.2.3 Coding Indicator ..... 2994
2.2.4 PDU ..... 2994
2.2.5 CRC ..... 2995
2.2.6 TERM1 and TERM2 ..... 2995
2.3 Advertising physical channel PDU ..... 2995
2.3.1 Advertising PDUs ..... 2997
2.3.1.1 ADV_IND ..... 2998
2.3.1.2 ADV_DIRECT_IND ..... 2998
2.3.1.3 ADV_NONCONN_IND ..... 2999
2.3.1.4 ADV_SCAN_IND ..... 2999
2.3.1.5 ADV_EXT_IND ..... 2999
2.3.1.6 AUX_ADV_IND ..... 3001
2.3.1.7 AUX_SYNC_IND ..... 3002
2.3.1.8 AUX_CHAIN_IND ..... 3002
2.3.1.9 AUX_SYNC_SUBEVENT_IND ..... 3003

2.3.1.10 AUX_SYNC_SUBEVENT_RSP ..... 3004
2.3.1.11 ADV_DECISION_IND ..... 3004
2.3.2 Scanning PDUs ..... 3006
2.3.2.1 SCAN_REQ and AUX_SCAN_REQ ..... 3006
2.3.2.2 SCAN_RSP ..... 3007
2.3.2.3 AUX_SCAN_RSP ..... 3007
2.3.3 Initiating PDUs ..... 3008
2.3.3.1 CONNECT_IND and AUX_CONNECT_REQ ..... 3008
2.3.3.2 AUX_CONNECT_RSP ..... 3010
2.3.4 Common Extended Advertising Payload Format ..... 3011
2.3.4.1 AdvA field ..... 3013
2.3.4.2 TargetA field ..... 3013
2.3.4.3 CTEInfo field ..... 3013
2.3.4.4 AdvDataInfo field ..... 3014
2.3.4.5 AuxPtr field ..... 3014
2.3.4.6 SyncInfo field ..... 3016
2.3.4.7 TxPower field ..... 3018
2.3.4.8 ACAD field ..... 3018
2.3.4.9 Host Advertising Data ..... 3018
2.4 Data Physical Channel PDU ..... 3019
2.4.1 LL Data PDU ..... 3021
2.4.2 LL Control PDU ..... 3021
2.4.2.1 LL_CONNECTION_UPDATE_IND ..... 3025
2.4.2.2 LL_CHANNEL_MAP_IND ..... 3026
2.4.2.3 LL_TERMINATE_IND ..... 3026
2.4.2.4 LL_ENC_REQ ..... 3026
2.4.2.5 LL_ENC_RSP ..... 3027
2.4.2.6 LL_START_ENC_REQ ..... 3027
2.4.2.7 LL_START_ENC_RSP ..... 3027
2.4.2.8 LL_UNKNOWN_RSP ..... 3027
2.4.2.9 LL_FEATURE_REQ ..... 3028
2.4.2.10 LL_FEATURE_RSP ..... 3028
2.4.2.11 LL_PAUSE_ENC_REQ ..... 3028
2.4.2.12 LL_PAUSE_ENC_RSP ..... 3028
2.4.2.13 LL_VERSION_IND ..... 3029
2.4.2.14 LL_REJECT_IND ..... 3029
2.4.2.15 LL_PERIPHERAL_FEATURE_REQ ..... 3030
2.4.2.16 LL_CONNECTION_PARAM_REQ ..... 3030
2.4.2.17 LL_CONNECTION_PARAM_RSP ..... 3031
2.4.2.18 LL_REJECT_EXT_IND ..... 3031
2.4.2.19 LL_PING_REQ ..... 3032
2.4.2.20 LL_PING_RSP ..... 3032

2.4.2.21 LL_LENGTH_REQ and LL_LENGTH_RSP ..... 3032
2.4.2.22 LL_PHY_REQ and LL_PHY_RSP ..... 3033
2.4.2.23 LL_PHY_UPDATE_IND ..... 3033
2.4.2.24 LL_MIN_USED_CHANNELS_IND ..... 3034
2.4.2.25 LL_CTE_REQ ..... 3034
2.4.2.26 LL_CTE_RSP ..... 3035
2.4.2.27 LL_PERIODIC_SYNC_IND ..... 3035
2.4.2.28 LL_CLOCK_ACCURACY_REQ and LL_CLOCK_ACCURACY_RSP ..... 3036
2.4.2.29 LL_CIS_REQ ..... 3037
2.4.2.30 LL_CIS_RSP ..... 3039
2.4.2.31 LL_CIS_IND ..... 3040
2.4.2.32 LL_CIS_TERMINATE_IND ..... 3040
2.4.2.33 LL_POWER_CONTROL_REQ ..... 3041
2.4.2.34 LL_POWER_CONTROL_RSP ..... 3042
2.4.2.35 LL_POWER_CHANGE_IND ..... 3042
2.4.2.36 LL_SUBRATE_REQ ..... 3043
2.4.2.37 LL_SUBRATE_IND ..... 3044
2.4.2.38 LL_CHANNEL_REPORTING_IND ..... 3044
2.4.2.39 LL_CHANNEL_STATUS_IND ..... 3045
2.4.2.40 LL_PERIODIC_SYNC_WR_IND ..... 3046
2.4.2.41 LL_FEATURE_EXT_REQ and LL_FEATURE_EXT_RSP ..... 3046
2.4.2.42 LL_CS_SEC_REQ ..... 3047
2.4.2.43 LL_CS_SEC_RSP ..... 3047
2.4.2.44 LL_CS_CAPABILITIES_REQ and LL_CS_CAPABILITIES_RSP ..... 3048
2.4.2.45 LL_CS_CONFIG_REQ ..... 3056
2.4.2.46 LL_CS_CONFIG_RSP ..... 3061
2.4.2.47 LL_CS_REQ ..... 3061
2.4.2.48 LL_CS_RSP ..... 3064
2.4.2.49 LL_CS_IND ..... 3064
2.4.2.50 LL_CS_TERMINATE_REQ and LL_CS_TERMINATE_RSP ..... 3065
2.4.2.51 LL_CS_FAE_REQ ..... 3065
2.4.2.52 LL_CS_FAE_RSP ..... 3066
2.4.2.53 LL_CS_CHANNEL_MAP_IND ..... 3066
2.4.2.54 LL_FRAME_SPACE_REQ ..... 3067
2.4.2.55 LL_FRAME_SPACE_RSP ..... 3068
2.4.2.56 LL_OTA_UTP_IND ..... 3068
2.4.2.57 LL_CONNECTION_RATE_REQ ..... 3068
2.4.2.58 LL_CONNECTION_RATE_IND ..... 3070

2.5 Constant Tone Extension and IQ sampling ..... 3071
2.5.1 Constant Tone Extension structure and types ..... 3071
2.5.2 CTEInfo field ..... 3072
2.5.3 Transmitting Constant Tone Extensions ..... 3073
2.5.4 IQ sampling ..... 3073
2.6 Isochronous Physical Channel PDU ..... 3074
2.6.1 Connected Isochronous PDU ..... 3075
2.6.2 Broadcast Isochronous PDU ..... 3076
2.6.3 BIG Control PDU ..... 3077
2.6.3.1 BIG_CHANNEL_MAP_IND ..... 3078
2.6.3.2 BIG_TERMINATE_IND ..... 3079

**3 Bit stream processing ..... 3080**
3.1 Error checking ..... 3080
3.1.1 CRC generation ..... 3080
3.2 Data whitening ..... 3081
3.3 Coding ..... 3082
3.3.1 Forward Error Correction encoder ..... 3083
3.3.2 Pattern mapper ..... 3083

**4 Air Interface protocol ..... 3084**
4.1 Frame Space ..... 3084
4.1.1 Inter Frame Space ..... 3084
4.1.2 Minimum AUX Frame Space ..... 3084
4.1.3 Minimum Isochronous Channel Subevent Space ..... 3085
4.1.4 Minimum Channel Sounding subevent space ..... 3086
4.1.5 Minimum Connection Event Spacing ..... 3087
4.2 Timing requirements ..... 3087
4.2.1 Active clock accuracy ..... 3087
4.2.2 Sleep clock accuracy ..... 3088
4.2.3 Range delay ..... 3089
4.2.4 Window widening ..... 3089
4.3 Link Layer device filtering ..... 3093
4.3.1 Filter Accept List ..... 3094
4.3.2 Advertising filter policy ..... 3094
4.3.3 Scanning filter policy ..... 3094
4.3.3.1 Extended scanning filter policies ..... 3095
4.3.3.2 Decision scanning filter policy modes ..... 3095
4.3.4 Initiator filter policy ..... 3096
4.3.5 Periodic sync establishment filter policy ..... 3096
4.4 Non-connected states ..... 3097
4.4.1 Standby state ..... 3097
4.4.2 Advertising state ..... 3097

4.4.2.1 Advertising channel index selection ..... 3099
4.4.2.2 Advertising events ..... 3100
4.4.2.3 Connectable and scannable undirected event type ..... 3105
4.4.2.4 Connectable directed event type ..... 3108
4.4.2.5 Scannable undirected event type ..... 3112
4.4.2.6 Non-connectable and non-scannable undirected event type ..... 3116
4.4.2.7 Connectable undirected event type ..... 3118
4.4.2.8 Scannable directed event type ..... 3120
4.4.2.9 Non-connectable and non-scannable directed event type ..... 3120
4.4.2.10 Advertising Sets ..... 3121
4.4.2.11 Using AdvDataInfo (ADI) ..... 3122
4.4.2.12 Periodic advertising ..... 3122
4.4.2.13 Requirements ..... 3127
4.4.3 Scanning state ..... 3127
4.4.3.1 Passive scanning ..... 3129
4.4.3.2 Active scanning ..... 3129
4.4.3.3 Advertising sets ..... 3130
4.4.3.4 Scanning for periodic advertisements ..... 3130
4.4.3.5 Advertising reports ..... 3130
4.4.3.6 Decision PDU scanning ..... 3131
4.4.3.7 Requirements ..... 3132
4.4.3.8 Monitoring Advertisers ..... 3133
4.4.4 Initiating state ..... 3133
4.4.4.1 Connect requests on the primary advertising physical channel ..... 3134
4.4.4.2 Connect requests on the secondary advertising physical channel ..... 3134
4.4.4.3 Requirements ..... 3135
4.4.5 Synchronization state ..... 3135
4.4.5.1 Periodic advertising trains ..... 3135
4.4.5.2 Broadcast Isochronous Streams ..... 3136
4.4.6 Isochronous Broadcasting state ..... 3138
4.4.6.1 Broadcast Isochronous Stream (BIS) ..... 3138
4.4.6.2 Broadcast Isochronous Group (BIG) ..... 3138
4.4.6.3 BIG parameters ..... 3139
4.4.6.4 BIG event ..... 3140
4.4.6.5 Broadcast Isochronous Data ..... 3142
4.4.6.6 BIS subevents ..... 3143
4.4.6.7 Control subevents ..... 3145
4.4.6.8 Channel indices ..... 3146

4.4.6.9 Associated periodic advertising train ..... 3147
4.4.6.10 Encryption ..... 3147
4.4.6.11 BIGInfo ..... 3148
4.5 Connection state ..... 3150
4.5.1 Connection events ..... 3151
4.5.2 Supervision timeout ..... 3156
4.5.3 Connection setup - connection event transmit window ..... 3157
4.5.4 Connection setup – Central Role ..... 3158
4.5.5 Connection setup – Peripheral Role ..... 3159
4.5.6 Closing connection events ..... 3161
4.5.7 Sleep clock accuracy ..... 3161
4.5.7.1 Sleep clock accuracy for Channel Sounding ..... 3162
4.5.8 General-purpose channel group index selection ..... 3162
4.5.8.1 Channel classification ..... 3162
4.5.8.2 Channel Selection algorithm #1 ..... 3163
4.5.8.3 Channel Selection algorithm #2 ..... 3165
4.5.9 Acknowledgment and flow control ..... 3171
4.5.9.1 Flow control ..... 3173
4.5.10 Data PDU length management ..... 3173
4.5.11 Control PDU length management ..... 3177
4.5.12 Connection termination and loss ..... 3177
4.5.13 Connected Isochronous Stream (CIS) ..... 3178
4.5.13.1 CIS parameters ..... 3178
4.5.13.2 CIS events and subevents ..... 3179
4.5.13.3 Connected Isochronous Data ..... 3181
4.5.13.4 Closing CIS events ..... 3182
4.5.13.5 Flush Timeout and Flush Points ..... 3183
4.5.13.6 Channel indices ..... 3184
4.5.13.7 CIS Encryption ..... 3184
4.5.14 Connected Isochronous Group (CIG) ..... 3184
4.5.14.1 CIG event ..... 3185
4.5.14.2 Arrangement of multiple CISes ..... 3186
4.5.14.3 States of a CIG ..... 3187
4.5.15 Power level management ..... 3188
4.5.16 Path loss monitoring ..... 3189
4.5.17 ACL data Host transport ..... 3191
4.5.17.1 Flushing of ACL data from Host to Controller ..... 3191
4.5.18 Channel Sounding ..... 3191
4.5.18.1 Channel Sounding procedures and subevents ..... 3192

4.5.18.2 Channel Sounding security ..... 3195
4.6 Feature support ..... 3196
4.6.1 LE Encryption ..... 3199
4.6.2 Connection Parameters Request procedure ..... 3199
4.6.3 Extended Reject Indication ..... 3200
4.6.4 Peripheral-initiated Features Exchange ..... 3200
4.6.5 LE Ping ..... 3200
4.6.6 LE Data Packet Length Extension ..... 3200
4.6.7 LL Privacy ..... 3200
4.6.8 Extended Scanning Filter Policies ..... 3201
4.6.9 Multiple PHYs ..... 3201
4.6.9.1 Symmetric and asymmetric connections .... 3201
4.6.10 Stable Modulation Index - Transmitter ..... 3201
4.6.11 Stable Modulation Index - Receiver ..... 3202
4.6.12 LE Extended Advertising ..... 3202
4.6.13 LE Periodic Advertising ..... 3203
4.6.14 Channel Selection Algorithm #2 ..... 3203
4.6.15 Minimum Number of Used Channels procedure ..... 3203
4.6.16 Connection CTE Request ..... 3203
4.6.17 Connection CTE Response ..... 3204
4.6.18 Connectionless CTE Transmitter ..... 3204
4.6.19 Connectionless CTE Receiver ..... 3204
4.6.20 Antenna Switching During CTE Transmission (AoD) 3204
4.6.21 Antenna Switching During CTE Reception (AoA) ..... 3205
4.6.22 Receiving Constant Tone Extensions ..... 3205
4.6.23 Periodic Advertising Sync Transfer - Sender ..... 3205
4.6.24 Periodic Advertising Sync Transfer - Recipient ..... 3205
4.6.25 Sleep Clock Accuracy Updates ..... 3205
4.6.26 Remote Public Key Validation ..... 3206
4.6.27 Connected Isochronous Stream - Central and Connected Isochronous Stream - Peripheral ..... 3206
4.6.28 Isochronous Broadcaster ..... 3206
4.6.29 Synchronized Receiver ..... 3207
4.6.30 [This section is no longer used] ..... 3207
4.6.31 LE Power Control Request ..... 3207
4.6.32 LE Path Loss Monitoring ..... 3207
4.6.33 Host-set feature bits ..... 3207
4.6.33.1 Connected Isochronous Stream (Host Support) ..... 3207
4.6.33.2 Connection Subrating (Host Support) ..... 3208
4.6.33.3 Advertising Coding Selection (Host Support) ..... 3208
4.6.33.4 Channel Sounding (Host Support) ..... 3208

4.6.33.5 Shorter Connection Intervals (Host Support) ..... 3208
4.6.34 Periodic Advertising ADI Support ..... 3208
4.6.35 Connection Subrating ..... 3209
4.6.36 Channel Classification ..... 3209
4.6.37 Advertising Coding Selection ..... 3209
4.6.38 Periodic Advertising with Responses - Advertiser ..... 3210
4.6.39 Periodic Advertising with Responses - Scanner ..... 3210
4.6.40 LL Extended Feature Set ..... 3210
4.6.41 Channel Sounding ..... 3210
4.6.42 Channel Sounding Tone Quality Indication ..... 3212
4.6.43 Decision-Based Advertising Filtering ..... 3212
4.6.44 ISOAL Unsegmented Framed Mode ..... 3212
4.6.45 Monitoring Advertisers ..... 3212
4.6.46 Frame Space Update ..... 3212
4.6.47 UTP OTA mode ..... 3213
4.6.48 UTP HCI mode ..... 3213
4.6.49 LL_OTA_UTP_IND maximum length ..... 3213
4.6.50 Shorter Connection Intervals ..... 3213
4.6.51 LE Flushable ACL Data ..... 3214
4.6.52 Channel Sounding Enhancement #1 ..... 3214
4.7 Resolving List ..... 3214

**5 Link Layer control ..... 3216**
5.1 Link Layer ACL control procedures ..... 3216
5.1.1 Connection Update procedure ..... 3216
5.1.2 Channel Map Update procedure ..... 3219
5.1.3 Encryption procedure ..... 3220
5.1.3.1 Encryption Start procedure ..... 3220
5.1.3.2 Encryption Pause procedure ..... 3223
5.1.4 Feature Exchange procedure ..... 3224
5.1.4.1 Central-initiated Feature Exchange procedure ..... 3225
5.1.4.2 Peripheral-initiated Feature Exchange procedure ..... 3225
5.1.4.3 Feature Page Exchange procedure ..... 3226
5.1.5 Version Exchange procedure ..... 3226
5.1.6 ACL Termination procedure ..... 3226
5.1.7 Connection Parameters Request procedure ..... 3227
5.1.7.1 Issuing an LL_CONNECTION_PARAM_REQ PDU ..... 3227

5.1.7.2 Responding to LL_CONNECTION_PARAM_REQ and LL_CONNECTION_PARAM_RSP PDUs ... 3229
5.1.7.3 Examples ... 3231
5.1.7.4 Packet transmit time restrictions ... 3237
5.1.8 LE Ping procedure ... 3238
5.1.9 Data Length Update procedure ... 3238
5.1.10 PHY Update procedure ... 3239
5.1.10.1 Packet transmit restrictions ... 3241
5.1.11 Minimum Number Of Used Channels procedure ... 3242
5.1.12 Constant Tone Extension Request procedure ... 3243
5.1.13 Periodic Advertising Sync Transfer procedure ... 3244
5.1.13.1 Timing considerations ... 3244
5.1.14 Sleep Clock Accuracy Update procedure ... 3247
5.1.15 Connected Isochronous Stream Creation procedure 3248
5.1.16 Connected Isochronous Stream Termination procedure ... 3250
5.1.17 Power Control Request procedure ... 3250
5.1.17.1 Acceptable power reduction ... 3251
5.1.18 Power Change Indication procedure ... 3252
5.1.19 Connection Subrate Update procedure ... 3253
5.1.20 Connection Subrate Request procedure ... 3255
5.1.21 Channel Classification Enable procedure ... 3256
5.1.22 Channel Classification Reporting procedure ... 3256
5.1.23 Channel Sounding Security Start procedure ... 3257
5.1.24 Channel Sounding Capabilities Exchange procedure 3259
5.1.25 Channel Sounding Configuration procedure ... 3259
5.1.26 Channel Sounding Start procedure ... 3262
5.1.27 Channel Sounding Procedure Repeat Termination procedure ... 3268
5.1.28 Channel Sounding Channel Map Update procedure . 3269
5.1.29 Channel Sounding Mode-0 FAE Table Request procedure ... 3270
5.1.30 Frame Space Update procedure ... 3271
5.1.30.1 Adjacent packets in the same connection event ... 3272
5.1.31 UTP OTA mode procedure ... 3273
5.1.32 Connection Rate Update procedure ... 3274
5.1.33 Connection Rate Request procedure ... 3276
5.2 Procedure response timeout ... 3277
5.3 Procedure collisions ... 3278
5.4 LE Authenticated Payload Timeout ... 3279
5.5 Procedures with Instants ... 3279

5.5.1 ACL control procedures ..... 3280
5.5.2 BIG control procedures ..... 3281
5.6 BIG control procedures ..... 3281
5.6.1 BIG Channel Map Update procedure ..... 3281
5.6.2 BIG Termination procedure ..... 3281

**6 Privacy ..... 3283**
6.1 Resolvable Private address generation interval ..... 3283
6.2 Privacy in the Advertising state ..... 3283
6.2.1 Connectable and scannable undirected event type ... 3283
6.2.2 Connectable directed event type ..... 3284
6.2.3 Non-connectable and non-scannable undirected and scannable undirected event types ..... 3285
6.2.4 Connectable undirected event type ..... 3286
6.2.5 Non-connectable and non-scannable directed and scannable directed event types ..... 3286
6.3 Privacy in the Scanning state ..... 3287
6.4 Privacy in the Initiating state ..... 3288
6.5 Privacy of the device ..... 3289
6.6 Privacy in the Synchronization State ..... 3289
6.6.1 Periodic advertising trains ..... 3289

**7 ISO Test Mode ..... 3291**
7.1 ISO Transmit test mode ..... 3291
7.2 ISO Receive test mode ..... 3292

**8 References ..... 3294**

# 1 GENERAL DESCRIPTION

## 1.1 Link Layer states

The operation of the Link Layer can be described in terms of a state machine with the following states:

- Standby State
- Advertising State
- Scanning State
- Initiating State
- Connection State
- Synchronization State
- Isochronous Broadcasting State

The Link Layer state machine allows only one state to be active at a time. The Link Layer shall have at least one Link Layer state machine that supports one of Advertising state or Scanning state. The Link Layer may have multiple instances of the Link Layer state machine.

The Link Layer in the Standby state does not transmit or receive any packets. The Standby state can be entered from any other state.

The Link Layer in the Advertising state will be transmitting advertising physical channel packets and possibly listening to and responding to responses triggered by these advertising physical channel packets. A device in the Advertising state is known as an advertiser. The Advertising state can be entered from the Standby state.

The Link Layer in the Scanning state will be listening for advertising physical channel packets from devices that are advertising. A device in the Scanning state is known as a scanner. The Scanning state can be entered from the Standby state.

The Link Layer in the Initiating state will be listening for advertising physical channel packets from a specific device(s) and responding to these packets to initiate a connection with another device. A device in the Initiating state is known as an initiator. The Initiating state can be entered from the Standby state.

The Connection state can be entered either from the Initiating state or the Advertising state. A device in the Connection state is known as being in a connection.

Within the Connection state, two roles are defined:

- Central Role
- Peripheral Role

When entered from the Initiating state, the Connection state shall be in the Central Role. When entered from the Advertising state, the Connection state shall be in the Peripheral Role.

The Link Layer in the Central Role will communicate with a device in the Peripheral Role and defines the timings of transmissions.

The Link Layer in the Peripheral Role will communicate with a single device in the Central Role.

The Link Layer in the Synchronization State will be listening for periodic physical channel packets forming a specific periodic advertising train, coming from a specified device that is transmitting periodic advertising. The Synchronization State can be entered from the Standby State. While in this State, the Host may direct the Link Layer to listen for isochronous data packets coming from a specified device that is transmitting a Broadcast Isochronous Group (BIG). A device that is in the Synchronization State and is receiving isochronous data packets is referred as a Synchronized Receiver.

The Link Layer in the Isochronous Broadcasting State will transmit isochronous data packets on an isochronous physical channel. The Isochronous Broadcasting State can be entered from the Standby State. A device that is in the Isochronous Broadcasting State is referred as an Isochronous Broadcaster.

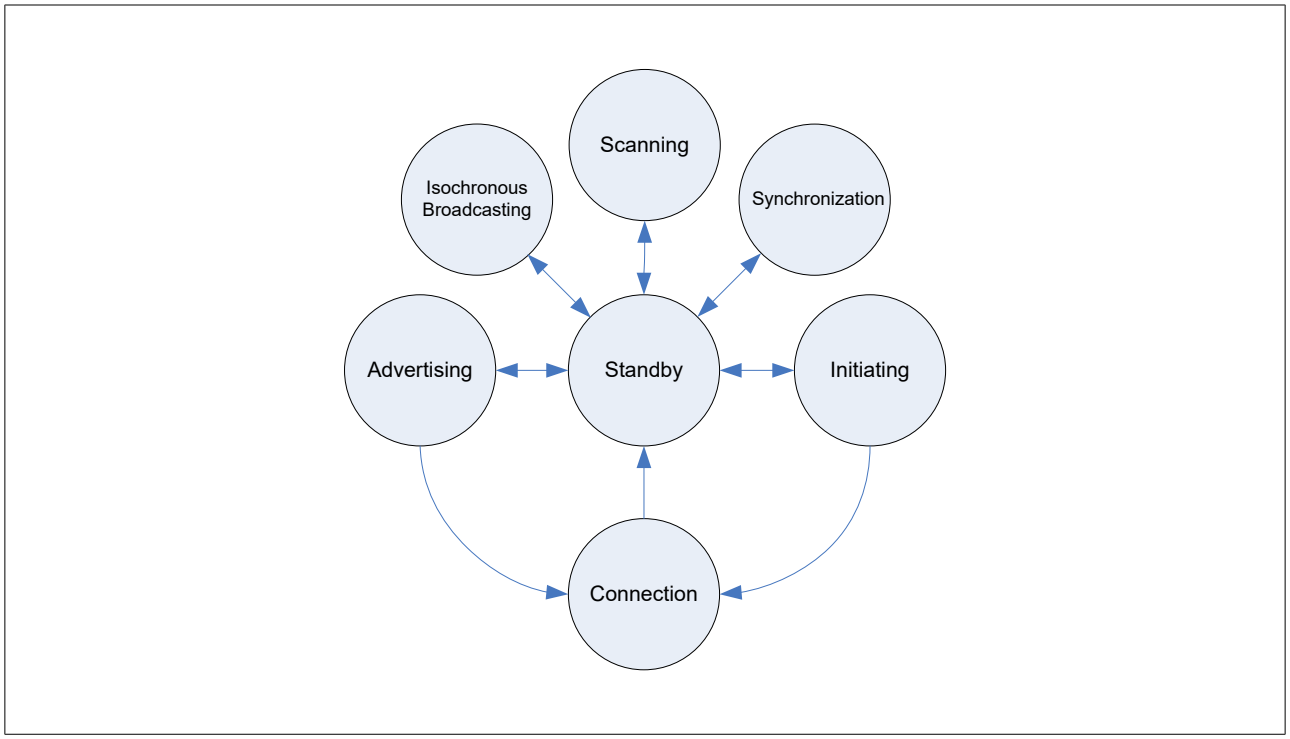



*Figure 1.1: State diagram of the Link Layer state machine*

### 1.1.1 Permitted state and role combinations

The Link Layer may support multiple state machines. If it does support multiple state machines, then any combination of states and roles may be supported. In particular, subject to the requirements in Section 4.5, the Link Layer may be in the Connection State multiple times with any mix of Central Role and Peripheral Role.

Note: A device supporting scanning for periodic advertising (see Section 4.4.3.4) must support at least two state machines.

A Link Layer implementation is not required to support all the possible state combinations that are allowed by the specification. However, if it supports a state or state combination given in the "combination A" column of Table 1.1, it shall also support the corresponding state or state combination in the "combination B" column.

| Combination A | Combination B |
|---|---|
| Initiating plus any combination C of other states | Connection (Central role) plus the same combination C |
| Connection (Central role) plus Initiating plus any combination C of other states | Connection (Central role) to more than one device in the Peripheral role plus the same combination C |
| A connectable Advertising state plus any combination C of other states | Connection (Peripheral role) plus the same combination C |

| Combination A | Combination B |
|---|---|
| Connection (Peripheral role) plus a connectable Advertising state plus any combination C of other states | Connection (Peripheral role) to more than one device in the Central role plus the same combination C |

*Table 1.1: Requirements on supported states and state combinations*

In each case, the combination of other states C may be empty. In the last two rows, "other states" includes other connectable Advertising states.

### 1.1.2 Devices supporting only some states

Devices supporting only some Link Layer states or only one of the two roles within the Connection state are not required to support features (including supporting particular PDUs, procedures, data lengths, or HCI commands or particular features of an HCI command) that are only used by a state or mode that the device does not support.

## 1.2 Bit ordering

The bit ordering when defining fields within the packet or Protocol Data Unit (PDU) in the Link Layer specification follows the little-endian format. The following rules apply:

- The Least Significant Bit (LSB) corresponds to $b_0$
- The LSB is the first bit sent over the air
- In illustrations, the LSB is shown on the left side

Furthermore, data fields defined in the Link Layer, such as the PDU header fields, shall be transmitted with the LSB first. For instance, a 3-bit parameter X=3 is sent as:

$$b_0 b_1 b_2 = 110$$

Over the air, 1 is sent first, 1 is sent next, and 0 is sent last. This is shown as 110 in the specification.

Binary field values specified in the specification that follow the format 0b10101010 (e.g., the advertising physical channel Access Address in Section 2.1.2) are written with the MSB to the left.

Multi-octet fields, with the exception of the Cyclic Redundancy Check (CRC) and the Message Integrity Check (MIC), shall be transmitted with the least significant octet first. Each octet within multi-octet fields, with the exception of the CRC (see Section 3.1.1), shall be transmitted in LSB first order. For example, the 48-bit addresses in the advertising physical channel PDUs shall be transmitted with the least significant octet first, followed by the remainder of the five octets in increasing order.

Multi-octet field values specified in the specification (e.g. the CRC initial value in Section 2.3.3.1) are written with the most significant octet to the left; for example in 0x112233445566, the octet 0x11 is the most significant octet.

Where a packet or PDU is shown in a figure as containing more than one field, the fields shall be transmitted in the order shown in the figure, leftmost first.

## 1.3 Device address

Devices are identified using a device address and an address type; the address type indicates either a public device address or a random device address. A public device address and a random device address are both 48 bits in length.

A device shall use at least one type of device address and may use both. The device may be addressed by any device address that it uses.

A device's Identity Address is a Public Device Address or Random Static Device Address that it uses in packets it transmits. If a device is using Resolvable Private Addresses, it shall also have an Identity Address.

Whenever two device addresses are compared, the comparison shall include the device address type (i.e. if the two addresses have different types, they are different even if the two 48-bit addresses are the same).

### 1.3.1 Public device address

The public device address shall be created in accordance with [Vol 2] Part B, Section 1.2, with the exception that the restriction on LAP values does not apply unless the public device address will also be used as a BD_ADDR for a BR/EDR Controller.

### 1.3.2 Random device address

The random device address may be of either of the following:

- Static address
- Private address.

The specific sub-type is indicated by the two most significant bits of the random device address as shown in Table 1.2.

| Address [47:46] | Sub-Type |
|---|---|
| 0b00 | Non-resolvable private address |
| 0b01 | Resolvable private address |

| Address [47:46] | Sub-Type |
|---|---|
| 0b10 | Reserved for future use |
| 0b11 | Static device address |

*Table 1.2: Sub-types of random device addresses*

The term random device address refers to both static and private address types.

The transmission of a random device address is optional. A device shall accept the reception of a random device address from a remote device.

#### 1.3.2.1 Static device address

A static address is a 48-bit randomly generated address and shall meet the following requirements:

- At least one bit of the random part of the address shall be 0
- At least one bit of the random part of the address shall be 1

The format of a static address is shown in Figure 1.2.

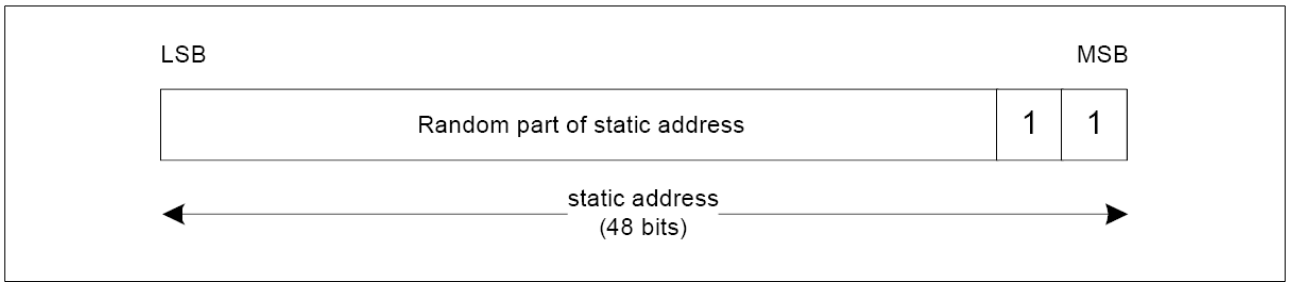



*Figure 1.2: Format of static address*

A device may choose to initialize its static address to a new value after each power cycle. A device shall not change its static address value once initialized until the device is power cycled.

Note: If the static address of a device is changed, then the address stored in peer devices will not be valid and the ability to reconnect using the old address will be lost.

#### 1.3.2.2 Private device address generation

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

To generate a non-resolvable address, the device shall generate a 48-bit address with the following requirements:

- At least one bit of the random part of the address shall be 1
- At least one bit of the random part of the address shall be 0
- The address shall not be equal to the public address

The format of a non-resolvable private address is shown in Figure 1.3.

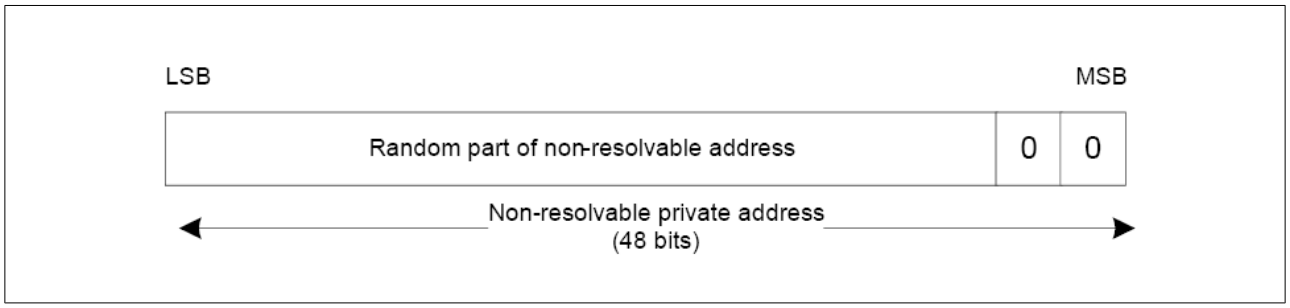



*Figure 1.3: Format of non-resolvable private address*

To generate a resolvable private address, the device must have either the Local Identity Resolving Key (IRK) or the Peer Identity Resolving Key (IRK). The resolvable private address shall be generated with the IRK and a randomly generated 24-bit number. The random number is known as *prand* and shall meet the following requirements:

- At least one bit of the random part of *prand* shall be 0
- At least one bit of the random part of *prand* shall be 1

The format of the resolvable private address is shown in Figure 1.4.

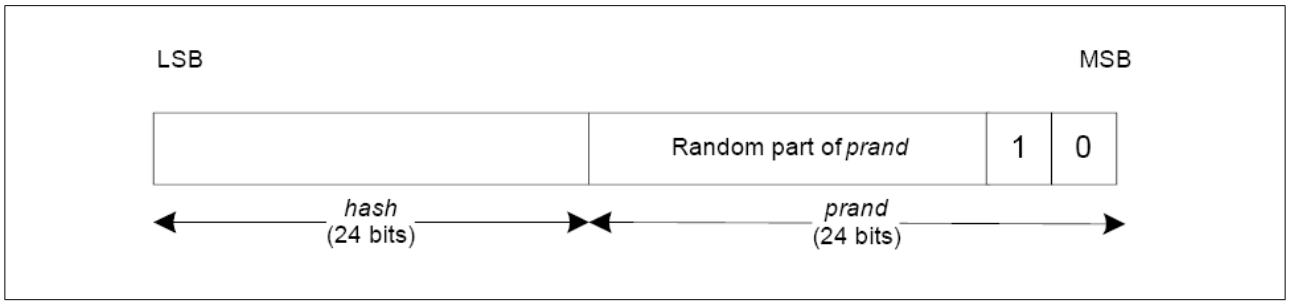



*Figure 1.4: Format of resolvable private address*

The hash is generated using the random address function *ah* defined in [Vol 3] Part H, Section 2.2.2 with the input parameter *k* set to the device's IRK and the input parameter *r* set to *prand*.

*hash* = *ah*(IRK, *prand*)

The *prand* and *hash* are concatenated to generate the random address in the following manner:

*randomAddress* = *prand* || *hash*

The least significant octet of *hash* becomes the least significant octet of *randomAddress* and the most significant octet of *prand* becomes the most significant octet of *randomAddress*.

#### 1.3.2.3 Private device address resolution

A resolvable private address may be resolved if the corresponding device's IRK is available using this procedure. If a resolvable private address is resolved, the device can associate this address with the peer device.

The resolvable private address (*RPA*) is divided into a 24-bit random part (*prand*) and a 24-bit hash part (*hash*). The least significant octet of the *RPA* becomes the least significant octet of *hash* and the most significant octet of *RPA* becomes the most significant octet of *prand*. A *localHash* value is then generated using the random address hash function *ah* defined in [Vol 3] Part H, Section 2.2.2 with the input parameter *k* set to IRK of the known device and the input parameter *r* set to the *prand* value extracted from the *RPA*.

*localHash* = *ah*(IRK, *prand*)

The *localHash* value is then compared with the *hash* value extracted from *RPA*. If the *localHash* value matches the extracted *hash* value, then the identity of the peer device has been resolved.

If a device has more than one stored IRK, the device repeats the above procedure for each stored IRK to determine if the received resolvable private address is associated with a stored IRK, until either address resolution is successful for one of the IRKs or all have been tried.

Note: Because the size of the hash in the RPA is less than the size of the IRK, the same RPA can resolve with more than one IRK (the likelihood of this increases with the number of advertisers within radio range of the observer). Therefore, additional steps might have to be carried out to identify a device uniquely.

A device that cannot resolve a private address within T_IFS_150 may respond on the reception of the next event using that address.

A non-resolvable private address cannot be resolved.

## 1.4 Physical channel

As specified in [Vol 6] Part A, Section 2, 40 RF channels are defined in the 2.4 GHz ISM band. These RF channels are divided into 3 RF channels known as the "primary advertising channels", used for initial advertising and all legacy advertising activities, and 37 RF channels known as the "general-purpose channels", used for the majority of communications. Each of these RF channels is allocated a unique channel index (see Section 1.4.1).

These two groups of RF channels are used in four LE physical channels: advertising, periodic, isochronous, and data. The advertising physical channel uses both groups for discovering devices, initiating a connection, and broadcasting data; within this, the primary advertising channels form the primary advertising physical channel and the general-purpose channels form the secondary advertising physical channel. The remaining physical channels only use the general-purpose channels.

Two devices that communicate use a shared physical channel. To achieve this, their transceivers must be tuned to the same RF channel at the same time.

Given that the number of RF channels is limited, and that many Bluetooth devices may be operating independently within the same spatial and temporal area, there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF channel, resulting in a physical channel collision. To mitigate the unwanted effects of this collision, each transmission on a physical channel starts with an Access Address that is used as a correlation code by devices tuned to the physical channel. This Access Address is a property of the physical channel. The Access Address is present at the start of every transmitted packet.

The Link Layer uses one physical channel at a given time.

Whenever the Link Layer is synchronized to the timing, frequency, and Access Address of a physical channel, it is said to be 'connected' on the data physical channel or ‘synchronized’ to the periodic physical channel or isochronous physical channel (whether or not it is actively involved in communications over the channel).

### 1.4.1 Physical channel indices

Table 1.3 shows the mapping from RF Channel to Physical Channel Index and RF Channel group. A ‘●’ in Table 1.3 indicates the RF channel and index are part of the specified channel group; a blank cell indicates that they are not part of that group.

| RF Channel (Center Frequency) | Physical Channel Index | RF Channel Group | |
|---|---|---|---|
| | | Primary Advertising | General purpose |
| 2402 MHz | 37 | ● | |
| 2404 MHz | 0 | | ● |
| 2406 MHz | 1 | | ● |
| ... | ... | ... | ... |
| 2424 MHz | 10 | | ● |
| 2426 MHz | 38 | ● | |
| 2428 MHz | 11 | | ● |
| 2430 MHz | 12 | | ● |
| ... | ... | ... | ... |
| 2478 MHz | 36 | | ● |
| 2480 MHz | 39 | ● | |

*Table 1.3: Mapping of RF channel to physical channel index and RF channel group*

# 2 AIR INTERFACE PACKETS

LE devices shall use the packets as defined in the following sections. There are two basic formats: one for the LE Uncoded PHYs, described in Section 2.1, and one for the LE Coded PHY, described in Section 2.2.

Additional packet formats specific to Channel Sounding are defined in [Vol 6] Part H, Section 2.

## 2.1 Packet format for the LE Uncoded PHYs

The following packet format is defined for the LE Uncoded PHYs (LE 1M and LE 2M) and is used for packets on all physical channels.

This packet format is shown in Figure 2.1. Each packet consists of four mandatory fields and one optional field. The mandatory fields are Preamble, Access Address, PDU, and CRC. The optional field is Constant Tone Extension.

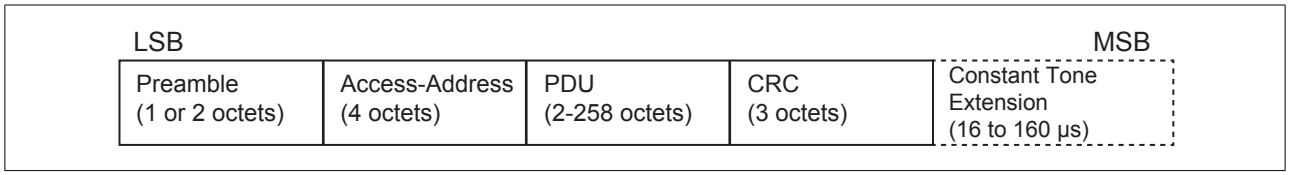



*Figure 2.1: Link Layer packet format for the LE Uncoded PHYs*

The preamble is 1 octet when transmitting or receiving on the LE 1M PHY and 2 octets when transmitting or receiving on the LE 2M PHY. The Access Address is 4 octets. The PDU range is from 2 to 258 octets. The CRC is 3 octets.

The Preamble is transmitted first, followed by the Access Address, PDU, CRC, and Constant Tone Extension (if present) in that order. The entire packet is transmitted at the same symbol rate (either 1 Msym/s or 2 Msym/s modulation).

Packets (not including the Constant Tone Extension) take between 44 and 2128 µs to transmit.

When the Constant Tone Extension is present, the Constant Tone Extension duration is between 16 and 160 µs, as shown in Figure 2.1.

### 2.1.1 Preamble

All Link Layer packets have a preamble which is used in the receiver to perform frequency synchronization, symbol timing estimation, and Automatic Gain Control training. The preamble is a fixed sequence of alternating 0 and 1 bits. For packets

transmitted on the LE 1M PHY, the preamble is 8 bits; for packets transmitted on the LE 2M PHY, the preamble is 16 bits. The first bit of the preamble (in transmission order) shall be the same as the LSB of the Access Address. The preamble is shown in Figure 2.2.

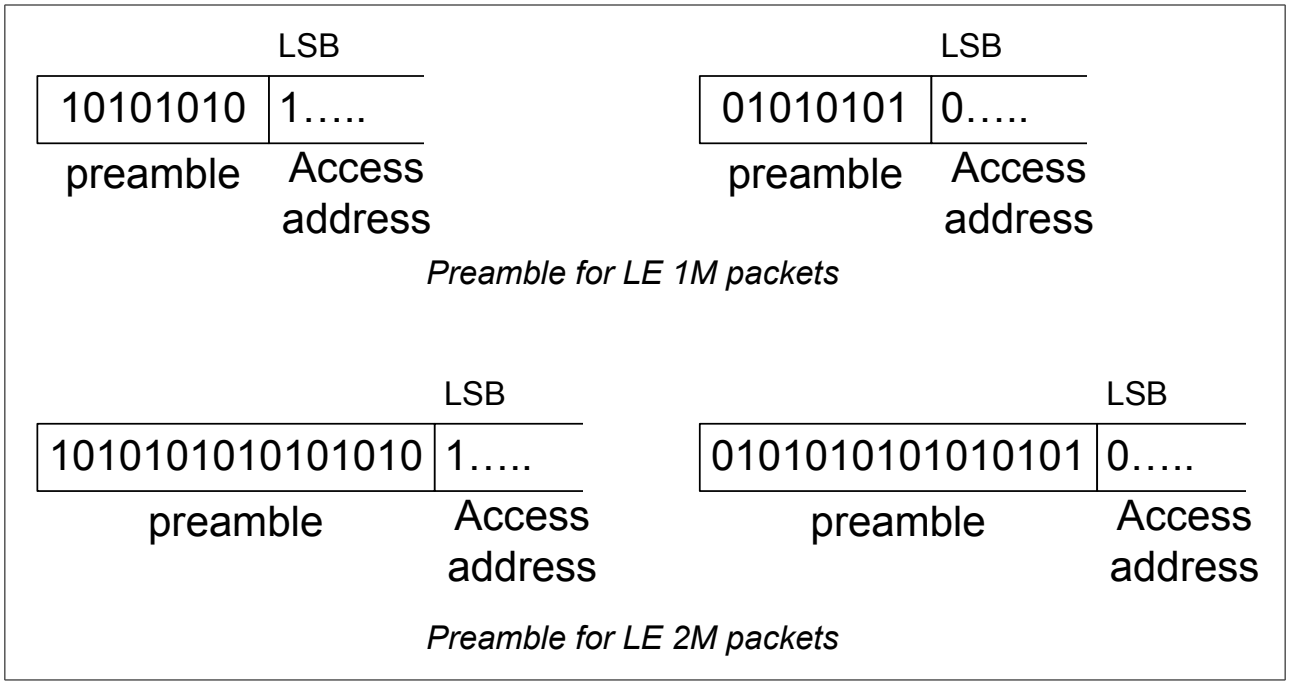



*Figure 2.2: Preamble*

### 2.1.2 Access Address

All AUX_SYNC_IND, AUX_CHAIN_IND, AUX_SYNC_SUBEVENT_IND, AUX_CONNECT_REQ, and AUX_CONNECT_RSP PDUs sent on a periodic advertising train shall use the Access Address (AA) value set in the SyncInfo field (see Section 2.3.4.6) contained in the AUX_ADV_IND PDU that describes the train. All AUX_SYNC_SUBEVENT_RSP PDUs sent on a periodic advertising with responses train shall use the Access Address value set in the Periodic Advertising Response Timing Information (see Section 1.24 of [1]) contained in the AUX_ADV_IND PDU that describes the train. For a given periodic advertising with responses train, the AA value of the AUX_SYNC_SUBEVENT_RSP PDUs shall be different from the AA value of the AUX_SYNC_SUBEVENT_IND PDUs.

The Access Address for all other advertising physical channel packets shall be 0b10001110_10001001_10111110_11010110 (0x8E89BED6).

It is intended that each Link Layer connection between any two devices, each BIS, and each periodic advertising train has a different Access Address.

The Link Layer in the Initiating state shall generate a new Access Address for each initiating PDU it sends (see Section 2.3.3.1). The Link Layer in the Advertising state shall generate a new Access Address each time that it enables a periodic advertising train and, for periodic advertising with responses, a second Access Address for the responses. The first address is sent in the SyncInfo field (see Section 2.3.4.6) of PDUs referring to that periodic advertising train and the second in the Periodic Advertising Response Timing Information (see Section 1.24 of [1]) for the train.

The Link Layer in the Central Role in the Connected State shall generate a new Access Address for each Connected Isochronous Stream (CIS) it creates. The address is sent in the Link Layer Control PDU that is used to create the CIS (see Section 2.4.2.31).

The Access Address shall be a 32-bit value. Each time it needs a new Access Address (except for a Broadcast Isochronous Stream (BIS)), the Link Layer shall generate a new random value.

Each Access Address shall meet the following requirements:

- It shall not be the Access Address for any existing ACL connection or CIS on this device.
- It shall not be the Access Address for any enabled periodic advertising train.
- It shall not be the Access Address for any existing BIS on this device.
- It shall not be the Access Address for any existing BIG Control logical link on this device.
- If it is the Access Address for a new CIS, it shall differ by more than one bit from any other Access Address being used on the same device.
- It shall have no more than six consecutive zeros or ones.
- It shall not be the advertising physical channel packets' Access Address.
- It shall not be a sequence that differs from the advertising physical channel packets' Access Address by only one bit.
- It shall not have all four octets equal.
- It shall have no more than 24 transitions.
- It shall have a minimum of two transitions in the most significant six bits.

The Link Layer in the Isochronous Broadcasting State shall generate a new Seed Access Address (SAA) for each BIG. The Access Addresses for the constituent BIS(es) are derived from the SAA. The SAA shall be a random number that meets the following requirements:

$$SAA_{19} = SAA_{15}$$
$$SAA_{22} = SAA_{16} \neq SAA_{15}$$

$SAA_{25}$ = 0
$SAA_{23}$ = 1

For any pair of BIGs transmitted by the same device, the $SAA_{15-0}$ values shall differ in at least two bits.

The Access Address for each BIS and for the BIG Control logical link (see Section 4.4.6.7) in a BIG shall be derived from the SAA for that BIG.

For each BIS logical transport, the Access Address shall be equal to the SAA bit-wise XORed with a diversifier word (DW) for that logical transport derived from a Diversifier (D) as follows:

D = ((35 × n) + 42) *mod* 128 where n is the BIS_Number, or 0 for the BIG Control logical link
DW = 0b$D_0D_0D_0D_0D_0D_0D_1D_6$_$D_10D_5D_40D_3D_20$_00000000_00000000

For example, if n=1, D=77=0b01001101 and DW = 0xFD060000.

The seed for the random number generator shall be from a physical source of entropy and should have at least 20 bits of entropy.

If the random number and, in the case of an SAA, the derived Access Addresses for the BIS and the BIG Control logical link do not meet the above requirements, new random numbers shall be generated until the requirements are met.

On an implementation that also supports the LE Coded PHY (see Section 2.2), the Access Address shall also meet the following requirements:

- It shall have at least three ones in the least significant 8 bits.
- It shall have no more than eleven transitions in the least significant 16 bits.

### 2.1.3 PDU

The preamble and Access Address are followed by a PDU. When a packet is transmitted on either the primary or secondary advertising physical channel or the periodic physical channel, the PDU shall be the Advertising Physical Channel PDU as defined in Section 2.3. When a packet is transmitted on the data physical channel, the PDU shall be the Data Physical Channel PDU as defined in Section 2.4. When a packet is transmitted on the isochronous physical channel, the PDU shall be one of the Isochronous Physical Channel PDUs as defined in Section 2.6.

### 2.1.4 CRC

The PDU is followed by a 24-bit CRC. It shall be calculated over the PDU. The CRC polynomial is defined in Section 3.1.1.

### 2.1.5 Constant Tone Extension

The CRC is followed by an optional Constant Tone Extension, which consists of a constantly modulated series of unwhitened 1s. The Constant Tone Extension is not included in CRC or MIC calculations. The Constant Tone Extension field shall not be present in a packet sent on the isochronous physical channel. The Constant Tone Extension is discussed further in Section 2.5.

## 2.2 Packet format for the LE Coded PHY

The following packet format is defined for the LE Coded PHY and is used for packets on all physical channels. This packet format is shown in Figure 2.3.

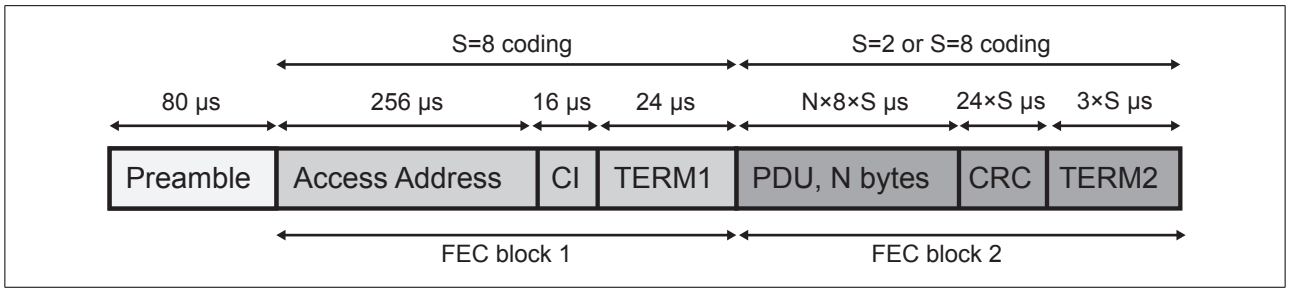



*Figure 2.3: Link Layer packet format for the LE Coded PHY*

Each packet consists of the Preamble, FEC block 1, and FEC block 2.

The Preamble is not coded.

The FEC block 1 consists of three fields: Access Address, Coding Indicator (CI), and TERM1. These shall use the S=8 coding scheme as defined in Section 3.3.1.

The CI determines which coding scheme is used for FEC block 2.

The FEC block 2 consists of three fields: PDU, CRC, and TERM2. These shall use either the S=2 or S=8 coding scheme as defined in Section 3.3, depending on the value of the CI.

The entire packet is transmitted with 1 Msym/s modulation.

Table 2.1 captures the size and duration of the data packet fields.

| | Fields | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Preamble** | **Access Address** | **CI** | **TERM1** | **PDU** | **CRC** | **TERM2** |
| Number of Bits | Uncoded | 32 | 2 | 3 | 16 – 2056 | 24 | 3 |
| Duration when using S=8 coding (µs) | 80 | 256 | 16 | 24 | 128 – 16448 | 192 | 24 |
| Duration when using S=2 coding (µs) | 80 | 256 | 16 | 24 | 32 – 4112 | 48 | 6 |

*Table 2.1: LE Coded PHY field sizes and durations in microseconds*

Packets take between 462 and 17040 µs to transmit.

Note: There is no Constant Tone Extension on the LE Coded PHY.

### 2.2.1 Preamble

The Preamble is 80 symbols in length and consists of 10 repetitions of the symbol pattern '00111100' (in transmission order).

### 2.2.2 Access Address

The Access Address is specified in Section 2.1.2.

### 2.2.3 Coding Indicator

The Coding Indicator (CI) consists of two bits as defined in Table 2.2.

| **CI** | **Meaning** |
|---|---|
| 0b00 | FEC Block 2 coded using S=8 |
| 0b01 | FEC Block 2 coded using S=2 |
| All other values | Reserved for future use |

*Table 2.2: Meaning of CI*

### 2.2.4 PDU

When a packet is transmitted on either the primary or secondary advertising physical channel or the periodic physical channel, the PDU shall be the Advertising Physical Channel PDU as defined in Section 2.3. When a packet is transmitted on the data physical channel, the PDU shall be the Data Physical Channel PDU as defined in Section 2.4. When a packet is transmitted on the isochronous physical channel, the PDU shall be one of the Isochronous Physical Channel PDUs as defined in Section 2.6.

### 2.2.5 CRC

The CRC is 24 bits in length and the value is calculated over all the PDU bits. The CRC generator polynomial is defined in Section 3.1.1.

### 2.2.6 TERM1 and TERM2

There is a terminator at the end of each FEC block referred to as TERM1 and TERM2. Each terminator is 3 bits long and forms the termination sequence defined in Section 3.3.1.

## 2.3 Advertising physical channel PDU

Note: Despite the name, the advertising physical channel PDU is also used on the periodic physical channel.

The advertising physical channel PDU has a 16-bit Header field and a variable size Payload field. Its format is as shown in Figure 2.4. The 16-bit Header field of the advertising physical channel PDU is as shown in Figure 2.5.

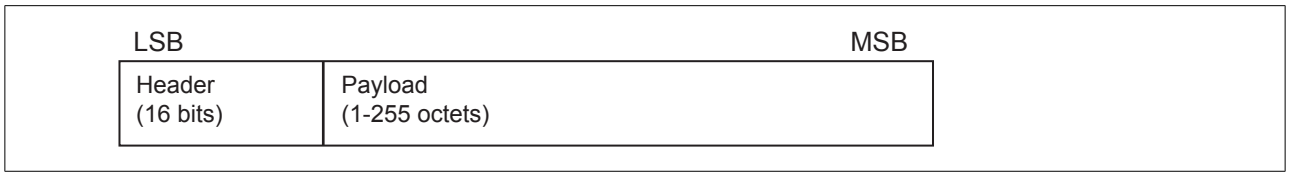



*Figure 2.4: Advertising physical channel PDU*

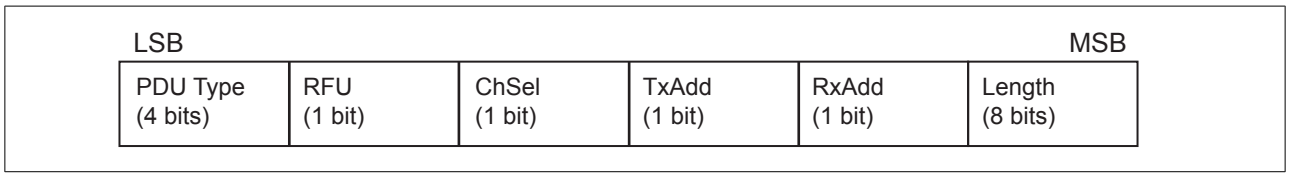



*Figure 2.5: Advertising physical channel PDU header*

The PDU Type field of the advertising physical channel PDU that is contained in the Header field indicates the PDU type as defined in Table 2.3, which also shows which channel and which PHYs the packet may appear on. A '●' in Table 2.3 indicates that the PDU may appear on that PHY; a blank cell indicates that the PDU shall not appear on that PHY.

| PDU Type | PDU Name | Physical Channel | Permitted PHYs | | |
|---|---|---|---|---|---|
| | | | LE 1M | LE 2M | LE Coded |
| 0b0000 | ADV_IND | Primary Advertising | • | | |
| 0b0001 | ADV_DIRECT_IND | Primary Advertising | • | | |
| 0b0010 | ADV_NONCONN_IND | Primary Advertising | • | | |
| 0b0011 | SCAN_REQ | Primary Advertising | • | | |
| | AUX_SCAN_REQ | Secondary Advertising | • | • | • |
| 0b0100 | SCAN_RSP | Primary Advertising | • | | |
| 0b0101 | CONNECT_IND | Primary Advertising | • | | |
| | AUX_CONNECT_REQ | Secondary Advertising | • | • | • |
| 0b0110 | ADV_SCAN_IND | Primary Advertising | • | | |
| 0b0111 | ADV_EXT_IND | Primary Advertising | • | | • |
| | AUX_ADV_IND | Secondary Advertising | • | • | • |
| | AUX_SCAN_RSP | Secondary Advertising | • | • | • |
| | AUX_SYNC_IND | Periodic | • | • | • |
| | AUX_CHAIN_IND | Secondary Advertising and Periodic | • | • | • |
| | AUX_SYNC_SUBEVENT_-IND | Periodic | • | • | • |
| | AUX_SYNC_SUBEVENT_-RSP | Periodic | • | • | • |
| 0b1000 | AUX_CONNECT_RSP | Secondary Advertising | • | • | • |
| 0b1001 | ADV_DECISION_IND | Primary Advertising | • | | • |
| All other values | Reserved for future use | | | | |

*Table 2.3: Advertising physical channel PDU header's PDU Type field encoding*

The ChSel, TxAdd and RxAdd fields of the advertising physical channel PDU that are contained in the Header field contain information specific to the PDU type defined for each advertising physical channel PDU separately. If the ChSel, TxAdd or RxAdd fields are not defined as used in a given PDU then they shall be considered reserved for future use.

The Length field of the advertising physical channel PDU Header field indicates the length of the Payload field in octets. The valid range of the Length field shall be 1 to 255 octets.

The Payload fields in the advertising physical channel PDUs are specific to the PDU Type and are defined in Section 2.3.1 to Section 2.3.4. The PDU Types marked as Reserved for future use shall not be sent and shall be ignored upon receipt.

Within advertising physical channel PDUs, advertising data or scan response data from the Host may be included in the Payload field in some PDU Types. The format of this data is defined in [Vol 3] Part C, Section 11.

Some advertising physical channel PDUs contain an AuxPtr field (see Section 2.3.4.5) which points to a packet containing another advertising physical channel PDU. In this case, the second packet and PDU are the *auxiliary packet* and *auxiliary PDU* of the original PDU, which in turn is the *superior packet* and *superior PDU* of the second one.

Note: A PDU can only have one auxiliary PDU but more than one superior PDU.

Given a packet, its *subordinate set* consists of its auxiliary packet, if any, and the subordinate set of the auxiliary packet. A packet without an AuxPtr field has an empty subordinate set.

### 2.3.1 Advertising PDUs

The following advertising physical channel PDU Types are called advertising PDUs:

- ADV_IND
- ADV_DIRECT_IND
- ADV_NONCONN_IND
- ADV_SCAN_IND
- ADV_EXT_IND
- ADV_DECISION_IND
- AUX_ADV_IND
- AUX_SYNC_IND
- AUX_CHAIN_IND
- AUX_SYNC_SUBEVENT_IND
- AUX_SYNC_SUBEVENT_RSP

These PDUs are sent by the Link Layer in the Advertising state and received by a Link Layer in the Scanning state or Initiating state. The ADV_IND, ADV_DIRECT_IND, ADV_NONCONN_IND, and ADV_SCAN_IND PDUs are called “legacy advertising PDUs”. The ADV_EXT_IND, ADV_DECISION_IND, AUX_ADV_IND, AUX_SYNC_IND, AUX_CHAIN_IND, AUX_SYNC_SUBEVENT_IND, and AUX_SYNC_SUBEVENT_RSP PDUs are called “extended advertising PDUs”. The ADV_DECISION_IND PDU is also

called a “decision PDU”. Advertising events using legacy advertising PDUs are called “legacy advertising events”.

#### 2.3.1.1 ADV_IND

The Payload field of the ADV_IND PDU is shown in Figure 2.6. The PDU shall be used in connectable and scannable undirected advertising events. The TxAdd in the advertising physical channel PDU Header field indicates whether the advertiser’s address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The ChSel field in the advertising physical channel PDU Header field shall be set to 1 if the advertiser supports the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3).

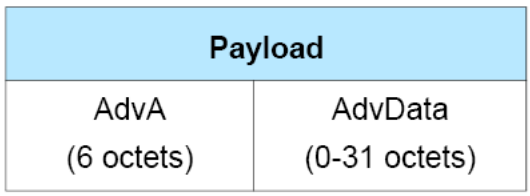



*Figure 2.6: ADV_IND PDU payload*

The AdvA field shall contain the advertiser’s public or random device address as indicated by TxAdd. The AdvData field, if not empty, shall contain Advertising Data from the advertiser’s Host.

#### 2.3.1.2 ADV_DIRECT_IND

The Payload field of the ADV_DIRECT_IND PDU is shown in Figure 2.7. The PDU shall only be used in connectable directed advertising events. The TxAdd in the advertising physical channel PDU Header field indicates whether the advertiser’s address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising physical channel PDU Header field indicates whether the target’s address in the TargetA field is public (RxAdd = 0) or random (RxAdd = 1). The ChSel field in the advertising physical channel PDU Header field shall be set to 1 if the advertiser supports the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3).

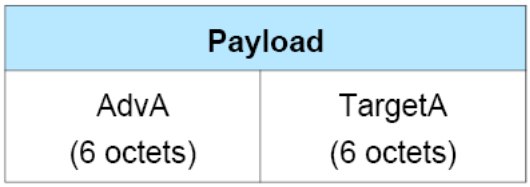



*Figure 2.7: ADV_DIRECT_IND PDU Payload*

The AdvA field shall contain the advertiser’s public or random device address as indicated by TxAdd. The TargetA field is the address of the device to which this PDU is addressed. The TargetA field shall contain the target’s public or random device address as indicated by RxAdd.

Note: This packet does not contain any Host data.

#### 2.3.1.3 ADV_NONCONN_IND

The Payload field of the ADV_NONCONN_IND PDU is shown in Figure 2.8. The PDU shall only be used in non-connectable and non-scannable undirected advertising events. The TxAdd in the advertising physical channel PDU Header field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1).

| Payload | |
|---|---|
| AdvA<br>(6 octets) | AdvData<br>(0-31 octets) |

*Figure 2.8: ADV_NONCONN_IND PDU Payload*

The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The AdvData field, if not empty, shall contain Advertising Data from the advertiser's Host.

#### 2.3.1.4 ADV_SCAN_IND

The Payload field of the ADV_SCAN_IND PDU is shown in Figure 2.9. The PDU shall only be used in scannable undirected advertising events. The TxAdd in the advertising physical channel PDU Header field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1).

| Payload | |
|---|---|
| AdvA<br>(6 octets) | AdvData<br>(0-31 octets) |

*Figure 2.9: ADV_SCAN_IND PDU Payload*

The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The AdvData field, if not empty, shall contain Advertising Data from the advertiser's Host.

#### 2.3.1.5 ADV_EXT_IND

The ADV_EXT_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU may be used in the advertising events indicated by the AdvMode field value. An advertising event using an ADV_EXT_IND PDU is directed

if, and only if, either the TargetA field is present or the AuxPtr field is present and points to a PDU where the TargetA field is present.

The Common Extended Advertising Payload Format fields permitted in the ADV_EXT_IND PDU are shown in Table 2.4.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Event Type** | **Adv Mode** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| Non-Connectable and Non-Scannable Undirected without auxiliary packet | 0b00 | M | X | X | X | X | X | O | X | X |
| Non-Connectable and Non-Scannable Undirected with auxiliary packet | 0b00 | C.1 | X | X | M | M | X | C.1 | X | X |
| Non-Connectable and Non-Scannable Directed without auxiliary packet | 0b00 | M | M | X | X | X | X | O | X | X |
| Non-Connectable and Non-Scannable Directed with auxiliary packet | 0b00 | C.1 | C.1 | X | M | M | X | C.1 | X | X |
| Connectable Undirected | 0b01 | X | X | X | M | M | X | C.1 | X | X |
| Connectable Directed | 0b01 | X | X | X | M | M | X | C.1 | X | X |
| Scannable Undirected | 0b10 | X | X | X | M | M | X | C.1 | X | X |
| Scannable Directed | 0b10 | X | X | X | M | M | X | C.1 | X | X |
| RFU | 0b11 | | | | | | | | | |
| C.1: This field is optional on the LE 1M PHY and reserved for future use on the LE Coded PHY. | | | | | | | | | | |

*Table 2.4: Common Extended Advertising Payload Format fields permitted in the ADV_EXT_IND PDU*

For the non-connectable and non-scannable directed and non-connectable and non-scannable undirected event types without ACAD or AdvData, the Controller can choose whether or not to use an auxiliary packet. See Section 4.4.2.6 and Section 4.4.2.9.

Fields reserved for future use shall not be present when the packet is sent and shall be ignored when received.

Any auxiliary packet shall be an AUX_ADV_IND packet.

#### 2.3.1.6 AUX_ADV_IND

The AUX_ADV_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU may be used in the advertising events indicated by the AdvMode field value.

The AdvMode field indicates the type of advertising event the AUX_ADV_IND packet is being used for and shall have the same value as the field in the superior PDU of this PDU.

The Common Extended Advertising Payload Format fields permitted in the AUX_ADV_IND PDU are shown in Table 2.5.

The PHY used for the AUX_ADV_IND shall be specified in the AuxPtr field of the superior PDU. The PHY specified in any AuxPtr field in an AUX_ADV_IND PDU shall be the same as the PHY the PDU was sent on.

If this PDU and its superior PDU both have an ADI field, the values in these fields shall be the same.

Note: The ADI field can be used to detect collisions.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

The SyncInfo field, when present, shall point to an AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Non-Connectable and Non-Scannable Undirected | C.1 | X | X | M | O | O | O | O | O |
| 0b00 | Non-Connectable and Non-Scannable Directed | C.1 | C.2 | X | M | O | O | O | O | O |
| 0b01 | Connectable Undirected | C.2 | X | X | M | X | X | O | O | O |
| 0b01 | Connectable Directed | M | M | X | M | X | X | O | O | O |
| 0b10 | Scannable Undirected | C.2 | X | X | M | X | X | O | O | X |
| 0b10 | Scannable Directed | M | M | X | M | X | X | O | O | X |
| 0b11 | RFU | | | | | | | | | |

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| C.1: This field is optional if the corresponding field in the superior PDU of this PDU is not present, otherwise it is reserved for future use. | | | | | | | | | | |
| C.2: This field is mandatory if the corresponding field in the superior PDU of this PDU is not present, otherwise it is reserved for future use. | | | | | | | | | | |

*Table 2.5: Common Extended Advertising Payload Format fields permitted in the AUX_ADV_IND PDU*

#### 2.3.1.7 AUX_SYNC_IND

The AUX_SYNC_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU is used in periodic advertising.

The AdvMode field shall be set to 0b00.

The Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_IND PDU are shown in Table 2.6.

The PHY used for the AUX_SYNC_IND PDU shall be that specified in Section 4.4.2.12.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

The Advertising SID subfield of the ADI field shall have the same value as that in the superior PDU of this PDU.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Non-Connectable and Non-Scannable Undirected | X | X | O | O | O | X | O | O | O |
| 0b01 to 0b11 | RFU | | | | | | | | | |

*Table 2.6: Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_IND PDU*

#### 2.3.1.8 AUX_CHAIN_IND

The AUX_CHAIN_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. The PDU is used to hold additional AdvData. Its superior PDU is an AUX_ADV_IND, AUX_SYNC_IND, AUX_SCAN_RSP or another AUX_CHAIN_IND PDU.

The AdvMode field shall be set to 0b00.

The Common Extended Advertising Payload Format fields permitted in the AUX_CHAIN_IND PDU are shown in Table 2.7.

The PHY used for the AUX_CHAIN_IND PDU shall be the same as the PHY used for its superior PDU.

The ADI field, when present, shall have the same value as the field in the superior PDU.

Note: The ADI field can be used to detect collisions.

Any auxiliary PDU shall be another AUX_CHAIN_IND PDU.

| | | **Common Extended Advertising Payload Format fields** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Chained data | X | X | C.2 | C.1 | O | X | O | X | O |
| 0b01 to 0b11 | RFU | | | | | | | | | |

C.1: This field is mandatory if the corresponding field in the superior PDU of this PDU is present, otherwise it is reserved for future use.

C.2: This field is optional if the corresponding field in the superior PDU of this PDU is present, otherwise it is reserved for future use.

*Table 2.7: Common Extended Advertising Payload Format fields permitted in the AUX_CHAIN_IND PDU*

#### 2.3.1.9 AUX_SYNC_SUBEVENT_IND

The AUX_SYNC_SUBEVENT_IND PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. This PDU is used in PAwR.

The AdvMode field shall be set to 0x00.

The Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_SUBEVENT_IND PDU are shown in Table 2.8.

The PHY used for the AUX_SYNC_SUBEVENT_IND shall be that specified in Section 4.4.2.12.

The Advertising SID subfield of the ADI field shall have the same value as that in the superior PDU of this PDU.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Subevent Indication | X | X | O | O | X | X | O | O | O |
| 0b01 to 0b11 | RFU | | | | | | | | | |

*Table 2.8: Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_SUBEVENT_IND PDU*

#### 2.3.1.10 AUX_SYNC_SUBEVENT_RSP

The AUX_SYNC_SUBEVENT_RSP PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4. This PDU is used in PAwR.

The AdvMode field shall be set to 0x00.

The Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_SUBEVENT_RSP PDU are shown in Table 2.9.

The PHY used for the AUX_SYNC_SUBEVENT_RSP shall be that specified in Section 4.4.2.12.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Subevent Response | O | X | O | X | X | X | O | O | O |
| 0b01 to 0b11 | RFU | | | | | | | | | |

*Table 2.9: Common Extended Advertising Payload Format fields permitted in the AUX_SYNC_SUBEVENT_RSP PDU*

#### 2.3.1.11 ADV_DECISION_IND

The Payload field of the ADV_DECISION_IND PDU is shown in Figure 2.10. The PDU may be used in the undirected advertising events indicated by the AdvMode field value.

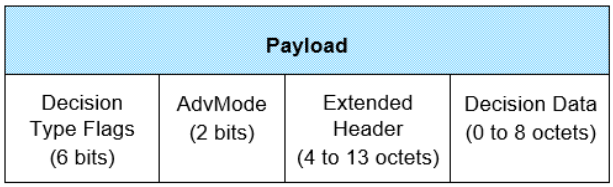



*Figure 2.10: ADV_DECISION_IND payload*

The Extended Header field is the same as the Extended Header field of the Common Extended Advertising Payload Format (see Section 2.3.4); the fields permitted in the ADV_DECISION_IND PDU are shown in Table 2.10.

| | | Common Extended Advertising Payload Format Extended Header fields | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTEInfo** | **ADI** | **AuxPtr** | **SyncInfo** | **TxPower** | **ACAD** |
| 0b00 | Non-Connectable and Non-Scannable Undirected | O | X | X | O | M | X | C.1 | X |
| 0b01 | Connectable Undirected | O | X | X | O | M | X | C.1 | X |
| 0b10 | Scannable Undirected | O | X | X | O | M | X | C.1 | X |
| 0b11 | RFU | | | | | | | | |
| C.1: This field is optional on the LE 1M PHY and reserved for future use on the LE Coded PHY. | | | | | | | | | |

*Table 2.10: Common Extended Advertising Payload Format fields permitted in the ADV_DECISION_IND PDU*

The format of the Decision Data field is shown in Figure 2.11:

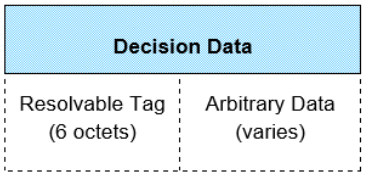



*Figure 2.11: Decision Data format*

The Decision Type Flags bit field definitions are listed in Table 2.11.

| Bit Number | Subfield |
|---|---|
| 0 | Resolvable Tag |
| All other bits | Reserved for future use |

*Table 2.11: Decision Type Flags field interpretation*

If a bit in the Decision Type Flags field is set to 1, then the corresponding subfield of Decision Data shall be present; otherwise, the corresponding subfield shall not be present. Therefore, the Decision Data field must be long enough to hold all the subfields indicated in the Decision Type Flags field. Any octets of the Decision Data following these subfields form the Arbitrary Data subfield.

A Resolvable Tag shall have the format shown in Figure 2.12.

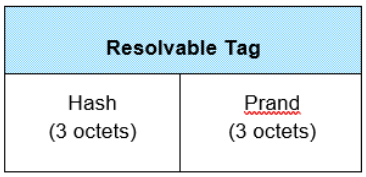



*Figure 2.12: Resolvable Tag format*

A Resolvable Tag is said to resolve against a key provided by the Host if the Hash field equals the value *ah*(*key, prand*), where *prand* is the value of the Prand field and *ah* is the function defined in [Vol 3] Part H, Section 2.2.2.

The auxiliary PDU shall be an AUX_ADV_IND PDU.

### 2.3.2 Scanning PDUs

The following advertising physical channel PDU types are called scanning PDUs.

- SCAN_REQ
- SCAN_RSP
- AUX_SCAN_REQ
- AUX_SCAN_RSP

The SCAN_REQ and AUX_SCAN_REQ PDUs are called scan request PDUs. The SCAN_RSP and AUX_SCAN_RSP PDUs are called scan response PDUs.

Where these PDUs are used to reply to a scannable advertisement, the PHY used for them shall be the same as the PHY used for the PDU that they reply to.

#### 2.3.2.1 SCAN_REQ and AUX_SCAN_REQ

The Payload field of the SCAN_REQ and AUX_SCAN_REQ PDUs is shown in Figure 2.13. The TxAdd in the advertising physical channel PDU Header field indicates whether the scanner's address in the ScanA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising physical channel PDU Header field indicates whether the advertiser's address in the AdvA field is public (RxAdd = 0) or random (RxAdd = 1).

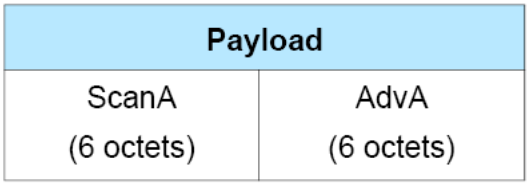



*Figure 2.13: SCAN_REQ and AUX_SCAN_REQ PDU Payload*

The ScanA field shall contain the scanner's public or random device address as indicated by TxAdd. The AdvA field is the address of the device to which this PDU is addressed. The AdvA field shall contain the advertiser's public or random device address as indicated by RxAdd.

Note: These PDUs do not contain any Host Data.

#### 2.3.2.2 SCAN_RSP

The Payload field of the SCAN_RSP PDU is shown in Figure 2.14. The TxAdd in the advertising physical channel PDU Header field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The Length field indicates the size of the Payload field (AdvA and ScanRspData) in octets.

| Payload | |
|---|---|
| AdvA<br>(6 octets) | ScanRspData<br>(0-31 octets) |

*Figure 2.14: SCAN_RSP PDU Payload*

The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The ScanRspData field may contain any data from the advertiser's Host.

#### 2.3.2.3 AUX_SCAN_RSP

The AUX_SCAN_RSP PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4.

The AdvMode field shall be set to 0b00.

The Common Extended Advertising Payload Format fields permitted in the AUX_SCAN_RSP PDU are shown in Table 2.12.

The ADI field, if present, shall have the same value as the field in the AUX_ADV_IND PDU that the scanner responded to.

Note: The ADI field can be used to detect collisions.

Any auxiliary PDU shall be an AUX_CHAIN_IND PDU.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Scan response | M | X | X | O | O | X | O | O | M |
| 0b01 to 0b11 | RFU | | | | | | | | | |

*Table 2.12: Common Extended Advertising Payload Format fields permitted in the AUX_SCAN_RSP PDU*

### 2.3.3 Initiating PDUs

The following advertising physical channel PDU Types are called initiating PDUs:

- CONNECT_IND
- AUX_CONNECT_REQ
- AUX_CONNECT_RSP

The CONNECT_IND and the AUX_CONNECT_REQ PDUs are sent by the Link Layer in the Initiating state and received by the Link Layer in the Advertising state. The AUX_CONNECT_RSP PDU is sent by the Link Layer in the Advertising state and received by the Link Layer in the Initiating state.

The PHY used for these PDUs shall be the same as the PHY used for the PDU that they reply to.

#### 2.3.3.1 CONNECT_IND and AUX_CONNECT_REQ

The Payload field of the CONNECT_IND and AUX_CONNECT_REQ PDUs is shown in Figure 2.15. TxAdd in the advertising physical channel PDU Header field indicates whether the address in the InitA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the advertising physical channel PDU Header field indicates whether the address in the AdvA field is public (RxAdd = 0) or random (RxAdd = 1).

The ChSel field in the CONNECT_IND PDU Header field shall be set to 1 if both the initiator and advertiser support the LE Channel Selection Algorithm #2 feature (see Section 4.5.8.3). If the initiator supports the LE Channel Selection Algorithm #2 feature but the advertiser does not, the initiator may set the ChSel field to 0 or 1. The ChSel field in the CONNECT_IND PDU Header field shall be set to 0 if the initiator does not support the LE Channel Algorithm #2 feature. The ChSel field in the AUX_CONNECT_REQ PDU Header field is reserved for future use.

| Payload | | |
|---|---|---|
| InitA (6 octets) | AdvA (6 octets) | LLData (22 octets) |

*Figure 2.15: CONNECT_IND and AUX_CONNECT_REQ PDU Payload*

The InitA field shall contain the Initiator's public or random device address as indicated by TxAdd. The AdvA field shall contain the advertiser's public or random device address as indicated by RxAdd.

The format of the LLData field is shown in Figure 2.16.

| LLData | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| AA (4 octets) | CRCInit (3 octets) | WinSize (1 octet) | WinOffset (2 octets) | Interval (2 octets) | Latency (2 octets) | Timeout (2 octets) | ChM (5 octets) | Hop (5 bits) | SCA (3 bits) |

*Figure 2.16: LLData field structure in CONNECT_IND and AUX_CONNECT_REQ PDU's Payload*

The AA field shall contain the ACL connection's Access Address determined by the Link Layer following the rules specified in Section 2.1.2.

The CRCInit field shall contain the initialization value for the CRC calculation for the ACL connection, as defined in Section 3.1.1. It shall be a random value, generated by the Link Layer. The seed for the random number generator shall be from a physical source of entropy and should have at least 20 bits of entropy.

The WinSize field shall be set to indicate the *transmitWindowSize* value, as defined in Section 4.5.3 in the following manner:

*transmitWindowSize* = WinSize × 1.25 ms.

The WinOffset field shall be set to indicate the *transmitWindowOffset* value, as defined in Section 4.5.3 in the following manner:

*transmitWindowOffset* = WinOffset × 1.25 ms.

The Interval field shall be set to indicate the *connInterval* as defined in Section 4.5.1 in the following manner:

*connInterval* = Interval × 1.25 ms.

The Latency field shall be set to indicate the *connPeripheralLatency* value, as defined in Section 4.5.1 in the following manner:

*connPeripheralLatency* = Latency.

The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined in Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

The ChM field shall contain the channel map indicating *Used* and *Unused* data channels. Every channel is represented with a bit positioned as per the data channel index as defined in Section 1.4.1. The LSB represents data channel index 0 and the bit in position 36 represents data channel index 36. A bit value of 0 indicates that the channel is *Unused*. A bit value of 1 indicates that the channel is *Used*. The bits in positions 37, 38 and 39 are reserved for future use.

Note: When mapping from RF channels to data channel index, there is a gap where advertising channel index 38 is placed.

The Hop field shall be set to indicate the *hopIncrement* used in the data channel selection algorithm as defined in Section 4.5.8.2. It shall have a random value in the range 5 to 16.

The SCA field shall be set to indicate the *centralSCA* used to determine the Central's worst case sleep clock accuracy as defined in Section 4.2.2. The value of the SCA field shall be set as defined in Table 2.13.

| SCA | *centralSCA or peripheralSCA* |
|---|---|
| 0 | 251 ppm to 500 ppm |
| 1 | 151 ppm to 250 ppm |
| 2 | 101 ppm to 150 ppm |
| 3 | 76 ppm to 100 ppm |
| 4 | 51 ppm to 75 ppm |
| 5 | 31 ppm to 50 ppm |
| 6 | 21 ppm to 30 ppm |
| 7 | 0 ppm to 20 ppm |

*Table 2.13: SCA field encoding*

#### 2.3.3.2 AUX_CONNECT_RSP

The AUX_CONNECT_RSP PDU uses the Common Extended Advertising Payload Format described in Section 2.3.4.

The AdvMode field shall be set to 0b00.

The Common Extended Advertising Payload Format fields permitted in the AUX_CONNECT_RSP PDU are shown in Table 2.14.

| | | Common Extended Advertising Payload Format fields | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Adv Mode** | **Event Type** | **AdvA** | **TargetA** | **CTE Info** | **ADI** | **Aux Ptr** | **Sync Info** | **Tx Power** | **ACAD** | **Adv Data** |
| 0b00 | Connection response | M | M | X | X | X | X | X | X | X |
| 0b01 to 0b11 | RFU | | | | | | | | | |

*Table 2.14: Common Extended Advertising Payload Format fields permitted in the AUX_CONNECT_RSP PDU*

### 2.3.4 Common Extended Advertising Payload Format

The following extended Advertising Physical Channel PDUs share the same Advertising Physical Channel PDU payload format, referred to in the specification as the “Common Extended Advertising Payload Format”:

- ADV_EXT_IND
- AUX_ADV_IND
- AUX_SCAN_RSP
- AUX_SYNC_IND
- AUX_CHAIN_IND
- AUX_CONNECT_RSP
- AUX_SYNC_SUBEVENT_IND
- AUX_SYNC_SUBEVENT_RSP

The common extended advertising payload format is shown in Figure 2.17.

| Payload | | | |
|---|---|---|---|
| Extended Header Length (6 bits) | AdvMode (2 bits) | Extended Header (0 - 63 octets) | AdvData (0 - 254 octets) |

*Figure 2.17: Common Extended Advertising Payload Format*

The Extended Header Length field contains a value between 0 and 63 and indicates the size of the variable length Extended Header field.

The AdvMode field indicates the mode of the advertisement. The value of the AdvMode field shall be set as defined in Table 2.15.

| Value | Mode | |
|---|---|---|
| 0b00 | Non-connectable | Non-scannable |
| 0b01 | Connectable | Non-scannable |
| 0b10 | Non-connectable | Scannable |
| 0b11 | Reserved for future use | |

*Table 2.15: AdvMode field encoding*

AdvData, if present, shall contain advertising data from the advertiser's Host. The maximum size of AdvData depends on the size of the Extended Header field. The size of the AdvData can be calculated by subtracting the length of the Extended Header field plus one octet from the Length field specified in the Advertising physical channel PDU Header field.

The Extended Header field is a variable length header that is present if, and only if, the Extended Header Length field is non-zero. The format of the Extended Header field is shown in Figure 2.18.

| Extended Header | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Extended Header Flags (1 octet) | AdvA (6 octets) | TargetA (6 octets) | CTEInfo (1 octet) | AdvData Info (ADI) (2 octets) | AuxPtr (3 octets) | SyncInfo (18 octets) | TxPower (1 octet) | ACAD (varies) |

*Figure 2.18: Extended Header*

The Extended Header Flags bit field definitions are shown in Table 2.16.

| Bit | Extended Header |
|---|---|
| 0 | AdvA |
| 1 | TargetA |
| 2 | CTEInfo |
| 3 | AdvDataInfo (ADI) |
| 4 | AuxPtr |
| 5 | SyncInfo |
| 6 | TxPower |
| 7 | Reserved for future use |

*Table 2.16: Extended Header Flags*

If a flag bit is set to 1, the corresponding Extended Header field is present; otherwise, the corresponding Extended Header field is not present. The Extended Header fields

that are present are always in the same order as the flags in the Extended Header flags (i.e., the AdvA field is first if present, then the TargetA field if present, etc.).

Whether an Extended Header flag and corresponding Extended Header field is mandatory, optional, or reserved for future use is dependent on the Advertising Physical Channel PDU in which the extended header is used.

#### 2.3.4.1 AdvA field

When present, the AdvA field is six octets with the format shown in Figure 2.19.

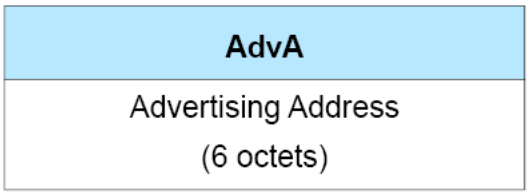



*Figure 2.19: AdvA field*

The Advertising Address field contains the advertiser's device address. If the AdvA field is present, TxAdd in the advertising physical channel PDU Header field indicates whether this address is public (TxAdd = 0) or random (TxAdd = 1).

#### 2.3.4.2 TargetA field

When present, the TargetA field is six octets with the format shown in Figure 2.20.

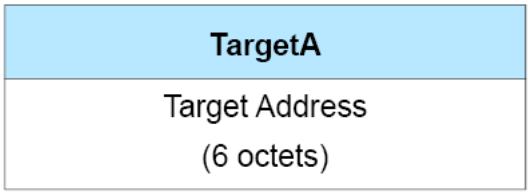



*Figure 2.20: TargetA field*

The Target Address field contains the scanner's or initiator's device address to which the advertisement is directed. If the TargetA field is present, RxAdd in the advertising physical channel PDU Header field indicates whether this address is public (RxAdd = 0) or random (RxAdd = 1).

#### 2.3.4.3 CTEInfo field

The presence of the CTEInfo field indicates that the packet includes a Constant Tone Extension. The CTEInfo field is defined in Section 2.5.2.

#### 2.3.4.4 AdvDataInfo field

When present, the AdvDataInfo (ADI) field is two octets with the format shown in Figure 2.21.

| AdvDataInfo | |
|---|---|
| Advertising Data ID (DID) (12 bits) | Advertising Set ID (SID) (4 bits) |

*Figure 2.21: AdvDataInfo field*

The Advertising Set ID (SID) is set by the advertiser's Host to identify an advertising set transmitted by this device.

The Advertising Data ID (DID) is set by the advertiser to indicate to the scanner whether it can assume that the data contents in the AdvData are a duplicate of the previous AdvData sent in an earlier packet.

#### 2.3.4.5 AuxPtr field

When present, the AuxPtr field is three octets with the format shown in Figure 2.22.

| AuxPtr | | | | |
|---|---|---|---|---|
| Channel Index (6 bits) | CA (1 bit) | Offset Units (1 bit) | AUX Offset (13 bits) | AUX PHY (3 bits) |

*Figure 2.22: AuxPtr field*

The presence of the AuxPtr field indicates that some or all of the advertisement data is in a subsequent auxiliary packet. The contents of the AuxPtr field describe this packet.

The Channel Index field contains the general-purpose channel index (see Section 1.4.1) used to transmit the auxiliary packet.

The Offset Units field indicates the units used by the Aux Offset Field. The value of the Offset Units field shall be set as defined in Table 2.17.

| Value | Units |
|---|---|
| 0 | 30 µs |
| 1 | 300 µs |

*Table 2.17: Offset Units field encoding*

The Aux Offset field contains the time from a reference point to the approximate start of the auxiliary packet, where the reference point is the start of the packet containing the AuxPtr field. The value of the AUX Offset field is in the unit of time indicated by the Offset Units field; the offset is determined by multiplying the value by the unit. The Aux Offset shall be at least the length of the packet plus T_MAFS (see Section 4.1.2). The Offset Units field shall be set to 0 if the Aux Offset is less than 245,700 µs.The auxiliary packet shall not start any earlier than the Aux Offset after the reference point and shall start no later than the Aux Offset plus one Offset Unit after the reference point. This allows the Link Layer to round the Aux Offset to the Offset Unit.

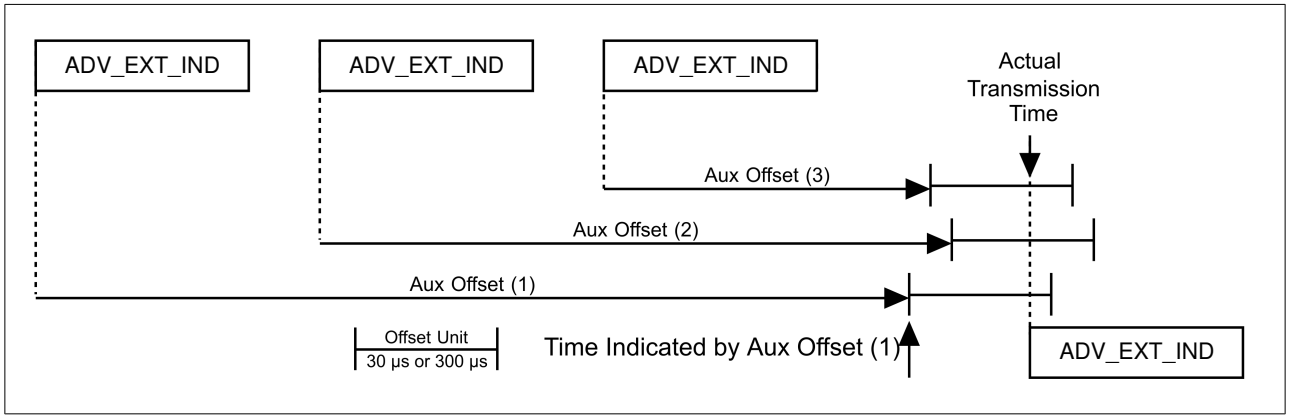



*Figure 2.23: Aux Offset transmission window*

The Aux PHY field indicates the PHY used to transmit the auxiliary packet. The value of the Aux PHY field shall be set as defined in Table 2.18.

| Value | PHY used |
|---|---|
| 0b000 | LE 1M |
| 0b001 | LE 2M |
| 0b010 | LE Coded |
| 0b011 – 0b111 | Reserved for future use |

*Table 2.18: Aux PHY field encoding*

The CA field contains the clock accuracy of the advertiser that will be used between the packet containing this data and the auxiliary packet. The value of the CA field shall be set as defined in Table 2.19.

| CA Value | Advertiser's Clock Accuracy |
|---|---|
| 0 | 51 ppm to 500 ppm |
| 1 | 0 ppm to 50 ppm |

*Table 2.19: Clock Accuracy field encoding*

An AuxPtr field with an Aux Offset of zero is permitted and indicates that no auxiliary packet will be transmitted but the Host advertising data in the current PDU is incomplete (see Section 2.3.4.9); it shall be treated as equivalent to one referring to an AUX_CHAIN_IND PDU that is never received. The remaining fields shall contain valid values.

### 2.3.4.6 SyncInfo field

When present, the SyncInfo field is 18 octets with the format shown in Figure 2.24.

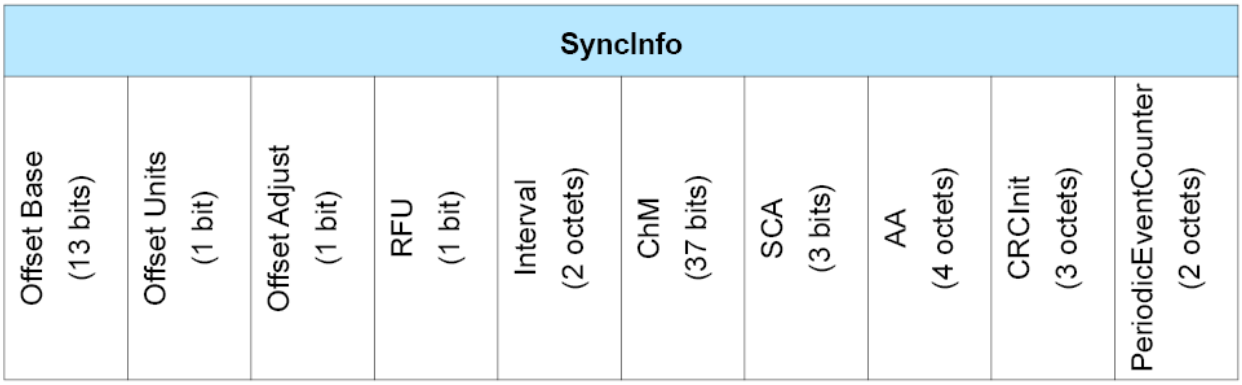



*Figure 2.24: SyncInfo field*

The presence of the SyncInfo field indicates the presence of a periodic advertising train (using AUX_SYNC_IND PDUs or AUX_SYNC_SUBEVENT_IND PDUs). The contents of the SyncInfo field describe this periodic advertising train.

The Offset Units field indicates the units used by the Offset Base field. The value of the Offset Units field shall be set as defined in Table 2.20.

| Value | Units |
|---|---|
| 0 | 30 µs |
| 1 | 300 µs |

*Table 2.20: Offset Units field encoding*

The SyncInfo field can appear in an advertising PDU or in an LL_PERIODIC_SYNC_IND PDU or LL_PERIODIC_SYNC_WR_IND PDU (see Section 2.4.2.27 and Section 2.4.2.40).

The syncPacketWindowOffset value is the time from a reference point to the start of the AUX_SYNC_IND or the AUX_SYNC_SUBEVENT_IND of subevent 0 packet that this SyncInfo field describes. If the SyncInfo appears in an advertising PDU, the reference point is the start of the packet containing it. If it appears in an LL_PERIODIC_SYNC_IND PDU or an LL_PERIODIC_SYNC_WR_IND PDU, the reference point is specified by that PDU. The value of syncPacketWindowOffset is

determined by multiplying the value of the Offset Base field by the unit of time indicated by the Offset Units field and then, if the Offset Adjust field is set to 1, adding 2.4576 seconds. The Offset Units field shall be set to 0 if syncPacketWindowOffset is less than 245,700 µs. The Offset Adjust field shall be set to 0 if the Offset Units field is set to 0 or if the SyncInfo field appears within an advertising PDU. As illustrated in Figure 2.25, the packet containing the AUX_SYNC_IND PDU or AUX_SYNC_SUBEVENT_IND PDU shall not start any earlier than syncPacketWindowOffset after the reference point and shall start no later than syncPacketWindowOffset plus one Offset unit after the reference point. This allows the Link Layer to round syncPacketWindowOffset to the Offset Unit.

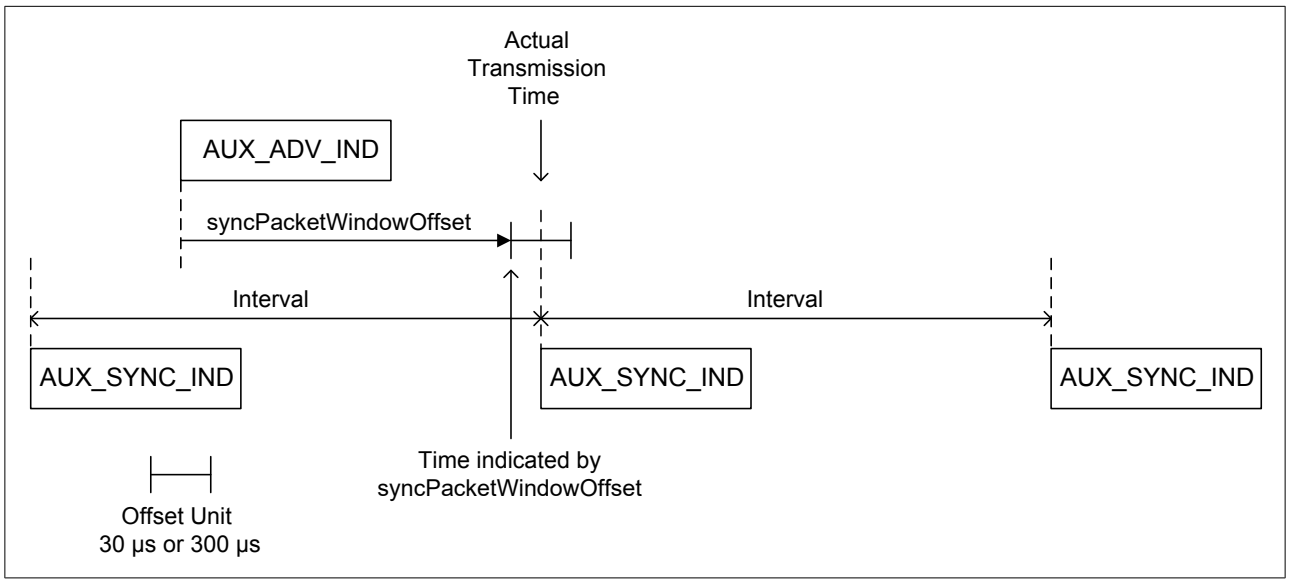



*Figure 2.25: Transmission window represented by syncPacketWindowOffset*

A value of 0 for syncPacketWindowOffset indicates that the time to the next AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND packet is greater than can be represented.

The Interval field contains the time, in 1.25 ms units, from the start of one packet of the periodic advertising train to the start of the next packet. The value shall not be less than 6 (7.5 ms).

The ChM field contains the channel map indicating *Used* and *Unused* RF channels on the periodic physical channel. Every channel is represented with a bit positioned as per the channel index as defined in Section 1.4.1. The LSB represents channel index 0 and the bit in position 36 represents channel index 36. A bit value of 0 indicates that the channel is *Unused*. A bit value of 1 indicates that the channel is *Used*.

The AA, CRCInit, and SCA fields have the same meaning as the corresponding fields in the CONNECT_IND PDU (see Section 2.3.3.1).

The PeriodicEventCounter field contains the value of *paEventCounter* (see Section 4.4.2.1) that applies to the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND packet that this SyncInfo field describes.

#### 2.3.4.7 TxPower field

When present, the TxPower is one octet with the format shown in Figure 2.26.

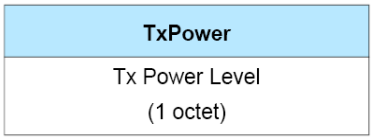



*Figure 2.26: TxPower field*

The Tx Power Level field is the same value defined for the Tx Power Advertising Data type defined in Section 1.5 of [1].

If the Host instructs the Controller to include the TxPower field in an advertisement, then it shall be included in the AUX_ADV_IND PDU if the extended advertising event contains one and in all the ADV_EXT_IND PDUs otherwise. Any AUX_CHAIN_IND PDUs should not contain a TxPower field. If the Host instructs the Controller to include the TxPower field in a periodic advertisement, then it shall be included in the AUX_SYNC_IND, AUX_SYNC_SUBEVENT_IND, or AUX_SYNC_SUBEVENT_RSP PDUs but not in any AUX_CHAIN_IND PDUs.

#### 2.3.4.8 ACAD field

The remainder of the Extended Header field forms the Additional Controller Advertising Data (ACAD) field. The length of this field is the Extended Header field length minus the sum of the size of the Extended Header flags (1 octet) and those fields indicated by the flags as present. ACAD cannot be fragmented across multiple advertising physical channel PDUs; it shall always fit inside a single advertising physical channel PDU.

The ACAD field, if present, shall hold data from the advertiser's Controller or intended to be used by the recipient's Controller. It uses the format described in [Vol 3] Part C, Section 11.

The ACAD type formats and meanings are defined in Section 1 of [1]. The ACAD type identifier values are defined in Assigned Numbers.

#### 2.3.4.9 Host Advertising Data

The portion of the PDU after the Extended Header field forms the AdvData field. The length of this field is specified in Section 2.3.4.

The AdvData field, if present, shall hold data from the advertiser's Host in the format described in [Vol 3] Part C, Section 11. If the Host does not provide any data, the

AdvData field shall be omitted but, for all other purposes in this Part, this shall be treated as if the Host had provided data.

The Controller may support fragmentation of Host Advertising Data. The total amount of Host Advertising Data before fragmentation shall not exceed 1650 octets. When the Link Layer fragments the Host advertising data, the number of fragments and the size of each fragment are chosen by the Controller. The Controller should minimize the number of fragments to ensure more reliability in delivering the entire Host advertising data. The Host may indicate a preference whether the Controller should fragment the Host advertising data, but the Controller may ignore the preference. If the amount of advertising or scan response data to be sent in an extended advertising or scanning PDU plus the Extended Header Length, AdvMode, and Extended Header fields exceed the maximum Advertising Physical Channel PDU Payload field (255 octets), the Link Layer shall fragment the Host advertising data.

The Link Layer shall place the multiple fragments in the AdvData field of different PDUs. When Host advertising data is fragmented the first fragment shall be placed in the AUX_ADV_IND, AUX_SYNC_IND or AUX_SCAN_RSP PDU while subsequent fragments shall be placed in AUX_CHAIN_IND PDUs. Each AUX_CHAIN_IND PDU is the auxiliary PDU of the PDU containing the previous fragment; AUX_CHAIN_IND PDUs not containing AdvData shall not have an auxiliary PDU containing AdvData.

If the Link Layer has fragmented the Host advertising data but is subsequently unable to transmit all the fragments, the last fragment that it is able to transmit should contain an AuxPtr field with an Aux Offset of zero so that scanners are aware that the data has been truncated.

## 2.4 Data Physical Channel PDU

The Data Physical Channel PDU has a 16 or 24 bit Header field, a variable size Payload field, and may include a Message Integrity Check (MIC) field.

The Data Physical Channel PDU is as shown in Figure 2.27.

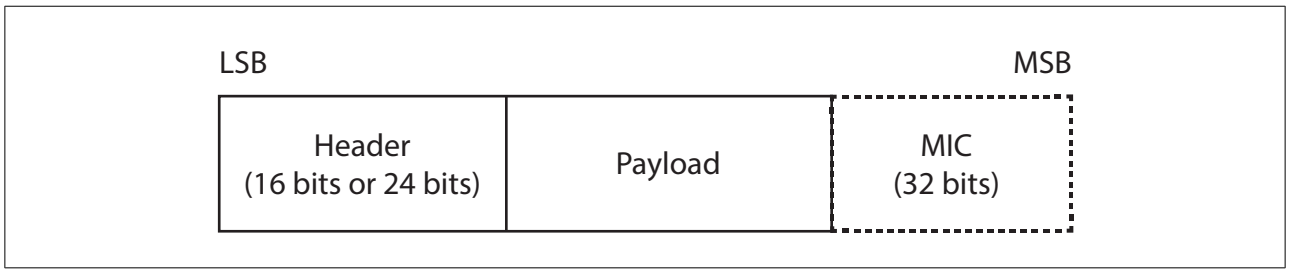



*Figure 2.27: Data Physical Channel PDU*

The Header field of the Data Physical Channel PDU is as shown in Figure 2.28.

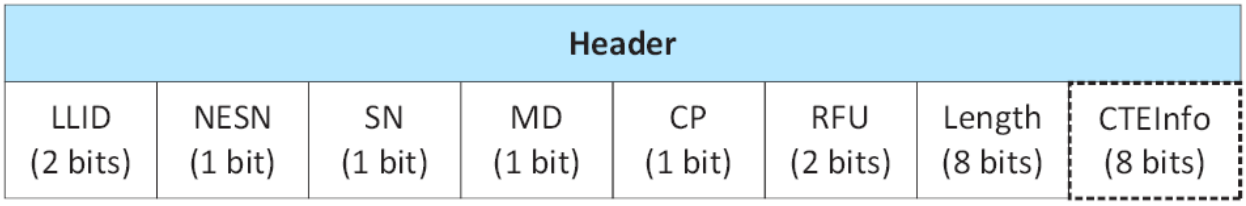



*Figure 2.28: Data Physical Channel PDU header*

The 16 or 24 bit Header field consists of 6 or 7 fields that are specified in Table 2.21.

The MIC field shall be included in all PDUs containing a non-zero length Payload field that are sent on an encrypted ACL connection. The MIC shall be calculated as specified in [Vol 6] Part E, Section 1. The MIC field shall not be included in any PDU that is sent on an unencrypted ACL connection or that has a zero-length Payload field.

The Payload field format depends on the LLID field of the Header field. If the LLID field is 0b01 or 0b10, the Data Physical Channel PDU Payload field contains an LL Data PDU as defined in Section 2.4.1. If the LLID field is 0b11 then the Data Physical Channel PDU Payload field contains an LL Control PDU as defined in Section 2.4.2.

The NESN bit of the Header field is defined in Section 4.5.9.

The SN bit of the Header field is defined in Section 4.5.9.

The MD bit of the Header field is defined in Section 4.5.6.

The CTEInfo Present (CP) field of the Header field indicates whether the Data Physical Channel PDU Header field has a CTEInfo field and therefore whether the data physical channel packet has a Constant Tone Extension. If the CP field is 0, then no CTEInfo field is present in the Data Channel PDU Header field and there is no Constant Tone Extension in the data physical channel packet. If the CP field is 1, then the CTEInfo field in the Data Physical Channel PDU Header field is present and the data physical channel packet includes a Constant Tone Extension.

The Length field of the Header field indicates the length of the Payload field and MIC if included. The Length field has the range 0 to 255 octets. The Payload field shall be less than or equal to 251 octets in length. The MIC is 4 octets in length.

The CTEInfo field of the Header field is present if indicated in the CP field. The CTEInfo field is defined in Section 2.5.2. The CTEInfo field can be present in the Header field of any Data Physical Channel PDU. However, the Link Layer shall not transmit a Data Physical Channel PDU containing a CTEInfo field unless it has determined that the peer device supports the Receiving Constant Tone Extensions feature (see Section 4.6.22).

Note: A Controller that transmits a PDU containing a CTEInfo field will not necessarily be able to receive one and vice-versa.

In this version of the specification, the only way to request a remote device to send a packet containing the CTEInfo field is via the Constant Tone Extension Request procedure.

| Field name | Description |
|---|---|
| **LLID** | The LLID indicates whether the packet is an LL Data PDU or an LL Control PDU.<br>0b00 = Reserved for future use<br>0b01 = LL Data PDU: Continuation fragment of an L2CAP message, or an Empty PDU.<br>0b10 = LL Data PDU: Start of an L2CAP message or a complete L2CAP message with no fragmentation.<br>0b11 = LL Control PDU |
| **NESN** | Next Expected Sequence Number |
| **SN** | Sequence Number |
| **MD** | More Data |
| **CP** | CTEInfo Present |
| **Length** | The Length field indicates the size, in octets, of the Payload field and MIC, if included. |
| **CTEInfo** | The CTEInfo field indicates the type and length of the Constant Tone Extension. |

*Table 2.21: Data Physical Channel PDU Header field*

### 2.4.1 LL Data PDU

An LL Data PDU is a Data Channel PDU that is used to send L2CAP data. The LLID field in the Header field shall be set to either 0b01 or 0b10.

An LL Data PDU with the LLID field in the Header field set to 0b01, and the Length field set to 0b00000000, is known as an Empty PDU. The Central's Link Layer may send an Empty PDU to the Peripheral to allow the Peripheral to respond with any Data Physical Channel PDU, including an Empty PDU.

An LL Data PDU with the LLID field in the Header set to 0b10 shall not have the Length field set to 0 and should not have it set to less than 4.

Note: If the Link Layer receives an HCI ACL Data Packet with Data_Total_Length equal to 0b00000000 and Packet_Boundary_Flag set to 0b00 (i.e., a start fragment), then the Link Layer cannot simply transmit the fragment over the air but, as this section requires, must instead combine it with one or more of the following continuation fragments to form a PDU with LLID set to 0b10 and non-zero length.

### 2.4.2 LL Control PDU

An LL Control PDU is a Data Physical Channel PDU that is used to control the Link Layer connection.

The Payload field of the LL Control PDU is shown in Figure 2.29.

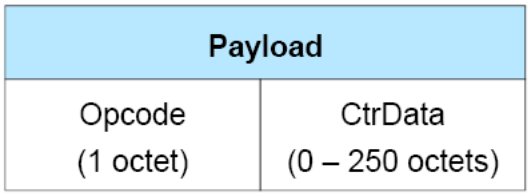



*Figure 2.29: LL Control PDU Payload*

An LL Control PDU shall not have the Length field set to 0b00000000.

The Opcode field identifies different types of LL Control PDU, as defined in Table 2.22.

The CtrData field in the LL Control PDU is specified by the Opcode field and is defined in the following subsections.

An LL Control PDU may have more than one format, indicated by “[v1]”, “[v2]”, etc. If so, then each field within the CtrData field has the same meaning in all formats it appears in. If a reference to such a PDU does not mention the version number, then any version that includes the fields mentioned may be used.

Where the description of a field within the CtrData field gives a range of valid values or other restrictions on a value (e.g. that field X is less than field Y), all other values shall be reserved for future use. The range may be directly specified in the relevant subsection for the LL Control PDU or indirectly, possibly in a section referenced by that subsection (e.g. the range of the WinSize field in Section 2.4.2.1 is derived from the range of the *transmitWindowSize* value specified in Section 4.5.3).

If no range is specified for a field, then all values for that field are valid.

Except where explicitly stated otherwise, all fields within the CtrData field in an LL Control PDU that hold an integer shall be interpreted as unsigned.

| Opcode | LL Control PDU Name |
|---|---|
| 0x00 | LL_CONNECTION_UPDATE_IND |
| 0x01 | LL_CHANNEL_MAP_IND |
| 0x02 | LL_TERMINATE_IND |
| 0x03 | LL_ENC_REQ |
| 0x04 | LL_ENC_RSP |
| 0x05 | LL_START_ENC_REQ |
| 0x06 | LL_START_ENC_RSP |

| Opcode | LL Control PDU Name |
|---|---|
| 0x07 | LL_UNKNOWN_RSP |
| 0x08 | LL_FEATURE_REQ |
| 0x09 | LL_FEATURE_RSP |
| 0x0A | LL_PAUSE_ENC_REQ |
| 0x0B | LL_PAUSE_ENC_RSP |
| 0x0C | LL_VERSION_IND |
| 0x0D | LL_REJECT_IND |
| 0x0E | LL_PERIPHERAL_FEATURE_REQ |
| 0x0F | LL_CONNECTION_PARAM_REQ |
| 0x10 | LL_CONNECTION_PARAM_RSP |
| 0x11 | LL_REJECT_EXT_IND |
| 0x12 | LL_PING_REQ |
| 0x13 | LL_PING_RSP |
| 0x14 | LL_LENGTH_REQ |
| 0x15 | LL_LENGTH_RSP |
| 0x16 | LL_PHY_REQ |
| 0x17 | LL_PHY_RSP |
| 0x18 | LL_PHY_UPDATE_IND |
| 0x19 | LL_MIN_USED_CHANNELS_IND |
| 0x1A | LL_CTE_REQ |
| 0x1B | LL_CTE_RSP |
| 0x1C | LL_PERIODIC_SYNC_IND |
| 0x1D | LL_CLOCK_ACCURACY_REQ |
| 0x1E | LL_CLOCK_ACCURACY_RSP |
| 0x1F | LL_CIS_REQ |
| 0x20 | LL_CIS_RSP |
| 0x21 | LL_CIS_IND |
| 0x22 | LL_CIS_TERMINATE_IND |
| 0x23 | LL_POWER_CONTROL_REQ |
| 0x24 | LL_POWER_CONTROL_RSP |
| 0x25 | LL_POWER_CHANGE_IND |
| 0x26 | LL_SUBRATE_REQ |
| 0x27 | LL_SUBRATE_IND |

| Opcode | LL Control PDU Name |
|---|---|
| 0x28 | LL_CHANNEL_REPORTING_IND |
| 0x29 | LL_CHANNEL_STATUS_IND |
| 0x2A | LL_PERIODIC_SYNC_WR_IND |
| 0x2B | LL_FEATURE_EXT_REQ |
| 0x2C | LL_FEATURE_EXT_RSP |
| 0x2D | LL_CS_SEC_RSP |
| 0x2E | LL_CS_CAPABILITIES_REQ |
| 0x2F | LL_CS_CAPABILITIES_RSP |
| 0x30 | LL_CS_CONFIG_REQ |
| 0x31 | LL_CS_CONFIG_RSP |
| 0x32 | LL_CS_REQ |
| 0x33 | LL_CS_RSP |
| 0x34 | LL_CS_IND |
| 0x35 | LL_CS_TERMINATE_REQ |
| 0x36 | LL_CS_FAE_REQ |
| 0x37 | LL_CS_FAE_RSP |
| 0x38 | LL_CS_CHANNEL_MAP_IND |
| 0x39 | LL_CS_SEC_REQ |
| 0x3A | LL_CS_TERMINATE_RSP |
| 0x3B | LL_FRAME_SPACE_REQ |
| 0x3C | LL_FRAME_SPACE_RSP |
| 0x3D | LL_OTA_UTP_IND |
| 0x3E | LL_CONNECTION_RATE_REQ |
| 0x3F | LL_CONNECTION_RATE_IND |
| 0xF0 to 0xFB | Reserved for specification development purposes |
| All other values | Reserved for future use |

*Table 2.22: LL Control PDU opcodes*

If an LL Control PDU is received that is not supported or reserved for future use, the Link Layer shall respond with an LL_UNKNOWN_RSP PDU. The UnknownType field of the LL_UNKNOWN_RSP PDU shall be set to the value of the Opcode in the received PDU.

If an LL Control PDU is received with the wrong length or with invalid CtrData fields, the Link Layer may continue with the relevant Link Layer procedure with an implementation-specific interpretation of the data (e.g., if the PDU is too long it can ignore the extra

data; if a field is out of range it can use the nearest permitted value). If it does not continue the procedure, it shall respond with an LL_UNKNOWN_RSP PDU or, if the relevant procedure allows it, an LL_REJECT_IND or LL_REJECT_EXT_IND PDU. The UnknownType field of the LL_UNKNOWN_RSP PDU or the RejectOpcode field of the LL_REJECT_EXT_IND PDU shall be set to the value of the Opcode in the received PDU.

Note: The device receiving an LL_UNKNOWN_RSP PDU is forbidden by Section 4.6 from initiating the same procedure later in the connection.

#### 2.4.2.1 LL_CONNECTION_UPDATE_IND

The format of the CtrData field is as shown in Figure 2.30.

| CtrData | | | | | |
|---|---|---|---|---|---|
| WinSize (1 octet) | WinOffset (2 octets) | Interval (2 octets) | Latency (2 octets) | Timeout (2 octets) | Instant (2 octets) |

*Figure 2.30: CtrData field of the LL_CONNECTION_UPDATE_IND PDU*

The WinSize field shall be set to indicate the *transmitWindowSize* value, as defined in Section 4.5.3 in the following manner:

*transmitWindowSize* = WinSize × 1.25 ms.

The WinOffset field shall be set to indicate the *transmitWindowOffset value,* as defined in Section 4.5.3, in the following manner:

*transmitWindowOffset* = WinOffset × 1.25 ms.

The Interval field shall be set to indicate the *connInterval* value, as defined in Section 4.5.1, in the following manner:

*connInterval* = Interval × 1.25 ms.

The Latency field shall be set to indicate the *connPeripheralLatency* value, as defined by Section 4.5.1, in the following manner:

*connPeripheralLatency* = Latency.

The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined by Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

The Instant field shall be set to indicate the instant described in Section 5.1.1.

#### 2.4.2.2 LL_CHANNEL_MAP_IND

The format of the CtrData field is shown in Figure 2.31.

| CtrData | |
|---|---|
| ChM<br>(5 octets) | Instant<br>(2 octets) |

*Figure 2.31: CtrData field of the LL_CHANNEL_MAP_IND PDU*

The ChM field shall contain the channel map indicating *Used* and *Unused* data channels. Every channel is represented with a bit positioned as per the data channel index defined by Section 1.4.1. The format of this field is identical to the ChM field in the CONNECT_IND PDU (see Section 2.3.3.1).

The Instant field shall be set to indicate the instant described in Section 5.1.2.

#### 2.4.2.3 LL_TERMINATE_IND

The format of the CtrData field is shown in Figure 2.32.

| CtrData |
|---|
| ErrorCode<br>(1 octet) |

*Figure 2.32: CtrData field of the LL_TERMINATE_IND PDU*

The ErrorCode field shall be set to inform the remote device why the connection is about to be terminated. See [Vol 1] Part F, Controller Error Codes for details.

#### 2.4.2.4 LL_ENC_REQ

The format of the CtrData field is shown in Figure 2.33.

| CtrData | | | |
|---|---|---|---|
| Rand<br>(8 octets) | EDIV<br>(2 octets) | SKD_C<br>(8 octets) | IV_C<br>(4 octets) |

*Figure 2.33: CtrData field of the LL_ENC_REQ PDU*

The Rand field shall contain a random number that is provided by the Host and used with EDIV (see [Vol 3] Part H, Section 2.4.4).

The EDIV field shall contain the encrypted diversifier.

The SKD_C field shall contain the Central's portion of the session key diversifier.

The IV_C field shall contain the Central's portion of the initialization vector.

#### 2.4.2.5 LL_ENC_RSP

The format of the CtrData field is shown in Figure 2.34.

| CtrData | |
|---|---|
| SKD_P (8 octets) | IV_P (4 octets) |

*Figure 2.34: CtrData field of the LL_ENC_ RSP PDU*

The SKD_P field shall contain the Peripheral's portion of the session key diversifier.

The IV_P field shall contain the Peripheral's portion of the initialization vector.

#### 2.4.2.6 LL_START_ENC_REQ

The LL_START_ENC_REQ PDU does not have a CtrData field.

#### 2.4.2.7 LL_START_ENC_RSP

The LL_START_ENC_RSP PDU does not have a CtrData field.

#### 2.4.2.8 LL_UNKNOWN_RSP

The format of the CtrData field is shown in Figure 2.35.

| CtrData |
|---|
| UnknownType (1 octet) |

*Figure 2.35: CtrData field of the LL_UNKNOWN_RSP PDU*

UnknownType shall contain the Opcode field value of the received LL Control PDU.

#### 2.4.2.9 LL_FEATURE_REQ

The format of the CtrData field is shown in Figure 2.36.

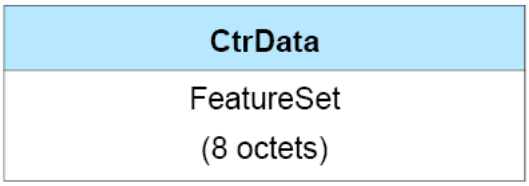



*Figure 2.36: CtrData field of the LL_FEATURE_REQ PDU*

FeatureSet shall contain the set of features supported by the Central's Link Layer as specified in Section 4.6.

#### 2.4.2.10 LL_FEATURE_RSP

The format of the CtrData field is shown in Figure 2.37.

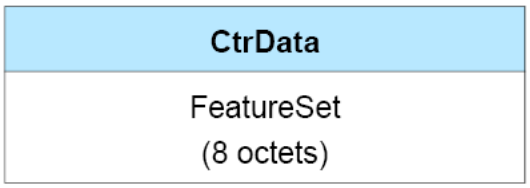



*Figure 2.37: CtrData field of the LL_FEATURE_RSP PDU*

FeatureSet[0] shall contain a set of features supported by the Link Layers of both the Central and Peripheral.

FeatureSet[1-7] shall contain a set of features supported by the Link Layer that transmits this PDU.

See Section 4.6 for the list of features.

#### 2.4.2.11 LL_PAUSE_ENC_REQ

The LL_PAUSE_ENC_REQ packet does not have a CtrData field.

#### 2.4.2.12 LL_PAUSE_ENC_RSP

The LL_PAUSE_ENC_RSP packet does not have a CtrData field.

#### 2.4.2.13 LL_VERSION_IND

The format of the CtrData field is shown in Figure 2.38.

| CtrData | | |
|---|---|---|
| Version (1 octet) | Company_Identifier (2 octets) | Subversion (2 octets) |

*Figure 2.38: CtrData field of the LL_VERSION_IND PDU*

Version field shall contain the version of the Bluetooth Link Layer specification supported by the Controller (see Assigned Numbers).

Company_Identifier field shall contain the company identifier of the manufacturer of the Bluetooth Controller (see Assigned Numbers).

Subversion field shall contain a unique value for each implementation or revision of an implementation of the Bluetooth Controller.

Note: A given value of Version does not indicate that the device supports all the features in the corresponding version of the specification; the relevant feature bits (see Section 4.6) must be checked instead.

Note: A larger value for Version does not necessarily indicate a higher version of the specification.

#### 2.4.2.14 LL_REJECT_IND

The format of the CtrData field is shown in Figure 2.39.

| CtrData |
|---|
| ErrorCode (1 octet) |

*Figure 2.39: CtrData field of the LL_ REJECT_IND*

ErrorCode shall contain the reason a request was rejected; see [Vol 1] Part F, Controller Error Codes.

#### 2.4.2.15 LL_PERIPHERAL_FEATURE_REQ

The format of the CtrData field is shown in Figure 2.40.

| CtrData |
| --- |
| FeatureSet (8 octets) |

*Figure 2.40: CtrData field of the LL_PERIPHERAL_FEATURE_REQ PDU*

FeatureSet shall contain the set of features supported by the Peripheral's Link Layer as specified in Section 4.6.

#### 2.4.2.16 LL_CONNECTION_PARAM_REQ

The format of the CtrData field is shown in Figure 2.41.

| CtrData | | | | | |
| --- | --- | --- | --- | --- | --- |
| Interval_Min (2 octets) | Interval_Max (2 octets) | Latency (2 octets) | Timeout (2 octets) | PreferredPeriodicity (1 octet) | ReferenceConnEventCount (2 octets) |

| CtrData (continued) | | | | | |
| --- | --- | --- | --- | --- | --- |
| Offset0 (2 octets) | Offset1 (2 octets) | Offset2 (2 octets) | Offset3 (2 octets) | Offset4 (2 octets) | Offset5 (2 octets) |

*Figure 2.41: CtrData field of the LL_CONNECTION_PARAM_REQ PDU*

The Interval_Min field shall be set to indicate the minimum value of *connInterval*, as defined in Section 4.5.1, in the following manner:

*connInterval* = Interval_Min × 1.25 ms.

The Interval_Max field shall be set to indicate the maximum value of *connInterval*, as defined in Section 4.5.1, in the following manner:

*connInterval* = Interval_Max × 1.25 ms.

The value shall not be less than the value of Interval_Min.

The Latency field shall be set to indicate the *connPeripheralLatency* value, as defined by Section 4.5.1, in the following manner:

*connPeripheralLatency* = Latency.

The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined by Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

The PreferredPeriodicity field shall be set to indicate a value the *connInterval* is preferred to be a multiple of. PreferredPeriodicity is in units of 1.25 ms. E.g. if the PreferredPeriodicity is set to 100, it implies that *connInterval* is preferred to be any multiple of 125 ms. A value of zero means no preference. The PreferredPeriodicity shall be less than or equal to Interval_Max.

The ReferenceConnEventCount field shall be set to indicate the value of the *connEventCounter* relative to which all the valid Offset0 to Offset5 fields have been calculated.

Note: The ReferenceConnEventCount field is independent of the Instant field in the LL_CONNECTION_UPDATE_IND PDU.

The Offset0, Offset1, Offset2, Offset3, Offset4, and Offset5 fields shall be set to indicate the possible values of the position of the anchor points of the LE connection with the updated connection parameters relative to the ReferenceConnEventCount. The Offset0 to Offset5 fields are in units of 1.25 ms and are in decreasing order of preference; that is, Offset0 is the most preferred value, followed by Offset1, and so on. Offset0 to Offset5 shall be less than Interval_Max. A value of 0xFFFF means not valid. Valid Offset0 to Offset5 fields shall contain unique values. Valid fields shall always be before invalid fields.

#### 2.4.2.17 LL_CONNECTION_PARAM_RSP

The format of the LL_CONNECTION_PARAM_RSP PDU is identical to the format of the LL_CONNECTION_PARAM_REQ PDU (see Section 2.4.2.16).

#### 2.4.2.18 LL_REJECT_EXT_IND

The format of the CtrData field is shown in Figure 2.42.

| CtrData | |
|---|---|
| RejectOpcode (1 octet) | ErrorCode (1 octet) |

*Figure 2.42: CtrData field of the LL_REJECT_EXT_IND PDU*

RejectOpcode shall contain the Opcode field value of the LL Control PDU being rejected.

ErrorCode shall contain the reason the LL Control PDU was being rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.

This PDU shall be issued only when the remote Link Layer supports the Extended Reject Indication Link Layer feature (Section 4.6). Otherwise, the LL_REJECT_IND PDU (Section 2.4.2.14) shall be issued instead.

#### 2.4.2.19 LL_PING_REQ

The LL_PING_REQ PDU does not have a CtrData field.

#### 2.4.2.20 LL_PING_RSP

The LL_PING_RSP PDU does not have a CtrData field.

#### 2.4.2.21 LL_LENGTH_REQ and LL_LENGTH_RSP

The format of the CtrData field for both the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs is shown in Figure 2.43.

| CtrData | | | |
|---|---|---|---|
| MaxRxOctets (2 octets) | MaxRxTime (2 octets) | MaxTxOctets (2 octets) | MaxTxTime (2 octets) |

*Figure 2.43: CtrData field of the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs*

MaxRxOctets shall be set to the sender's *connMaxRxOctets* value, as defined in Section 4.5.10. The MaxRxOctets field shall have a value not less than 27 octets.

MaxRxTime shall be set to the sender's *connMaxRxTime* value, as defined in Section 4.5.10. The MaxRxTime field shall have a value not less than 328 µs.

MaxTxOctets shall be set to the sender's *connMaxTxOctets* value, as defined in Section 4.5.10. The MaxTxOctets field shall have a value not less than 27 octets.

MaxTxTime shall be set to the sender's *connMaxTxTime* value, as defined in Section 4.5.10. The MaxTxTime field shall have a value not less than 328 µs.

#### 2.4.2.22 LL_PHY_REQ and LL_PHY_RSP

The format of the CtrData field for both the LL_PHY_REQ and LL_PHY_RSP PDUs is shown in Figure 2.44.

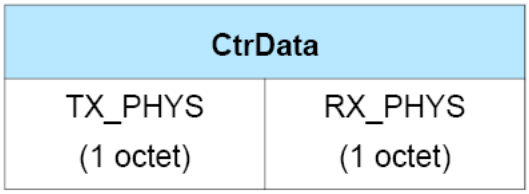



*Figure 2.44: CtrData field of the LL_PHY_REQ and LL_PHY_RSP PDUs*

TX_PHYS shall be set to indicate the transmitter PHYs that the sender prefers to use.

RX_PHYS shall be set to indicate the receiver PHYs that the sender prefers to use.

These fields each consist of 8 bits; at least one bit in each field shall be set to 1. The bits that are set indicate which PHY or PHYs the sender prefers to use, as specified in Table 2.23.

| Bit Number | Meaning |
|---|---|
| 0 | LE 1M PHY |
| 1 | LE 2M PHY |
| 2 | LE Coded PHY |
| All other bits | Reserved for future use |

*Table 2.23: PHY field bit meanings*

#### 2.4.2.23 LL_PHY_UPDATE_IND

The format of the CtrData field is shown in Figure 2.45.

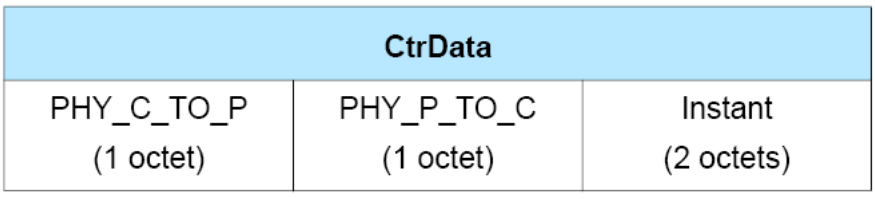



*Figure 2.45: CtrData field of the LL_PHY_UPDATE_IND PDU*

PHY_C_TO_P shall be set to indicate the PHY that shall be used for packets sent from the Central to the Peripheral. PHY_P_TO_C shall be set to indicate the PHY that shall

be used for packets sent from the Peripheral to the Central. These fields each consist of 8 bits. If a PHY is changing, the bit corresponding to the new PHY (as specified in Table 2.23) shall be set to 1 and the remaining bits to 0; if a PHY is remaining unchanged, then the corresponding field shall be set to the value 0.

Instant shall be set to indicate the instant described in Section 5.1.10.

If both the PHY_C_TO_P and PHY_P_TO_C fields are zero then there is no Instant and the Instant field is reserved for future use.

#### 2.4.2.24 LL_MIN_USED_CHANNELS_IND

The format of the CtrData field is as shown in Figure 2.46.

| CtrData | |
|---|---|
| PHYS<br>(1 octet) | MinUsedChannels<br>(1 octet) |

*Figure 2.46: CtrData field of the LL_MIN_USED_CHANNELS_IND PDU*

The PHYS field shall be set to the PHY(s) for which the Peripheral has a minimum number of used channels requirement. The PHYS field consists of 8 bits as specified in Table 2.23. At least one bit in the field shall be set to 1.

The MinUsedChannels field contains the minimum number of channels to be used on the specified PHY. The MinUsedChannels field shall have a value in the range 2 to 37.

#### 2.4.2.25 LL_CTE_REQ

The format of the CtrData field is shown in Figure 2.47.

| CtrData | | |
|---|---|---|
| MinCTELenReq<br>(5 bits) | RFU<br>(1 bit) | CTETypeReq<br>(2 bits) |

*Figure 2.47: CtrData field of the LL_CTE_REQ PDU*

MinCTELenReq shall contain the minimum length of the Constant Tone Extension (see Section 2.5.1) requested from the remote device, in 8 µs units. The value of the field shall be in the range 2 to 20.

CTETypeReq shall contain the type of the Constant Tone Extension requested from the remote device as specified in Table 2.24.

| Value | Meaning |
|---|---|
| 0 | AoA Constant Tone Extension |
| 1 | AoD Constant Tone Extension with 1 µs slots |
| 2 | AoD Constant Tone Extension with 2 µs slots |
| 3 | Reserved for future use |

*Table 2.24: CTETypeReq field bit meanings*

#### 2.4.2.26 LL_CTE_RSP

The LL_CTE_RSP PDU does not have a CtrData field.

#### 2.4.2.27 LL_PERIODIC_SYNC_IND

The format of the CtrData field is shown in Figure 2.48. This field is also embedded in the LL_PERIODIC_SYNC_WR_IND PDU (see Section 2.4.2.40).

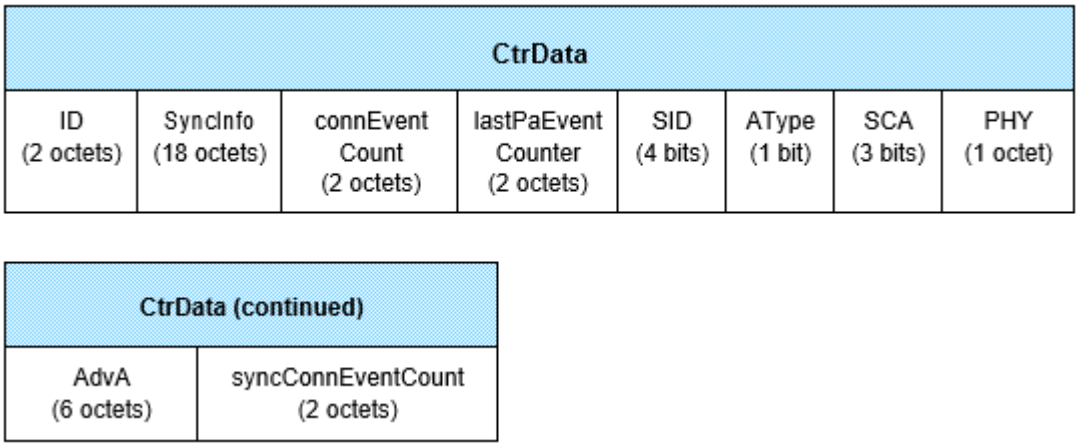



*Figure 2.48: CtrData field of the LL_PERIODIC_SYNC_IND PDU*

ID shall be set to an identifier provided by the Host. This is for use by higher-level protocols and its value is not specified or used by this specification.

SyncInfo has the meaning and format specified in Section 2.3.4.6, except that the reference point for syncPacketWindowOffset is the anchor point of the nearest connection event with the counter value specified in the connEventCount field and a zero offset does not have a special meaning.

connEventCount shall be set to a connection event counter value that meets the requirement currEvent - $2^{14}$ < connEventCount < currEvent + $2^{14}$ (*mod* 65536), where currEvent is the counter value for the connection event when the LL_PERIODIC_SYNC_IND PDU is being transmitted (or retransmitted).

In an LL_PERIODIC_SYNC_IND PDU, lastPaEventCounter shall be set to the paEventCounter applying to the AUX_SYNC_IND PDU used to determine the contents of the SyncInfo; the device sending this PDU shall have actually transmitted or received the AUX_SYNC_IND PDU.

When embedded in an LL_PERIODIC_SYNC_WR_IND PDU, lastPaEventCounter shall be set to the paEventCounter applying to the AUX_SYNC_SUBEVENT_IND PDU in subevent 0 used to determine the contents of the SyncInfo; the device sending the LL_PERIODIC_SYNC_WR_IND PDU shall have actually transmitted or received an AUX_SYNC_SUBEVENT_IND PDU in a subevent of that periodic advertising event.

The lastPaEventCounter field and the PeriodicEventCounter field of the SyncInfo shall be equal, shall have a difference of 1 (*mod* 65536), or shall represent times not more than 5 seconds apart.

SID shall be set to the Advertising SID subfield of the advertising set pointing to the periodic advertising.

AType shall be 0 if the AdvA field holds a public address and 1 if it holds a random address.

SCA shall be set to indicate the sleep clock accuracy of the device sending this PDU. The value shall be represented in the same way as in the CONNECT_IND PDU (see Section 2.3.3.1).

PHY shall be set to indicate the PHY used by the periodic advertising. The bit corresponding to the PHY (as specified in Table 2.23) shall be set to 1 and the remaining bits to 0.

If the advertiser's address in the advertising set pointing to the periodic advertising is a resolvable private address, AdvA shall be set to any resolvable private address that was generated using the same IRK. Otherwise, AdvA shall be set to the advertiser's address in the advertising set.

syncConnEventCount shall be set to the connection event counter for the connection event that the sending device used in determining the contents of this PDU. This shall be a connection event where the sending device received a packet from the device it will send the LL_PERIODIC_SYNC_IND PDU to and, if the sending device is the Peripheral on the piconet containing those two devices, it used the received packet to synchronize its anchor point (see Section 4.5.7). This means that the connection must be established before this PDU can be transmitted.

#### 2.4.2.28 LL_CLOCK_ACCURACY_REQ and LL_CLOCK_ACCURACY_RSP

The format of the CtrData field is shown in Figure 2.49.

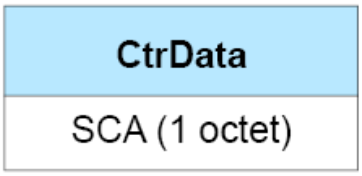



*Figure 2.49: CtrData field of the LL_CLOCK_ACCURACY_REQ and LL_CLOCK_ACCURACY_RSP PDUs*

The SCA field shall indicate the current *centralSCA* (if the PDU is sent by the Central) or *peripheralSCA* (if the PDU is sent by the Peripheral) used to determine the worst case sleep clock accuracy of the sending device as defined in Section 4.2.2. The format of this field is identical to the SCA field in the CONNECT_IND PDU (see Section 2.3.3.1).

#### 2.4.2.29 LL_CIS_REQ

The format of the CtrData field is shown in Figure 2.50.

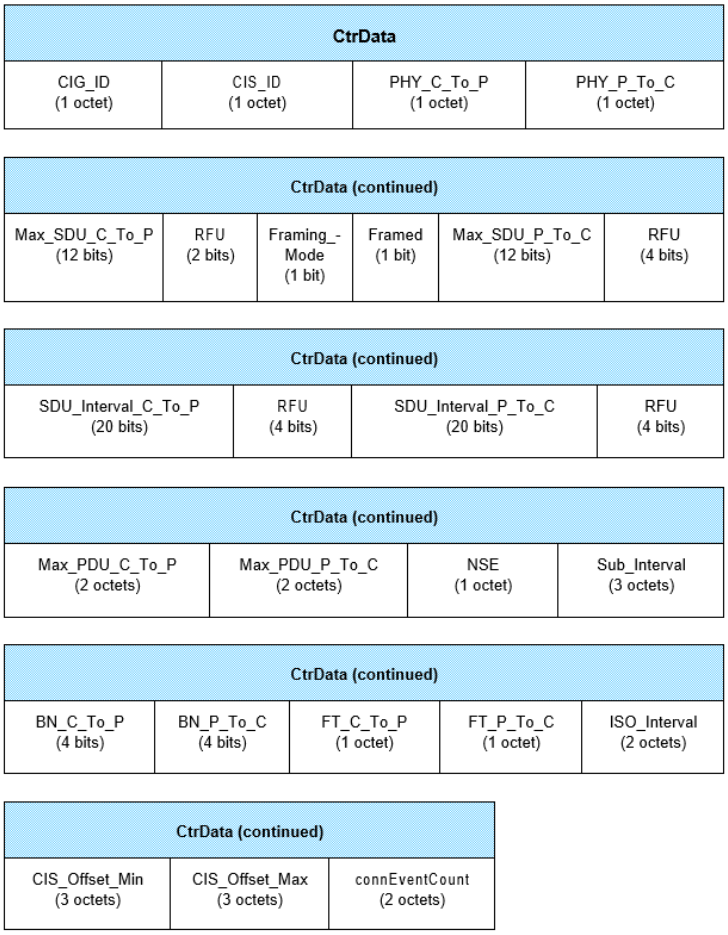



*Figure 2.50: CtrData field of the LL_CIS_REQ PDU*

CIG_ID shall be set to the CIG identifier as defined in Section 4.5.14.

CIS_ID shall be set to the CIS identifier as defined in Section 4.5.13.1

PHY_C_To_P shall be set to indicate the PHY that shall be used from the Central to the Peripheral, as defined in Table 2.23. Exactly 1 bit shall be set.

PHY_P_To_C shall be set to indicate the PHY that shall be used from the Peripheral to the Central, as defined in Table 2.23. Exactly 1 bit shall be set.

Max_SDU_C_To_P shall be set to the maximum size of an SDU, in octets, from the Central's Host.

Framed shall be set to 0 for unframed Data PDUs and 1 for framed Data PDUs.

For framed Data PDUs, Framing_Mode shall be set to 0 if Segmentable mode is being used and to 1 if Unsegmented mode is being used. For unframed Data PDUs, Framing_Mode is RFU.

Max_SDU_P_To_C shall be set to the maximum size of an SDU, in octets, from the Peripheral's Host.

SDU_Interval_C_To_P shall be set to the time, in microseconds, between two consecutive SDUs from the Central's Host. The value shall be between 255 and 1048575.

SDU_Interval_P_To_C shall be set to the time, in microseconds, between two consecutive SDUs from the Peripheral's Host. The value shall be between 255 and 1048575.

Max_PDU_C_To_P shall be set to the maximum Payload field size, in octets, from the Central to the Peripheral. The value shall be between 0 and 251 octets. This field shall be set to 0 if and only if BN_C_To_P is set to 0.

Max_PDU_P_To_C shall be set to the maximum Payload field size, in octets, from the Peripheral to the Central. The value shall be between 0 and 251 octets. This field shall be set to 0 if and only if BN_P_To_C is set to 0.

NSE shall be set to the maximum number of subevents in each CIS event. The value shall be between 1 and 31.

Sub_Interval shall be set to the time, in microseconds, between the start of a subevent and the start of the next subevent for that CIS in the same CIS event. The value shall be set to 0 if the NSE field is set to 1; otherwise the value shall be at least 400 µs and shall be less than ISO_Interval.

BN_C_To_P shall be set to the BN parameter value used from the Central to Peripheral, as defined in Section 4.5.13.2. The value shall be between 0 and 15.

BN_P_To_C shall be set to the BN parameter value used from the Peripheral to Central, as defined in Section 4.5.13.2. The value shall be between 0 and 15.

FT_C_To_P shall be set to the FT parameter value used from the Central to the Peripheral, as defined in Section 4.5.13.2. The value shall be between 1 and 255.

FT_P_To_C shall be set to the FT parameter value used from the Peripheral to the Central, as defined in Section 4.5.13.2. The value shall be between 1 and 255.

ISO_Interval shall be set to the time between two consecutive CIS anchor points in units of 1.25 ms. The value shall be between 4 and 3200 (i.e. 5 ms to 4 s).

CIS_Offset_Min shall be set to the proposed minimum time, in microseconds, between the ACL anchor point of the connection event with the connection event counter equal to *connEventCount* and the first CIS anchor point. The value shall be at least 500 µs.

CIS_Offset_Max shall be set to the proposed maximum time, in microseconds, between the ACL anchor point of the connection event with the connection event counter equal to *connEventCount* and the first CIS anchor point. The value shall be greater than or equal to CIS_Offset_Min and less than connInterval. CIS_Offset_Max should be less than (connInterval – ((NSE – 1) × Sub_Interval + MPT_C + T_IFS_CIS + MPT_P + T_MSS_CIS)), where MPT_C and MPT_P are defined in Section 4.5.13.1. For the first CIS in a CIG it should be less than (connInterval – (CIG_Sync_Delay + T_MSS_CIS)).

*connEventCount* shall be set to a connection event counter value that meets the requirement currEvent – $2^{14}$ < *connEventCount* < currEvent + $2^{14}$ (*mod* 65536), where *currEvent* is the counter value for the connection event where the PDU containing this field is being transmitted or retransmitted. *connEventCount* should be set to a value greater than *currEvent* of the event in which the LL_CIS_REQ PDU is first transmitted.

#### 2.4.2.30 LL_CIS_RSP

The format of the CtrData field is shown in Figure 2.51.

| CtrData | | |
|---|---|---|
| CIS_Offset_Min (3 octets) | CIS_Offset_Max (3 octets) | connEventCount (2 octets) |

*Figure 2.51: CtrData field of the LL_CIS_RSP PDU*

Each field has the same meaning as the field with the same name in the LL_CIS_REQ PDU.

#### 2.4.2.31 LL_CIS_IND

The format of the CtrData field is shown in Figure 2.52.

| CtrData | | | | |
|---|---|---|---|---|
| AA<br>(4 octets) | CIS_Offset<br>(3 octets) | CIG_Sync_Delay<br>(3 octets) | CIS_Sync_Delay<br>(3 octets) | connEventCount<br>(2 octets) |

*Figure 2.52: CtrData field of the LL_CIS_IND PDU*

AA shall be set to the Access Address of the CIS, generated by the Link Layer following the rules specified in Section 2.1.2.

CIS_Offset shall be set to the time, in microseconds, from the ACL anchor point of the connection event that is referenced by *connEventCount* to the first CIS anchor point.

CIG_Sync_Delay shall be set to the value of CIG_Sync_Delay, as defined in Section 4.5.14.1, in microseconds.

CIS_Sync_Delay shall be set to the value of CIS_Sync_Delay, as defined in Section 4.5.14.1, in microseconds.

*connEventCount* shall be set to a connection event counter value that meets the requirement $currEvent - 2^{14} < connEventCount < currEvent + 2^{14}$ (*mod* 65536), where *currEvent* is the ACL connection event counter value for the connection event where the LL_CIS_IND PDU is being transmitted. *connEventCount* should be set to a value greater than *currEvent* of the event in which the LL_CIS_IND PDU is first transmitted.

#### 2.4.2.32 LL_CIS_TERMINATE_IND

The format of the CtrData field is shown in Figure 2.53.

| CtrData | | |
|---|---|---|
| CIG_ID<br>(1 octet) | CIS_ID<br>(1 octet) | ErrorCode<br>(1 octet) |

*Figure 2.53: CtrData field of the LL_CIS_TERMINATE_IND PDU*

CIG_ID shall be set to the identifier of the CIG containing the CIS being terminated.

CIS_ID shall be set to the identifier of the CIS being terminated.

ErrorCode shall be set to inform the peer why the CIS is about to be terminated (see [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions).

#### 2.4.2.33 LL_POWER_CONTROL_REQ

The format of the CtrData field is shown in Figure 2.54.

| CtrData | | |
|---|---|---|
| PHY<br>(1 octet) | Delta<br>(1 octet) | TxPower<br>(1 octet) |

*Figure 2.54: CtrData field of the LL_POWER_CONTROL_REQ PDU*

PHY shall be set to indicate the PHY for which the request is being made. The bit corresponding to the PHY (as specified in Table 2.25) shall be set to 1 and the remaining bits to 0.

| Bit Number | Meaning |
|---|---|
| 0 | LE 1M PHY |
| 1 | LE 2M PHY |
| 2 | LE Coded PHY with S=8 data coding |
| 3 | LE Coded PHY with S=2 data coding |
| All other bits | Reserved for future use |

*Table 2.25: PHY field bit meanings*

Delta shall be set to the requested change in the recipient's transmit power level, in dB, for the PHY indicated. The value is a signed integer: a positive value indicates a request to increase the transmit power level, a negative value indicates a request to decrease it, and zero indicates that no change is being requested. A value of 0x7F indicates a request to increase to the maximum power level.

TxPower shall be set to the sender's transmit power level for the PHY indicated. The value is in dBm, represented as a signed integer. When set to 127, it indicates that the value is unavailable. It shall not be set to 126.

#### 2.4.2.34 LL_POWER_CONTROL_RSP

The format of the CtrData field is shown in Figure 2.55.

| CtrData | | | | | |
|---|---|---|---|---|---|
| Min (1 bit) | Max (1 bit) | RFU (6 bits) | Delta (1 octet) | TxPower (1 octet) | APR (1 octet) |

*Figure 2.55: CtrData field of the LL_POWER_CONTROL_RSP PDU*

Min shall be set if the sender is currently at the minimum supported power level.

Max shall be set if the sender is currently at the maximum supported power level.

Note: if Min and Max are both set then the sender has a single fixed transmit power level.

Delta shall be set to the actual change in the sender's transmit power level, in dB, that the sender has made for the PHY requested. The value is a signed integer, with a positive value indicating a power increase, and a negative value indicating a power decrease. Zero indicates that there was no change.

TxPower shall be set to the sender's transmit power level for the PHY requested. The value is in dBm, represented as a signed integer. When set to 127, it indicates that the value is unavailable. When set to 126, it indicates that the sender is not currently managing power for the requested PHY; in this case all other fields shall be ignored.

APR (Acceptable Power Reduction) shall be set to the maximum decrease in radiative transmit power level of the peer device in dB, for the PHY requested, that is acceptable to the device sending this PDU. When set to 0xFF, it indicates that the sender is unable to determine a value.

#### 2.4.2.35 LL_POWER_CHANGE_IND

The format of the CtrData field is shown in Figure 2.56.

| CtrData | | | | | |
|---|---|---|---|---|---|
| PHY (1 octet) | Min (1 bit) | Max (1 bit) | RFU (6 bits) | Delta (1 octet) | TxPower (1 octet) |

*Figure 2.56: CtrData field of the LL_POWER_CHANGE_IND PDU*

PHY shall be set to indicate the PHY(s) for which the power level has changed, using the values listed in Table 2.25. The PHY field may have more than one bit set if the other fields have the same values for the PHYs being reported.

Min shall be set if the sender is currently at the minimum supported power level.

Max shall be set if the sender is currently at the maximum supported power level.

Note: if Min and Max are both set then the sender has a single fixed transmit power level.

Delta shall be set to the change in the sender's transmit power level, if any, for the PHY(s) indicated. The value is a signed integer in dB, with a positive value indicating a power increase, and a negative value indicating a power decrease. Zero indicates that there was no change.

TxPower shall be set to the sender's transmit power level for the PHY(s) indicated. The value is in dBm, represented as a signed integer. When set to 127, it indicates that the value is unavailable. When set to 126, it indicates that the sender has stopped managing power for the indicated PHY(s); in this case all other fields shall be ignored.

#### 2.4.2.36 LL_SUBRATE_REQ

The format of the CtrData field is shown in Figure 2.57.

| CtrData | | | | |
|---|---|---|---|---|
| SubrateFactorMin (2 octets) | SubrateFactor-Max (2 octets) | Max_Latency (2 octets) | Continuation-Number (2 octets) | Timeout (2 octets) |

*Figure 2.57: CtrData field of the LL_SUBRATE_REQ PDU*

The SubrateFactorMin field shall be set to the minimum requested *connSubrateFactor*, as defined in Section 4.5.1.

The SubrateFactorMax field shall be set to the maximum requested *connSubrateFactor*, as defined in Section 4.5.1.

The Max_Latency field shall be set to the maximum requested *connPeripheralLatency*, as defined in Section 4.5.1, in subrated events. The same maximum shall apply irrespective of the subrating factor actually chosen. The value of SubrateFactorMax × (Max_Latency + 1) shall be less than or equal to than 500.

The ContinuationNumber field shall be set to the minimum requested *connContinuationNumber*, as defined in Section 4.5.1.

The Timeout field shall be set to indicate the requested *connSupervisionTimeout* value, as defined in Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

#### 2.4.2.37 LL_SUBRATE_IND

The format of the CtrData field is shown in Figure 2.58.

| CtrData | | | | |
|---|---|---|---|---|
| SubrateFactor (2 octets) | SubrateBaseEvent (2 octets) | Latency (2 octets) | Continuation-Number (2 octets) | Timeout (2 octets) |

*Figure 2.58: CtrData field of the LL_SUBRATE_IND PDU*

The SubrateFactor field shall be set to the new *connSubrateFactor*, as defined in Section 4.5.1, to be used on the connection.

The SubrateBaseEvent field shall be set to the new *connSubrateBaseEvent,* as defined in Section 4.5.1, to be used on the connection.

The Latency field shall be set to the new *connPeripheralLatency* in subrated events, as defined in Section 4.5.1, to be used on the connection.

The ContinuationNumber field shall be set to the new *connContinuationNumber*, as defined in Section 4.5.1, to be used on the connection.

The Timeout field shall be set to indicate the new *connSupervisionTimeout* value, as defined in Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

#### 2.4.2.38 LL_CHANNEL_REPORTING_IND

The format of the CtrData field is shown in Figure 2.59.

| CtrData | | |
|---|---|---|
| Enable (1 octet) | Min_Spacing (1 octet) | Max_Delay (1 octet) |

*Figure 2.59: CtrData field of the LL_CHANNEL_REPORTING_IND*

Enable shall be set to indicate whether channel classification reporting needs to be enabled or disabled as specified in Table 2.26.

| Value | Meaning |
|---|---|
| 0x00 | Disable channel classification reporting |
| 0x01 | Enable channel classification reporting |
| All other values | Reserved for future use |

*Table 2.26: Enable field values*

Min_Spacing shall be set to indicate, in units of 200 ms, the minimum amount of time from the last LL_CHANNEL_STATUS_IND PDU that was sent before the next LL_CHANNEL_STATUS_IND PDU may be sent. The value shall be between 5 (1 second) and 150 (30 seconds).

Max_Delay shall be set to indicate, in units of 200 ms, the maximum amount of time between the change in the channel classification being detected by a Peripheral and its generation of an LL_CHANNEL_STATUS_IND PDU. The value shall be between 5 (1 second) and 150 (30 seconds). Max_Delay shall be greater than or equal to Min_Spacing.

#### 2.4.2.39 LL_CHANNEL_STATUS_IND

The format of the CtrData field is shown in Figure 2.60.

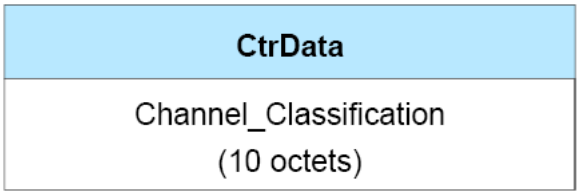



*Figure 2.60: CtrData field of the LL_CHANNEL_STATUS_IND*

Channel_Classification has the type uint2[37]. The $n^{th}$ (numbering from 0) element defines the classification of data channel index n. The value of each element indicates:

- 0 = *unknown*
- 1 = *good*
- 2 = reserved for future use
- 3 = *bad*

#### 2.4.2.40 LL_PERIODIC_SYNC_WR_IND

The format of the CtrData field is shown in Figure 2.61.

| CtrData | | | | | |
|---|---|---|---|---|---|
| CtrData of LL_PERIODIC_SYNC_-IND (34 octets) | RspAA (4 octets) | numSub-events (1 octet) | sub-event-Interval (1 octet) | response-SlotDelay (1 octet) | response-SlotSpacing (1 octet) |

*Figure 2.61: CtrData field of the LL_PERIODIC_SYNC_WR_IND PDU*

The CtrData of LL_PERIODIC_SYNC_IND field has the meaning and format specified in Section 2.4.2.27.

The RspAA field shall be set to the Access Address to be used by the device when it transmits a response packet to the periodic advertiser.

The numSubevents field shall be set to the number of subevents.

The subeventInterval field shall be set to the time, in 1.25 ms units, from the start of one subevent to the start of the next subevent.

The responseSlotDelay field shall be set to the time, in 1.25 ms units, from the start of one subevent to the first response slot.

responseSlotSpacing shall be set to the time, in 0.125 ms units, from the start of one response slot to the start of the next response slot.

#### 2.4.2.41 LL_FEATURE_EXT_REQ and LL_FEATURE_EXT_RSP

The format of the CtrData field is shown in Figure 2.62.

| CtrData | | |
|---|---|---|
| MaxPage (1 octet) | PageNumber (1 octet) | FeaturePage (24 octets) |

*Figure 2.62: CtrData field of the LL_FEATURE_EXT_REQ and LL_FEATURE_EXT_RSP PDUs*

MaxPage shall contain the number of the highest page of the sender's FeatureSet, with certain bits set to 0 as specified in Section 4.6, containing at least one bit set to 1.

PageNumber shall contain the page number of the page stored in FeaturePage, in the range 0x01 to 0x0A.

FeaturePage shall contain the set of features in page PageNumber of the sender's FeatureSet, with certain bits set to 0, as specified in Section 4.6.

#### 2.4.2.42 LL_CS_SEC_REQ

The format of the CtrData field is shown in Figure 2.63.

| CtrData | | |
|---|---|---|
| CS_IV_C<br>(8 octets) | CS_IN_C<br>(4 octets) | CS_PV_C<br>(8 octets) |

*Figure 2.63: CtrData field of the LL_CS_SEC_REQ PDU*

The CS_IV_C field shall contain the Central's portion of the initialization vector used for CS security.

The CS_IN_C field shall contain the Central's portion of the instantiation nonce used for CS security.

The CS_PV_C field shall contain the Central's portion of the personalization vector used for CS security.

#### 2.4.2.43 LL_CS_SEC_RSP

The format of the CtrData field is shown in Figure 2.64.

| CtrData | | |
|---|---|---|
| CS_IV_P<br>(8 octets) | CS_IN_P<br>(4 octets) | CS_PV_P<br>(8 octets) |

*Figure 2.64: CtrData field of the LL_CS_SEC_RSP PDU*

The CS_IV_P field shall contain the Peripheral's portion of the initialization vector used for CS security.

The CS_IN_P field shall contain the Peripheral's portion of the instantiation nonce used for CS security.

The CS_PV_P field shall contain the Peripheral's portion of the personalization vector used for CS security.

#### 2.4.2.44 LL_CS_CAPABILITIES_REQ and LL_CS_CAPABILITIES_RSP

The format of the CtrData field is shown in Figure 2.65.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| **All versions** | | | | | | |
| Mode_-Types (1 octet) | RTT_-Capability (1 octet) | RTT_AA_-Only_N (1 octet) | RTT_-Sounding_N (1 octet) | RTT_-Random_-Sequence_N (1 octet) | NADM_-Sounding_-Capability (2 octets) | NADM_-Random_-Capability (2 octets) |

| CtrData (Continued) | | | | | | |
|---|---|---|---|---|---|---|
| **All versions** | | | | | | |
| CS_-SYNC_-PHY_-Capability (1 octet) | Num_Ant (4 bits) | Max_Ant_-Path (4 bits) | Role (2 bits) | IPT (1 bit) | No_FAE (1 bit) | Channel Selection #3c (1 bit) |

| CtrData (Continued) | | | | | | |
|---|---|---|---|---|---|---|
| **All versions** | | | | | | |
| Sounding_-PCT_-Estimate (1 bit) | RFU (2 bits) | Num_-Configs (1 octet) | Max_-Procedures_-Supported (2 octets) | T_SW (1 octet) | T_IP1_-Capability (2 octets) | T_IP2_-Capability (2 octets) |

| CtrData (Continued) | | | | | |
|---|---|---|---|---|---|
| **All versions** | | | | **[v2] and above** | |
| T_FCS_-Capability (2 octets) | T_PM_-Capability (2 octets) | RTT_PHY (1 bit) | TX_SNR_-Capability (7 bits) | T_IP2_IPT_-Capability (2 octets) | T_SW_IPT (1 octet) |

| CtrData (Continued) | | |
|---|---|---|
| **[v2] and above** | | |
| RTT_2M_AA_-Only_N (1 octet) | RTT_2M_-Sounding_N (1 octet) | RTT_2M_Random_-Sequence_N (1 octet) |

*Figure 2.65: CtrData field of the LL_CS_CAPABILITIES_REQ PDU*

The Mode_Types field is a bit field that shall be set to indicate which of the optional CS modes are supported by the Link Layer transmitting this PDU, as described in [Vol 6] Part H, Section 4.4. The meaning of each bit is specified in Table 2.27.

| Bit Number | Meaning |
|---|---|
| 0 | Mode-3 |
| All other bits | Reserved for future use |

*Table 2.27: Mode_Types field values*

A Controller that supports mode-3 shall support the following section of this document:

- Channel Sounding step mode-3 ([Vol 6] Part H, Section 4.3.4)

The RTT_Capability field shall be set to indicate which of the time-of-flight accuracy requirements described in [Vol 6] Part H, Section 3.1.2 are met by the corresponding “N” count values (e.g., RTT_AA_Only_N, RTT_2M_AA_Only_N, RTT_Sounding_N, RTT_2M_Sounding_N, RTT_Random_Sequence_N, RTT_2M_Random_Sequence_N) for at least one combination of a specific PHY and Payload field length by that device. If a bit is set to 1, then the corresponding “N” value reflects the 10 ns requirement for that RTT variant. If a bit is set to 0 (default), then either the corresponding “N” value reflects the 150 ns requirement for that RTT variant or that variant is not supported. The respective bits are only valid if the corresponding "N" value is not 0. If the corresponding “N” value is set to 0, then the respective bit shall be set to 0. One or more bits in this field may be set; the meaning of each bit is specified in Table 2.28.

| Bit Number | Meaning |
|---|---|
| 0 | 0 = The value in the RTT_AA_Only_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_AA_Only_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS _CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is applicable to all mandatory and optional PHYs.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 1M PHY only. |
| 1 | 0 = The value in the RTT_Sounding_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_Sounding_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is applicable to all mandatory and optional PHYs.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 1M PHY only. |

| Bit Number | Meaning |
|---|---|
| 2 | 0 = The value in the RTT_Random_Sequence_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_Random_Sequence_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is applicable to all mandatory and optional PHYs.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 1M PHY only. |
| 3 | 0 = The value in the RTT_2M_AA_Only_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_2M_AA_Only_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is RFU.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 2M and LE 2M 2BT PHYs only. |
| 4 | 0 = The value in the RTT_2M_Sounding_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_2M_Sounding_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is RFU.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 2M and LE 2M 2BT PHYs only. |
| 5 | 0 = The value in the RTT_2M_Random_Sequence_N field refers to the “N” value that meets the 150 ns accuracy requirement.<br>1 = The value in the RTT_2M_Random_Sequence_N field refers to the “N” value that meets the 10 ns accuracy requirement.<br>For LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], the bit is RFU.<br>For LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2], the bit is applicable to the LE 2M and LE 2M 2BT PHYs only. |
| All other bits | Reserved for future use |

*Table 2.28: RTT_Capability field values*

A device that supports the RTT_Capability shall support the following section of this document:

- [Vol 6] Part H, Section 3.1

The IPT bit shall be set to 1 if the CS reflector supports the Inline PCT Transfer (IPT) (see [Vol 6] Part A, Section 6.5). It shall be set to 0 otherwise.

A device that supports the IPT capability shall support the following section of this document:

- Inline PCT Transfer ([Vol 6] Part A, Section 6.5)

The RTT_AA_Only_N, RTT_Sounding_N, and RTT_Random_Sequence_N fields shall hold the values specified by the product manufacturer to satisfy the accuracy requirements described in [Vol 6] Part H, Section 3.1.2. For the LL_CS_CAPABILITIES_REQ [v1] and LL_CS_CAPABILITIES_RSP [v1], if an implementation supports optional transmit and receive PHYs as indicated by the CS_SYNC_PHY_Capability field, then the corresponding field shall be set to the highest "N" value derived when tested across all mandatory and supported optional PHYs. If the RTT_PHY bit is set, then the corresponding field shall be set to the "N" value derived when tested on the LE 1M PHY. The value of RTT_AA_Only_N shall be non-zero. If an implementation does not support another RTT type, then the corresponding field shall be set to 0.

A device that supports any of the RTT types shall support the following sections of this document:

- [Vol 6] Part H, Section 2
- [Vol 6] Part H, Section 3.1

The NADM_Sounding_Capability and NADM_Random_Capability fields are bit-mapped fields that shall be set to indicate the Normalized Attack Detector Metric support described in [Vol 6] Part H, Section 3.5.1, that shall be applied by the local Controller whenever a CS_SYNC with a sounding sequence or random sequence is received, respectively. The meaning of each bit is specified in Table 2.29. A value of 0 indicates no NADM support availability for either of the two CS_SYNC Payload field types.

| Bit Number | Meaning |
|---|---|
| 0 | Phase-based NADM (see [Vol 6] Part H, Section 3.5.3) |
| 1 | Amplitude-based NADM (see [Vol 6] Part H, Section 3.5.4) |
| All other bits | Reserved for future use |

*Table 2.29: NADM_Sounding_Capability and NADM_Random_Capability field values*

A device that supports the NADM shall support the following section of this document:

- [Vol 6] Part H, Section 3.5.1

A device that supports phase-based NADM shall support the following section of this document.

- [Vol 6] Part H, Section 3.5.3

A device that supports the amplitude-based NADM shall support the following section of this document:

- [Vol 6] Part H, Section 3.5.4

The CS_SYNC_PHY_Capability field is a bit-mapped field that shall be set to indicate which of the optional transmit and receive PHYs are supported by the Link Layer transmitting this PDU for CS_SYNC exchanges during mode-0, mode-1, and mode-3 steps as described in [Vol 6] Part H, Section 4.3. The meaning of each bit is specified in Table 2.30.

| Bit Number | Meaning |
|---|---|
| 1 | LE 2M PHY |
| 2 | LE 2M 2BT PHY |
| All other bits | Reserved for future use |

*Table 2.30: CS_SYNC_PHY field values*

A device that supports the LE 2M PHY for CS_SYNC exchanges shall support the following section of this document:

- [Vol 6] Part A, Section 3.1

A device that supports the LE 2M 2BT PHY for CS_SYNC exchanges shall support the following sections of this document:

- [Vol 6] Part A, Section 3.1
- [Vol 6] Part A, Section 3.1.2

The Num_Ant field shall be set to the number of antenna elements that are available for CS tone exchanges. The Num_Ant field shall be set to a value between 1 and 4. Each antenna element reflected by this Num_Ant field is ordered by the local device from the first ordered antenna element to the Num_Ant ordered element. This ordering is implementation specific but should remain constant.

A device that supports a Num_Ant value greater than 1 shall support the following sections of this document:

- [Vol 6] Part A, Section 5.3
- [Vol 6] Part H, Section 4.7

The Max_Ant_Path field shall be set to the maximum number of antenna paths that are supported by the device for CS tone exchanges. The field shall be set to a value between 1 and 4 and shall be greater than or equal to the value set for the Num_Ant field.

A device that supports a Max_Ant_Path value greater than 1 shall support the following sections of this document:

- [Vol 6] Part A, Section 5.3
- [Vol 6] Part H, Section 4.7

The Role field shall be set to the CS role options that the Link Layer transmitting this PDU supports, as described in Section 5.1.25. The meaning of each bit is specified in Table 2.31. At least one bit shall be set in this field.

| Bit Number | Meaning |
|---|---|
| 0 | Initiator |
| 1 | Reflector |
| All other bits | Reserved for future use |

*Table 2.31: Role field values*

A device that supports either the initiator or reflector role shall support the following section of this document:

- [Vol 6] Part H

The No_FAE bit shall be set to 1 if the transmitting device supports only an FAE of 0 (see Section 2.4.2.52) for all allowed CS channels as specified in [Vol 6] Part H, Section 1, for the device's mode-0 transmissions when in the reflector role. Otherwise, the No_FAE bit shall be set to 0.

A device shall set the No_FAE bit in accordance with the requirements described in the following sections of this document:

- Section 2.4.2.52 and
- [Vol 6] Part A, Section 3.5.1

The Channel Selection Algorithm #3c bit shall be set to 1 if Channel Selection Algorithm #3c is supported.

A device that supports Channel Selection Algorithm #3c shall support the following section of this document:

- [Vol 6] Part H, Section 4.1.4.2

The Sounding_PCT_Estimate bit shall be set to 1 if PCT estimates from a sounding sequence are supported and RTT_Sounding_N is set to a value greater than 0.

A device that supports PCT estimates from a sounding sequence shall support the following section of this document:

- [Vol 6] Part H, Section 3.3.1

The Num_Configs field shall be set to the number of independent CS configurations supported by the Link Layer transmitting this PDU for this specific ACL connection. The Num_Configs field shall be set to a value greater than 0 and less than or equal to 4.

The Max_Procedures_Supported field shall be set to the maximum possible value supported by the Link Layer transmitting this PDU for N_PROCEDURE_COUNT value as described in Section 4.5.18.1. The Max_Procedures_Supported field shall be set to a value greater than or equal to 0, where 0 indicates the capability to run CS procedures repetitions until terminated.

The T_SW field shall be set to one of the valid values in units of microseconds, for the duration of the antenna switch period used by the local Controller, when antenna switching is performed during active transmissions as described in [Vol 6] Part H, Section 4.7.

The T_IP1_Capability field shall contain the supported optional durations for the T_IP1 parameter. Each supported duration is represented by a bit positioned according to the T_IP1 index defined in [Vol 6] Part H, Section 4.3.1. The LSB represents T_IP1 index 0, with the next adjacent bit represented by T_IP1 index 1, and so on. A bit value of 0 indicates that the specific T_IP1 duration is not supported. A bit value of 1 indicates that the specific T_IP1 duration is supported. Bit positions beyond the maximum T_IP1 index are reserved for future use.

The T_IP2_Capability field shall contain the supported optional durations for the T_IP2 parameter. Each supported duration is represented by a bit positioned according to the T_IP2 index defined in [Vol 6] Part H, Section 4.3.3. The LSB represents T_IP2 index 0, with the next adjacent bit represented by T_IP2 index 1, and so on. A bit value of 0 indicates that the specific T_IP2 duration is not supported. A bit value of 1 indicates that the specific T_IP2 duration is supported. Bit positions beyond the maximum T_IP2 index are reserved for future use.

The T_FCS_Capability field shall contain the supported optional durations for the T_FCS parameter. Each supported duration is represented by a bit positioned according to the T_FCS index defined in [Vol 6] Part H, Section 4.3. The LSB represents T_FCS index 0, with the next adjacent bit represented by T_FCS index 1, and so on. A bit value of 0 indicates that the specific T_FCS duration is not supported. A bit value of 1 indicates that the specific T_FCS duration is supported. Bit positions beyond the maximum T_FCS index are reserved for future use.

The T_PM_Capability field shall contain the supported optional durations for the T_PM parameter. Each supported duration is represented by a bit positioned according to the T_PM index defined in [Vol 6] Part H, Section 4.3.3. The LSB represents T_PM index 0, with the next adjacent bit represented by T_PM index 1, and so on. A bit value of 0 indicates that the specific T_PM duration is not supported. A bit value of 1 indicates that the specific T_PM duration is supported. Bit positions beyond the maximum T_PM index are reserved for future use.

The RTT_PHY bit may be set to 1 only if the Channel Sounding Enhancements #1 feature bit is set in the Controller. Otherwise the RTT_PHY bit shall be set to 0. Additionally, the RTT_PHY bit shall be set to 1 only if the controller supports optional transmit and receive PHYs as indicated by the CS_SYNC_PHY_Capability field and the controller also supports the setting of the RTT accuracy requirements described in [Vol 6] Part H, Section 3.1.2 for the optional LE 2M and LE 2M 2BT PHYs separate from the accuracy requirements for the mandatory LE 1M PHY.

A device that supports the RTT_PHY bit shall also support the LL_CS_CAPABILITIES_REQ [v2] and LL_CS_CAPABILITIES_RSP [v2] PDU.

The TX_SNR_Capability field shall contain the supported optional levels of TX SNR control for the transmission of CS_SYNC packets used in mode-1 and mode-3 steps. Each supported level is represented by a bit positioned according to the SNR Output Index defined in [Vol 6] Part A, Section 3.1.3. The LSB represents the SNR output index 0, with the next adjacent bit represented by SNR output index 1, and so on. A bit value of 0 indicates that the specific SNR output level is not supported. A bit value of 1 indicates that the specific SNR output level is supported. Bit positions beyond the maximum SNR output index are reserved for future use.

A device that supports TX_SNR_CAPABILITY shall support the following section of this document:

- [Vol 6] Part A, Section 3.1.3

The T_IP2_IPT_Capability field shall contain the supported optional durations for the T_IP2 parameter if the CS reflector supports the Channel Sounding Enhancement #1 feature and the IPT capability. Each supported duration is represented by a bit positioned according to the T_IP2 index defined in [Vol 6] Part H, Section 4.3.3. If IPT is

not supported in the reflector role, then the value of T_IP2_IPT_Capability shall be set to 0 and ignored by the receiver.

The T_SW_IPT field shall be set to one of the valid values, in units of microseconds, of the duration of the antenna switch period used by the local Controller in the reflector role. If IPT is not supported in the reflector role, then the value of T_SW_IPT shall be set to 0 and ignored by the receiver.

The RTT_2M_AA_Only_N, RTT_2M_Sounding_N, and RTT_2M_Random_-Sequence_N fields shall be set to 0 if the RTT_PHY bit is not set. Otherwise, the RTT_2M_AA_Only_N, RTT_2M_Sounding_N, and RTT_2M_Random_Sequence_N fields shall hold the values specified by the product manufacturer to satisfy the accuracy requirements described in [Vol 6] Part H, Section 3.1.2. If an implementation supports optional transmit and receive PHYs as indicated by the CS_SYNC_PHY_Capability field, then the corresponding field shall be set to the highest “N” value derived when tested across all supported optional LE 2M and LE 2M 2BT PHYs. In this case, the value of RTT_2M_AA_Only_N shall be non-zero. If an implementation does not support another RTT type, then the corresponding field shall be set to 0.

#### 2.4.2.45 LL_CS_CONFIG_REQ

The format of the CtrData field is shown in Figure 2.66.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| Config_ID (6 bits) | Action (2 bits) | ChM (10 octets) | ChM_-Repetition (1 octet) | Main_Mode (1 octet) | Sub_Mode (1 octet) | Main_-Mode_Min_-Steps (1 octet) |

| CtrData (continued) | | | | | | |
|---|---|---|---|---|---|---|
| Main_-Mode_-Max_Steps (1 octet) | Main_-Mode_-Repetition (1 octet) | Mode_0_-Steps (1 octet) | CS_-SYNC_-PHY (1 octet) | RTT_Type (4 bits) | Role (2 bits) | IPT (1 bit) |

| CtrData (continued) | | | | | | |
|---|---|---|---|---|---|---|
| RFU (1 bit) | ChSel (4 bits) | Ch3cShape (4 bits) | Ch3cJump (1 octet) | T_IP1 (1 octet) | T_IP2 (1 octet) | T_FCS (1 octet) |

| CtrData (continued) | |
|---|---|
| T_PM (1 octet) | RFU (1 octet) |

*Figure 2.66: CtrData field of the LL_CS_CONFIG_REQ PDU*

The Config_ID field shall be set to the specific CS configuration ID that is being configured, as described in Section 5.1.25. The meaning of each value is specified in Table 2.32.

| Value | Meaning |
|---|---|
| 0 to 3 | CS Configuration ID |
| All other values | Reserved for future use |

*Table 2.32: Config_ID values*

The Action field indicates whether or not this specific configuration ID is being created or removed, as described in Section 5.1.25. The meaning of each value is specified in Table 2.33. If the Action field is set to removed, then all fields except for the Config_ID and the Action fields in this PDU are RFU.

| Value | Meaning |
|---|---|
| 0b00 | Configuration is to be removed |
| 0b01 | Configuration is to be created |
| All other values | Reserved for future use |

*Table 2.33: Action field values*

The ChM field contains the channel map indicating which channels shall be used and unused within the CS procedure. Every channel is represented by a bit positioned according to the CS channel index defined in [Vol 6] Part H, Section 1. The LSB represents CS channel index 0 and bit 78 represents CS channel index 78. ChM bits beyond this index range are RFU. A bit value of 1 indicates that the channel shall be used. A bit value of 0 indicates that the channel shall not be used. At least 15 channels shall be marked as used. Some channels are not used in CS procedures, as described in [Vol 6] Part H, Section 1. These channels shall be marked as unused.

The value in the ChM_Repetition field shall indicate the number of times the ChM field is cycled through for non-mode-0 steps within a CS procedure. This value shall be greater than or equal to 1.

The Main_Mode and the Sub_Mode fields shall be set to indicate which CS modes are to be used within the CS procedure, as described in [Vol 6] Part H, Section 4.4. The meaning of each value is specified in Table 2.34.

| Value | Meaning |
|---|---|
| 0x01 | Mode-1 |
| 0x02 | Mode-2 |
| 0x03 | Mode-3 |
| 0xFF | None; applies only to Sub_Mode selection and RFU for Main_Mode selection |
| All other values | Reserved for future use |

*Table 2.34: Main_Mode and Sub_Mode field values*

The Main_Mode_Min_Steps and Main_Mode_Max_Steps fields shall be set to the range of Main_Mode steps to be executed before a Sub_Mode step is executed within the CS procedure, as described in [Vol 6] Part H, Section 4.4. If the Sub_Mode field has been set to None, then these two fields shall be reserved for future use. Otherwise, Main_Mode_Min_Steps shall be greater than or equal to 1 and less than or equal to N_STEPS_MAX - 1. Main_Mode_Max_Steps shall be greater than or equal to Main_Mode_Min_Steps and less than or equal to N_STEPS_MAX - 1.

The Main_Mode_Repetition field shall be set to the number of Main_Mode steps taken from the end of the last CS subevent to be repeated at the beginning of the next CS subevent directly after the last mode-0 step of that subevent, as described in [Vol 6] Part H, Section 4.4. This field shall be set with a value from 0 to 3.

The Mode_0_Steps field shall be set to the number of mode-0 steps to be included at the beginning of each CS Subevent, as described in [Vol 6] Part H, Section 4.4. Mode_0_Steps shall be greater than or equal to 1 and less than or equal to 3.

The CS_SYNC_PHY field shall be set to the transmit and receive PHY to be used for CS_SYNC exchanges during mode-0, mode-1, and mode-3 steps, as described in [Vol 6] Part H, Section 4.3. The bit corresponding to that PHY, as specified in Table 2.35, shall be set to 1 and the remaining bits shall be set to 0.

| Bit Number | Meaning |
|---|---|
| 0 | LE 1M PHY |
| 1 | LE 2M PHY |
| 2 | LE Coded PHY |
| 3 | LE 2M 2BT PHY |
| All other bits | Reserved for future use |

*Table 2.35: CS_SYNC_PHY field bit meanings*

The RTT_Type field shall be set to indicate which RTT variant is to be used within the CS procedure, as described in [Vol 6] Part H, Section 2. The meaning of each value is specified in Table 2.36.

| Value | Meaning |
|---|---|
| 0x00 | RTT CS AA-only timing |
| 0x01 | RTT with 32-bit sounding sequence |
| 0x02 | RTT with 96-bit sounding sequence |
| 0x03 | RTT with 32-bit random sequence |
| 0x04 | RTT with 64-bit random sequence |
| 0x05 | RTT with 96-bit random sequence |
| 0x06 | RTT with 128-bit random sequence |
| All other values | Reserved for future use |

*Table 2.36: RTT_Type field values*

The Role field shall be set to the CS role that the Link Layer transmitting this PDU shall use, as described in Section 5.1.25. The meaning of each value is specified in Table 2.37.

| Value | Meaning |
|---|---|
| 0b00 | Initiator |
| 0b01 | Reflector |
| All other values | Reserved for future use |

*Table 2.37: Role field values*

The IPT bit shall be set to 1 to enable the IPT in the CS reflector and 0 otherwise.

The ChSel field shall be set to the channel selection algorithm to be used within the CS procedure for non-mode-0 steps, as described in [Vol 6] Part H, Section 4.1. The meaning of each value is specified in Table 2.38.

| Value | Meaning |
|---|---|
| 0b0000 | Channel Selection Algorithm #3b |
| 0b0001 | Channel Selection Algorithm #3c |
| All other values | Reserved for future use |

*Table 2.38: ChSel field values*

The Ch3cShape field shall be set to the selected shape to be rendered when Channel Selection Algorithm #3c is used, as described in [Vol 6] Part H, Section 4.1.4.2. The meaning of each value is specified in Table 2.39. The Ch3cShape field is valid only if the ChSel field has been set to Channel Selection Algorithm #3c; otherwise, the field is reserved for future use.

| Value | Meaning |
|---|---|
| 0x0 | Hat shape |
| 0x1 | “X” shape |
| All other values | Reserved for future use |

*Table 2.39: Ch3cShape field values*

The Ch3cJump field shall be set to one of the valid CSChannelJump values defined in [Vol 6] Part H, Table 4.2. This field is valid only if the ChSel field has been set to Channel Selection Algorithm #3c; otherwise, the field is reserved for future use.

The T_IP1 field shall be set to the T_IP1 index as defined in [Vol 6] Part H, Section 4.3.1, which shows the selected duration of the interlude period used between CS_SYNC packets used in mode-0 and mode-1 steps.

The T_IP2 field shall be set to the T_IP2 index as defined in [Vol 6] Part H, Section 4.3.3, which shows the selected duration of the interlude period used between CS tones.

The T_FCS field shall be set to the T_FCS index as defined in [Vol 6] Part H, Table 4.3, which shows the selected duration used for frequency changes.

The T_PM field shall be set to the T_PM index as defined in [Vol 6] Part H, Section 4.3.3, which shows the selected duration used for the phase measurement period of CS tones.

### 2.4.2.46 LL_CS_CONFIG_RSP

The format of the CtrData field is shown in Figure 2.67.

| CtrData | |
|---|---|
| Config_ID (6 bits) | RFU (2 bits) |

*Figure 2.67: CtrData field of the LL_CS_CONFIG_RSP PDU*

The Config_ID field has the same meaning as the Config_ID field in the LL_CS_CONFIG_REQ PDU (see Section 2.4.2.45).

### 2.4.2.47 LL_CS_REQ

The format of the CtrData field is shown in Figure 2.68.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| Config_ID (6 bits) | RFU (2 bits) | connEventCount (2 octets) | Offset_Min (3 octets) | Offset_Max (3 octets) | Max_Procedure_Len (2 octets) | Event_Interval (2 octets) |

| CtrData (continued) | | | | | | |
|---|---|---|---|---|---|---|
| Subevents_Per_Event (1 octet) | Subevent_Interval (2 octets) | Subevent_Len (3 octets) | Procedure_Interval (2 octets) | Procedure_Count (2 octets) | ACI (1 octet) | Preferred_Peer_Ant (1 octet) |

| CtrData (continued) | | | |
|---|---|---|---|
| PHY (1 octet) | Pwr_Delta (1 octet) | TX_SNR_I (4 bits) | TX_SNR_R (4 bits) |

*Figure 2.68: CtrData field of the LL_CS_REQ PDU*

The Config_ID field shall be used to select a created CS procedure configuration parameter set, as described in Section 5.1.25.

The connEventCount field shall be set to a connection event counter value that meets the requirement *currEvent* – $2^{14}$ < *connEventCount* < *currEvent* + $2^{14}$ (***mod*** 65536),

where *currEvent* is the counter value for the connection event in which the PDU containing this field is being transmitted or retransmitted. The connEventCount field should be set to a value greater than the *currEvent* value of the event in which the LL_CS_REQ is first transmitted.

The Offset_Min field shall be set to the proposed minimum time, in microseconds, from the ACL anchor point of the connection event that is referenced by connEventCount to the start of the first CS subevent. The Offset_Min value shall be greater than or equal to 500 µs and less than 4 seconds.

The Offset_Max field shall be set to the proposed maximum time, in microseconds, between the ACL anchor point of the connection event from which the first CS step of the first CS subevent is offset. The value shall be greater than or equal to the Offset_Min value and shall be less than the LE connection interval.

The Max_Procedure_Len field shall be set to the proposed maximum extent of the entire CS procedure in units of 625 microseconds. This value is equivalent to the time extent from the beginning of the transmission of the first CS step to the end of the transmission of the final CS step.

The Event_Interval field shall be set to time, in units of connection intervals between the start of two consecutive CS events that are offset directly from ACL anchor points as described in Section 5.1.26. This value shall be greater than or equal to 1 and less than or equal to 65535.

The Subevents_Per_Event field shall be set to indicate the number of CS subevents that are to be included in each CS event. This value shall be greater than or equal to 1 and is limited by the maximum number of subevents allowed in a CS procedure as described in Section 4.5.18.1. The use of this parameter is further described in Section 5.1.26.

The Subevent_Interval field shall be set to indicate the gap, in units of 625 microseconds, between the start of two consecutive CS subevents that are anchored starting from the same ACL anchor point. The Subevent_Interval field is valid only if Subevents_Per_Event > 1, otherwise this field shall be set to 0. This field is further defined in Section 5.1.26.

The Subevent_Len field shall be set to indicate the maximum duration of each CS subevent in microseconds and shall be greater than or equal to 1250 microseconds and less than 4 seconds. The Subevent_Len is further defined in Section 5.1.26.

The Procedure_Interval field shall be set to indicate the time in units of connection intervals between the start of consecutive CS procedures. The Procedure_Interval field shall be set to a value from 0 to 65535. This value shall be set to 0 if the procedure is only to be run once. Otherwise the time span resulting from the Procedure_Interval

value shall be greater than the time span denoted by Max_Procedure_Len added to Offset_Max and T_MES.

The Procedure_Count field shall be set to indicate the number of consecutive CS procedures to invoke. The Procedure_Count field shall be set to a value from 0 to 65535. A value of 0 indicates that CS procedures shall continue to be invoked until terminated, as described in Section 5.1.27.

The ACI field shall indicate the preferred ACI to use in the CS procedure, as described in [Vol 6] Part A, Section 5.3, where device A represents the initiator.

The Preferred_Peer_Ant field is a bit-mapped field that shall indicate the preferred peer-ordered antenna elements to be used by the peer for the antenna configuration denoted by the ACI field. The meaning of each bit is specified in Table 2.40.

| Bit Number | Meaning |
|---|---|
| 0 | Use first ordered antenna element |
| 1 | Use second ordered antenna element |
| 2 | Use third ordered antenna element |
| 3 | Use fourth ordered antenna element |
| All other bits | Reserved for future use |

*Table 2.40: Preferred_Peer_Ant field values*

The PHY field shall be set to indicate the remote device's Tx PHY, to which the Pwr_Delta field in this PDU applies. The bit corresponding to that PHY as specified in Table 2.25 shall be set to 1 and the remaining bits shall be set to 0.

The Pwr_Delta field shall represent the difference between the remote device's transmit power level for the CS tones and CS_SYNC packets, and the transmit power level for the PHY indicated by the PHY field. The value in the Pwr_Delta field is a signed integer in dB, with a positive value indicating a higher transmit power level for the CS tones and CS_SYNC packets, and a negative value indicating a lower transmit power level for the CS tones and CS_SYNC packets, compared to the transmit power level for the PHY indicated by the PHY field. A Pwr_Delta value of 0x00 indicates that the two transmit power levels are the same.

The TX_SNR_I field shall be set to indicate the SNR output index to be used by the initiator in the transmission of all CS_SYNC packets used in mode-1 and mode-3 steps, as described in [Vol 6] Part A, Section 3.1.3. This field shall be set to 0xF if SNR control is not to be used or not supported by the Channel Sounding initiator device.

The TX_SNR_R field shall be set to indicate the SNR output index to be used by the reflector in the transmission of all CS_SYNC packets used in mode-1 and mode-3

steps, as described in [Vol 6] Part A, Section 3.1.3. This field shall be set to 0xF if SNR control is not to be used or not supported by the Channel Sounding reflector device.

#### 2.4.2.48 LL_CS_RSP

The format of the CtrData field is shown in Figure 2.69.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| Config_ID (6 bits) | RFU (2 bits) | connEventCount (2 octets) | Offset_Min (3 octets) | Offset_Max (3 octets) | Event_Interval (2 octets) | Subevents_Per_-Event (1 octet) |

| CtrData (continued) | | | | | |
|---|---|---|---|---|---|
| Subevent_Interval (2 octets) | Subevent_Len (3 octets) | ACI (1 octet) | PHY (1 octet) | Pwr_Delta (1 octet) | RFU (1 octet) |

*Figure 2.69: CtrData field of the LL_CS_RSP PDU*

The LL_CS_RSP CtrData fields have the same meaning as the fields in the LL_CS_REQ PDU CtrData field. The values of each field are selected as described in Section 5.1.26.

#### 2.4.2.49 LL_CS_IND

The format of the CtrData field is shown in Figure 2.70.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| Config_ID (6 bits) | RFU (2 bits) | connEventCount (2 octets) | Offset (3 octets) | Event_Interval (2 octets) | Subevents_Per_-Event (1 octet) | Subevent_Interval (2 octets) |

| CtrData (continued) | | | | |
|---|---|---|---|---|
| Subevent_Len (3 octets) | ACI (1 octet) | PHY (1 octet) | Pwr_Delta (1 octet) | RFU (1 octet) |

*Figure 2.70: CtrData field of the LL_CS_IND PDU*

The Config_ID, Event_Interval, Subevents_Per_Event, Subevent_Interval, Subevent_Len, ACI, PHY, and Pwr_Delta fields have the same meaning as the fields in the LL_CS_REQ PDU CtrData field.

The connEventCount field shall be set to a connection event counter value that meets the requirement $currEvent - 2^{14} < connEventCount < currEvent + 2^{14}$ ($mod$ 65536), where *currEvent* is the counter value for the connection event in which the PDU containing this field is being transmitted or retransmitted. The *connEventCount* value should be set to a value greater than the *currEvent* value of the event in which the LL_CS_IND is first transmitted.

The Offset field shall be set to the time, in microseconds, from the ACL anchor point of the connection event that is referenced by connEventCount to the start of the first CS subevent.

#### 2.4.2.50 LL_CS_TERMINATE_REQ and LL_CS_TERMINATE_RSP

The format of the CtrData field is shown in Figure 2.71.

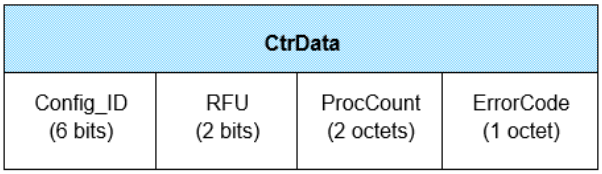



*Figure 2.71: CtrData field of the LL_CS_TERMINATE_REQ and LL_CS_TERMINATE_RSP PDU*

The Config_ID field shall be used to identify the CS procedure configuration parameter that was used to start the CS procedure repeat that is to be terminated.

The ProcCount field shall be set to the value of CSProcCount at the time this PDU is transmitted.

The ErrorCode field shall be set to inform the peer why the CS procedure repeat is about to be terminated (see [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions).

#### 2.4.2.51 LL_CS_FAE_REQ

The LL_CS_FAE_REQ PDU does not have a CtrData field.

#### 2.4.2.52 LL_CS_FAE_RSP

The format of the CtrData field is shown in Figure 2.72.

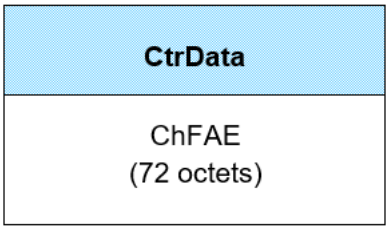



*Figure 2.72: CtrData field of the LL_CS_FAE_RSP PDU*

The ChFAE field contains the per-channel mode-0 FAE table of the local Controller. Every per-channel mode-0 FAE value is represented by an 8-bit signed integer, positioned per the CS channel index table in [Vol 6] Part H, Section 1. Only the allowed channels are represented, shifted lower to fill in the indices left void by the channel indices that are not allowed. Each FAE value spans the [-4, +3.96875] ppm range, with a resolution of 0.03125 ppm. A value of -128 then represents -4 ppm and a value of 127 represents 3.96875 ppm. Each FAE value is multiplied by 32 and then rounded to the nearest integer value within the signed integer range of [-128, 127]. A value of 0 then represents a range of -0.015625 ppm to 0.015625 ppm. The FAE is defined in [Vol 6] Part A, Section 3.5.1.

#### 2.4.2.53 LL_CS_CHANNEL_MAP_IND

The format of the CtrData field is shown in Figure 2.73.

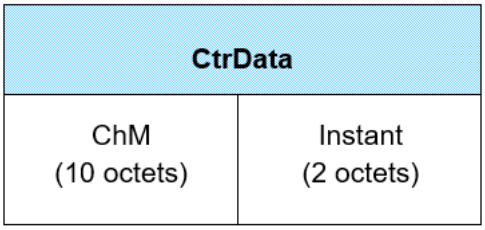



*Figure 2.73: CtrData field of the LL_CS_CHANNEL_MAP_IND PDU*

The ChM field contains the channel map indicating which channels shall be used and unused within the CS procedure. Every channel is represented by a bit positioned according to the CS channel index in [Vol 6] Part H, Section 1. The format of the field is identical to the ChM field in the LL_CS_CONFIG_REQ PDU (see Section 2.4.2.45).

The Instant field shall be set to a connection event counter value that meets the requirement $currEvent - 2^{14} < Instant < currEvent + 2^{14}$ (*mod* 65536), where *currEvent* is the counter value for the connection event in which the PDU containing this field is being transmitted or retransmitted. The Instant field should be set to a value greater

than the *currEvent* value of the event in which the LL_CS_CHANNEL_MAP_IND is first transmitted.

#### 2.4.2.54 LL_FRAME_SPACE_REQ

The format of the CtrData field is shown in Figure 2.74.

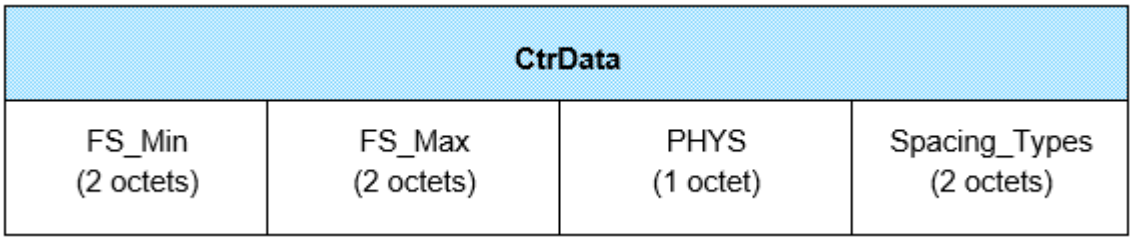



*Figure 2.74: CtrData field of the LL_FRAME_SPACE_REQ*

The FS_Min field shall be set to the minimum frame space value being requested by the Controller, in microseconds.

The FS_Max field shall be set to the maximum frame space value being requested by the Controller, in microseconds. The value shall be greater than or equal to FS_Min and shall not be greater than 10 ms.

The PHYS field shall be set to indicate the PHYs for which the request is being made. The bit corresponding to each PHY for which the request is being made (as specified in Table 2.23) shall be set to 1, and the remaining bits shall be set to 0. At least one bit shall be set to 1.

The Spacing_Types field shall be set to indicate the Frame Space (as defined in Section 4.1) for which the request is being made. The bit corresponding to each frame spacing type for which the request is being made (as specified in Table 2.41) shall be set to 1, and the remaining bits shall be set to 0. At least one bit shall be set to 1.

| Bit Number | Meaning |
|---|---|
| 0 | T_IFS_ACL_CP |
| 1 | T_IFS_ACL_PC |
| 2 | T_MCES |
| 3 | T_IFS_CIS |
| 4 | T_MSS_CIS |
| All other bits | Reserved for future use |

*Table 2.41: Spacing type*

#### 2.4.2.55 LL_FRAME_SPACE_RSP

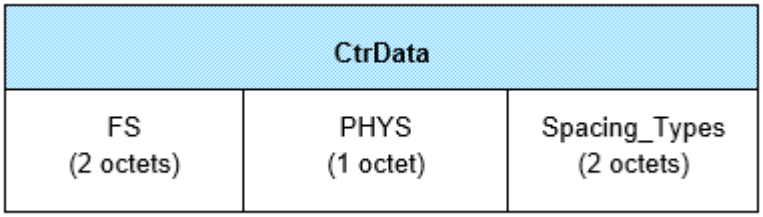



*Figure 2.75: CtrData field of the LL_FRAME_SPACE_RSP*

The FS field shall be set to the frame space value selected by the Controller, in microseconds.

The PHYS field indicates the PHYs for which the response is being sent. The bit corresponding to each PHY for which the response is being made (as specified in Table 2.23) shall be set to 1, and the remaining bits shall be set to 0. If all the frame space values for a PHY will remain unchanged, then the corresponding field shall be set to 0.

The Spacing_Types field indicates the spacing types for which the response is to be set. The bit corresponding to each spacing type for which the response is to be set (as specified in Table 2.41) shall be set to 1, and the remaining bits shall be set to 0. If all the frame space values for a spacing type will remain unchanged, then the corresponding field shall be set to 0.

#### 2.4.2.56 LL_OTA_UTP_IND

The format of the CtrData field is shown in Figure 2.76.

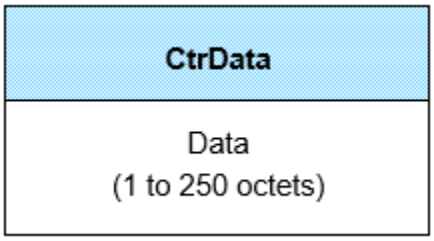



*Figure 2.76: CtrData field of the LL_OTA_UTP_IND PDU*

The Data field shall be set to UTP data (see [Vol 6] Part F, Section 5). This is not required to contain complete UTP messages.

#### 2.4.2.57 LL_CONNECTION_RATE_REQ

The format of the CtrData field is shown in Figure 2.77.

| CtrData | | | | | |
|---|---|---|---|---|---|
| Interval_Min (2 octets) | Interval_Max (2 octets) | SubrateFactor-Min (2 octets) | SubrateFactor-Max (2 octets) | Max_Latency (2 octets) | Continuation-Number (2 octets) |

| CtrData (continued) | | | | | | |
|---|---|---|---|---|---|---|
| Timeout (2 octets) | Preferred-Periodicity (2 octets) | Reference-ConnEventCount (2 octets) | Offset0 (2 octets) | Offset1 (2 octets) | Offset2 (2 octets) | Offset3 (2 octets) |

*Figure 2.77: CtrData field of the LL_CONNECTION_RATE_REQ PDU*

The Interval_Min field shall be set to indicate the minimum value of *connInterval*, as defined in Section 4.5.1, in the following manner:

*connInterval* ≥ Interval_Min × 125 µs.

The Interval_Max field shall be set to indicate the maximum value of *connInterval*, as defined in Section 4.5.1, in the following manner:

*connInterval* ≤ Interval_Max × 125 µs.

The value of Interval_Max shall be greater than or equal to Interval_Min.

The SubrateFactorMin field shall be set to the minimum requested *connSubrateFactor*, as defined in Section 4.5.1.

The SubrateFactorMax field shall be set to the maximum requested *connSubrateFactor*, as defined in Section 4.5.1.

The Max_Latency field shall be set to the maximum requested *connPeripheralLatency*, as defined in Section 4.5.1, in subrated events. The same maximum shall apply irrespective of the subrating factor chosen.

The value of SubrateFactorMax × (Max_Latency + 1) shall be less than or equal to 500.

The ContinuationNumber field shall be set to the minimum requested *connContinuationNumber*, as defined in Section 4.5.1.

The Timeout field shall be set to indicate the requested *connSupervisionTimeout* value, as defined in Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

The PreferredPeriodicity field shall be set to indicate a preference of the *connInterval* value to a subset of the *connInterval* supported range. The preferred range is expressed

by setting PreferredPeriodicity to be a multiple of the *connInterval*. PreferredPeriodicity is in units of 125 µs. For example, if the PreferredPeriodicity field is set to 5, *connInterval* should be set to any multiple of 625 µs. A PreferredPeriodicity field value of zero indicates no preference for the value of *connInterval*. The PreferredPeriodicity field shall be less than or equal to Interval_Max.

The ReferenceConnEventCount field shall be set to indicate the value of the *connEventCounter* relative to the proposed anchor point position.

Note: The ReferenceConnEventCount field is independent of the Instant field in the LL_CONNECTION_RATE_IND PDU.

The Offset0, Offset1, Offset2, and Offset3 fields shall be set to determine the possible proposed values of the position of the anchor points of the LE connection with the updated connection parameters relative to the ReferenceConnEventCount. The Offset0 to Offset3 fields are in units of 125 µs and are in decreasing order of preference; i.e., Offset0 is the most preferred value, followed by Offset1, and so on. Offset0 to Offset3 shall be less than Interval_Max. A value of 0xFFFF means not valid. Valid Offset0 to Offset3 fields shall contain unique values. Valid offset fields shall appear in the PDU before invalid offset fields.

#### 2.4.2.58 LL_CONNECTION_RATE_IND

The format of the CtrData field is shown in Figure 2.78.

| CtrData | | | | | | |
|---|---|---|---|---|---|---|
| WinOffset (2 octets) | Interval (2 octets) | Instant (2 octets) | SubrateFactor (2 octets) | Latency (2 octets) | ContinuationNumber (2 octets) | Timeout (2 octets) |

*Figure 2.78: CtrData field of the LL_CONNECTION_RATE_IND PDU*

The WinOffset field shall be set to indicate the *transmitWindowOffset* value, as defined in Section 5.1.32, in the following manner:

*transmitWindowOffset* = WinOffset × 125 µs.

The value of the Interval field shall be set to indicate the *connInterval value*, as defined in Section 4.5.1, in the following manner:

*connInterval* = Interval × 125 µs.

The Instant field shall be set to indicate the instant described in Section 5.1.32.

The SubrateFactor field shall be set to the *connSubrateFactor*, as defined in Section 4.5.1, to be used on the connection.

The Latency field shall be set to indicate the *connPeripheralLatency* value, as defined in Section 4.5.1, in the following manner:

*connPeripheralLatency* = Latency.

The ContinuationNumber field shall be set to the *connContinuationNumber*, as defined in Section 4.5.1, to be used on the connection.

The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined in Section 4.5.2, in the following manner:

*connSupervisionTimeout* = Timeout × 10 ms.

## 2.5 Constant Tone Extension and IQ sampling

### 2.5.1 Constant Tone Extension structure and types

The Constant Tone Extension has a variable length; it shall be at least 16 µs and not greater than 160 µs. The contents are a constantly modulated series of 1s and no whitening shall be applied to them.

The first 4 µs of the Constant Tone Extension are termed the guard period and the next 8 µs are termed the reference period. After the reference period, the Constant Tone Extension consists of a sequence of alternating switch slots and sample slots, each either 1 µs or 2 µs long as specified by the Host. The 2-µs-long switch and sample slots are mandatory to support; the 1-µs-long switch and sample slots are optional to support. However, if a device supports 1-µs-long switch and sample slots, it shall support them on all supported PHYs that allow Constant Tone Extensions. The structure of the Constant Tone Extension is shown in Figure 2.79.

The Constant Tone Extension can be one of two types: AoA or AoD.

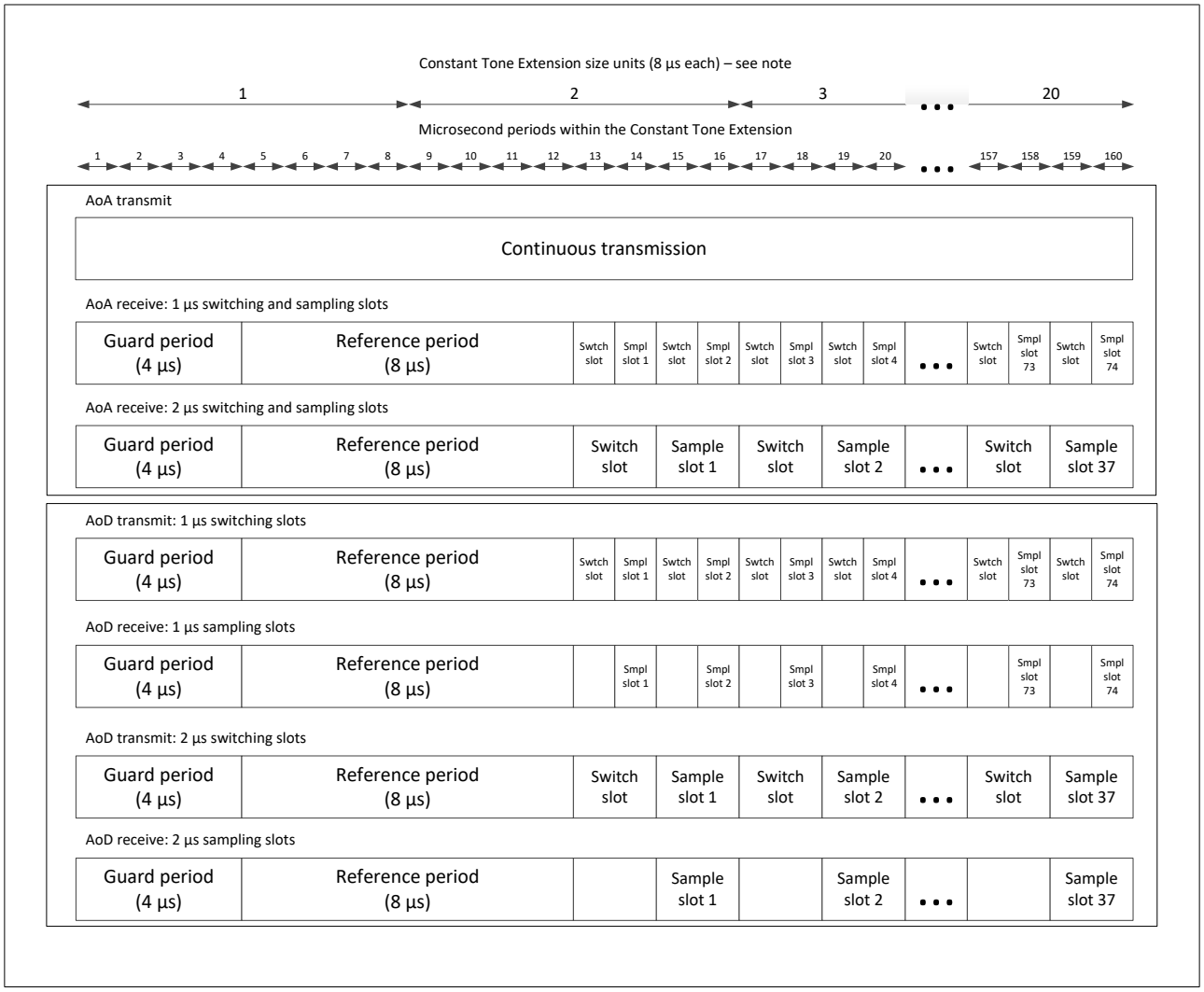



*Figure 2.79: Constant Tone Extension structure*

Note: See Section 2.4.2.25 for examples of the use of these units.

### 2.5.2 CTEInfo field

When present, the CTEInfo field is one octet with the format shown in Figure 2.80.

| CTEInfo | | |
|---|---|---|
| CTETime (5 bits) | RFU (1 bit) | CTEType (2 bits) |

*Figure 2.80: CTEInfo field*

The presence of the CTEInfo field indicates that the packet includes a Constant Tone Extension.

The CTETime field defines the length of the Constant Tone Extension in 8 µs units. The value of the CTETime field shall be between 2 and 20; all other values are reserved for future use.

The CTEType field defines the type of the Constant Tone Extension and the duration of the switching slots. The value of the field is specified in Table 2.42 and the various possible formats are specified in Section 2.5.1.

| CTEType value | Description |
|---|---|
| 0 | AoA Constant Tone Extension |
| 1 | AoD Constant Tone Extension with 1 µs slots |
| 2 | AoD Constant Tone Extension with 2 µs slots |
| 3 | Reserved for future use |

*Table 2.42: CTEType field encoding*

### 2.5.3 Transmitting Constant Tone Extensions

When transmitting a packet that contains an AoA Constant Tone Extension, the transmitter shall not switch antennae. When transmitting a packet that contains an AoD Constant Tone Extension, the transmitter shall perform antenna switching at the rate and according to the switching pattern configured by the Host (see [Vol 6] Part A, Section 5).

### 2.5.4 IQ sampling

When requested by the Host, the receiver shall perform IQ sampling when receiving a valid packet that contains a Constant Tone Extension and may perform IQ sampling when receiving a packet that contains a Constant Tone Extension but an incorrect CRC. The remainder of this section shall apply whenever the receiver performs IQ sampling on a packet.

When receiving a packet that contains an AoD Constant Tone Extension, the receiver does not need to switch antennae. When receiving a packet that contains an AoA Constant Tone Extension, the receiver shall perform antenna switching at the rate and according to the switching pattern configured by the Host (see [Vol 6] Part A, Section 5). In both cases, the receiver shall take an IQ sample each microsecond during the reference period and an IQ sample each sample slot (thus there will be 8 reference IQ samples, 1 to 37 IQ samples with 2 µs slots, and 2 to 74 IQ samples with 1 µs slots, meaning 9 to 82 samples in total). The Controller shall report the IQ samples to the Host. The receiver shall sample the entire Constant Tone Extension, irrespective of length, unless this conflicts with other activities.

In order to obtain good quality data for angle estimation, IQ samples should be taken at the same point within each IQ Sampling Window; this starts 0.125 µs after the beginning and ends 0.125 µs before the end of each microsecond period (see Figure 2.81). If 2 µs sample slots are used, the sampling should be done during the latter microsecond (see Figure 2.82).

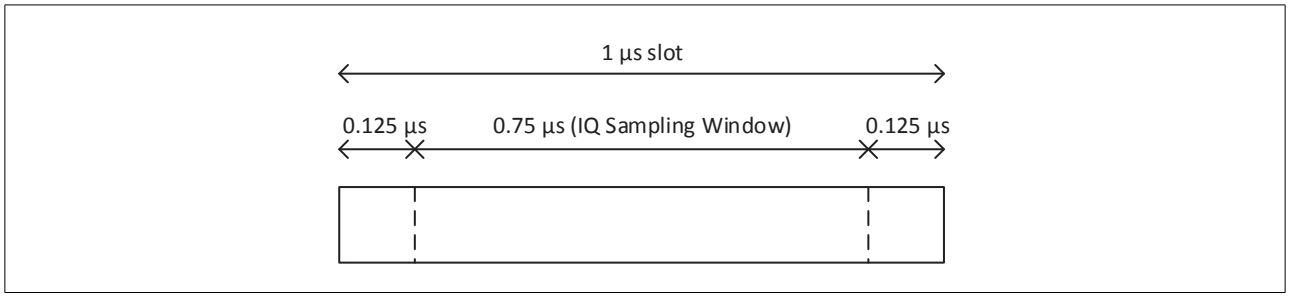



*Figure 2.81: IQ Sampling Window for 1 µs sample slots*

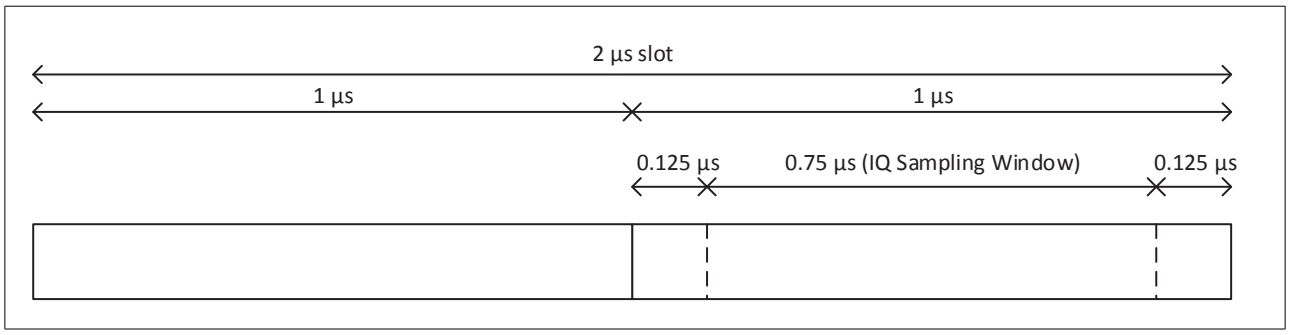



*Figure 2.82: IQ Sampling Window for 2 µs sample slots*

A Controller that supports IQ Sampling shall be able to measure the RSSI of received packets on any antenna used for receiving the body of the packet (in both cases ignoring any Constant Tone Extension) and be able to receive and sample a Constant Tone Extension of any valid length.

If the Controller has insufficient resources to perform sampling on all Constant Tone Extensions it receives, it may stop sampling after it has reported at least one set of IQ samples to the Host. If the Controller stops sampling, it shall report this to the Host and should resume sampling at the start of the next periodic advertising event or connection event. If the required resources are not yet available, then the Controller may, but should not, report this to the Host again.

## 2.6 Isochronous Physical Channel PDU

The Isochronous Physical Channel PDU has a 16-bit Header field, a variable size Payload field, and may include a Message Integrity Check (MIC) field.

The Isochronous Physical Channel PDU is shown in Figure 2.83.

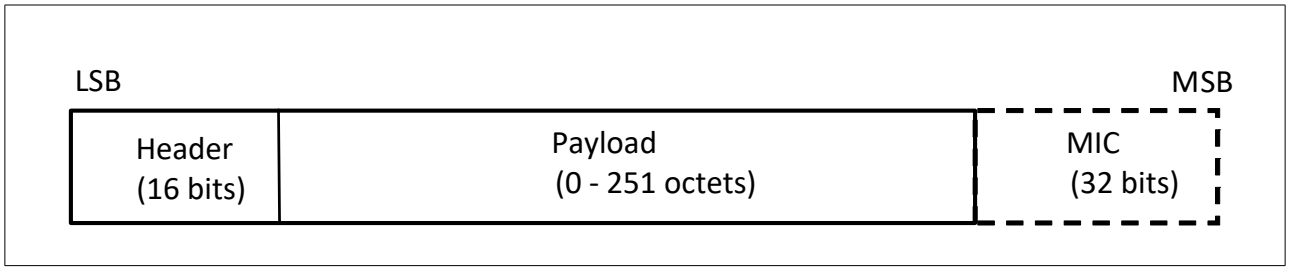



*Figure 2.83: Isochronous Physical Channel PDU*

The format of the Header field and the Payload field depends on the type of the Isochronous Physical Channel PDU that is being used. When used on a CIS, an Isochronous Physical Channel PDU shall be a Connected Isochronous PDU as defined in Section 2.6.1. When used on a BIS, it shall be a Broadcast Isochronous PDU as defined in Section 2.6.2.

The MIC field shall be included in all PDUs containing a non-zero length Payload field that are sent on an encrypted CIS or BIS. The MIC shall be calculated as specified in [Vol 6] Part E, Section 1. The MIC is a 4-octet field. The MIC field shall not be included in any PDU that is sent on an unencrypted CIS or BIS or that has a zero-length Payload field.

### 2.6.1 Connected Isochronous PDU

A Connected Isochronous PDU (CIS PDU) shall be either a CIS Data PDU or a CIS Null PDU. A CIS Data PDU is used to carry isochronous data. A CIS Null PDU is used when there is no data to be sent.

The Header field of the Connected Isochronous PDU is shown in Figure 2.84.

| Header | | | | | | | |
|---|---|---|---|---|---|---|---|
| LLID (2 bits) | NESN (1 bit) | SN (1 bit) | CIE (1 bit) | RFU (1 bit) | NPI (1 bit) | RFU (1 bit) | Length (8 bits) |

*Figure 2.84: Connected Isochronous PDU Header field*

The 16-bit Header field consists of the fields that are shown in Table 2.43.

- The LLID field is RFU in a CIS Null PDU.
- The Next Expected Sequence Number (NESN) field shall be set as defined in Section 4.5.9.
- The Sequence Number (SN) field shall be set as defined in Section 4.5.9 in a CIS Data PDU. This field is RFU in a CIS Null PDU.

- The Close Isochronous Event (CIE) field shall be set as defined in Section 4.5.13.4.
- The Null PDU Indicator (NPI) field shall be set for every CIS Null PDU.
- The Length field shall be set to 0 for a CIS Null PDU.

| Field Name | Description |
|---|---|
| LLID | The LLID field indicates the type of content of the Payload field of the CIS Data PDU.<br>0b00 = Unframed CIS Data PDU; end fragment of an SDU or a complete SDU.<br>0b01 = Unframed CIS Data PDU; start or continuation fragment of an SDU.<br>0b10 = Framed CIS Data PDU.<br>0b11 = Reserved for future use. |
| NESN | Next Expected Sequence Number |
| SN | Sequence Number |
| CIE | Close Isochronous Event |
| NPI | The Null PDU Indicator (NPI) indicates whether the packet is a CIS Data PDU or a CIS Null PDU. |
| Length | The Length field indicates the size, in octets, of the Payload field and MIC, if included. |

*Table 2.43: Connected Isochronous PDU Header field*

See [Vol 6] Part G, Section 1 for more details about fragmentation and segmentation of SDUs.

### 2.6.2 Broadcast Isochronous PDU

A Broadcast Isochronous PDU (BIS PDU) shall be either a BIS Data PDU or BIG Control PDU. A BIS Data PDU is used to carry isochronous data. A BIG Control PDU is used to send control information for a BIG.

The Header field of the Broadcast Isochronous PDU is shown in Figure 2.85.

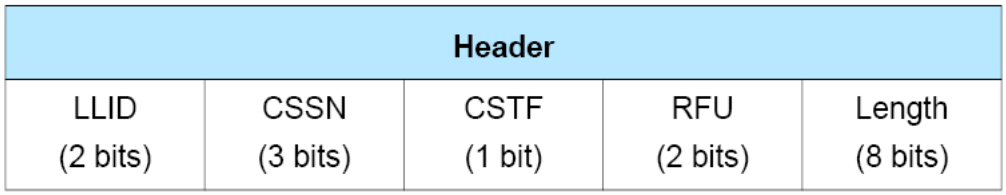



*Figure 2.85: Broadcast Isochronous PDU Header field*

The 16-bit Header field consists of the fields that are specified in Table 2.44.

The Control Subevent Sequence Number (CSSN) field shall be set as defined in Section 4.4.6.7.

The Control Subevent Transmission Flag (CSTF) field shall be set as defined in Section 4.4.6.7.

| Field Name | Description |
|---|---|
| LLID | The LLID field indicates the type of content of the Payload field of the PDU.<br>0b00 = Unframed BIS Data PDU; end fragment of an SDU or a complete SDU.<br>0b01 = Unframed BIS Data PDU; start or continuation fragment of an SDU.<br>0b10 = Framed BIS Data PDU.<br>0b11 = BIG Control PDU. |
| CSSN | Control Subevent Sequence Number |
| CSTF | Control Subevent Transmission Flag |
| Length | The Length field indicates the size, in octets, of the Payload field and MIC, if included. |

*Table 2.44: Broadcast Isochronous PDU Header field*

See [Vol 6] Part G, Section 1 for more details about fragmentation and segmentation of SDUs.

### 2.6.3 BIG Control PDU

A BIG Control PDU shall be used to send control information in a BIG (see Section 4.4.6.7).

The format of the Payload field in a BIG Control PDU is shown in Figure 2.86.

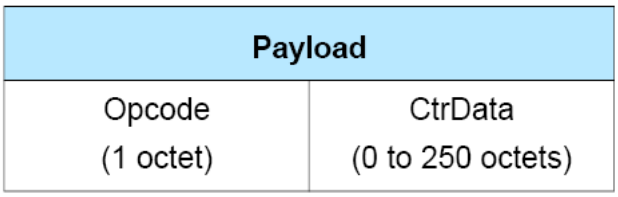



*Figure 2.86: Format of the Payload field of a BIG Control PDU*

A BIG Control PDU shall not have the Length field (see Section 2.6.2) set to 0b00000000.

The Opcode field identifies different types of BIG Control PDUs, as defined in Table 2.45.

The CtrData field in the BIG Control PDU is specified by the Opcode field and is defined in the following subsections. For a given Opcode the length of the CtrData field is fixed.

Where the description of a field within the CtrData field gives a range of valid values or other restrictions on a value (e.g. that field X is less than field Y), all other values

shall be reserved for future use. The range may be directly specified in the relevant subsection for the BIG Control PDU or indirectly, possibly in a section referenced by that subsection.

Except where explicitly stated otherwise, all fields within the CtrData field in a BIG Control PDU that hold an integer shall be interpreted as unsigned.

| Opcode | BIG Control PDU Name |
|---|---|
| 0x00 | BIG_CHANNEL_MAP_IND |
| 0x01 | BIG_TERMINATE_IND |
| 0xF8 to 0xFB | Reserved for specification development purposes |
| All other values | Reserved for future use |

*Table 2.45: BIG Control PDU opcodes*

If a BIG Control PDU is received that is not supported or is reserved for future use, the Link Layer shall ignore it.

If a BIG Control PDU is received with the wrong length or with invalid CtrData fields, the Link Layer may continue with the relevant BIG Control procedure with an implementation-specific interpretation of the data (e.g., if the PDU is too long, it can ignore the extra data; if a field is out of range, it can use the nearest permitted value). If it does not continue the procedure, it shall ignore the PDU.

#### 2.6.3.1 BIG_CHANNEL_MAP_IND

The format of the CtrData field is shown in Figure 2.87.

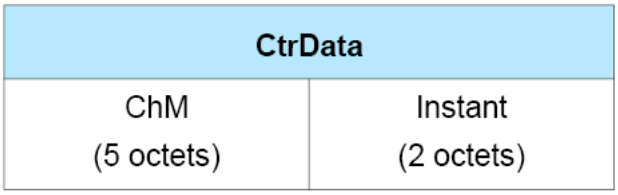



*Figure 2.87: CtrData field of the BIG_CHANNEL_MAP_IND PDU*

The BIG_CHANNEL_MAP_IND CtrData consists of two fields:

- The ChM field shall be set to the channel map indicating Used and Unused data channels. Every channel is represented with a bit positioned as per the channel index defined by Section 1.4.1. The format of this field is identical to the ChM field in the CONNECT_IND PDU (see Section 2.3.3.1).
- The Instant field shall be set to the value of $bigEventCounter_{15-0}$ (see Section 4.4.6.3) when the channel map takes effect.

#### 2.6.3.2 BIG_TERMINATE_IND

The format of the CtrData field is shown in Figure 2.88.

| CtrData | |
|---|---|
| Reason (1 octet) | Instant (2 octets) |

*Figure 2.88: CtrData field of the BIG_TERMINATE_IND PDU*

The BIG_TERMINATE_IND CtrData consists of two fields:

- The Reason field shall be set to inform the Synchronized Receiver(s) why the BIG is about to be terminated. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions.
- The Instant field shall be set to the value of $bigEventCounter_{15\text{-}0}$ (see Section 4.4.6.3) when the BIG will be terminated.

# 3 BIT STREAM PROCESSING

Bluetooth devices shall use the bit stream processing schemes as defined in the following sections.

Figure 3.1 shows the bit stream processing for PDUs on the LE Uncoded PHYs.

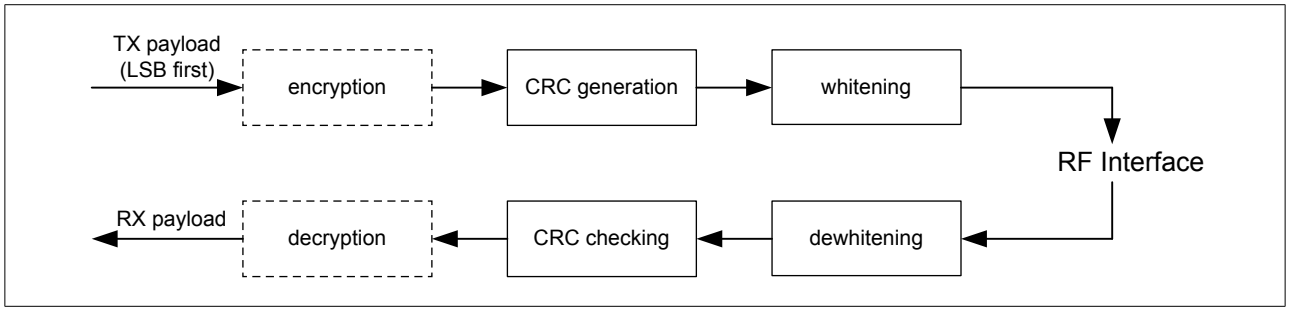



*Figure 3.1: Payload bit processes for the LE Uncoded PHYs*

Figure 3.2 shows the bit stream processing for PDUs on the LE Coded PHYs.

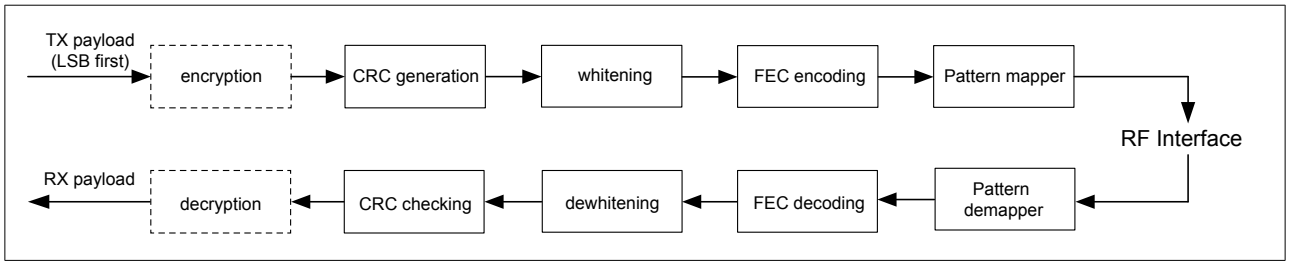



*Figure 3.2: Bit stream processing for the LE Coded PHYs*

## 3.1 Error checking

At packet reception, the Access Address shall be checked first. If the Access Address is incorrect, the packet shall be rejected, otherwise the packet shall be considered received. If the CRC is incorrect, the packet shall be rejected, otherwise the packet shall be considered successfully received and therefore valid. A packet shall only be processed if the packet is considered valid, except that the receiver may carry out IQ sampling (see Section 2.5.4) even if the CRC is incorrect. A packet with an incorrect CRC may cause a connection event to continue, as specified in Section 4.5.1.

### 3.1.1 CRC generation

The CRC shall be calculated on the PDU of all Link Layer packets. If the PDU is encrypted, then the CRC shall be calculated after encryption of the PDU has been performed.

The CRC polynomial is a 24-bit CRC and all bits in the PDU shall be processed in transmitted order starting from the least significant bit. The polynomial has the form of $x^{24} + x^{10} + x^9 + x^6 + x^4 + x^3 + x + 1$. For every Data Physical Channel PDU and Connected Isochronous PDU, the shift register shall be preset with the CRC initialization value set for the ACL connection and communicated in the CONNECT_IND or AUX_CONNECT_REQ PDU. For every PDU sent on a periodic advertising train (including AUX_CONNECT_REQ and AUX_CONNECT_RSP PDUs), the shift register shall be preset with the CRCInit value set in the SyncInfo field (see Section 2.3.4.6) that describes the periodic advertising train. For all other Advertising Physical Channel PDUs, the shift register shall be preset with 0x555555. For every Broadcast Isochronous PDU, the shift register shall be preset with the BaseCRCInit value from the BIGInfo data (see Section 4.4.6.11) in the most significant 2 octets and the BIS_Number for the specific BIS in the least significant octet. For BIG Control PDUs, the least significant octet shall be 0.

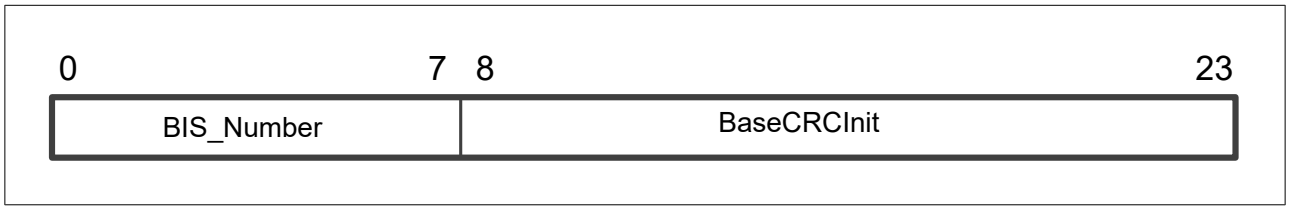



*Figure 3.3: Generation of CRCInit for Broadcast Isochronous PDUs*

Position 0 shall be set as the least significant bit and position 23 shall be set as the most significant bit of the initialization value. The CRC is transmitted most significant bit first, i.e. from position 23 to position 0 (see Section 1.2).

Figure 3.4 shows an example linear feedback shift register (LFSR) to generate the CRC.

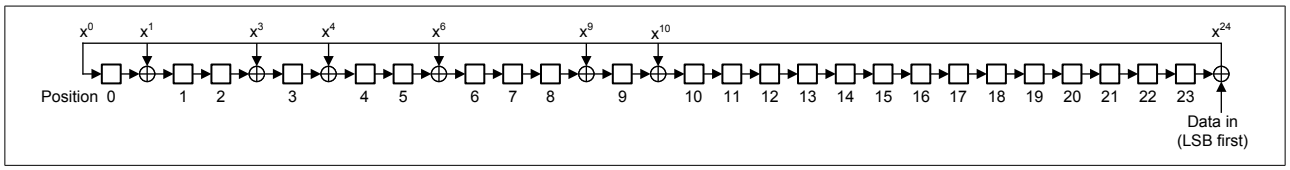



*Figure 3.4: The LFSR circuit generating the CRC*

## 3.2 Data whitening

Data whitening is used to avoid long sequences of zeros or ones, e.g., 0b0000000 or 0b1111111, in the data bit stream. Whitening shall be applied on the PDU and CRC of all Link Layer packets and is performed after the CRC generation in the transmitter. No other parts of the packets are whitened. De-whitening is performed before the CRC checking in the receiver (see Figure 3.1).

The whitener and de-whitener are defined the same way, using a 7-bit linear feedback shift register with the polynomial $x^7 + x^4 + 1$. Before whitening or de-whitening, the shift register is initialized with a sequence that is derived from the physical channel index in which the packet is transmitted in the following manner:

- Position 0 is set to one.
- Positions 1 to 6 are set to the channel index of the channel used when transmitting or receiving, from the most significant bit in position 1 to the least significant bit in position 6.

For example, if the channel index = 23 (0x17), the positions would be set as follows:

Position 0 = 1
Position 1 = 0
Position 2 = 1
Position 3 = 0
Position 4 = 1
Position 5 = 1
Position 6 = 1

Figure 3.5 shows an example linear feedback shift register (LFSR) to generate data whitening.

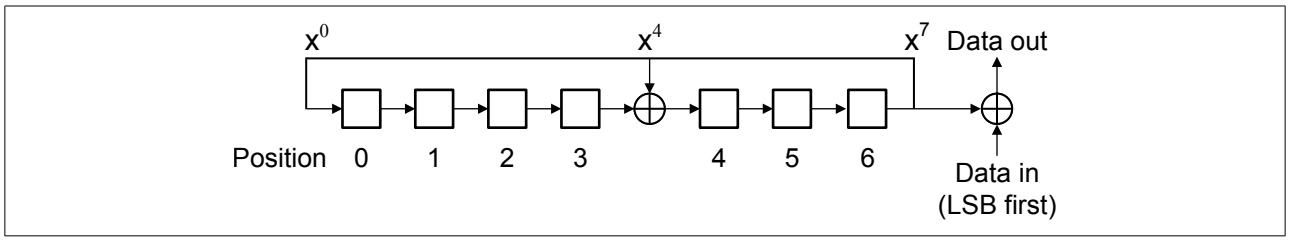



*Figure 3.5: The LFSR circuit to generate data whitening*

## 3.3 Coding

Coding only applies to the LE Coded PHY.

Coding consists of two processes. Data is first coded by the Forward Error Correction (FEC) convolutional encoder as defined in Section 3.3.1 and then spread by the pattern mapper as defined in Section 3.3.2.

### 3.3.1 Forward Error Correction encoder

The convolutional FEC encoder uses a non-systematic, non-recursive rate ½ code with constraint length K=4. The generator polynomials are:

$G_0(x) = 1 + x + x^2 + x^3$

$G_1(x) = 1 + x^2 + x^3$

The bit coming from generator polynomial $G_0$ ($a_0$) is transmitted first; the bit coming from generator polynomial $G_1$ ($a_1$) is transmitted second.

The initial state of the convolutional FEC encoder is set to all zeros. An input sequence of three consecutive zeros always brings the convolutional FEC encoder back to its original state. This sequence is known as the termination sequence.

Figure 3.6 illustrates operation of the convolutional FEC encoder. Squares represent bit storage operations and circles represent XOR.

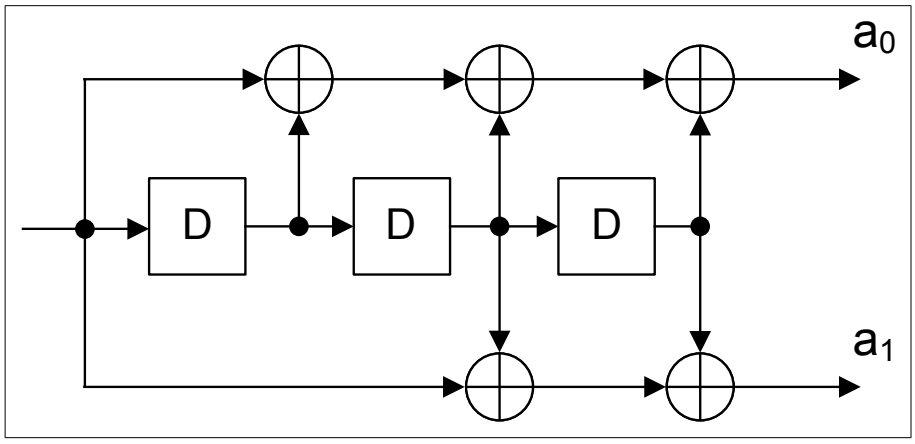



*Figure 3.6: Convolutional Forward Error Correction encoder*

### 3.3.2 Pattern mapper

The pattern mapper converts each bit from the convolutional FEC encoder into P symbols, where the value of P depends on the coding scheme in use, according to Table 3.1 (the output sequences are in transmission order):

| Input bit from the convolutional FEC encoder | Output sequence when P=1 (used by S=2) | Output sequence when P=4 (used by S=8) |
|---|---|---|
| 0 | 0 | 0011 |
| 1 | 1 | 1100 |

*Table 3.1: Pattern mapper inputs and outputs*

# 4 AIR INTERFACE PROTOCOL

The air interface protocol consists of the multiple access scheme, device discovery and Link Layer connection methods.

## 4.1 Frame Space

### 4.1.1 Inter Frame Space

The time interval between two consecutive packets on the same channel index is called the Inter Frame Space. It is defined as the time from the end of the last bit of the previous packet to the start of the first bit of the subsequent packet. The Inter Frame Space is designated “T_IFS” followed by a suffix to indicate its purpose. The different types of Inter Frame Space are:

- T_IFS_ACL_CP – the time between the end of a transmission by the Central and the following transmission by the Peripheral on an ACL. This value is selected based on the Peripheral’s transmission PHY (see Section 4.5.1).
- T_IFS_ACL_PC – the time between the end of a transmission by the Peripheral and the following transmission by the Central on an ACL. This value is selected based on the Central’s transmission PHY (see Section 4.5.1).
- T_IFS_CIS – the time between the end of a transmission by the Central and the following transmission by the Peripheral in a CIS sub-event. This value is selected based on the Peripheral’s transmission PHY (see Section 4.2).
- T_IFS_150 – various times that cannot be negotiated.

These values shall default to 150 µs but may be changed for a specific connection and specific PHYs on that connection, using the Frame Space Update procedure (see Section 5.1.30), except for T_IFS_150, which remains fixed.

### 4.1.2 Minimum AUX Frame Space

The minimum time interval between a packet containing an AuxPtr and the auxiliary packet it indicates is called the Minimum AUX Frame Space. It is defined as the minimum time from the end of the last bit of the packet containing the AuxPtr to the start of the auxiliary packet. The Minimum AUX Frame Space is designated “T_MAFS” and shall be 300 µs.

Figure 4.1 illustrates an example where the Minimum AUX Frame Space applies.

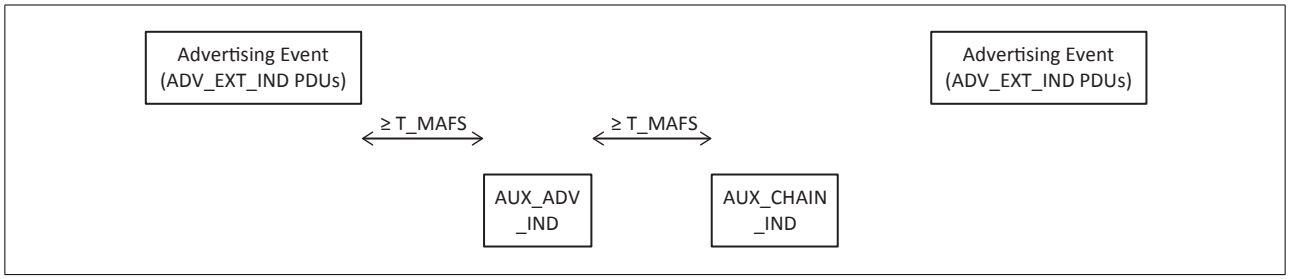



*Figure 4.1: Example where the Minimum AUX Frame Space applies*

### 4.1.3 Minimum Isochronous Channel Subevent Space

The minimum time interval between the end of the last bit of the last packet in one subevent and the start of the first bit of the first packet in the next subevent is called the Minimum Subevent Space.

The Minimum Subevent Space is designated "T_MSS_CIS" for CISes and “T_MSS_150” for BISes and shall default to 150 µs. The value of T_MSS_CIS is selected based on the Central’s transmission PHY (see Section 4.5.1). The value of T_MSS_CIS may be changed using the Frame Space Update procedure (see Section 5.1.30). The value of T_MSS_150 cannot be changed.

Figure 4.2 illustrates an example where the Minimum Subevent Space applies in a CIS.

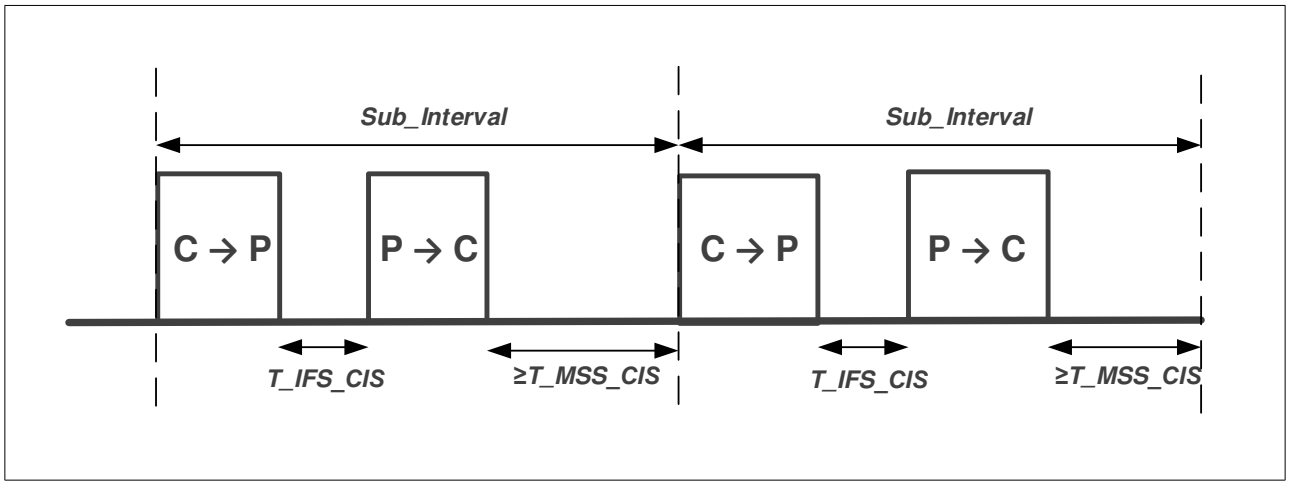



*Figure 4.2: Minimum Subevent Space in a CIS*

Figure 4.3 illustrates an example where the Minimum Subevent Space applies in a BIS.

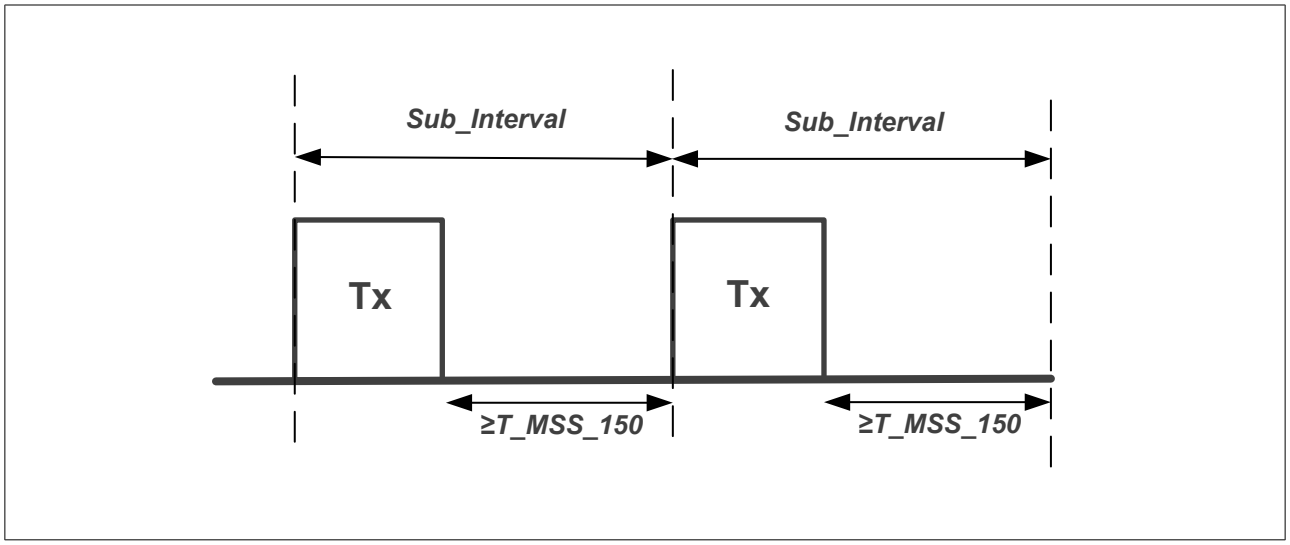



*Figure 4.3: Minimum Subevent Space in a BIS*

### 4.1.4 Minimum Channel Sounding subevent space

The minimum time interval between the end of a CS subevent and the start of the next CS subevent whose subevent length is defined by T_SUBEVENT_LEN and whose periodicity is defined by T_SUBEVENT_INTERVAL, as described in Section 4.5.18.1, is called the minimum subevent space.

The minimum subevent space is designated T_MES and shall be 150 µs.

Figure 4.4 shows an example in which the minimum subevent space applies between CS subevents.

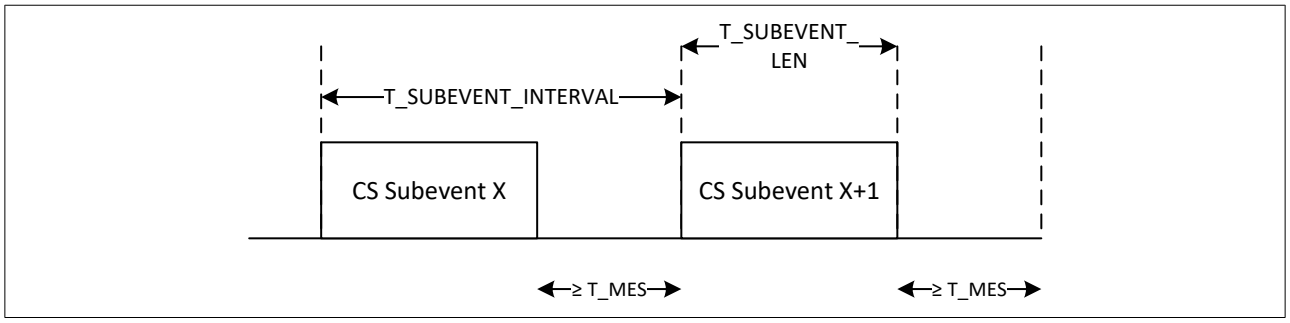



*Figure 4.4: Minimum subevent spacing in a CS procedure*

Similarly, T_MES shall also be the minimum time interval between the last CS subevent within a CS event and the first CS subevent of the next CS event, as well as the last CS subevent within the current CS procedure and the first CS subevent within the next CS procedure.

### 4.1.5 Minimum Connection Event Spacing

The minimum time interval between the last PDU sent from the peripheral and the anchor point of the next connection event is called the Minimum Connection Event Spacing. It is defined as the minimum time from the end of the last bit of the packet sent from the Peripheral to the anchor point of the next connection event.

The Minimum Connection Event Spacing is designated “T_MCES” and shall default to 150 µs. The value of T_MCES is selected based on the Central’s transmission PHY (see Section 4.5.1). The value of T_MCES may be changed using the Frame Space Update procedure (see Section 5.1.30).

## 4.2 Timing requirements

The Link Layer shall use one of two possible clock accuracies: in the circumstances described in Section 4.2.1 it shall use the active clock accuracy; otherwise it shall use the sleep clock accuracy.

The clock accuracies specified in this section apply to the time between two specific events and only apply to devices when they transmit a packet. The clock used to time packet reception may have any accuracy but the receiving device will need to allow for this. For example, if the receiving device clock has an accuracy of 2000 ppm and a maximum jitter of 45 µs, then, for an event 1 second after the last event where timings were synchronized, the device will need to start listening at least 2045 µs earlier and continue listening until at least 2045 µs later than it would otherwise have done.

An implementation may use either a single clock or several clocks. For example, one implementation could use a single clock with variable accuracy whereas a different implementation could use one clock that only runs during periods where an event timed using active clock accuracy is pending and another clock with lower accuracy that runs at all times.

### 4.2.1 Active clock accuracy

The average timing of packet transmission during a connection, BIG, or CIG event during active scanning, during a periodic advertising with responses subevent, and when requesting a connection is determined using the active clock accuracy, with a drift less than or equal to ±50 ppm. All instantaneous timings shall not deviate more than 2 µs from the average timing.

The average timing of CS tone transmission during a CS subevent is determined using the same active clock accuracy but using the instantaneous timing requirements specified in [Vol 6] Part H, Section 4.5.

More specifically, these requirements apply to the intervals between:

- adjacent packets in the same connection event
- packets in the same BIG or CIG event, even if they are in different BISes or CISes or in different subevents
- an advertising packet and a packet containing a SCAN_REQ, AUX_SCAN_REQ, CONNECT_IND, or AUX_CONNECT_REQ PDU
- a packet containing a SCAN_REQ and the response packet containing a SCAN_RSP PDU
- a packet containing an AUX_SCAN_REQ PDU and the response packet containing an AUX_SCAN_RSP PDU
- a packet containing an AUX_CONNECT_REQ PDU and the response packet containing an AUX_CONNECT_RSP PDU
- packets in the same periodic advertising with responses subevent
- the transmission and reception of the content of CS steps within a CS subevent.

### 4.2.2 Sleep clock accuracy

The average timing of all other activities is determined using the sleep clock accuracy, with a drift less than or equal to ±500 ppm. All instantaneous timings shall not deviate more than 16 µs from the average timing. The worst-case drift and instantaneous deviation of the active clock shall be less than or equal to those of the sleep clock.

In the specification, the Central's current sleep clock accuracy is referred to as *centralSCA* and the Peripheral's current sleep clock accuracy as *peripheralSCA*.

On a connection a device shall not use a sleep clock accuracy that is worse than the worst case indicated in the SCA field of the most recently sent LL_CLOCK_ACCURACY_REQ or LL_CLOCK_ACCURACY_RSP PDU. If the Link Layer has not initiated or responded to the Sleep Clock Accuracy Update procedure (see Section 5.1.14) in the current connection, the Central shall use a sleep clock accuracy that is better than or equal to the worst case indicated in the SCA field of the CONNECT_IND or AUX_CONNECT_REQ PDU used to create the connection and the Peripheral shall use a sleep clock accuracy of ±500 ppm or better.

Note: These requirements therefore apply to:

- The time between ACL anchor points (see Section 4.5.7).
- The time between CIS anchor points (see Section 4.5.14.1).
- The time between BIG anchor points (see Section 4.4.6.4).

- The time between CS subevents (see Section 4.5.18.1).
- The advertising and periodic advertising intervals and the advDelay value (see Section 4.4.2.2).
- All intervals between packets in the same extended advertising event or periodic advertising event.
- All offsets specified by the AuxPtr and SyncInfo fields of advertising PDUs.

Note: This means that a 2 s connection interval with a 500 ppm Central sleep clock accuracy will require a window widening either side of the anchor point of 1 ms plus 16 µs plus any allowance for the accuracy of the clock actually used by the Peripheral during the connection interval.

### 4.2.3 Range delay

Where two devices are more than a few meters apart the time taken for a signal to propagate between them will be significant compared with the Active Clock Accuracy defined in Section 4.2.1. When a device is listening for a packet that might be up to D meters away, it should listen for an extra 2D × 4 ns after the nominal latest time (e.g. T_IFS_ACL_CP + 2 µs) that the packet would have been transmitted.

(1÷c ~= 3.3 × refractive index ns/m, so 4 ns gives a conservative allowance.)

Figure 4.5 shows the range delays relative to a Central packet transmission in an ACL.

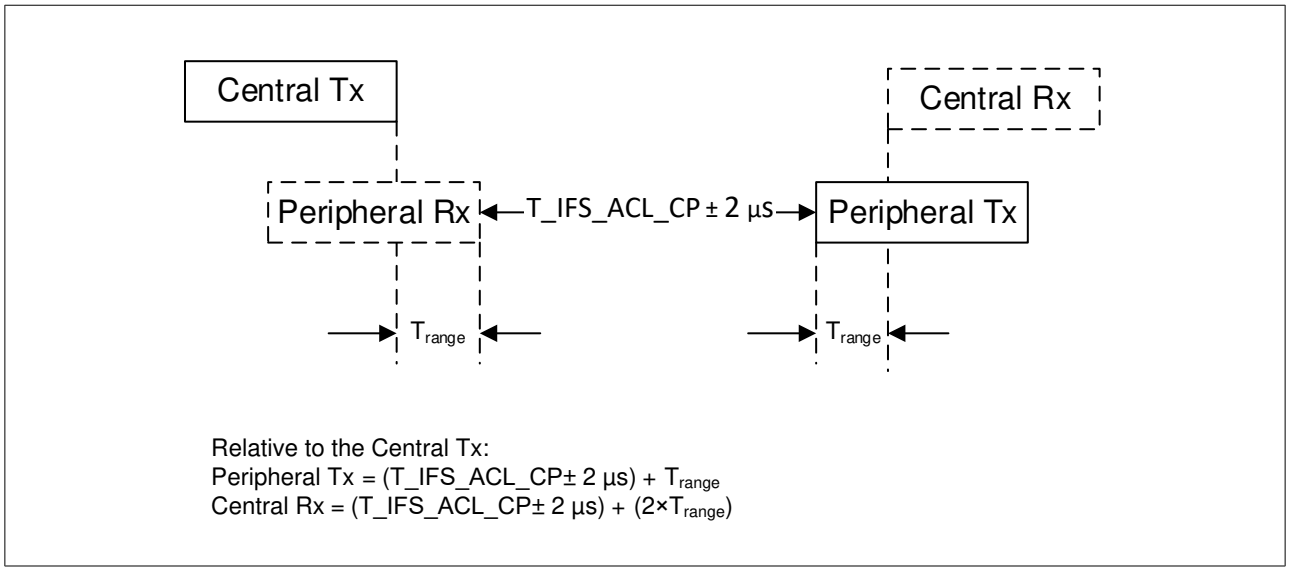



*Figure 4.5: Range delays relative to a Central packet transmission in an ACL*

### 4.2.4 Window widening

In various circumstances a Link Layer is expecting to receive a packet within a certain window (extending from *receiveWindowStart* to *receiveWindowEnd*) or at a certain time

(in which case *receiveWindowStart* and *receiveWindowEnd* are both that time) but, because of active clock accuracies (see Section 4.2.1) and sleep clock accuracies (see Section 4.2.2), there is uncertainty as to the exact timing of that window at the sending Link Layer. The recipient shall therefore adjust its listening time to allow for this uncertainty.

The increase in listening time is called the window widening. Assuming the clock inaccuracies are purely given in parts per million (ppm), it is calculated as follows:

*transmitterAllowance* = (*txCA* ÷ 1000000) × (*receiveWindowEnd* − *timeOfLastSync*) + J µs

where J shall be 2 when the active clock applies and 16 when the sleep clock applies and the other parameters are specified in Table 4.1.

Note: *txCA* is the clock accuracy of the transmitting Link Layer and *timeOfLastSync* is the most recent time that the receiving Link Layer was able to synchronize its clock to that of the transmitting Link Layer.

| **Activity** | ***txCA*** | ***timeOfLastSync*** | ***receiveWindowStart*** | ***receiveWindowEnd*** |
|---|---|---|---|---|
| Receiving any auxiliary packet | CA field of the relevant AuxPtr field | Start of the packet containing that field | AUX Offset from the start of that packet | One Offset Unit after *receiveWindowStart* |
| Advertiser performing connection setup - Peripheral role | SCA field of the relevant CONNECT_IND or AUX_CONNECT_REQ PDU. | Start of the packet containing that PDU. | Start of the transmit window (see Section 4.5.3) | End of the transmit window |
| Peripheral performing connection parameter update | Current *centralSCA* (see Section 4.5.7) | Last ACL or CIS anchor point where a packet was received from the Central | Start of the transmit window (see Section 5.1.1 and Section 5.1.7) | End of the transmit window |
| Peripheral receiving the next connection event | Current *centralSCA* (see Section 4.5.7) | Last ACL or CIS anchor point where a packet was received from the Central | Time of the scheduled connection event anchor point | |

| **Activity** | ***txCA*** | ***timeOfLastSync*** | ***receiveWindowStart*** | ***receiveWindowEnd*** |
|---|---|---|---|---|
| Adjacent packets in the same connection event (see also Section 5.1.30.1) | 50 | End of the previous packet | T_IFS_ACL_CP or T_IFS_ACL_PC after timeOfLastSync | |
| Adjacent packets in the same CIS subevent | 50 | End of the previous packet | T_IFS_CIS after *timeOfLastSync* | |
| Synchronization state - synchronizing | SCA field of the relevant SyncInfo field | Start of the packet containing that field | syncPacketWindowOffset from the start of that packet | One Offset Unit after *receiveWindowStart* |
| Synchronization state - synchronized | *txCA* used while synchronizing | Start of the last packet received containing an AUX_SYNC_IND PDU | Time of the scheduled periodic advertising event | |
| Synchronization state - listening for subevent | txCA used while synchronizing | Start of the last packet received containing an AUX_SYNC_SUBEVENT_IND or AUX_CONNECT_REQ PDU | Time of the scheduled periodic advertising subevent | |
| Periodic Advertising state - listening to response slot | 50 | Start of the last packet transmitted containing an AUX_SYNC_SUBEVENT_IND PDU | Time of the scheduled subevent response slot | |
| Peripheral receiving the next CIS event | Current *centralSCA* (see Section 4.5.7) | Last ACL anchor point or the start of a CIS subevent where a packet was received from the Central | Time of the scheduled CIS subevent | |
| Reception of a BIS Data PDU in the first BIS event | SCA from SyncInfo | Start of the last packet received on the associated periodic advertising train | Offset in BIGInfo from the start of that packet plus any BIS_Spacing applied | One offset Unit after ReceiveWindowStart |

| Activity | *txCA* | *timeOfLastSync* | *receiveWindowStart* | *receiveWindowEnd* |
|---|---|---|---|---|
| Reception of the next BIS event | txCA used while synchronizing | Start of the last packet received in any BIS event on the same BIG | Time of the scheduled BIS subevent | |
| Reception of subsequent sub-events | 50 | Start of the last subevent where a packet was received | Time of the start of the scheduled subevent | |
| Peripheral as CS reflector receiving the next CS subevent | Same as current *centralSCA* (see Section 4.5.7.1) | Last ACL anchor point prior to the start of a CS event or the start of a CS subevent in which a packet was received from the initiator | Time of the scheduled CS subevent | |
| Central as CS reflector receiving the next CS subevent | Same as the current peripheralSCA (see Section 4.5.7.1) | Last ACL anchor point prior to the start of a CS event or the start of a CS subevent in which a packet was received from the initiator. | Time of the scheduled CS subevent | |
| Adjacent CS SYNCs or CS tones in the same CS subevent | 50 | Start of the CS subevent | Reception windows are defined in [Vol 6] Part H, Section 4.5. | |

*Table 4.1: Parameters for window widening*

The rows relating to a Peripheral receiving a CIS event or a device receiving a BIS event only apply until a packet has been received in the event from the correct sender, regardless of a CRC match, and the timing re-synchronized. Once this has happened, the row "Reception of subsequent sub-events" applies for the rest of the event.

If the recipient has more accurate information about the transmitting Link Layer's clock, it may select a smaller value for *transmitterAllowance*.

*windowWidening = transmitterAllowance + receiverAllowance*

where *receiverAllowance* is the allowance made by the receiver for its own clock accuracy.

If the recipient listens for a packet, it shall listen starting at *windowWidening* before *receiveWindowStart* and until *windowWidening* after *receiveWindowEnd*.

The *windowWidening* in an ACL connection shall be smaller than ((*connInterval* ÷ 2) - T_MCES µs). If the *windowWidening* reaches ((*connInterval* ÷ 2) − T_MCES µs) in magnitude, the connection should be considered lost (see Section 4.5.12).

If the *windowWidening* for a CIS with NSE < 3 reaches ((ISO_Interval ÷ 2) − T_MSS_CIS) µs in magnitude, the Link Layer should terminate the CIS (see Section 5.1.16).

If the *windowWidening* for a BIS with NSE < 3 reaches ((ISO_Interval ÷ 2) − T_MSS_150) µs in magnitude, then the Link Layer should stop synchronization with the BIG.

If the *windowWidening* for a CIS or BIS with NSE ≥ 3 reaches Sub_Interval in magnitude, the Link Layer should terminate the CIS or stop synchronization with the BIG. The value of Sub_Interval should be chosen to ensure that this limit is not reached within 2 × ISO_Interval. For example, if *txCA* equals 150 ppm and *receiverAllowance* equals *transmitterAllowance*, the parameters should be chosen so that Sub_Interval ÷ ISO_Interval > 0.0006.

The *windowWidening* in a CS procedure shall be smaller than ((T_EVENT_INTERVAL ÷ 2) − T_MES µs). T_EVENT_INTERVAL is defined in Section 5.1.26 and T_MES is defined in Section 4.1.4. If the *windowWidening* reaches ((T_EVENT_INTERVAL ÷ 2) − T_MES µs) in magnitude, then the CS procedure should be aborted. When the CS procedure is aborted, the Host shall be notified.

## 4.3 Link Layer device filtering

The Link Layer may perform device filtering based on the device address of the peer device. Link Layer device filtering is used by the Link Layer to minimize the number of devices to which it responds.

A Link Layer shall support Link Layer device filtering unless it only supports the Advertising state and only supports non-connectable and non-scannable advertising.

The filter policies for the Advertising state, Scanning state, Initiating state, and Periodic Sync Establishment are independent of each other. When the Link Layer is in the Advertising state, the advertising filter policy shall be used. When the Link Layer is in the Scanning state, the scanning filter policy shall be used. When the Link Layer is in the Initiating state, the initiator filter policy shall be used. When the Link Layer is performing Periodic Sync Establishment, the periodic sync establishment filter policy shall be used. If the Link Layer does not support the Advertising state, Scanning state, Initiating state, or Periodic Sync Establishment, the corresponding filter policy is not required to be supported.

### 4.3.1 Filter Accept List

The set of devices that the Link Layer uses for device filtering is called the Filter Accept List.

A Filter Accept List contains a set of records used for Link Layer device filtering. A Filter Accept List record contains both the device address and the device address type (public or random). There is also a special device address type "anonymous"; an entry with this type matches all advertisements sent with no address. All Link Layers supporting Link Layer device filtering shall support a Filter Accept List capable of storing at least one record.

On reset, the Filter Accept List shall be empty.

The Filter Accept List is configured by the Host and is used by the Link Layer to filter advertisers, scanners, or initiators, but not periodic sync establishment. This allows the Host to configure the Link Layer to act on a request without awakening the Host.

All the device filter policies shall use the same Filter Accept List.

### 4.3.2 Advertising filter policy

The advertising filter policy determines how the advertiser's Link Layer processes scan and/or connection requests.

When the Link Layer is using non-connectable and non-scannable directed advertising events, scannable directed advertising events, and connectable directed advertising events the advertising filter policy shall be ignored. Otherwise the Link Layer shall use one of the following advertising filter policy modes which are configured by the Host:

- The Link Layer shall process scan and connection requests from all devices (i.e. the Filter Accept List is not in use). This is the default on reset.
- The Link Layer shall process connection requests from all devices and shall only process scan requests from devices that are in the Filter Accept List.
- The Link Layer shall process scan requests from all devices and shall only process connection requests from devices that are in the Filter Accept List.
- The Link Layer shall process scan and connection requests only from devices in the Filter Accept List.

Only one advertising filter policy mode per advertising set shall be supported at a time.

### 4.3.3 Scanning filter policy

The scanning filter policy determines how the scanner's Link Layer processes advertising and scan response PDUs. The Link Layer shall use one of the following

scanning filter policies as selected by the Host. Only one scanning filter policy shall be supported at a time.

There is a choice of two primary filter policies:

- Unfiltered: The Link Layer shall process all advertising and scan response PDUs (i.e., the Filter Accept List is not used). This is the default on reset.
- Filtered: The Link Layer shall process decision PDUs and their subordinate sets from all devices and other advertising PDUs and any scan response PDUs only from devices in the Filter Accept List.

In the basic scanning filter policy mode, a directed advertising PDU accepted by the primary filter policy shall nevertheless be ignored unless either:

- the TargetA field is identical to the scanner's device address, or
- the TargetA field is a resolvable private address, address resolution is enabled, and the address is resolved successfully.

Note: The scanning filter policy does not affect initiation or periodic sync establishment even though they involve scanning for advertising PDUs.

#### 4.3.3.1 Extended scanning filter policies

If the Link Layer supports the extended scanning filter policies, then extended mode shall also be supported. This is identical to basic mode except that a directed advertising PDU accepted by the primary filter policy shall nevertheless be ignored unless either:

- the TargetA field is identical to the scanner's device address, or
- the TargetA field is a resolvable private address.

#### 4.3.3.2 Decision scanning filter policy modes

The Link Layer shall also use one of the following filter policy modes which are configured by the Host. These only apply to advertising PDUs received on the primary advertising physical channel. They apply in addition to the primary filter policy in use.

- No decisions: The Link Layer shall ignore decision PDUs and shall process other advertising PDUs. This is the default on reset and is the only mode that shall be used if the Link Layer does not support the Decision-Based Advertising Filtering feature.
- Decisions only: The Link Layer shall only process decision PDUs and shall ignore all other advertising PDUs.
- All PDUs: The Link Layer shall process all advertising PDUs.

### 4.3.4 Initiator filter policy

The initiator filter policy determines how an initiator's Link Layer processes advertising PDUs. The Link Layer shall use one of the following initiator filter policy modes which are configured by the Host:

- The Link Layer shall ignore the Filter Accept List and process connectable advertising PDUs, other than decision PDUs, from a specific single device specified by the Host.
- The Link Layer shall ignore the Filter Accept List and only process connectable decision PDUs from any device.
- The Link Layer shall process connectable advertising PDUs from all devices in the Filter Accept List.
- The Link Layer shall process connectable advertising PDUs, other than decision PDUs, from all devices in the Filter Accept List.
- The Link Layer shall process connectable decision PDUs and their subordinate sets from all devices and other connectable advertising PDUs from all devices in the Filter Accept List.

If the Link Layer receives a connectable directed advertising PDU that is not allowed by the initiator filter policy, the connectable directed advertising PDU shall be ignored.

Only one initiator filter policy mode shall be supported at a time.

### 4.3.5 Periodic sync establishment filter policy

The periodic sync establishment filter policy determines how a scanner's Link Layer processes advertising PDUs when attempting to synchronize to a periodic advertising train. The Link Layer shall use one of the following periodic sync establishment filter policy modes which are configured by the Host:

- The Link Layer shall ignore the Periodic Advertiser List and process advertising PDUs from a specific single device specified by the Host.
- The Link Layer shall process advertising PDUs from all devices in the Periodic Advertiser List.

If the Link Layer receives an advertising PDU which contains a SyncInfo field from an advertiser that is not contained in the Periodic Advertiser List or the single address specified by the Host, or if the advertising has an Advertising SID which is not the one specified in the list entry or by the Host, the SyncInfo field shall be ignored.

Only one periodic sync establishment filter policy mode shall be supported at a time.

Synchronization to periodic advertising takes place at the same time as scanning, but the filter policies for the two activities are independent. The periodic sync establishment

filter policy, and not the scanning filter policy, shall determine which advertising PDUs are used to synchronize to a periodic advertising train (successful synchronization is then reported to the Host). If a received PDU only matches one of the two policies, it shall only be processed for the purpose that uses that policy and not for the other.

The Link Layer shall ignore the Periodic Advertiser List when synchronizing to a periodic advertising train where it received the synchronization information using the Periodic Advertising Sync Transfer procedure (see Section 5.1.13).

## 4.4 Non-connected states

The Host may specify which coding to use when transmitting packets on the LE Coded PHY. If it does not, then the Controller determines the coding of each packet. The coding is indicated by the CI as defined in Section 2.2.3 and may be different in each direction and in adjacent packets in a given direction.

### 4.4.1 Standby state

The Standby state is the default state in the Link Layer. The Link Layer shall not send or receive packets in the Standby state. The Link Layer may leave the Standby state to enter the Advertising state, Scanning state, Initiator state, Synchronization state, or Isochronous Broadcasting state.

### 4.4.2 Advertising state

The Link Layer shall enter the Advertising state when directed by the Host. When placed in the Advertising state, the Link Layer shall send advertising PDUs (see Section 2.3.1) in advertising events, periodic advertising events, or both.

Each advertising event is composed of one or more advertising PDUs sent on used primary advertising channel indices. The advertising event shall be closed after one advertising PDU has been sent on each of the used primary advertising channel indices (see Section 4.4.2.1). Some advertising PDUs in an advertising event may be omitted, causing the advertising event to begin late or close early, or the entire advertising event may be omitted to accommodate other functionality.

The time between two consecutive advertising events is defined in Section 4.4.2.2.

An advertising event can be one of the following types:

- a connectable and scannable undirected event
- a connectable undirected event
- a connectable directed event
- a non-connectable and non-scannable undirected event

- a non-connectable and non-scannable directed event
- a scannable undirected event
- a scannable directed event

At most one advertising PDU shall be sent on each used primary advertising channel index in an advertising event. Unless specified otherwise, the primary advertising channel indices may be used in any order. The order may be different in different events.

The advertising event type determines the allowable response PDUs. Table 4.2 specifies the allowable responses for each advertising event.

Connectable and scannable undirected, connectable undirected, and connectable directed events are collectively referred to as connectable events; the remaining events (non-connectable and non-scannable undirected, non-connectable and non-scannable directed, scannable undirected, and scannable directed) are collectively referred to as non-connectable events. Connectable and scannable undirected, scannable undirected, and scannable directed events are collectively referred to as scannable events; the remaining events (non-connectable and non-scannable undirected, non-connectable and non-scannable directed, connectable undirected, and connectable directed) are collectively referred to as non-scannable events.

| | | **Allowable response PDUs** | | | |
|---|---|---|---|---|---|
| **Advertising Event Type** | **Type of PDU being responded to** | **SCAN_-REQ[1]** | **CONNECT_-IND[1]** | **AUX_-SCAN_-REQ** | **AUX_-CONNECT_-REQ** |
| Connectable and Scannable Undirected event | ADV_IND | YES | YES | NO | NO |
| Connectable Undirected event | ADV_EXT_IND or ADV_DECISION_-IND | NO | NO | NO | NO |
| | AUX_ADV_IND | NO | NO | NO | YES |
| Connectable Directed event | ADV_DIRECT_IND | NO | YES[2] | NO | NO |
| | ADV_EXT_IND | NO | NO | NO | NO |
| | AUX_ADV_IND | NO | NO | NO | YES[2] |
| Non-Connectable and Non-Scannable Undirected event | ADV_NONCONN_-IND | NO | NO | NO | NO |
| | ADV_EXT_IND or ADV_DECISION_-IND | NO | NO | NO | NO |

| | | Allowable response PDUs | | | |
|---|---|---|---|---|---|
| **Advertising Event Type** | **Type of PDU being responded to** | **SCAN_-REQ[1]** | **CONNECT_-IND[1]** | **AUX_-SCAN_-REQ** | **AUX_-CONNECT_-REQ** |
| | AUX_ADV_IND | NO | NO | NO | NO |
| Non-Connectable and Non-Scanna-ble Directed event | ADV_EXT_IND | NO | NO | NO | NO |
| | AUX_ADV_IND | NO | NO | NO | NO |
| Scannable Undir-ected event | ADV_SCAN_IND | YES | NO | NO | NO |
| | ADV_EXT_IND or ADV_DECISION_-IND | NO | NO | NO | NO |
| | AUX_ADV_IND | NO | NO | YES | NO |
| Scannable Directed event | ADV_EXT_IND | NO | NO | NO | NO |
| | AUX_ADV_IND | NO | NO | YES[3] | NO |

*Table 4.2: Advertising event types, PDUs used, and allowable response PDUs*

[1]Not permitted on the LE Coded PHY.

[2]Initiators other than the correctly addressed initiator shall not respond.

[3]Scanners other than the correctly addressed scanner shall not respond.

If the advertiser receives a PDU for the advertising event that is not explicitly allowed it shall be ignored. If no PDU is received or the received PDU was ignored, the advertiser shall either send an advertising PDU on the next used primary advertising channel index or close the advertising event.

The Controller shall not use two different types of PDU on the primary advertising physical channel in the same advertising event. The Controller may use the ADV_DECISION_IND PDU instead of the ADV_EXT_IND PDU in any undirected advertising event.

#### 4.4.2.1 Advertising channel index selection

Advertising events use three predefined primary advertising physical channels. Primary advertising channel indices are either used or unused.

For AUX_ADV_IND and AUX_CHAIN_IND PDUs, the secondary advertising channel index used in the Channel Index subfield of the AuxPtr field is implementation specific. It is recommended that sufficient channel diversity is used to avoid collisions.

Each periodic advertising train shall have a 16-bit event counter (*paEventCounter*). The initial value of this counter is implementation specific. The counter shall be incremented by one for each Periodic Advertising Interval (see Section 4.4.2.3), whether or not the AUX_SYNC_IND PDU is actually transmitted; the *paEventCounter* shall wrap from

0xFFFF to 0x0000. AUX_SYNC_IND PDUs shall use the Channel Selection Algorithm #2 (see Section 4.5.8.3) with this event counter.

When periodic advertising subevents are used, the train has a 7-bit value (*paSubEventCounter*). The *paSubEventCounter* is set to 0 at the start of each periodic advertising event and incremented by one for each subevent. AUX_SYNC_SUBEVENT_IND PDUs shall use the Channel Selection Algorithm #2 with the *paEventCounter* and *paSubEventCounter*.

The response slots shall use the same channel as the AUX_SYNC_SUBEVENT_IND packet.

The Link Layer shall use the primary and secondary advertising channel indices as specified by the Host, and the used primary and secondary advertising channel indices shall take effect when the Advertising state is entered. The Link Layer need not use all the secondary channels that the Host has marked as "unknown".

#### 4.4.2.2 Advertising events

Advertising events are defined as one or more advertising PDUs sent on the primary advertising physical channel. At most one PDU shall be sent on each used advertising channel index; usually a PDU is sent on all the used advertising indices in each advertising event. The advertising event can be closed early after a CONNECT_IND is received or when a SCAN_RSP is sent.

Advertising packets sent on the secondary advertising physical channel are not part of the advertising event. Advertising events that use the ADV_EXT_IND PDU may also be part of an extended advertising event. All ADV_EXT_IND PDUs containing an AuxPtr field in the same advertising event shall point to the same AUX_ADV_IND packet.

##### 4.4.2.2.1 Advertising interval

For all undirected advertising events or connectable directed advertising events used in a low duty cycle mode, the time between the start of two consecutive advertising events (*T_advEvent*) for the same advertising set (see Section 4.4.2.10) is computed as follows for each advertising event:

$$T_advEvent = advInterval + advDelay$$

The advertising interval (*advInterval*) shall be an integer multiple of 0.625 ms in the range 20 ms to 10,485.759375 s.

The *advDelay* shall be a (pseudo-)random value with a range 0 ms to 10 ms generated by the Link Layer for each advertising event.

As illustrated in Figure 4.6, the advertising events shall be perturbed in time using the advDelay.

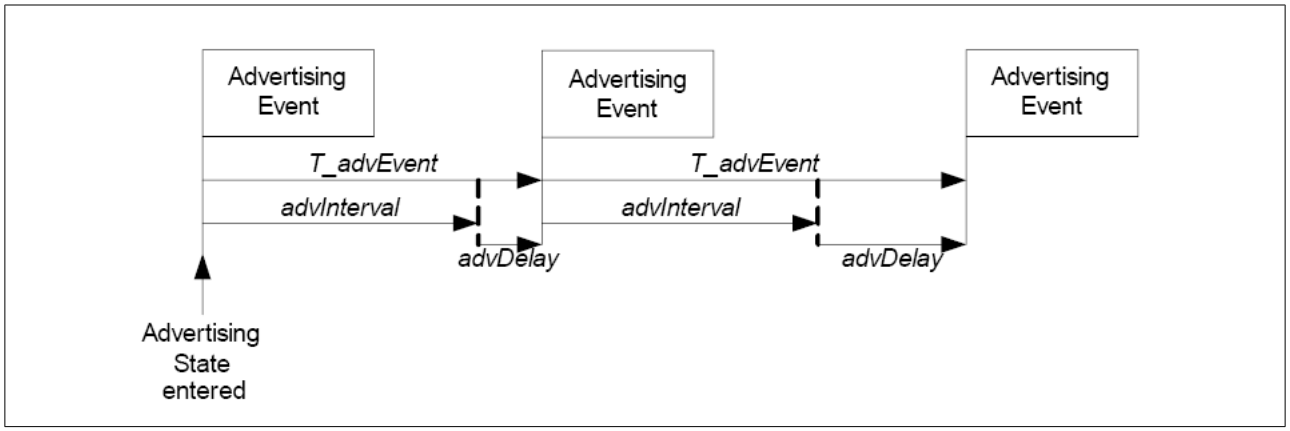



*Figure 4.6: Advertising events perturbed in time using advDelay*

#### 4.4.2.2.2 Extended advertising event

An extended advertising event begins at the start of an advertising event and consists of the PDUs in that advertising event plus their subordinate sets. The extended advertising event ends with the last such PDU.

Multiple extended advertising events may overlap with each other. This can occur when ADV_EXT_IND PDUs containing an AuxPtr field in multiple advertising events point to the same AUX_ADV_IND packet, or when a different advertising event is interposed between the ADV_EXT_IND PDUs and the AUX_ADV_IND PDU.

T_advEvent, advInterval and advDelay have the same meaning as in Section 4.4.2.2.1.

Figure 4.7 illustrates an example of overlapping extended advertising events.

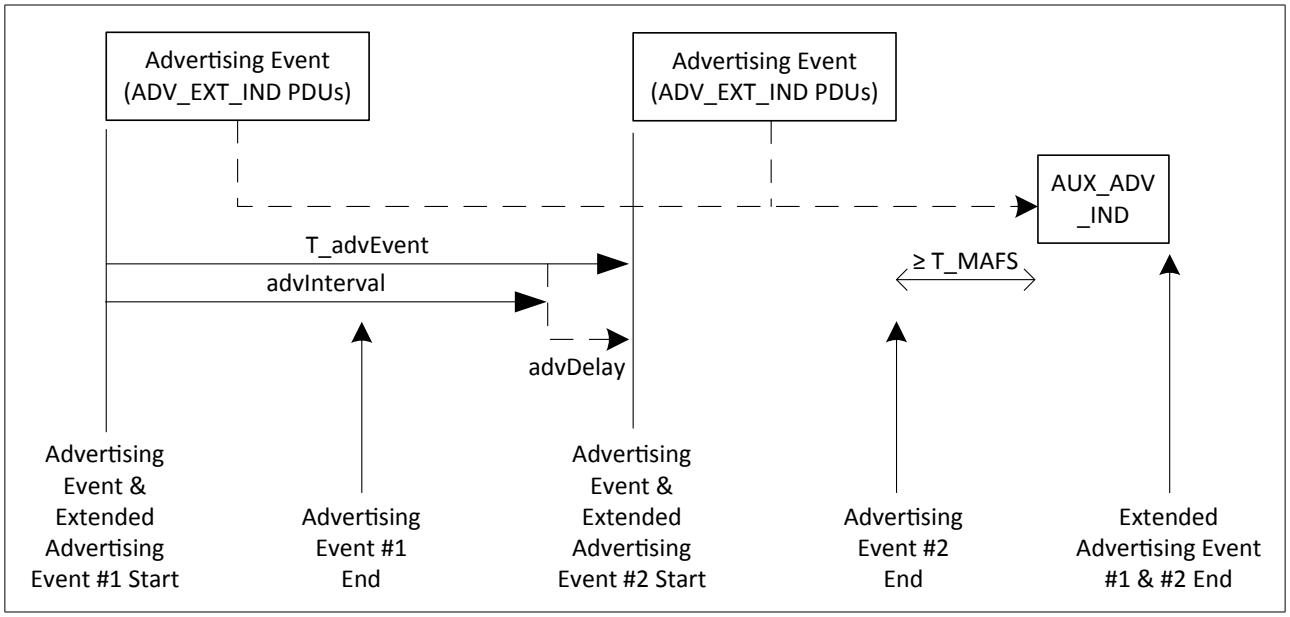



*Figure 4.7: Example of overlapping extended advertising events*

An auxiliary advertising segment starts with the first AUX_ADV_IND PDU in an extended advertising event and ends at the end of the extended advertising event (an auxiliary advertising segment can belong to more than one extended advertising event). Two auxiliary advertising segments for the same advertising set shall not overlap each other.

Figure 4.8 illustrates an example of auxiliary advertising segments belonging to multiple extended advertising events.

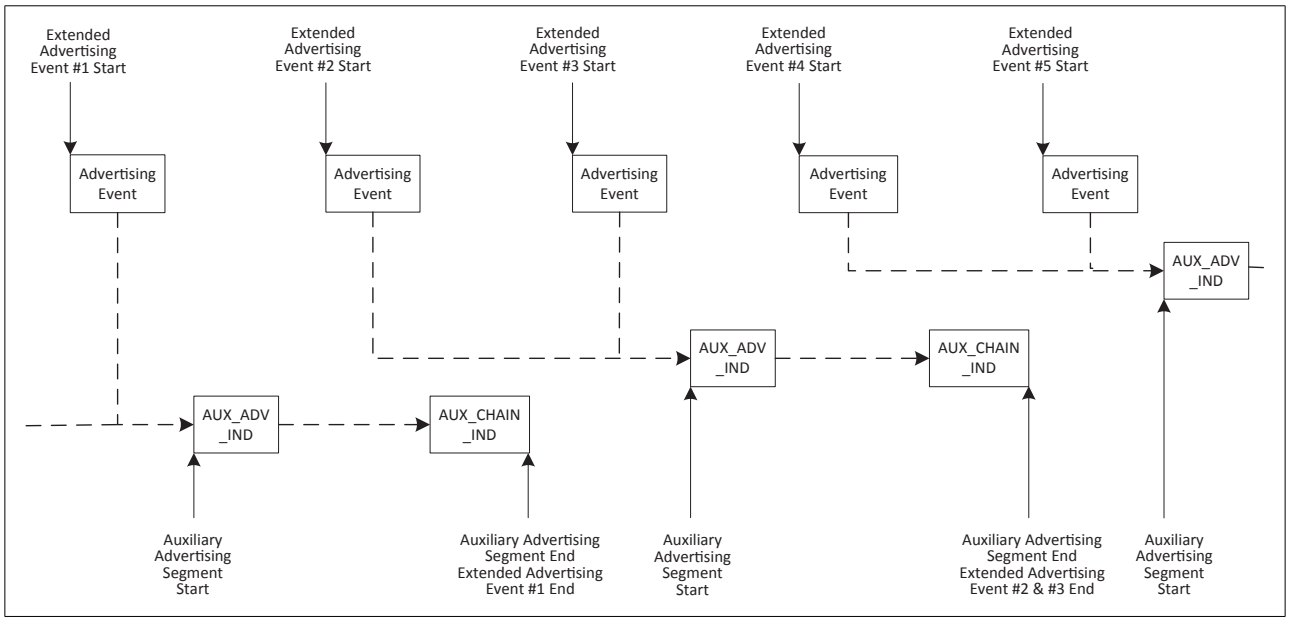



*Figure 4.8: Example of auxiliary advertising segments*

An advertiser should not space PDUs within the auxiliary advertising segment so that two of them would be within the same receive window. If an advertiser transmits a PDU with an AuxPtr field containing an offset of T milliseconds, then it should not start to transmit any other packet on the same RF channel as the auxiliary packet within 2.5×T microseconds of the start of the auxiliary packet of the original PDU.

#### 4.4.2.2.3 Periodic advertising events

The Periodic Advertising Interval is the interval between the start of two scheduled AUX_SYNC_IND PDUs from the same advertising set (even if the PDUs are not transmitted for some reason). The Periodic Advertising Interval shall be an integer multiple of 1.25 ms in the range 7.5 ms to 81.91875 s.

A periodic advertising event consists of an AUX_SYNC_IND PDU and its subordinate set.

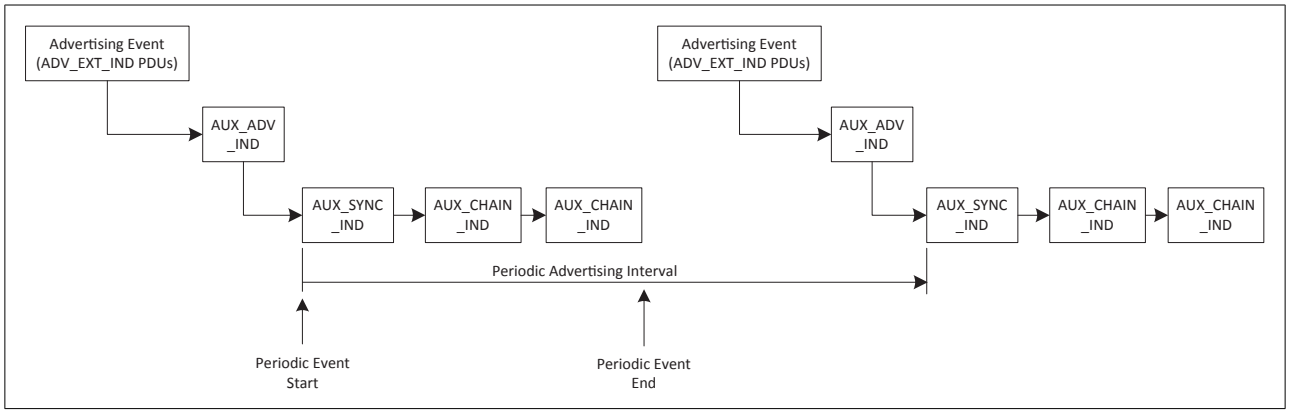



*Figure 4.9: Example of periodic advertising events from the same advertising set*

Two periodic advertising events for the same periodic advertising train shall not overlap each other. The periodic advertising interval shall not change while the periodic advertising is enabled.

#### 4.4.2.2.4 Periodic advertising with responses events

A periodic advertising with responses event (PAwR event) consists of one or more subevents. Each subevent consists of an AUX_SYNC_SUBEVENT_IND PDU and one or more AUX_SYNC_SUBEVENT_RSP PDUs (see Figure 4.11).

The duration of the AUX_SYNC_SUBEVENT_IND PDU shall be smaller than Periodic Advertising Response Slot Delay minus T_IFS_150 if one or more response slots are defined and smaller than Periodic Advertising Subevent Interval minus T_IFS_150 otherwise. The duration of the AUX_SYNC_SUBEVENT_RSP PDU shall be smaller than Periodic Advertising Response Slot Spacing minus T_IFS_150. The gap between the end of one AUX_SYNC_SUBEVENT_IND

or AUX_SYNC_SUBEVENT_RSP packet and the start of the next AUX_SYNC_SUBEVENT_IND or AUX_SYNC_SUBEVENT_RSP packet shall be at least T_IFS_150.

The AUX_SYNC_SUBEVENT_RSP PDU shall be sent in response to the AUX_SYNC_SUBEVENT_IND PDU in a response slot and subevent provided by the Host. The data in the AUX_SYNC_SUBEVENT_RSP PDU shall contain the response to the data received in the AUX_SYNC_SUBEVENT_IND PDU. The Controller shall send the response no later than the Periodic Advertising Interval after the start of the received AUX_SYNC_SUBEVENT_IND PDU (therefore, if there is only one subevent in each event then the response must be sent in the same subevent).

The Periodic Advertising Interval is the interval between the start of two scheduled AUX_SYNC_SUBEVENT_IND PDUs with the same subevent number from the same advertising set (even if the PDUs are not transmitted for some reason). The Periodic Advertising Interval shall be an integer multiple of 1.25 ms in the range 7.5 ms to 81.91875 s.

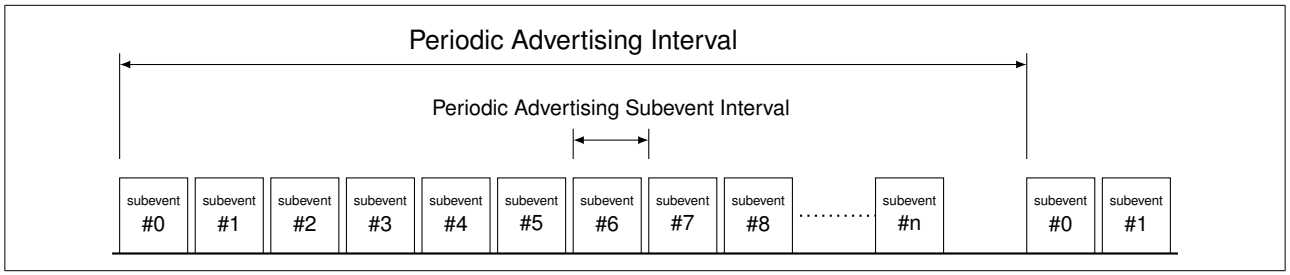



*Figure 4.10: Periodic Advertising Subevents*

The Periodic Advertising Subevent Interval is the interval between the start of two scheduled adjacent AUX_SYNC_SUBEVENT_IND PDUs in the same PAwR event. The Periodic Advertising Subevent Interval shall be an integer multiple of 1.25 ms in the range 7.5 ms to 318.75 ms. The subevent ends Periodic Advertising Subevent Interval after the start of the subevent. Up to 128 subevents can be used in a Periodic Advertising Interval. The subevent(s) to synchronize to is provided by the Host. If the Host has not provided the subevent(s), the Controller may synchronize with any subevent.

One or more response slots are defined when a device that is synchronized with this periodic advertising train may transmit a response packet. A synchronized device may only transmit a response packet in at most one response slot per subevent. Different synchronized devices may transmit response packets in different response slots in the same subevent. The Periodic Advertising Response Slot Delay is the interval between the start of an AUX_SYNC_SUBEVENT_IND PDU and the first response slot. The Periodic Advertising Response Slot Delay shall be an integer multiple of 1.25 ms in the range 1.25 ms to 317.5 ms. The Periodic Advertising Response Slot Spacing is

the interval between the start of two adjacent response slots in a given subevent. The Periodic Advertising Response Slot Spacing shall be an integer multiple of 0.125 ms in the range 0.25 ms to 31.875 ms and shall not be greater than (Periodic Advertising Subevent Interval − Periodic Advertising Response Slot Delay) ÷ number of response slots.

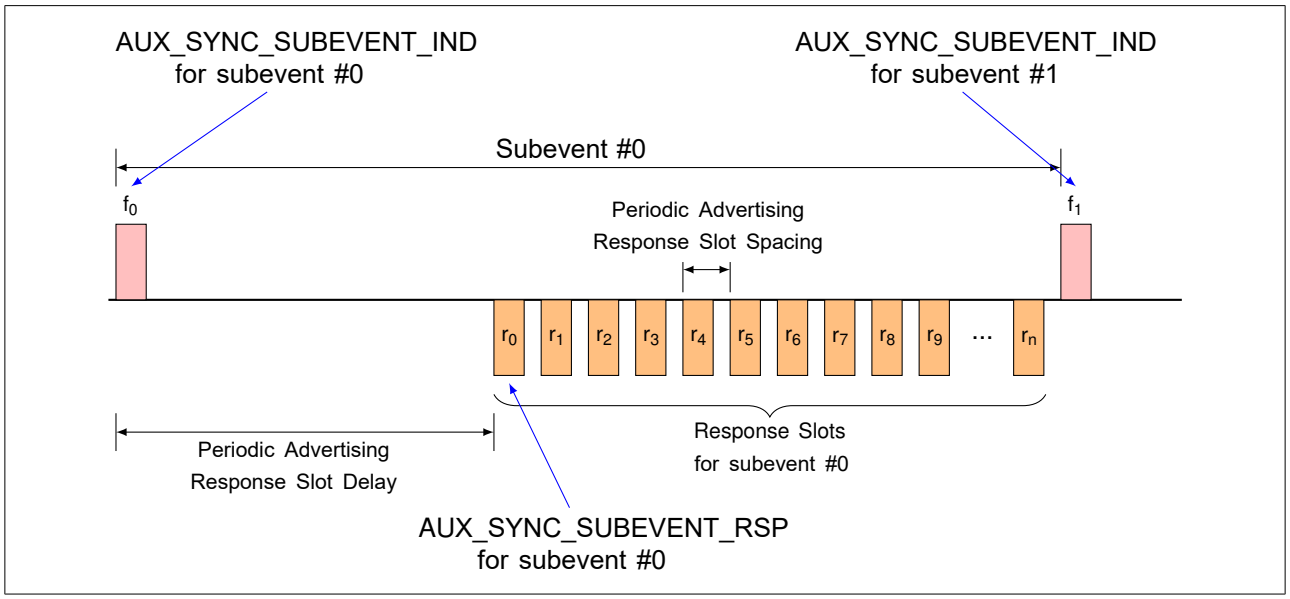



*Figure 4.11: Periodic Advertising Subevent Response Slots*

Two periodic advertising events for the same PAwR train shall not overlap each other. Two PAwR subevents shall not overlap each other. The Periodic Advertising Interval, Periodic Advertising Subevent Interval, Periodic Advertising Response Slot Delay, and Periodic Advertising Response Slot Interval shall not change while PAwR is enabled for that advertising set.

#### 4.4.2.3 Connectable and scannable undirected event type

When the connectable and scannable undirected advertising event type is used, advertising indications (ADV_IND PDUs) are sent by the Link Layer.

The connectable and scannable undirected advertising event type allows a scanner or initiator to respond with either a scan request or connect request. A scanner may send a scan request (SCAN_REQ PDU) to request additional information about the advertiser. An initiator may send a connect request (CONNECT_IND PDU) to request the Link Layer to enter the Connection state.

The Link Layer shall listen on the same primary advertising channel index for requests from scanners or initiators.

If the advertiser receives a SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy, it shall reply with a SCAN_RSP PDU on

the same primary advertising channel index. After the SCAN_RSP PDU is sent, or if the advertising filter policy did not allow processing the SCAN_REQ PDU, the advertiser shall either move to the next used primary advertising channel index to send another ADV_IND PDU, or close the advertising event.

If the advertiser receives a CONNECT_IND PDU that contains its device address, from an initiator allowed by the advertising filter policy, and is not already connected to the same device address, then the Link Layer shall exit the Advertising state and transition to the Connection state in the Peripheral Role as defined in Section 4.5.5. If the advertising filter policy did not allow processing the received CONNECT_IND PDU, the advertiser shall either move to the next used primary advertising channel index to send another ADV_IND PDU, or close the advertising event.

The time between the beginning of two consecutive ADV_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

An illustration of an advertising event using all the primary advertising channel indices and in which no SCAN_REQ or CONNECT_IND PDUs are received is shown in Figure 4.12.

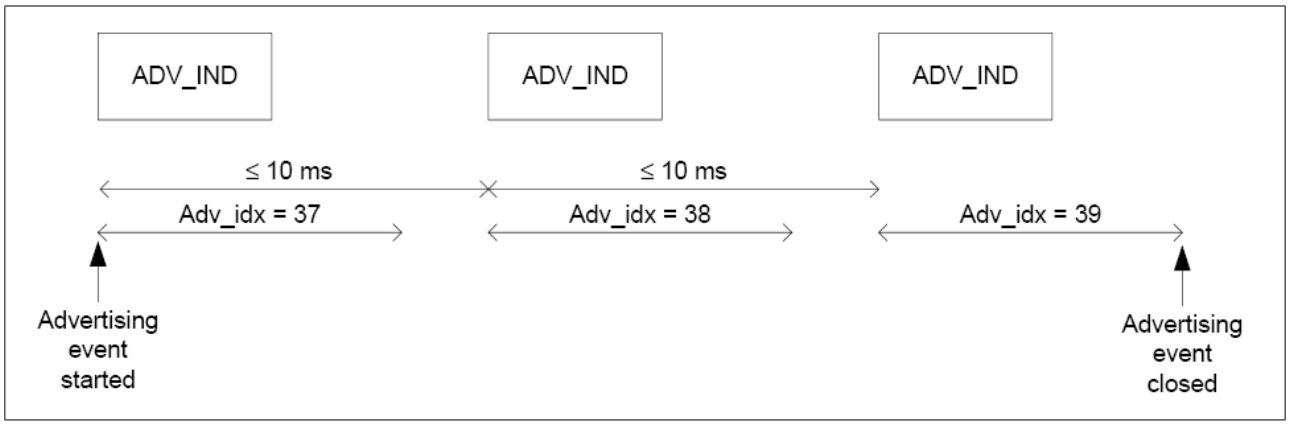



*Figure 4.12: Connectable and scannable undirected advertising event with only advertising PDUs*

Two illustrations of advertising events using all the primary advertising channel indices during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.13 and in Figure 4.14.

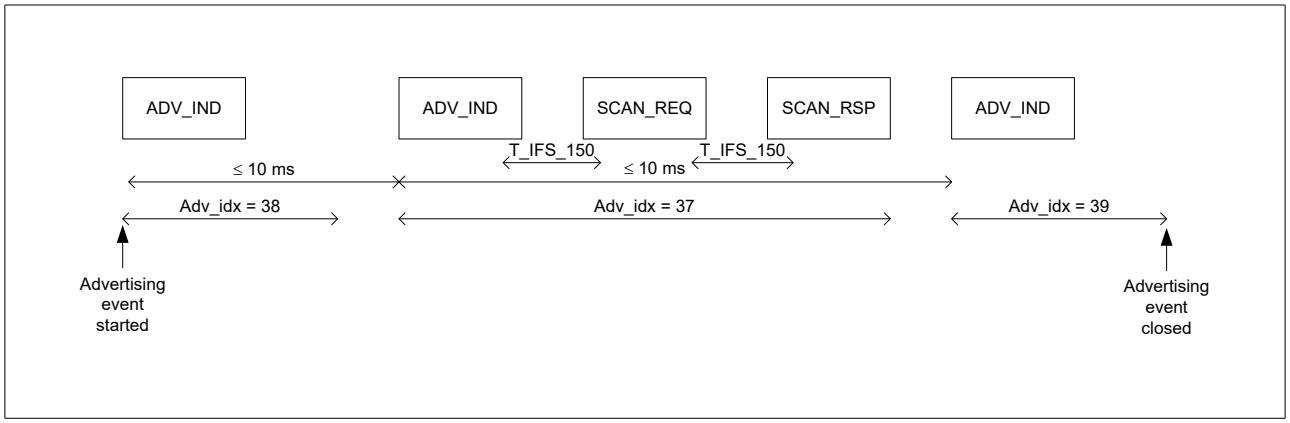



*Figure 4.13: Connectable and scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event*

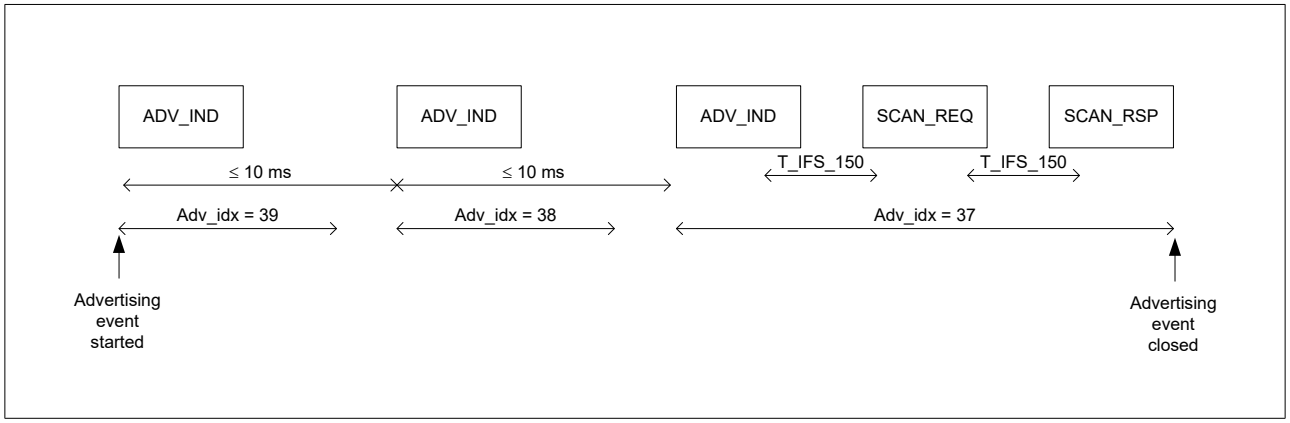



*Figure 4.14: Connectable and scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs at the end of an advertising event*

Figure 4.15 illustrates an advertising event during which a CONNECT_IND PDU is received on the second primary advertising channel index.

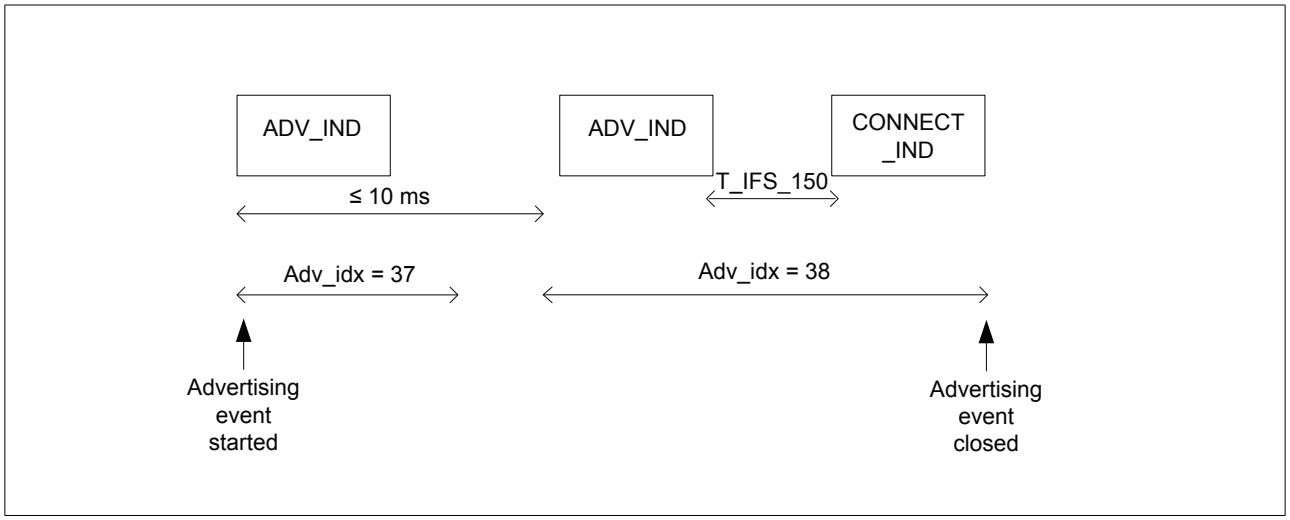



*Figure 4.15: Connectable and scannable undirected advertising event when a CONNECT_IND PDU is received*

If the Controller supports LL Privacy, then the requirements in Section 6.2.1 shall also be followed.

#### 4.4.2.4 Connectable directed event type

When the connectable directed advertising event type is used, directed advertising indications are sent by the Link Layer.

The connectable directed advertising event type allows an initiator to respond so that both the advertiser and initiator will enter the Connection state.

The connectable directed advertising event type may use either the ADV_DIRECT_IND PDU (see Section 4.4.2.4.1 to Section 4.4.2.4.3) or the ADV_EXT_IND PDU (see Section 4.4.2.4.4).

If the Controller supports LL Privacy, then the requirements in Section 6.2.2 shall also be followed.

##### 4.4.2.4.1 Connectable directed event type using ADV_DIRECT_IND

The connectable directed advertising event type using ADV_DIRECT_IND allows an initiator to respond with a connect request on the primary advertising physical channel to establish an ACL connection.

The ADV_DIRECT_IND PDU contains both the initiator's device address and the advertiser's device address. Only the addressed initiator may initiate an ACL connection with the advertiser by sending a CONNECT_IND PDU to the advertiser.

After every ADV_DIRECT_IND PDU sent by the advertiser, the advertiser shall listen for CONNECT_IND PDUs on the same primary advertising channel index. Any SCAN_REQ PDUs received shall be ignored.

If the advertiser receives a CONNECT_IND PDU that contains its device address, the initiator device address is contained in the ADV_DIRECT_IND PDU, and the advertiser is not already connected to that initiator device address, then the Link Layer shall exit the Advertising state and transition to the Connection state in the Peripheral Role as defined in Section 4.5.5.

Otherwise, the advertiser shall either move to the next used primary advertising channel index to send another ADV_DIRECT_IND PDU, or close the Advertising event.

Connectable directed advertising may be either used in a low duty cycle or high duty cycle mode; these are described in the next two sections. Low duty cycle connectable directed advertising is designed for cases where reconnection with a specific device is required, but time is not of the essence or it is not known if the Central is in range or not. High duty cycle connectable directed advertising is designed for cases in which fast ACL connection setup is essential (for example, a reconnection).

Note: High duty cycle connectable directed advertising is a power and bandwidth intensive advertising scheme that is only intended for use when fast connection setup is required.

#### 4.4.2.4.2 Low duty cycle connectable directed advertising

In low duty cycle connectable directed advertising, the time between the start of two consecutive ADV_DIRECT_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

An illustration of an advertising event using all primary advertising channel indices and in which no CONNECT_IND PDUs are received is shown in Figure 4.16.

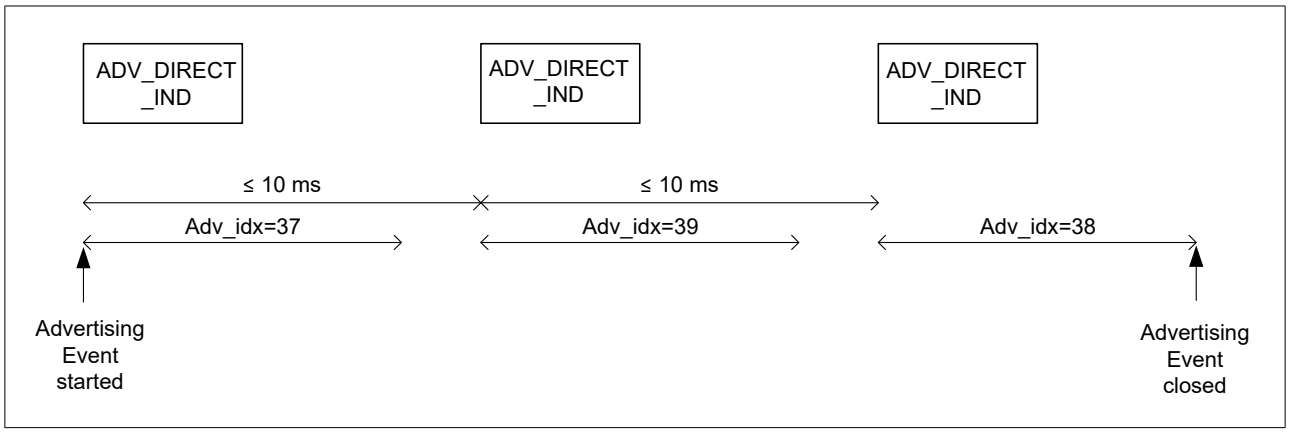



*Figure 4.16: Low duty cycle connectable directed advertising event with only advertising PDUs*

Figure 4.17 illustrates an advertising event using ADV_DIRECT_IND advertising PDUs during which a CONNECT_IND PDU is received on the second primary advertising channel index.

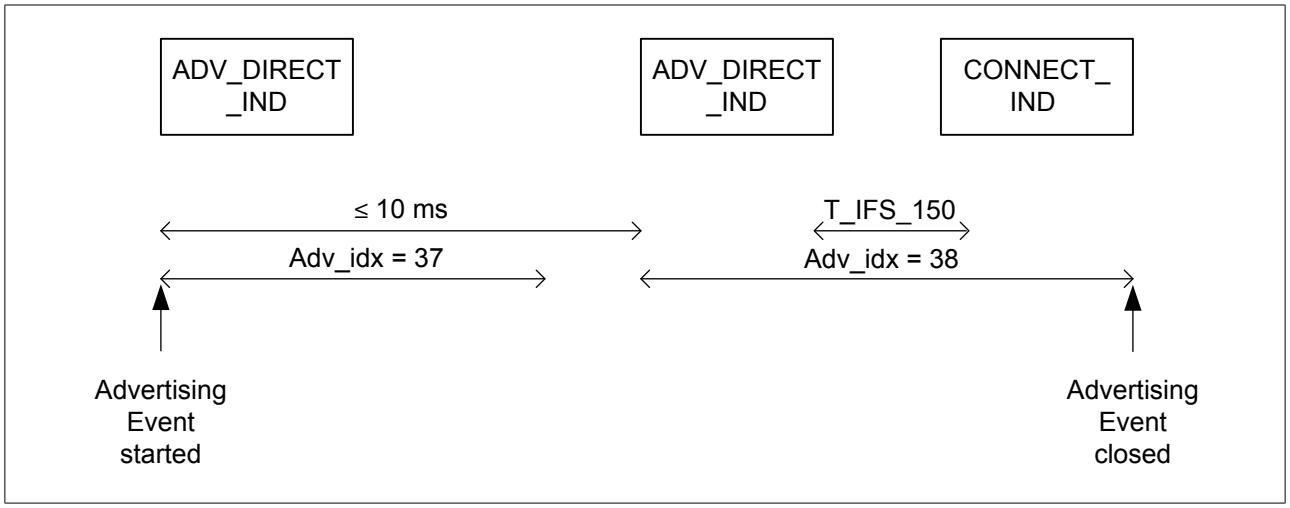



*Figure 4.17: Low duty cycle connectable directed advertising event during which a CONNECT_IND PDU is received*

#### 4.4.2.4.3 High duty cycle connectable directed advertising

In high duty cycle connectable directed advertising mode, the time between the start of two consecutive ADV_DIRECT_IND PDUs sent on the same advertising channel index shall be less than or equal to 3.75 ms.

The Link Layer shall exit the Advertising state no later than 1.28 s after the Advertising state was entered.

The Link Layer shall start each advertising event with the lowest used primary advertising channel index and move sequentially through the other used primary advertising indices.

A sequence of five ADV_DIRECT_IND PDUs in two Advertising events without CONNECT_IND PDUs is shown in Figure 4.18 for the case in which all the primary advertising physical channels are used.

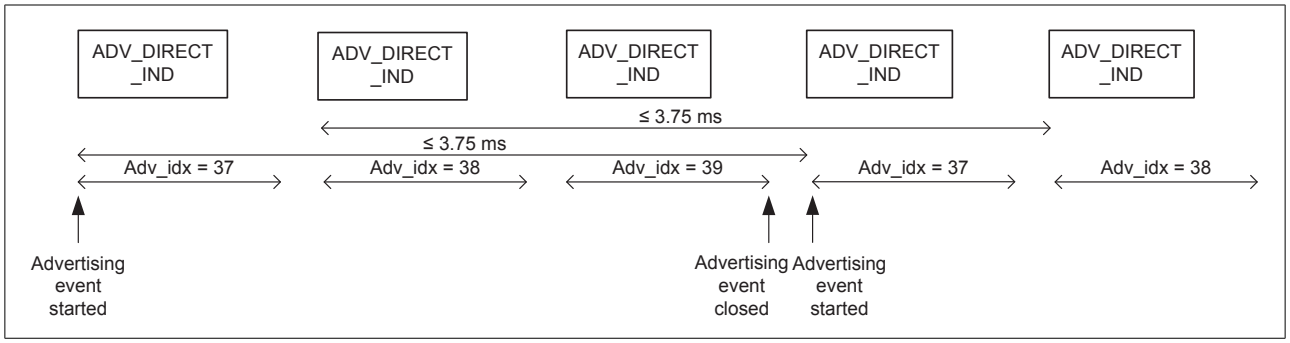



*Figure 4.18: High duty cycle connectable directed advertising event with only advertising PDUs*

#### 4.4.2.4.4 Connectable directed event type using ADV_EXT_IND

The connectable directed advertising event type using ADV_EXT_IND allows an initiator to respond with a connect request on the secondary advertising physical channel to establish an ACL connection.

After every AUX_ADV_IND PDU related to this event it sends, the advertiser shall listen for AUX_CONNECT_REQ PDUs on the same secondary advertising channel index. Any AUX_SCAN_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_CONNECT_REQ PDU that contains its device address, the initiator's device address was contained in the AUX_ADV_IND PDU, and the advertiser is not already connected to that initiator device address, then the advertiser shall reply with an AUX_CONNECT_RSP PDU on the same secondary advertising channel index. If the advertiser's Link Layer does not support LL Privacy, then it shall use those addresses in the AUX_CONNECT_RSP PDU. After the AUX_CONNECT_RSP PDU is sent the Link Layer shall exit the Advertising state and transition to the Connection state in the Peripheral Role as defined in Section 4.5.5. Any AUX_SCAN_REQ PDUs received on the secondary advertising physical channel shall be ignored.

The time between the start of two consecutive connectable directed ADV_EXT_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

The channel index on the secondary channel, SAdv_idx, is contained in the AuxPtr field of the ADV_EXT_IND PDU.

Figure 4.19 shows an advertising event in which no AUX_CONNECT_REQ PDU is received.

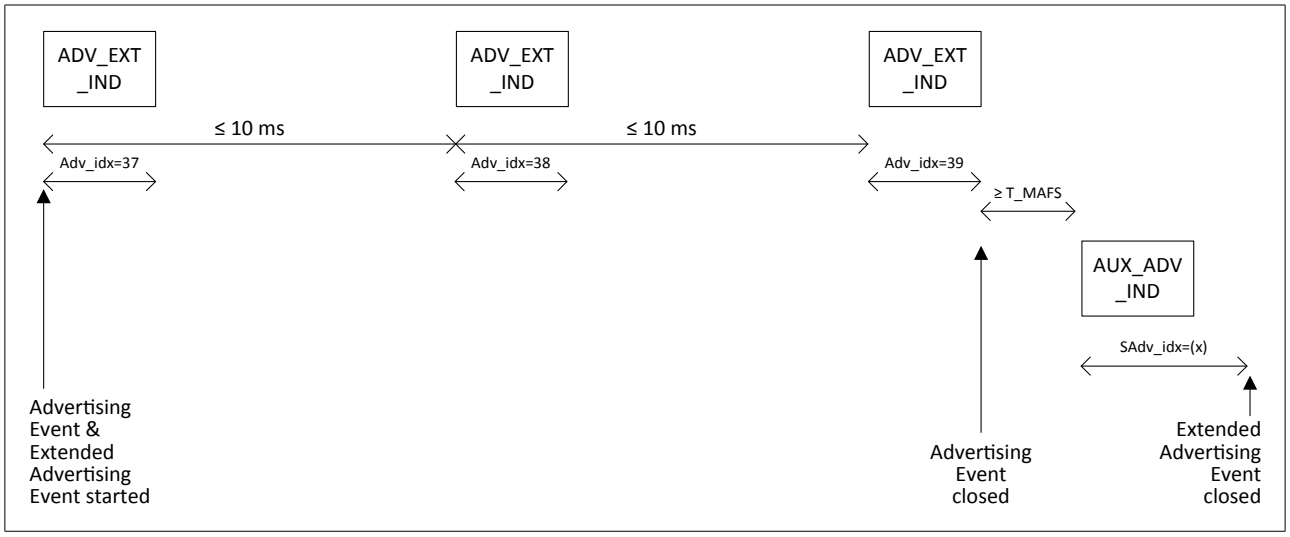



*Figure 4.19: Connectable directed advertising event using the ADV_EXT_IND PDUs and AUX_ADV_IND PDU containing advertising data*

Figure 4.20 illustrates an advertising event using connectable directed ADV_EXT_IND advertising PDUs during which an AUX_CONNECT_REQ PDU is received on the second secondary advertising channel index.

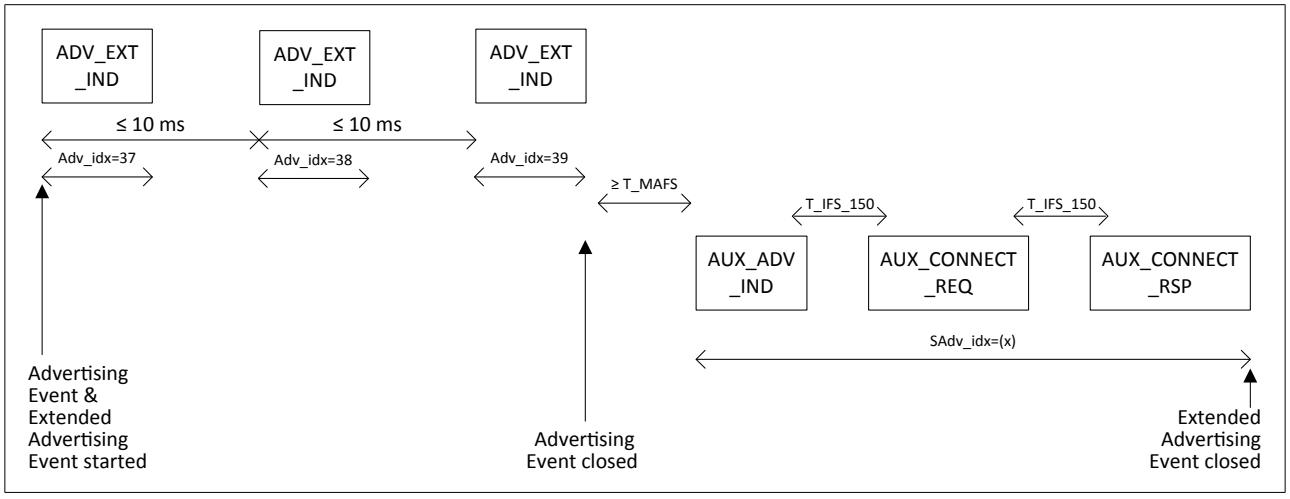



*Figure 4.20: Connectable directed advertising event using ADV_EXT_IND PDUs and AUX_ADV_IND PDUs containing advertising data with an AUX_CONNECT_REQ PDU*

#### 4.4.2.5 Scannable undirected event type

When the scannable undirected advertising event type is used, scannable undirected advertising indications (ADV_SCAN_IND or scannable undirected ADV_EXT_IND or ADV_DECISION_IND PDUs) are sent by the Link Layer.

If the Controller supports LL Privacy, then the requirements in Section 6.2.3 shall also be followed.

#### 4.4.2.5.1 Scannable undirected event type using ADV_SCAN_IND

The scannable undirected event type allows a scanner to respond with a scan request (SCAN_REQ PDU) to request additional information about the advertiser.

The Link Layer shall listen on the same primary advertising channel index for requests from scanners. Any CONNECT_IND PDUs received shall be ignored.

If the advertiser receives a SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy it shall reply with a SCAN_RSP PDU on the same advertising channel index. After the SCAN_RSP PDU is sent or if the advertising filter policy did not allow processing the SCAN_REQ PDU the advertiser shall either move to the next used primary advertising channel index to send another ADV_SCAN_IND PDU, or close the advertising event.

The time between the beginning of two consecutive ADV_SCAN_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

The structure of an advertising event in which no SCAN_REQ PDU was received is shown in Figure 4.21 for the case in which all the primary advertising physical channels are used.

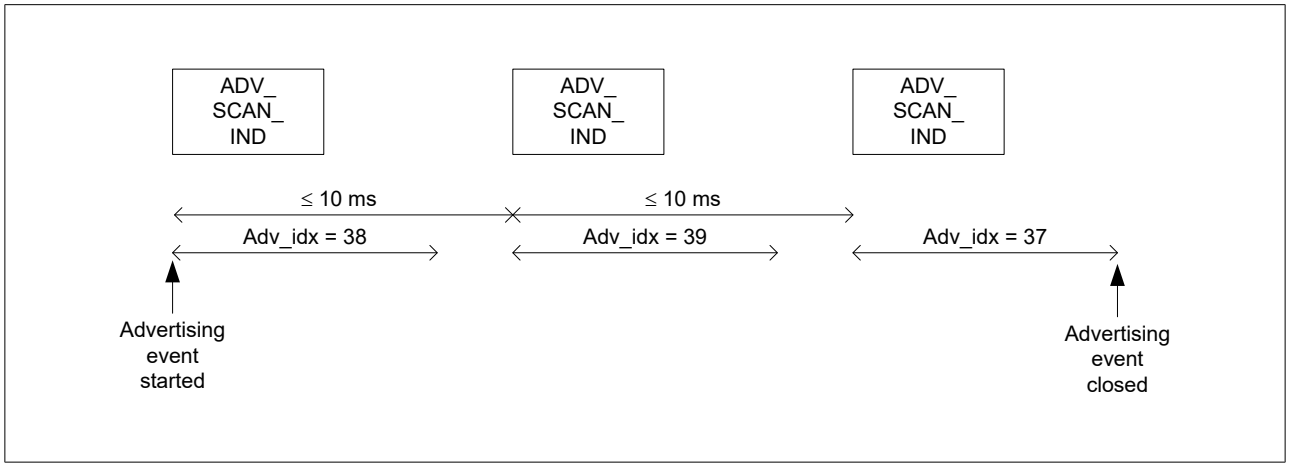



*Figure 4.21: Scannable undirected advertising event with only advertising PDUs*

Two example advertising events during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.22 and in Figure 4.23 for the case in which all the primary advertising physical channels are used.

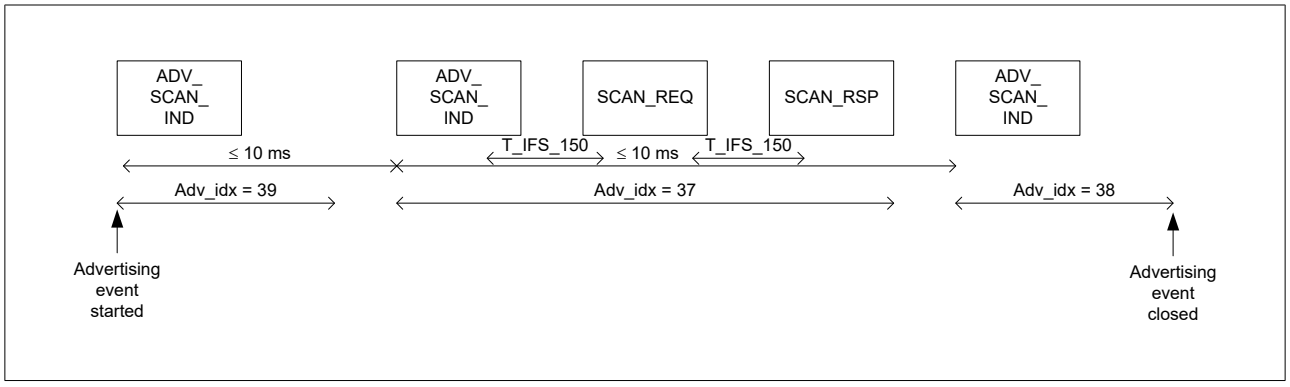



*Figure 4.22: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event*

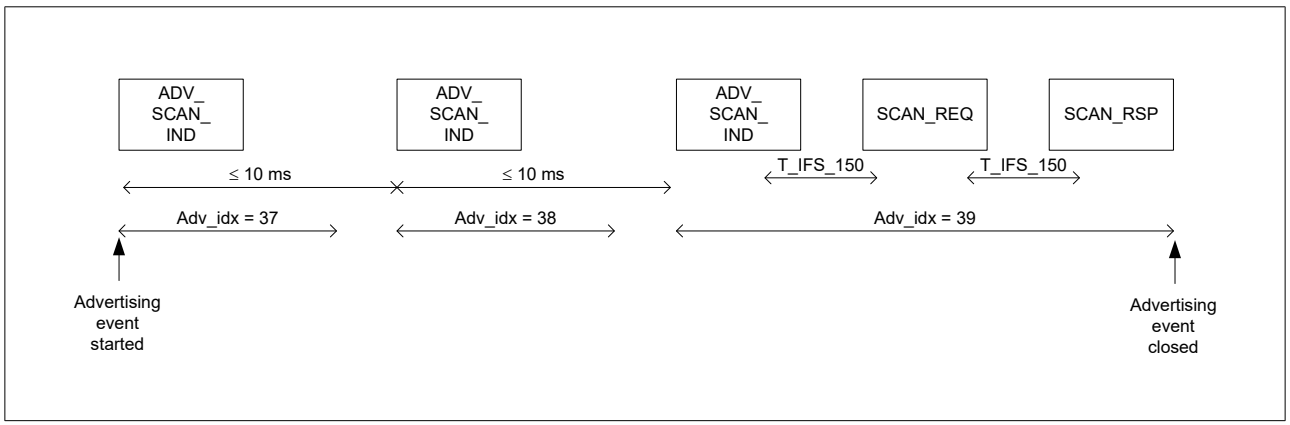



*Figure 4.23: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs at the end of an advertising event*

#### 4.4.2.5.2 Scannable undirected event type using ADV_EXT_IND or ADV_DECISION_IND

The scannable undirected event type using the ADV_EXT_IND or ADV_DECISION_IND PDU allows any scanner to respond with a scan request to receive scan response data on the secondary advertising physical channel.

A scanner may send a scan request using the AUX_SCAN_REQ PDU on the same secondary advertising channel index as the received AUX_ADV_IND PDU pointed to by the ADV_EXT_IND PDU.

After every AUX_ADV_IND PDU sent by the advertiser, the advertiser shall listen for AUX_SCAN_REQ PDUs on the same secondary advertising channel index from scanners. Any AUX_CONNECT_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy, it shall reply with an AUX_SCAN_RSP PDU on the same secondary advertising channel index prior to the start of the next advertising event. After the AUX_SCAN_RSP PDU is sent, or if the

advertising filter policy prohibits processing the AUX_SCAN_REQ PDU, the advertising event shall be closed. Any AUX_CONNECT_REQ PDUs on the secondary advertising physical channel shall be ignored.

The time between the beginning of two consecutive ADV_EXT_IND or ADV_DECISION_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

Figure 4.24 shows an advertising event in which no AUX_SCAN_REQ PDU is received.

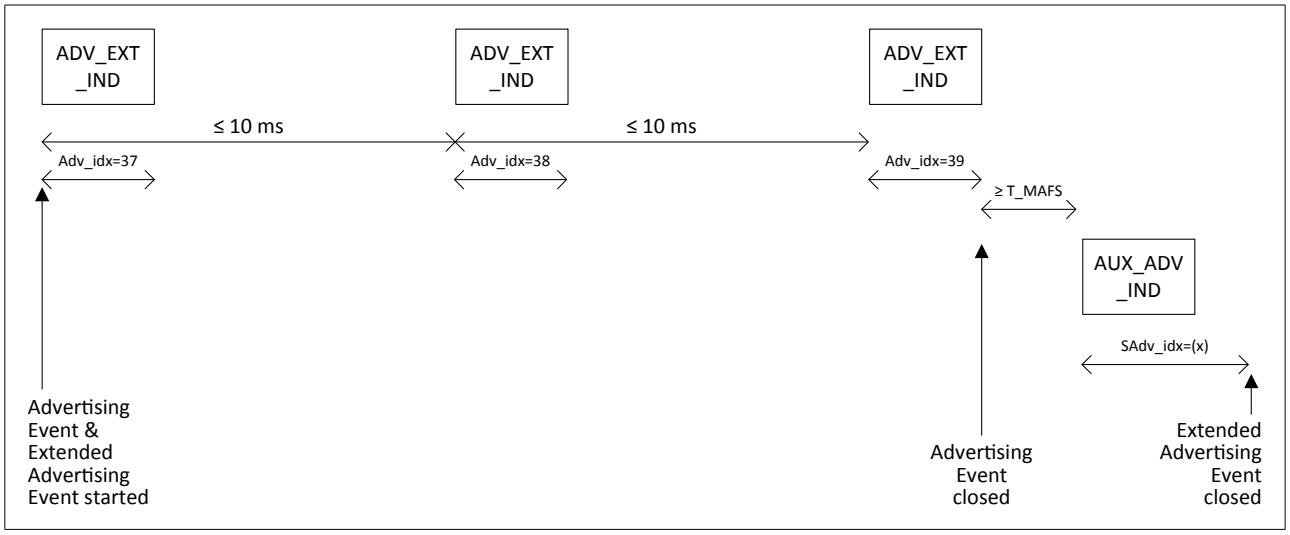



*Figure 4.24: Scannable undirected advertising event using the ADV_EXT_IND PDU and AUX_ADV_IND PDUs where no scan request is received*

An example advertising event during which an AUX_SCAN_REQ PDU is received and an AUX_SCAN_RSP PDU is sent on the secondary advertising physical channel is shown in Figure 4.25.

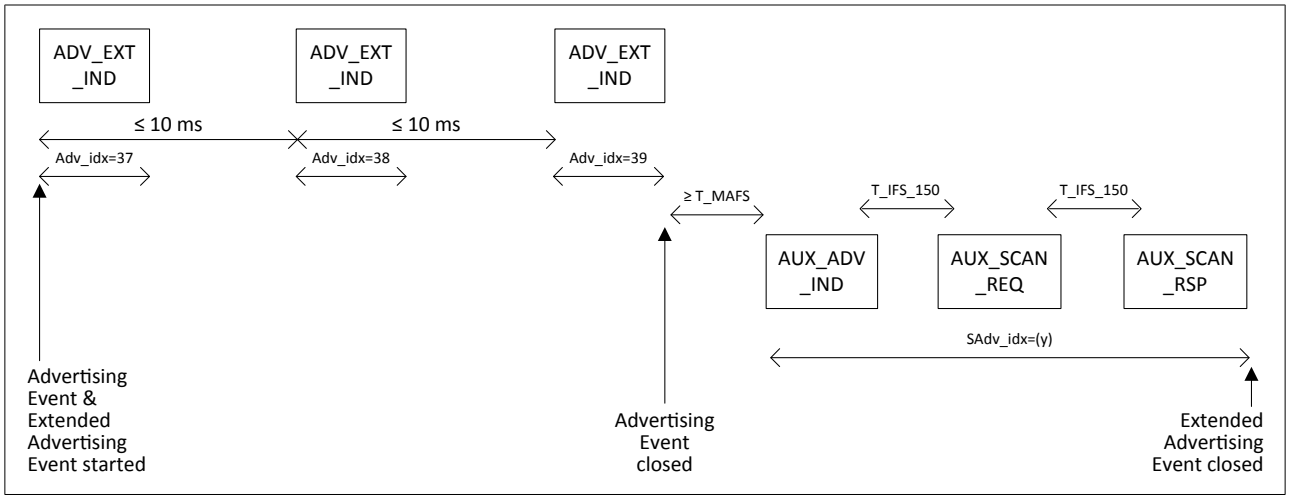



*Figure 4.25: Scannable undirected advertising event with ADV_EXT_IND and AUX_ADV_IND PDUs where a scan request is received*

#### 4.4.2.6 Non-connectable and non-scannable undirected event type

When the non-connectable and non-scannable undirected advertising event type is used, non-connectable and non-scannable undirected advertising indications (ADV_NONCONN_IND or non-connectable and non-scannable undirected ADV_EXT_IND or ADV_DECISION_IND PDUs, either with or without an auxiliary AUX_ADV_IND PDU) are sent by the Link Layer.

The non-connectable and non-scannable undirected event type allows a scanner to receive information from the advertiser. This information is contained either in the ADV_NONCONN_IND PDU or in an AUX_ADV_IND PDU pointed to by the AuxPtr field of the ADV_EXT_IND or ADV_DECISION_IND PDU.

The advertiser shall either move to the next used primary advertising channel index or close the advertising event after each ADV_NONCONN_IND, ADV_EXT_IND, or ADV_DECISION_IND PDU that is sent. The Link Layer does not listen, and therefore cannot receive any requests from scanners or initiators.

The time between the beginning of two consecutive ADV_NONCONN_IND, ADV_EXT_IND, or ADV_DECISION_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

When using an ADV_EXT_IND PDU with an AUX_ADV_IND PDU, the Controller shall decide which PDU contains the AdvA field and should make this choice based on overall efficient use of the medium.

An illustration of a non-connectable and non-scannable undirected advertising event is shown in Figure 4.26 for the case in which all the primary advertising physical channels are used.

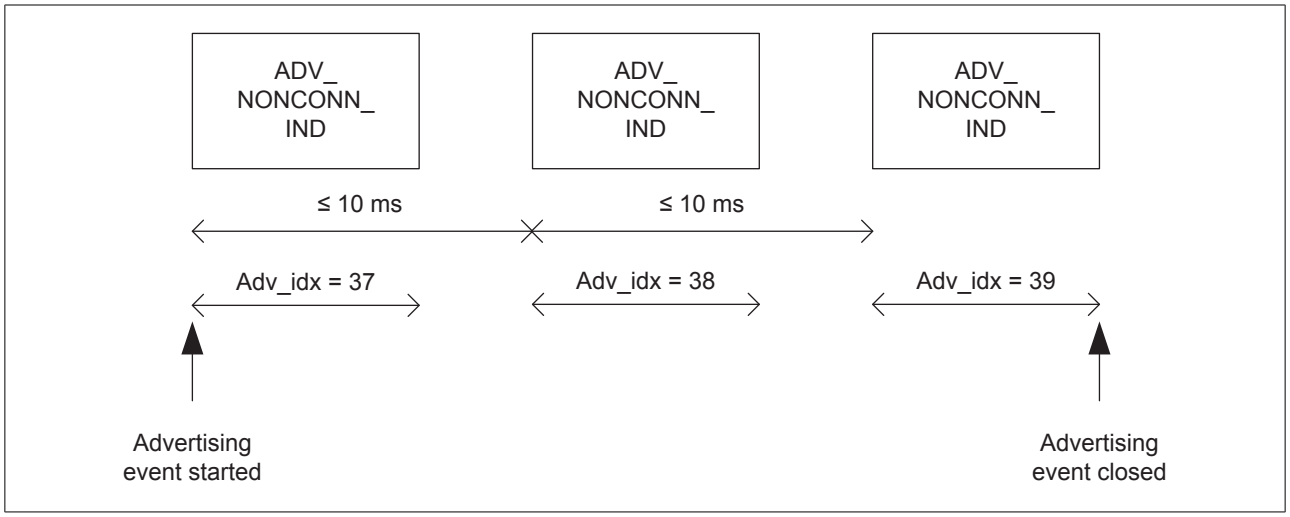



*Figure 4.26: Non-connectable and non-scannable undirected advertising event using ADV_NONCONN_IND PDUs*

Figure 4.27 shows an example of a non-connectable and non-scannable undirected ADV_EXT_IND PDU.

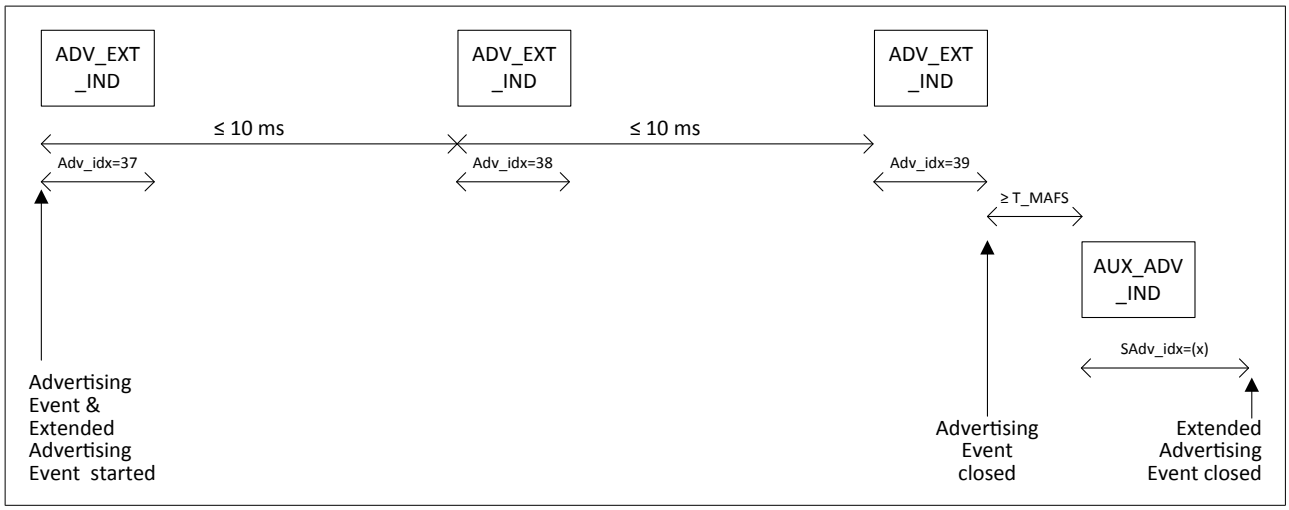



*Figure 4.27: Non-connectable and non-scannable undirected advertising event using the ADV_EXT_IND PDU*

Figure 4.28 shows an example of a non-connectable and non-scannable undirected ADV_EXT_IND PDU where the Host advertising data is fragmented using the AUX_CHAIN_IND PDU.

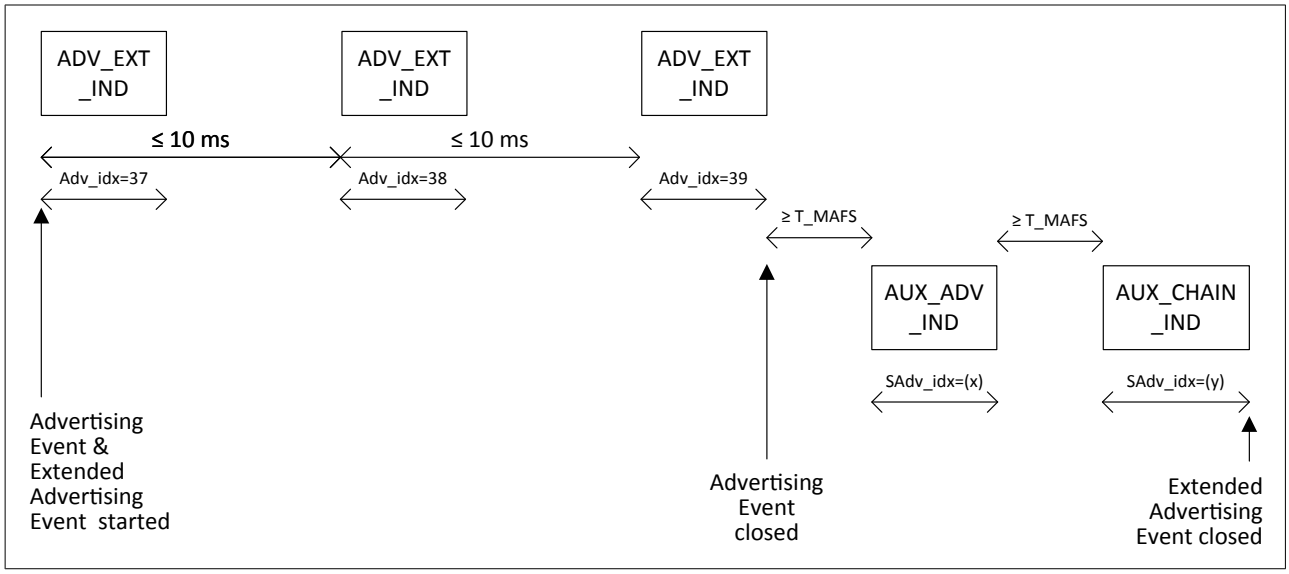



*Figure 4.28: Non-connectable and non-scannable undirected advertising event using the ADV_EXT_IND PDU with fragmented Host advertising data*

If the Controller supports LL Privacy, then the requirements in Section 6.2.3 shall also be followed.

#### 4.4.2.7 Connectable undirected event type

The connectable undirected advertising event type using the ADV_EXT_IND or ADV_DECISION_IND PDU allows an initiator to respond with a connect request to establish an ACL connection on the secondary advertising physical channel.

An initiator may send a connect request using the AUX_CONNECT_REQ PDU on the same secondary advertising physical channel as the AUX_ADV_IND PDU to request the Link Layer to enter the Connection state.

After every AUX_ADV_IND PDU sent related to this event by the advertiser, the advertiser shall listen for AUX_CONNECT_REQ PDUs on the same secondary advertising channel index. Any AUX_SCAN_REQ PDUs received shall be ignored.

If the advertiser receives an AUX_CONNECT_REQ PDU that contains its device address from an initiator allowed by the advertising filter policy and if the advertiser is not already connected to that initiator device address, then it shall reply with an AUX_CONNECT_RSP PDU on the same secondary advertising channel index. After the AUX_CONNECT_RSP PDU is sent the Link Layer shall exit the Advertising state and transition to the Connection state in the Peripheral Role as defined in Section 4.5.5.

The time between the beginning of two consecutive ADV_EXT_IND or ADV_DECISION_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

Figure 4.29 shows an advertising event in which no AUX_CONNECT_REQ PDU is received.

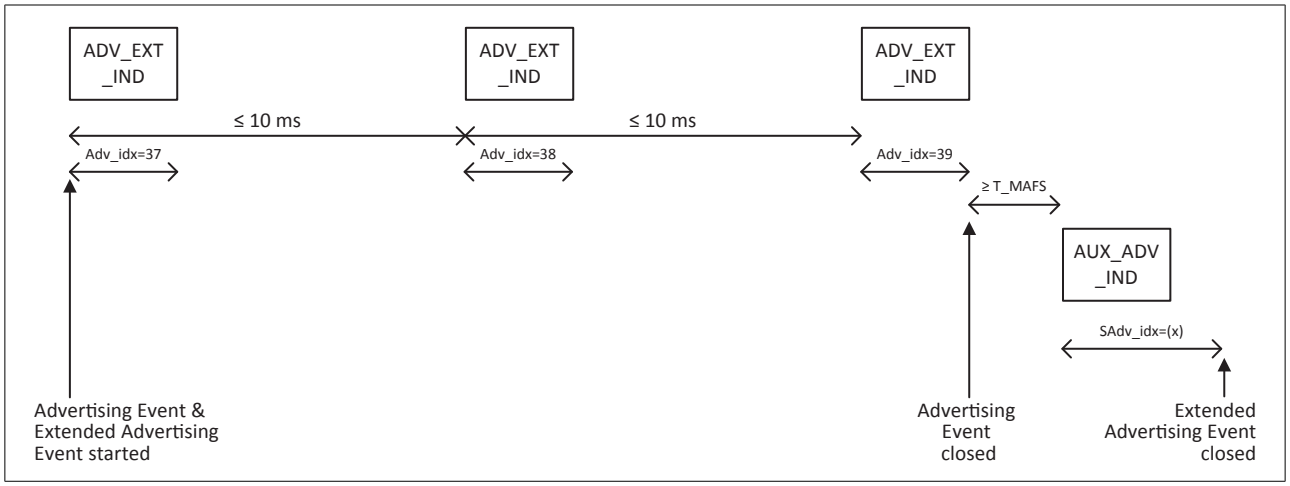



*Figure 4.29: Connectable undirected advertising event using the ADV_EXT_IND PDUs and AUX_ADV_IND PDU containing advertising data*

Figure 4.30 illustrates an advertising event during which an AUX_CONNECT_REQ PDU is received and an AUX_CONNECT_RSP PDU is sent on the secondary advertising channel index.

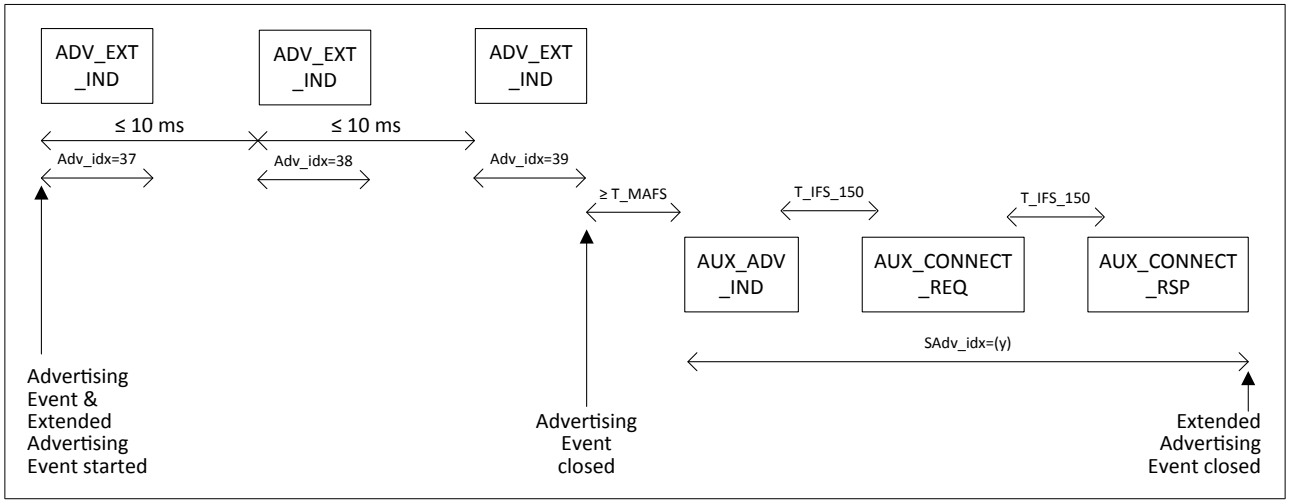



*Figure 4.30: Connectable undirected advertising using ADV_EXT_IND PDUs when an AUX_CONNECT_REQ PDU is received*

If the Controller supports LL Privacy, then the requirements in Section 6.2.4 shall also be followed.

#### 4.4.2.8 Scannable directed event type

The scannable directed advertising event type using the ADV_EXT_IND PDU allows a specific scanner to respond with a scan request to receive scan response data on the secondary advertising physical channel.

After every AUX_ADV_IND PDU sent, the advertiser shall listen for an AUX_SCAN_REQ PDU on the same secondary advertising channel index from the targeted scanner.

If the advertiser receives an AUX_SCAN_REQ PDU that contains its device address and the scanner's device address is contained in the AUX_ADV_IND PDU, it shall reply with an AUX_SCAN_RSP PDU on the same secondary advertising channel index prior to the next advertising event. The AUX_SCAN_RSP PDU shall contain the scan response data. AUX_SCAN_REQ PDUs from any other scanner or any AUX_CONNECT_REQ PDUs received shall be ignored.

The time between the beginning of two consecutive ADV_EXT_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

See Section 4.4.2.5.2 for a description on how the AUX_SCAN_REQ packet is used in conjunction with the AUX_SCAN_RSP packets on the secondary advertising physical channel.

If the Controller supports LL Privacy, then the requirements in Section 6.2.5 shall also be followed.

#### 4.4.2.9 Non-connectable and non-scannable directed event type

The non-connectable and non-scannable directed advertising event type using the ADV_EXT_IND PDU (either with or without an auxiliary AUX_ADV_IND PDU) allows an advertiser to send non-connectable and non-scannable directed ADV_EXT_IND PDUs on the primary advertising physical channel with any advertising data sent on the secondary advertising physical channel targeted for a specific scanner.

The Controller should choose which PDU contains the AdvA or TargetA field based on overall efficient use of the medium.

The Link Layer does not listen, and therefore cannot receive any requests from scanners or initiators.

If the Controller supports LL Privacy, then the requirements in Section 6.2.5 shall also be followed.

#### 4.4.2.10 Advertising Sets

The advertiser's Host may instruct the Link Layer to interleave advertising events. Advertising data belonging together is called an advertising set. The Link Layer may support multiple advertising sets, with each set having different advertising parameters such as advertising PDU type, advertising interval, and PHY.

When advertising with the ADV_EXT_IND, AUX_ADV_IND, AUX_SYNC_IND, or AUX_SYNC_SUBEVENT_IND PDUs, the advertising set or group of sets is identified by the Advertising SID subfield of the ADI field. The Link Layer shall set the Advertising SID subfield as directed by the Host.

The scanner may filter advertisements based on the Advertising SID.

The advertising events for each advertising set are considered a separate instance of the Advertising State and each have their own Advertising Interval (see Section 4.4.2.2.1).

Figure 4.31 illustrates an example of advertising using multiple advertising sets.

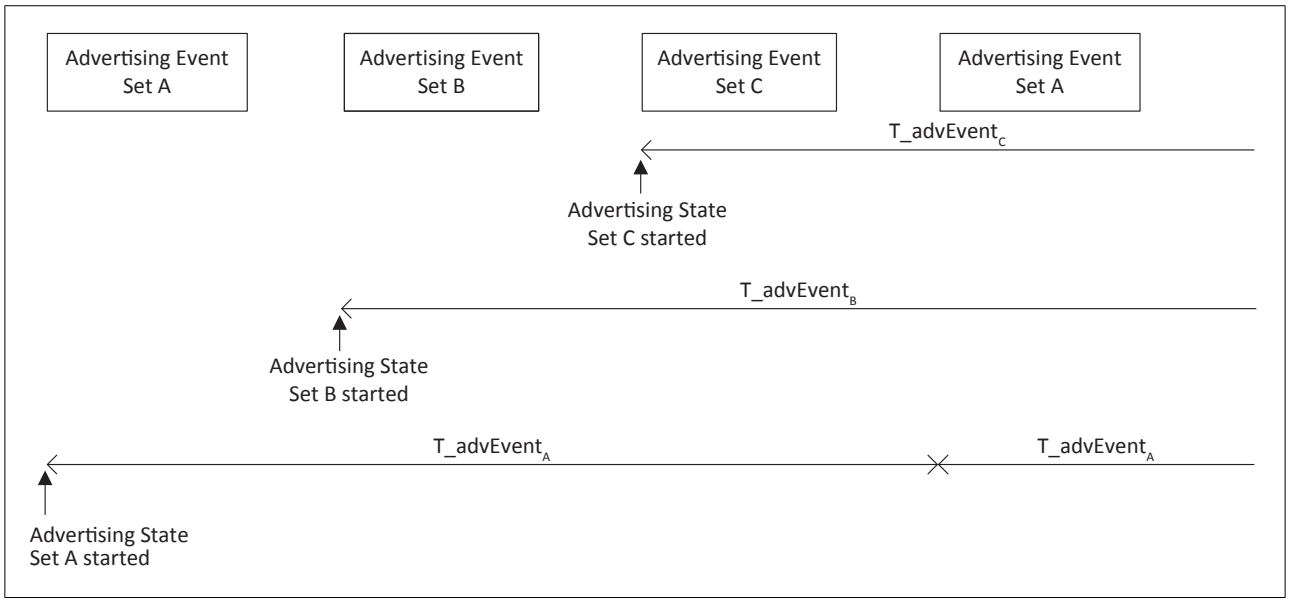



*Figure 4.31: Multiple advertising sets example*

On reset, all advertising sets are destroyed.

When an advertising set is created that includes advertising data, the Controller shall reserve sufficient resources to allow the set to contain at least 31 octets of advertising data. The Controller may release or reuse any unused portion of those resources at any time after the Host first specifies advertising data for that set or creates another advertising set.

When an advertising set is created that is scannable, the Controller shall reserve sufficient resources to allow the set to contain at least 31 octets of scan response data. The Controller may release or reuse those resources if the set is made non-scannable and may release or reuse any unused portion of those resources at any time after the Host first specifies scan response data for that set or creates another advertising set.

#### 4.4.2.11 Using AdvDataInfo (ADI)

The AdvDataInfo (ADI) field is used to identify advertising sets and duplicate AdvData in the AUX_ADV_IND, AUX_SYNC_IND, AUX_SYNC_SUBEVENT_IND, and AUX_SCAN_RSP PDUs. For scannable advertising events using the ADV_EXT_IND PDU, AdvData is not permitted in the AUX_ADV_IND PDU so the ADI only refers to the AdvData contained in the AUX_SCAN_RSP PDU. The Advertising SID is used to distinguish different advertising sets or groups of advertising sets from the same device. The advertiser may use the same Advertising SID for more than one advertising set, resulting in a group of advertising sets that cannot be distinguished by scanning devices using only the SID.

The Advertising DID for a given advertising set or periodic advertising train shall be initialized with a randomly chosen value. The advertising set and any associated periodic advertising shall have separate Advertising DIDs. Whenever the Host provides new advertising data, periodic advertising data, or scan response data for a given advertising set (whether it is the same as the previous data or not), the Advertising DID shall be updated. The new value shall be a randomly chosen value that is not the same as the most recently used value.

Note: Choosing Advertising DID field values randomly reduces the possibility of PDUs from different advertisers containing the same ADI field value.

Advertisers should update the Advertising DID when a periodic advertising train is enabled. Alternatively, the Host should enable periodic advertising before enabling advertising.

Note: If the Advertising DID does not change when a SyncInfo field is added to or removed from an advertising set, then scanners may fail to synchronize to periodic advertising because entries in the Advertising DID cache (see Section 4.4.3) mean they ignore the advertisements containing the SyncInfo field.

#### 4.4.2.12 Periodic advertising

When advertising data is required to be sent regularly at a fixed interval, periodic advertising is used. Two forms of periodic advertising are defined: periodic advertising without responses and periodic advertising with responses.

Periodic advertising without responses consists of advertisements sent at a fixed interval with the advertisement data changing from time to time. The AUX_SYNC_IND

and AUX_CHAIN_IND PDUs making up such a sequence of advertisements form a periodic advertising train.

Note: No AUX_SYNC_SUBEVENT_IND and AUX_SYNC_SUBEVENT_RSP PDUs are sent on a periodic advertising without responses train

Periodic Advertising with Responses (PAwR) consists of advertisements sent at a fixed interval with the advertisement data changing frequently, with responses being sent in response. The AUX_SYNC_SUBEVENT_IND and AUX_SYNC_SUBEVENT_RSP PDUs making up such a sequence of advertisements form a PAwR train.

Note: No AUX_SYNC_IND and AUX_CHAIN_IND PDUs are sent on a PAwR train.

The advertising set containing the Periodic Advertising is identified by the AdvDataInfo field of ADV_EXT_IND PDUs that point to AUX_ADV_IND PDUs containing the SyncInfo field. The AUX_SYNC_IND PDUs, the AUX_SYNC_SUBEVENT_IND PDUs, and the PDUs that point to them shall always be sent on the same PHY. The PHY used, the Access Address, and the CRCInit value of the AUX_SYNC_IND PDUs and the AUX_SYNC_SUBEVENT_IND PDUs for a periodic advertising train shall not change while that train is enabled. Advertising pointing to a periodic advertising train shall not be anonymous. Each time that a periodic advertising train is enabled, the Controller shall transmit at least one AUX_ADV_IND PDU pointing to the first AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU of that train; after this, there is no requirement whether or when to transmit advertising PDUs pointing to the train. If the Periodic Advertising is with responses, then the ACAD field of the AUX_ADV_IND PDUs shall contain Periodic Advertising Response Timing Information (see Section 1.24 of [1]).

Note: The SyncInfo field is only allowed in non-scannable and non-connectable advertisements.

#### 4.4.2.12.1 Trains without responses

When periodic advertising without responses takes place, the advertiser shall send AUX_SYNC_IND PDUs at regular intervals (the periodic advertising interval - see Section 4.4.2.2.3), which are described in the SyncInfo field of AUX_ADV_IND PDUs.

The Host may send periodic advertising data to the Link Layer. This advertising data is placed by the Link Layer in the periodic AUX_SYNC_IND PDUs and their subordinate sets. The Link Layer shall repeat the last advertising data sent by the Host until it receives new advertising data. The AUX_SYNC_IND PDUs are continuously sent out until the Host directs the Link Layer to terminate the periodic advertising train. When including the ADI field is enabled by the Host and where the periodic advertising data is not changed for every periodic advertising event, the Controller should include the ADI field in the AUX_SYNC_IND PDU.

Packets in a periodic advertising train may include a Constant Tone Extension. The Host may enable this before the periodic advertising train starts or may enable or disable inclusion of the Constant Tone Extension while the periodic advertising is in progress.

When an advertising set is first configured for periodic advertising, the Controller shall either reserve sufficient resources to allow the set to contain at least 31 octets of advertising data or else shall not allow periodic advertising on that advertising set. The Controller may release or reuse any unused portion of those resources at any time after the Host first specifies periodic advertising data for that set or creates another advertising set.

Figure 4.32 illustrates an example of periodic advertising.

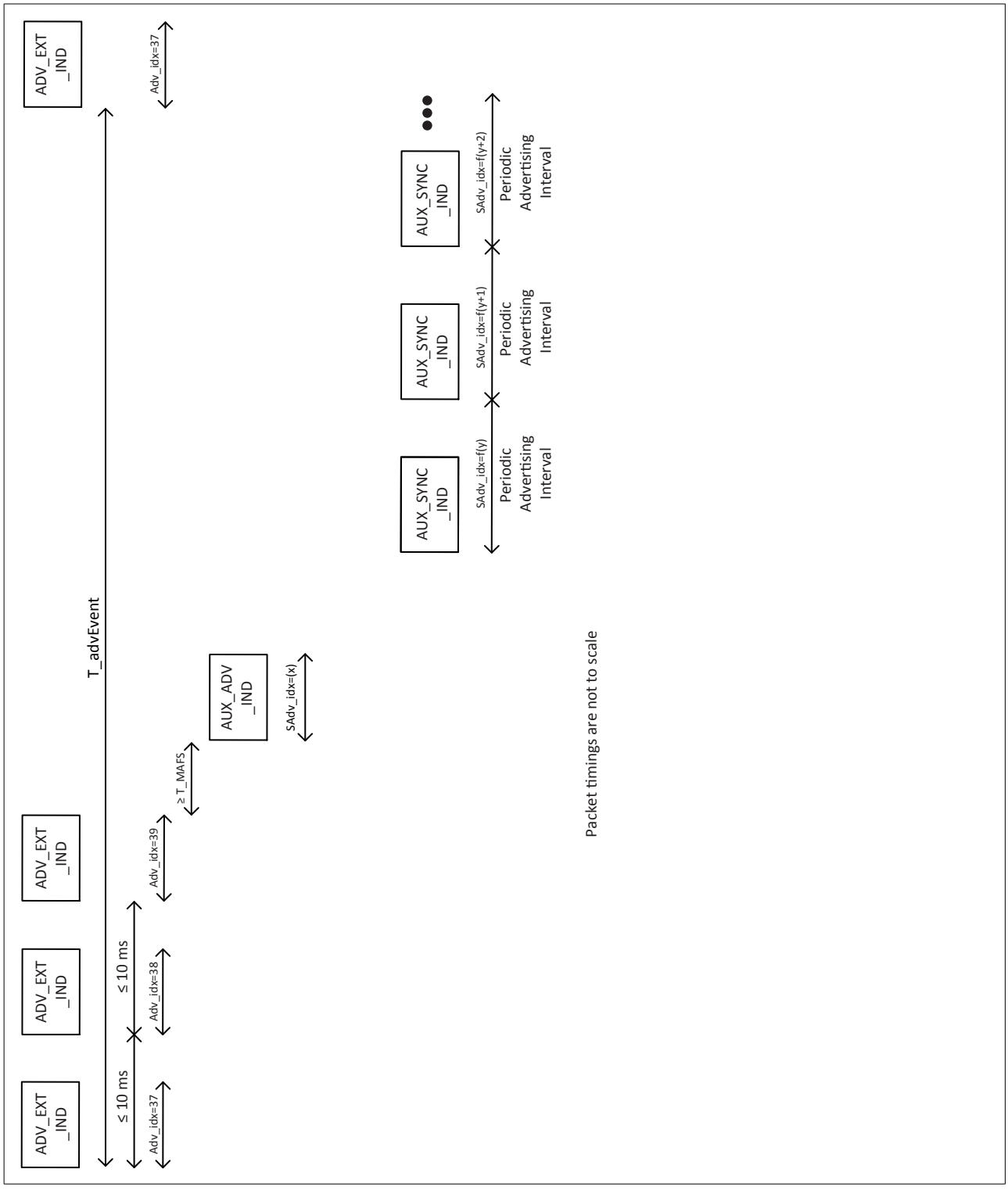



*Figure 4.32: Example of periodic advertising without responses*

#### 4.4.2.12.2 Trains with responses

When PAwR takes place, the advertiser shall send AUX_SYNC_SUBEVENT_IND PDUs at regular intervals defined by the periodic advertising interval and the periodic advertising subevent interval (see Section 4.4.2.2.4). The PHY used and the CRCInit value of the AUX_SYNC_SUBEVENT_IND and AUX_SYNC_SUBEVENT_RSP PDUs for a periodic advertising train shall not change while that train is enabled. The Access Address value of the AUX_SYNC_SUBEVENT_IND shall not change while that train is enabled. The Access Address value of the AUX_SYNC_SUBEVENT_RSP shall be set by each synchronized device to the RspAA.

The Host may send periodic advertising data to the Link Layer. This advertising data is placed by the Link Layer in the periodic AUX_SYNC_SUBEVENT_IND PDUs. The Link Layer shall send the advertising data sent by the Host once and then immediately discard the advertising data. The AUX_SYNC_SUBEVENT_IND PDUs should be continuously sent out until the Host directs the Link Layer to terminate the periodic advertising train. If the Host has not provided any data to the Link Layer, then an AUX_SYNC_SUBEVENT_IND_PDU should still be sent with an empty Payload field.

When instructed by the Host, the Link Layer shall transmit an AUX_CONNECT_REQ PDU instead of an AUX_SYNC_SUBEVENT_IND PDU, with the InitA field set to the address of the advertiser and the AdvA field set to the address of the synchronized device. The Link Layer shall transmit the AUX_CONNECT_REQ PDU even if the Host also instructed it to transmit data on the same subevent. After sending the AUX_CONNECT_REQ PDU, the periodic advertiser shall wait for the synchronized device to send an AUX_CONNECT_RSP PDU T_IFS_150 from the end of the AUX_CONNECT_REQ PDU on the same secondary advertising channel index. If an AUX_CONNECT_RSP PDU is received from the synchronized device and the periodic advertiser is not already connected to the synchronized device's device address, then the Link Layer of the periodic advertiser shall start a new state machine that immediately transitions to the Connection state in the Central role as defined in Section 4.5.4; the existing state machine shall remain in the Advertising state. If an AUX_CONNECT_RSP PDU is not received from the synchronized device, then the request to connect to the synchronized device is discarded.

If the synchronized device receives an AUX_CONNECT_REQ PDU from the periodic advertiser that contains its device address and is not already connected to the same device address, then the synchronized device shall reply with an AUX_CONNECT_RSP PDU on the same secondary advertising channel index with the AdvA field set to the address of the synchronized device and the TargetA field set to the address of the advertiser. After the AUX_CONNECT_RSP PDU is sent, the Link Layer shall start a new state machine that immediately transitions to the Connection state in the Peripheral role as defined in Section 4.5.5; the existing state machine shall remain in the Synchronization state.

#### 4.4.2.13 Requirements

A device that supports Advertising State shall support transmitting advertising event types as specified in Table 4.3.

| Advertising event type | Requirements |
|---|---|
| Non-connectable non-scannable undirected | May be supported; shall be supported if the device supports Periodic Advertising or does not support Connection State (Peripheral role). |
| Non-connectable non-scannable directed | Shall be supported if and only if the device supports both LE Extended Advertising and non-connectable non-scannable undirected type. |
| Scannable undirected | May be supported. |
| Scannable directed | Shall be supported if and only if the device supports both LE Extended Advertising and scannable undirected type. |
| Connectable directed<br>Connectable and scannable undirected | Shall be supported if and only if the device supports Connection State (Peripheral role). |
| Connectable undirected | Shall be supported if and only if the device supports both LE Extended Advertising and connectable directed type. |

*Table 4.3: Requirements for supporting advertising event types*

If a device supports the LE Extended Advertising feature, then for each advertising event type that it supports which can use either a legacy PDU or ADV_EXT_IND (i.e., non-connectable and non-scannable undirected, scannable undirected, and connectable directed), the device shall support transmitting that advertising event type using both PDU types.

### 4.4.3 Scanning state

The Link Layer shall enter the Scanning state when directed by the Host. When scanning, the Link Layer shall listen on the primary advertising physical channel for the types of PDU and on the PHYs that have been indicated by the Host. There are two types of scanning, determined by the Host: passive and active.

There are no strict timing or advertising channel index selection rules for scanning.

During scanning, the Link Layer should listen on a primary advertising channel index for the duration of the scan window, *scanWindow*. The scan interval, *scanInterval*, is defined as the interval between the start of two consecutive scan windows.

The Link Layer should listen for the complete *scanWindow* every *scanInterval* as directed by the Host unless there is a scheduling conflict. In each scan window, the Link Layer should scan on a different primary advertising channel index than the one

used in the previous scan window. The Link Layer shall use all the primary advertising channel indices.

The *scanWindow* and *scanInterval* parameters shall be less than 40.96 s. The *scanWindow* shall be less than or equal to the *scanInterval*. If the *scanWindow* and the *scanInterval* parameters are set to the same value by the Host, the Link Layer should scan continuously.

The scanning filter policy shall apply when receiving an advertising PDU or scan response PDU when scanning.

On receiving a PDU with the AuxPtr field present, the scanner should also listen for the auxiliary PDU it points to (provided that it supports the PHY specified in the AuxPtr field) and should then attempt to receive the entire subordinate set of the PDU. While doing so, it shall perform the window widening specified in Section 4.2.4. If it does not support the specified PHY or the value is reserved for future use, it shall not listen for the auxiliary PDU and shall behave as if it listened for it but failed to receive it.

When a scanner receives ADV_EXT_IND PDUs that contain an AuxPtr field, it may either always listen for the auxiliary packet or may sometimes skip listening. In the latter case, the following requirements shall apply.

For each Advertising SID value received:

- The Controller shall keep a cache of one or more recent Advertising DID values used by each advertising device (for this purpose, all anonymous advertising is treated as being from a single device different to all real devices) and shall update them whenever a PDU containing an ADI field is received. The Controller may delete any cache entry at any time. The Controller should delete the cache entry relating to an ADV_EXT_IND PDU if it fails to receive that PDU's entire subordinate set.
- The Controller may only skip listening for the auxiliary packet if the cache has an entry specifying the Advertising DID value in the ADI field being used by a device; if the ADV_EXT_IND PDU contains an AdvA field, the entry shall be for that device. Otherwise, the Controller shall not skip listening for the auxiliary packet.
- Irrespective of the cache contents, the Controller should sometimes listen for the AUX_ADV_IND PDU in case another advertiser has started using the same Advertising DID value or the existing advertiser has made a significant change to the Extended Header field (e.g., included the SyncInfo field).

If the Controller supports LL Privacy, then the requirements in Section 6.3 shall also be followed.

#### 4.4.3.1 Passive scanning

When in passive scanning, the Link Layer will only receive packets; it shall not send any packets.

#### 4.4.3.2 Active scanning

In active scanning, the Link Layer shall listen for advertising PDUs and, depending on the advertising PDU type, it may request an advertiser to send additional information.

After entering the Scanning State, if the Link Layer receives a scannable PDU (i.e. an ADV_IND, ADV_SCAN_IND, or scannable AUX_ADV_IND PDU) from an advertiser allowed by the scanning filter policy, it shall respond with a scan request PDU and then listen for the scan response PDU. It shall continue to respond to the same advertiser until it has successfully received the scan response PDU. It may then either respond to or ignore subsequent scannable PDUs from the same advertiser. It should ignore them if either they are legacy PDUs or if the Advertising DID field has not changed since the last advertisement from the same advertiser with the same Advertising SID field; it should not ignore them otherwise.

The Link Layer shall only send a SCAN_REQ PDU to an advertiser from which an ADV_IND PDU or ADV_SCAN_IND PDU is received. The Link Layer shall only send an AUX_SCAN_REQ PDU to an advertiser from which a scannable AUX_ADV_IND is received. The Link Layer shall ignore a scannable AUX_ADV_IND PDU if the TargetA field is present and it does not match the Link Layer's device address.

The scanner shall run a backoff procedure to minimize collisions of scan request PDUs from multiple scanners. An example of such a procedure is given in the following paragraphs.

The backoff procedure uses two parameters, *backoffCount* and *upperLimit,* to restrict the number of scan request PDUs sent when collisions occur on scan response PDUs. Upon entering the Scanning State, the *upperLimit* and *backoffCount* are set to one.

On every received ADV_IND, ADV_SCAN_IND, or scannable AUX_ADV_IND PDU that is allowed by the scanning filter policy and for which a scan request PDU is to be sent, the *backoffCount* is decremented by one until it reaches the value of zero. The scan request PDU is only sent when *backoffCount* becomes zero.

After sending a scan request PDU the Link Layer listens for a scan response PDU from that advertiser. If the scan response PDU was not received from that advertiser, it is considered a failure; otherwise it is considered a success. On every two consecutive failures, the *upperLimit* is doubled until it reaches the value of 256. On every two consecutive successes, the *upperLimit* is halved until it reaches the value of one. After success or failure of receiving the scan response PDU, the Link Layer sets *backoffCount* to a new (pseudo-)random integer between one and *upperLimit*.

If a device uses a different backoff algorithm it shall share the medium responsibly.

Two illustrations of advertising events using all the advertising channel indices during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.13 and in Figure 4.14.

#### 4.4.3.3 Advertising sets

The ADV_EXT_IND PDU may contain an ADI field. When the ADI field is present, it can be used to identify advertisement data that belong to the same set or group of sets. This is specified in the Advertising SID subfield in the ADI. The Advertising SID is set by the Host of the advertiser.

#### 4.4.3.4 Scanning for periodic advertisements

When instructed by the Host, the scanner shall look for periodic advertising synchronization information located in the SyncInfo field and, for PAwR, the ACAD field of AUX_ADV_IND PDUs. If the syncPacketWindowOffset value of the SyncInfo is zero, the periodic advertising synchronization information is incomplete and the scanner should listen for a subsequent advertisement to be able to obtain the complete information. When it has received the complete information it shall start a new state machine that immediately transitions to the Synchronization State; the existing state machine shall remain in the Scanning State.

The Host may instruct the Controller not to synchronize to periodic advertising trains with certain types of Constant Tone Extensions or without a Constant Tone Extension. If the Controller receives an AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND with such a Constant Tone Extension while synchronizing, the synchronization attempt has failed and shall cease. Once synchronized, the presence or type of Constant Tone Extension shall not affect synchronization.

#### 4.4.3.5 Advertising reports

The Link Layer shall send an advertising report to the Host for each advertising PDU on the primary advertising physical channel that is accepted by the scanning filter policy (see Section 4.3.3) and for each scan response PDU from an advertiser that is accepted by the scanning filter policy, except where stated otherwise in this section. The Controller shall send the report even if it does not respond to the advertiser.

If the Controller receives an ADV_EXT_IND PDU with an AuxPtr field, it shall delay the report until after the corresponding AUX_ADV_IND PDU has been received and the report shall combine the information in the PDUs; if the Controller does not listen for or does not receive the AUX_ADV_IND PDU, no report shall be generated. The advertising report shall contain at least the advertiser's device address and all of the advertising data or scan response data (including any data in any subordinate AUX_CHAIN_IND PDUs that were received) if present.

Note: The Controller can use more than one HCI event to send the report if, for example, the total data does not fit within a single event.

Note: The requirements above and those in Section 4.6.12 mean that the Controller must be able to store and report to the Host at least 251 octets of data from a single advertisement or scan response (irrespective of the number of PDUs used to transmit the data) if the Controller supports LE extended advertising and 31 octets otherwise.

The Host may request that duplicate advertising reports are filtered and so not sent. The Controller may nevertheless send a duplicate of any report.

Where a received ADV_EXT_IND PDU contains an ADI field, a duplicate advertising report is an advertising report for the same device address where the previous report that contained an ADI value with the same Advertising SID also had the same Advertising DID. For this purpose, all anonymous advertising is treated as being from a single device different to all non-anonymous devices.

Where the ADV_EXT_IND PDU does not contain an ADI field or a legacy PDU was received, a duplicate advertising report is an advertising report for the same device address while the Link Layer stays in the Scanning state.

Advertising data reports and scan data reports shall be processed separately when determining duplicate advertising reports; i.e., an advertising data report shall not be treated as a duplicate of a scan response report or vice versa.

In either case the actual data may change; advertising data or scan response data is not considered significant when determining duplicate advertising reports. However, if not all the subordinate set of an advertisement or scan response was received (i.e., an incomplete report), a subsequent report that contains more of the data should not be treated as a duplicate of the incomplete report.

#### 4.4.3.6 Decision PDU scanning

The scanning or initiating filter policy may direct the Link Layer to listen for decision PDUs and, if so, may direct it to ignore other types of PDU on the primary advertising physical channel.

If a device has more than one active state machine, the following requirements apply to each state machine independently; for example, the same PDU can be ignored by one state machine but processed by another.

When the Link Layer receives a decision PDU when scanning or initiating and is not ignoring all decision PDUs for that purpose, it shall use decision instructions provided by the Host to determine whether to accept or reject the PDU. If the Controller does not support HCI then the decision instructions are specified in a vendor-specific way. If

the Host has not provided any decision instructions, then the Link Layer shall ignore all decision PDUs.

If the decision instructions indicate that the Link Layer shall accept a particular decision PDU, the Link Layer shall then behave in the same way as if it had received an ADV_EXT_IND PDU with the same Extended Header and AdvMode fields, except that the filter policy for decision PDUs continues to apply. If the decision instructions indicate that the Link Layer shall reject a particular decision PDU, or the filter policy means the Link Layer is ignoring the decision PDU, the Link Layer shall then behave as if it had not received that PDU: if scanning, it shall not report the advertisement to the Host, if active scanning, it shall not send a scan request PDU to the advertiser, and if initiating, it shall not connect to the advertiser.

#### 4.4.3.7 Requirements

A device that supports Scanning State shall support receiving advertising event types as specified in Table 4.4.

| Advertising event type | Requirements |
|---|---|
| Non-connectable non-scannable undirected | May be supported; shall be supported if the device supports Periodic Advertising or does not support either Connection State (Central role) or Active Scanning. |
| Non-connectable non-scannable directed | Shall be supported if and only if the device supports both LE Extended Advertising and non-connectable non-scannable undirected type. |
| Scannable undirected | May be supported; shall be supported if the device supports Active Scanning. |
| Scannable directed | Shall be supported if and only if the device supports both LE Extended Advertising and scannable undirected type. |
| Connectable directed | May be supported; shall be supported if the device supports Connection State (Central role). |
| Connectable undirected | Shall be supported if and only if the device supports both LE Extended Advertising and connectable directed type. |
| Connectable and scannable undirected | May be supported; shall be supported if the device supports Connection State (Central role) or Active Scanning. |

*Table 4.4: Requirements for supporting advertising event types*

If a device supports the LE Extended Advertising feature, then for each advertising event type that it supports which can use either a legacy PDU or ADV_EXT_IND (i.e., non-connectable and non-scannable undirected, scannable undirected, and connectable directed), the device shall support receiving that advertising event type using both PDU types.

#### 4.4.3.8 Monitoring Advertisers

When instructed by the Host, the Link Layer shall keep track of devices in the Monitored Advertisers List. When a device is entered in the Monitored Advertisers List and monitoring advertisers is enabled, a timer for that device is started. For each device in the Monitored Advertisers List, the Controller shall generate an event if, during the timeout period set by the Host, an advertisement has not been received from that device with an RSSI value above the RSSI low threshold set by the Host. These conditions are referred to as the loss-of-signal criteria, after which the entry shall be marked as awaiting an RSSI value above the RSSI high threshold and the timer shall be reset but not run. The timeout period is a value set in units of seconds by the Host.

If a device in the Monitored Advertisers List is awaiting an RSSI high threshold and an advertising packet is received with an RSSI value greater than or equal to the RSSI high threshold value, then the Controller shall generate an event, the entry is marked as not awaiting an RSSI high threshold, and the timer for this entry in the list is reset and runs.

Figure 4.33 shows the operation of monitoring advertisers as a state machine.

The Monitoring Advertisers feature operates independently from the Filter Accept List. For example, it is possible to use the Monitoring Advertisers feature on devices that are not in the Filter Accept List even if the Filter Accept List is enabled.

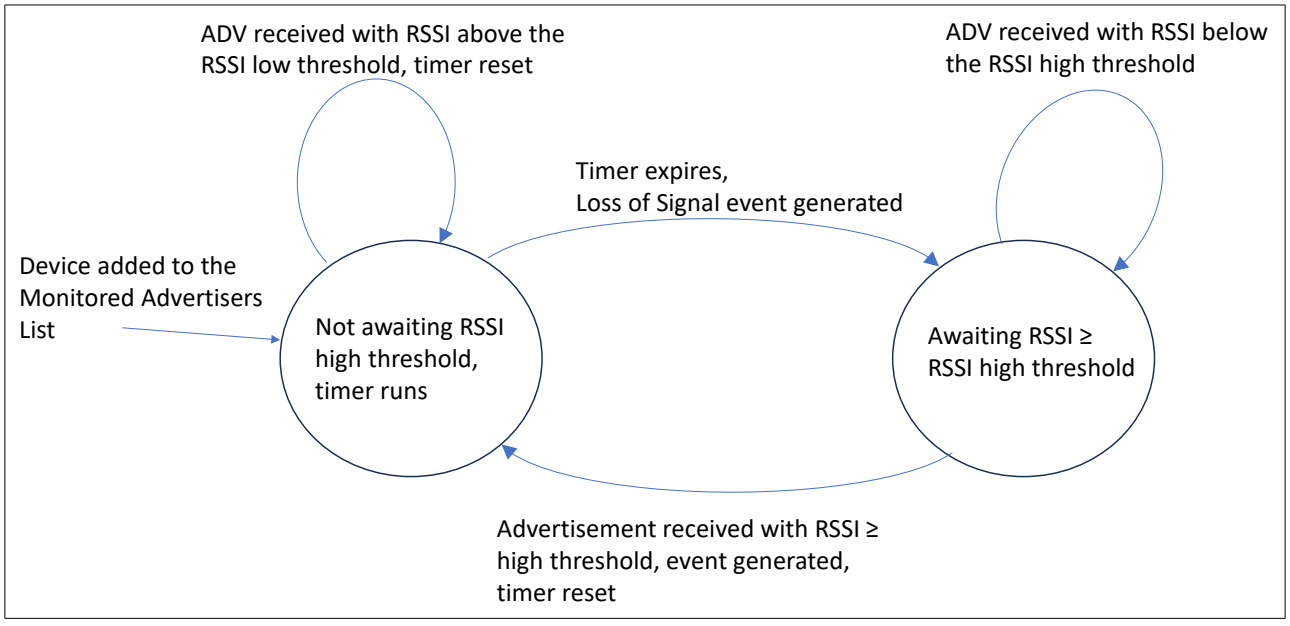



*Figure 4.33: State machine*

### 4.4.4 Initiating state

The Link Layer shall enter the Initiating state when directed by the Host. When initiating, the Link Layer shall listen on the primary advertising physical channel.

There are no strict timing or advertising channel index selection rules for initiators.

During initiating, the Link Layer listens on a primary advertising channel index for the duration of the scan window, *scanWindow*. The scan interval, *scanInterval*, is defined as the interval between the start of two consecutive scan windows.

The Link Layer should listen for the complete *scanWindow* every *scanInterval* as directed by the Host unless there is a scheduling conflict. In each scan window, the Link Layer should listen on a different primary advertising channel index. The Link Layer shall use all the primary advertising channel indices.

The *scanWindow* and *scanInterval* parameters shall be less than or equal to 40.96 s. The *scanWindow* shall be less than or equal to the *scanInterval*. If the *scanWindow* and the *scanInterval* parameters are set to the same value by the Host, the Link Layer should listen continuously.

Connection indications or requests in response to a connectable advertisement shall be sent on either the primary or secondary advertising physical channel depending on which advertising PDU contains an AdvA field. The following subsections describe the two procedures.

The initiator filter policy shall apply when receiving an advertising PDU while initiating a connection. If the Controller supports LL Privacy, then the requirements in Section 6.4 shall also be followed.

Note: If the Link Layer supports the Decision-Based Advertising Filtering feature, Section 4.4.3.6 also applies to initiating.

#### 4.4.4.1 Connect requests on the primary advertising physical channel

If an ADV_IND PDU is received that is allowed by the initiator filter policy, the initiator shall send a CONNECT_IND PDU to the advertiser. If an ADV_DIRECT_IND PDU containing the initiator's Link Layer device address and allowed by the initiator filter policy is received, the initiator shall send a CONNECT_IND PDU to the advertiser; otherwise it shall be ignored. However, in both cases, the initiator shall not send a CONNECT_IND PDU to a device address that it is already connected to.

After sending the CONNECT_IND PDU, the Link Layer shall exit the Initiating State, and shall transition to the Connection State in the Central Role as defined in Section 4.5.4.

#### 4.4.4.2 Connect requests on the secondary advertising physical channel

If a connectable ADV_EXT_IND PDU is received, the initiator shall listen for the connectable AUX_ADV_IND on the secondary advertising physical channel; while doing so, it shall perform the window widening specified in Section 4.2.4. If a connectable undirected AUX_ADV_IND PDU, or a connectable directed AUX_ADV_IND PDU

containing the initiator's Link Layer device address, is received and is allowed by the initiator filter policy, and if the initiator is not already connected to the advertiser's device address, then the initiator shall send an AUX_CONNECT_REQ PDU to the advertiser; otherwise, it shall be ignored.

After sending the AUX_CONNECT_REQ PDU, the initiator shall wait for the advertiser to send an AUX_CONNECT_RSP PDU. Once an AUX_CONNECT_RSP PDU is received, the Link Layer shall exit the Initiating State and shall transition to the Connection State in the Central Role as defined in Section 4.5.4. If the initiator does not receive an AUX_CONNECT_RSP PDU from the advertiser, it shall use the back-off algorithm described for SCAN_REQ in Section 4.4.3.2 before responding to the next connectable AUX_ADV_IND PDU.

#### 4.4.4.3 Requirements

A device that supports Initiating State shall support receiving the connectable directed advertising event type using ADV_DIRECT_IND PDUs and the connectable and scannable undirected event type using ADV_IND PDUs. If the device supports the LE Extended Advertising feature, then it shall also support receiving the connectable directed and connectable undirected advertising event types using ADV_EXT_IND PDUs.

### 4.4.5 Synchronization state

In the Synchronization state, the Link Layer listens to regular broadcasts from another device. There are two types of such broadcasts: periodic advertising trains and broadcast isochronous streams.

Synchronization state has two sub-states: synchronizing and synchronized. The Link Layer shall enter the Synchronization state in the synchronizing sub-state when directed by the Host, provided that it is in possession of the necessary information to locate the regular broadcasts. Once it has successfully received a PDU from the broadcast, it transitions to the synchronized sub-state and is then said to be synchronized to the broadcast; until then, it is said to be synchronizing.

Once synchronized, if it fails to receive any PDUs forming the broadcast for a period specified by the Host, it shall transition to the Standby state and notify the Host.

#### 4.4.5.1 Periodic advertising trains

To receive periodic advertising trains the Link Layer must obtain periodic advertising synchronization information. This information may be obtained from the SyncInfo field of an AUX_ADV_IND PDU (see Section 4.4.3.4) or from an LL_PERIODIC_SYNC_IND or an LL_PERIODIC_SYNC_WR_IND PDU sent by a connected device.

While in this state, the Link Layer shall listen on the secondary advertising channel indices specified in Section 4.4.2.1 for the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDUs forming the periodic advertising train specified in the synchronization information. In the synchronizing sub-state, if the Controller does not receive any of these AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDUs within 6 periodic advertising events, starting with the first periodic advertising event it listened for, it shall notify the Host and transition to the Standby State.

A device shall not attempt to synchronize to a periodic advertising train with the same address, address type, and Advertising SID as one that it is already synchronized to. A device should not attempt to synchronize to a periodic advertising train with the same Access Address as one that it is already synchronized to.

The Link Layer shall perform the window widening specified in Section 4.2.4 while listening. If the *windowWidening* reaches ((*periodicInterval*÷2) − T_IFS_150 µs) in magnitude, the Controller should notify the Host and transition to the Standby State.

#### 4.4.5.1.1 Trains without responses

If requested by the Host, the Link Layer shall report the advertising data received in the periodic advertisements to the Host. The Host may specify that not all such data is reported or that duplicate periodic advertising reports are filtered and so not sent to the Host. A duplicate periodic advertising report is a report for the same periodic advertising train where the previous report for that train that contained an ADI field had the same Advertising SID and DID. The Link Layer need not listen to AUX_SYNC_IND or AUX_CHAIN_IND PDUs where it will not be reporting the data or samples from Constant Tone Extensions to the Host, other than as necessary to maintain synchronization with the advertiser's clock or to receive any channel map updates.

#### 4.4.5.1.2 Trains with responses

If requested by the Host, the Link Layer shall report the advertising data received in the periodic advertisements to the Host. The Host may specify that not all such data is reported. The Link Layer need not listen to an AUX_SYNC_SUBEVENT_IND PDU where it would not be reporting the data to the Host, other than as necessary to maintain synchronization with the advertiser's clock or to receive any channel map updates.

A synchronized device may transmit an AUX_SYNC_SUBEVENT_RSP PDU when instructed to by its Host. The response slot to use is determined by the Host.

### 4.4.5.2 Broadcast Isochronous Streams

To receive broadcast isochronous streams, the Link Layer must obtain the BIGInfo describing the streams (see Section 4.4.6.11). This information may be obtained from

the ACAD of periodic advertising. If the PHY field of the BIGInfo specifies a PHY that the Link Layer does not support or is reserved for future use, the Link Layer shall ignore the BIGInfo, shall not report the BIGInfo to the Host, and shall not enter the Synchronization state for the BIG specified in the BIGinfo.

Note: The Link Layer can still synchronize to a BIS even if it does not support the exact combination of BN, IRC, NSE, and PTO specified in the BIGInfo, provided that it can still receive every payload at least once. For example, if the BIGInfo has BN=2, IRC=2, NSE=8, and PTO=5, a Link Layer that synchronizes using any of the following combinations will receive fewer transmissions of each payload but will still be capable of receiving every payload:

- BN=2, IRC=2, NSE=6, PTO=5
- BN=2, IRC=2, NSE=4
- BN=2, IRC=1, NSE=2

While in this state, the Link Layer shall listen on the isochronous channel indices specified in Section 4.4.6.8 for BIS Data PDUs forming the BIG specified in the BIGInfo. In the synchronizing sub-state, if the Link Layer does not receive a BIS Data PDU within 6 BIG events, starting with the first BIG event it listened for, it shall notify the Host and transition to the Standby state.

Once in the synchronized sub-state, the Link Layer shall listen for the Isochronous Broadcaster at least once within any 6 consecutive BIS events.

A device shall not attempt to synchronize to a BIG with the same associated periodic advertising train as a BIG that it is already synchronized to.

A device that has synchronized to a BIG is called a Synchronized Receiver. A Synchronized Receiver may, but it is not required to, remain synchronized to the periodic advertising train.

If requested by the Host, the Link Layer shall report the isochronous data received in the BIS Data PDUs forming the BIG to the Host. The Host may specify that only the data from specific BISes within the BIG are reported. The Link Layer shall listen to and act on the contents of new BIG Control PDUs where it has received at least one valid BIS Data PDU in the same BIG event and where all valid BIS Data PDUs received in the same BIG event have the CSTF bit set to 1. The Link Layer shall not listen to any other BIG Control PDU.

The Link Layer need not listen to Broadcast Isochronous PDUs that are retransmissions of PDUs already successfully received, or to BIS Data PDUs where it will not be reporting the data to the Host, other than as necessary to maintain synchronization with the Isochronous Broadcaster’s clock.

The Link Layer shall perform the window widening specified in Section 4.2.4 while listening.

The Link Layer shall stop listening to the BIG no later than when the *bisPayloadCounter* equals $2^{39} - 1$.

### 4.4.6 Isochronous Broadcasting state

The Link Layer shall enter the Isochronous Broadcasting state when directed by the Host, provided that it is able to schedule the BIG the Host is requesting to be transmitted. While in this state the Link Layer shall transmit BIS PDUs as described in the following subsections.

When the Link Layer enters the Isochronous Broadcasting state, it shall notify the Host.

In this state, the Host may disable and subsequently re-enable the periodic advertising train associated with the BIG.

Each instance of the Link Layer state machine in the Isochronous Broadcasting state shall transmit a BIG made up of one or more BISes. Each BIS carries a separate isochronous data flow. There shall be at most 31 BISes in a BIG.

Note: The Isochronous Broadcasting state is per BIG (i.e. every new BIG instantiates a new Link Layer state machine).

#### 4.4.6.1 Broadcast Isochronous Stream (BIS)

A BIS is a logical transport that enables a device to transfer isochronous data. The isochronous data can be either framed or unframed.

A BIS supports variable size packets and the transmission of one or more packets in each BIS event, allowing a range of data rates to be supported. The data traffic is unidirectional from the broadcasting device; hence there is no acknowledgment protocol and broadcast isochronous traffic is inherently unreliable. To improve the reliability of packet delivery, the BIS supports multiple retransmissions.

#### 4.4.6.2 Broadcast Isochronous Group (BIG)

A BIG consists of either two or more BISes that have the same ISO_Interval and are expected to have a time relationship at the application layer, or of a single BIS. The maximum number of BISes in a BIG shall be 31. A BIG also contains control subevents (see Section 4.4.6.7).

The packets forming the BIG should be transmitted at the same power level as those forming the associated periodic advertising train (see Section 4.4.6.9).

#### 4.4.6.3 BIG parameters

Each BIG is defined by the following parameters:

- Num_BIS is the number of BISes in the BIG. The BISes in a BIG are each assigned a different BIS_Number from 1 to Num_BIS.
- ISO_Interval is the time between two adjacent BIG anchor points, in units of 1.25 ms. The value shall be between 4 and 3200 (i.e. 5 ms to 4 s).
- BIS_Spacing is the time between the start of corresponding subevents in adjacent BISes in the BIG.
- Sub_Interval is the time between the start of two consecutive subevents of each BIS.
- Max_PDU is the maximum number of data octets (omitting the MIC, if any) that can be carried in each BIS Data PDU in the BIG. The value shall be between 1 and 251 octets.
- Max_SDU is the maximum size of an SDU on this BIG (see [Vol 6] Part G, Section 1). The value shall be between 1 and 4095 octets.
- MPT shall equal the time taken to transmit a packet containing a BIS Data PDU with a Payload field of Max_PDU octets on the PHY being used for the BIS; on the LE Coded PHY, the S=8 coding shall be assumed.
- BN, PTO, and IRC control which data is transmitted in each BIG event. The value of BN shall be between 1 and 7. The value of PTO shall be between 0 and 15. The value of IRC shall be between 1 and 15.
- NSE is the number of subevents per BIS in each BIG event. The value shall be between 1 and 31 and shall be an integer multiple of BN.
- Framed indicates whether the BIG carries framed or unframed data.
- Framing_Mode only applies when framed data is being carried and indicates whether Segmentable or Unsegmented mode is being used.
- Encrypted indicates whether the BIG is encrypted or not.

These parameters shall not change during the lifetime of the BIG. They are discussed further in Section 4.4.6.4 to Section 4.4.6.11. The mandatory range for each parameter is the entire range of valid values except for the following parameters, where only the listed values are mandatory:

- Num_BIS: 1
- BN: 1
- NSE: all supported values of BN
- PTO: 0

- IRC: all supported values of GC (see Section 4.4.6.6)

Each BIG shall have a 39-bit counter *bigEventCounter* associated with it. This shall be set to 0 for the first BIG event of a BIG and be incremented by 1 for each BIG event, whether or not the Isochronous Broadcaster transmits any Broadcast Isochronous PDUs during the event.

Each BIS shall have a 39-bit counter *bisPayloadCounter* associated with it, described further in Section 4.4.6.5. The Link Layers of an Isochronous Broadcaster and a Synchronized Receiver shall terminate the BIG no later than when the *bisPayloadCounter* equals $2^{39} – 1$.

Note: At the start of any BIG event, all the BISes in a BIG will have the same value for *bisPayloadCounter* and, in addition, *bigEventCounter* × BN = *bisPayloadCounter*.

#### 4.4.6.4 BIG event

A BIG event consists of one or more BIS PDUs. The Link Layer shall transmit BIS PDUs only in BIG events. The Link Layer shall transmit only BIS PDUs as part of a BIG event.

Each BIG event is divided into Num_BIS separate BIS events and a control subevent if present. Each BIS event is divided into NSE subevents.

Each BIS event starts at a moment called the BIS anchor point and ends after its last subevent. Each BIG event starts at a moment called the BIG anchor point and ends after the control subevent, if there is one, and otherwise at the end of the last constituent BIS event. The BIG anchor points shall be spaced regularly, ISO_Interval apart. The BIS anchor points for BIS *n* of a BIG shall be (*n* − 1) × BIS_Spacing after the BIG anchor points and so are also spaced regularly, ISO_Interval apart. The subevents of each BIS shall be Sub_Interval apart. The Isochronous Broadcaster shall close each BIG event at least T_MSS_150 before the BIG anchor point of the next BIG event. Figure 4.34 shows a BIS event with subevents.

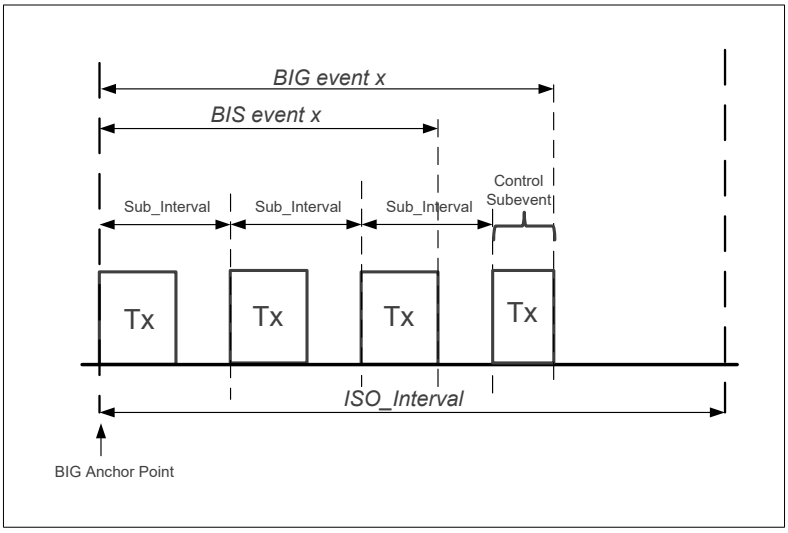



*Figure 4.34: Example of BIG and BIS events*

The BISes in a BIG shall be arranged either sequential or interleaved by setting the values of the Sub_Interval and BIS_Spacing parameters appropriately. If they are sequential, BIS_Spacing shall be greater than or equal to NSE × Sub_Interval and so all the subevents of a BIS event occur together. If they are interleaved, Sub_Interval shall be greater than or equal to Num_BIS × BIS_Spacing and so the first subevents of all BISes are adjacent, followed by the second subevents of all BISes, and so on. In each case, the minimum value for BIS_Spacing should be used. Figure 4.35 shows each arrangement for a BIG where Num_BIS = 2 and NSE = 2.

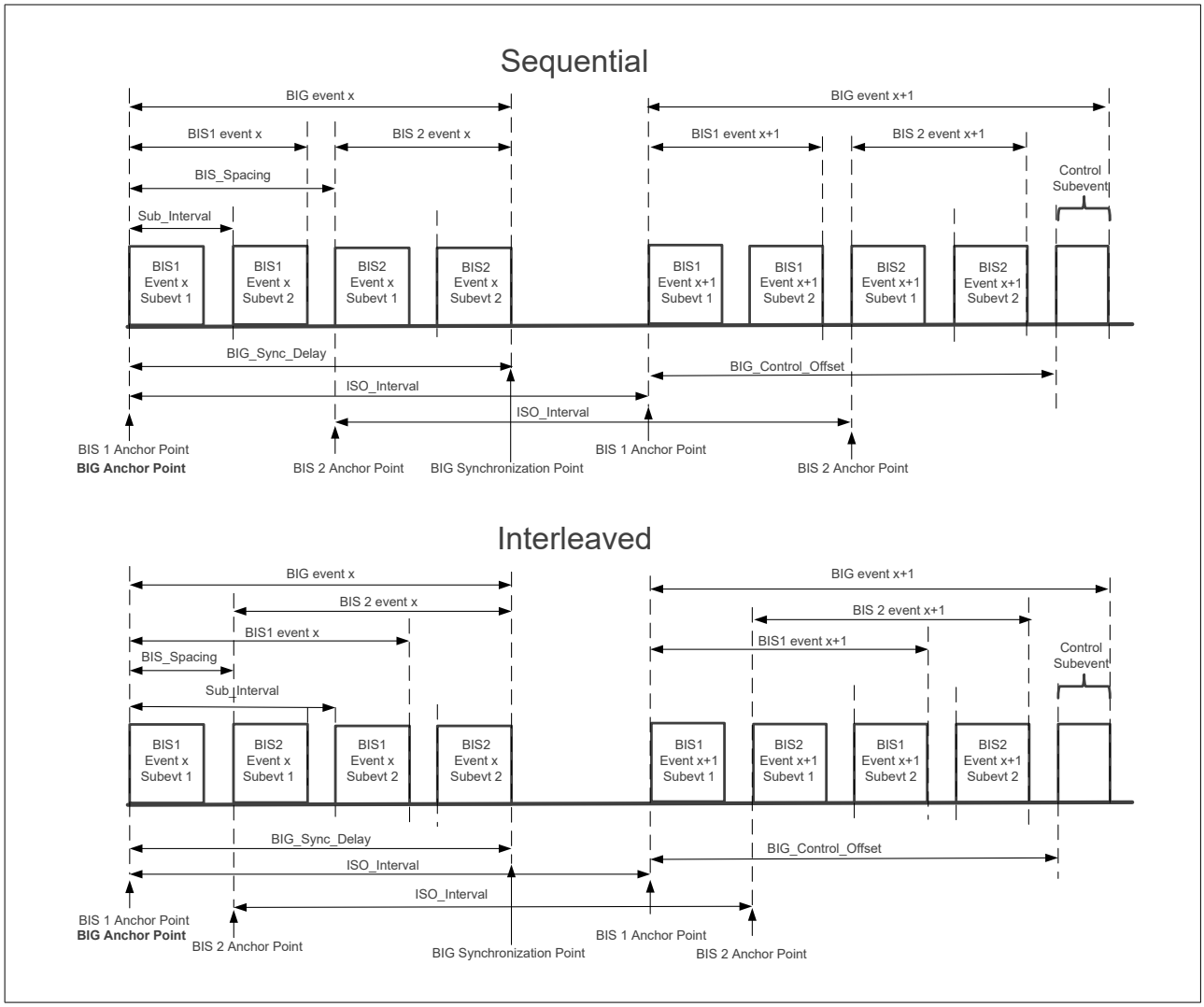



*Figure 4.35: Two BISes in sequential and interleaved arrangement*

The maximum possible length for the data portion of a BIG event (thus omitting any control subevent) is denoted as BIG_Sync_Delay. The value of BIG_Sync_Delay shall equal the time from the anchor point to the BIG Synchronization point, which is the instant at the end of a packet containing a Payload field of Max_PDU octets transmitted in the last subevent, as shown in Figure 4.35. Therefore, the BIG_Sync_Delay equals (Num_BIS – 1) × BIS_Spacing + (NSE – 1) × Sub_Interval + MPT.

### 4.4.6.5 Broadcast Isochronous Data

A BIS carries a single stream of isochronous data provided for broadcast. The data may be divided into payloads of at most Max_PDU octets, each of which is transmitted in a single BIS Data PDU; the payloads need not all be the same size and can be zero length. The BISes in a BIG carry separate but associated streams of data.

The Framed parameter of a BIG shall indicate whether all the constituent BISes are framed or unframed. Framed BIGs shall only use framed BIS Data PDUs to carry data; unframed BIGs shall only use unframed BIS Data PDUs to carry data.

Both BIS_Spacing and Sub_Interval shall be at least T_MSS_150 + MPT.

For each BIS, the payloads shall be numbered in the order provided, starting at zero. This number shall be used as the value of *bisPayloadCounter* for the PDU containing that payload. If the source of the data fails to provide BN payloads for a BIS event, *bisPayloadCounter* shall continue to increment as if it had provided the missing payloads and the Link Layer should transmit empty PDUs.

#### 4.4.6.6 BIS subevents

A BIS subevent is an opportunity for an Isochronous Broadcaster to transmit a Broadcast Isochronous BIS PDU and a Synchronized Receiver to receive it. The Link Layer should transmit one BIS Data PDU at the start of each subevent of the isochronous broadcasting event unless, for example, it has scheduling conflicts, but shall transmit at least one BIS PDU within any 6 consecutive BIS events on a given BIS. Where it does not transmit a PDU, the Link Layer shall behave for all other purposes (e.g. the timing of packets and the choice of payload) as if it had done so.

Each BIS subevent ends at the end of the transmitted PDU or, if the Link Layer does not transmit a PDU, the subevent is MPT in duration.

For each BIS event the source of the data shall supply a burst of data consisting of BN (“Burst Number”) payloads, each of which in turn shall hold either a single fragment or one or more SDU segments. This burst is associated with the corresponding BIS event but may be transmitted in earlier events as well. Each PDU containing a given payload shall have the same LLID value but can have different CSSN and CSTF values (see Section 4.4.6.7).

Note: The burst associated with a BIS event consists of payloads with *bisPayloadCounter* between *bigEventCounter* × BN and (*bigEventCounter* + 1) × BN – 1.

The subevents of each BIS event are partitioned into groups of BN subevents each. Therefore, there are Group Count (GC) groups, where GC = NSE ÷ BN.

IRC ("Immediate Repetition Count") specifies the number of groups that carry the data associated with the current BIS event; the remaining groups carry data associated with the future BIS events specified by PTO ("Pre-Transmission Offset"). IRC shall be greater than zero and not greater than GC. If IRC = GC then PTO shall be ignored. Otherwise PTO shall be greater than zero.

The groups of subevents are numbered using g from 0 to GC – 1 in order.

- If g < IRC, then group g shall contain the data associated with the current BIS event.
- If g ≥ IRC, then group g shall contain the data associated with the future BIS event that is PTO × (g - IRC + 1) BIS events after the current BIS event.

The payloads in each burst shall always be transmitted in the same order.

Note: Setting GC to a value greater than 1 provides redundant transmissions to compensate for the lack of acknowledgments when broadcasting, while setting IRC to a value less than GC (called pre-transmission) provides a greater time diversity among the redundant copies of the data.

For example, Figure 4.36, Figure 4.37, and Figure 4.38 show the allocation of payloads to subevents for three different BISes.

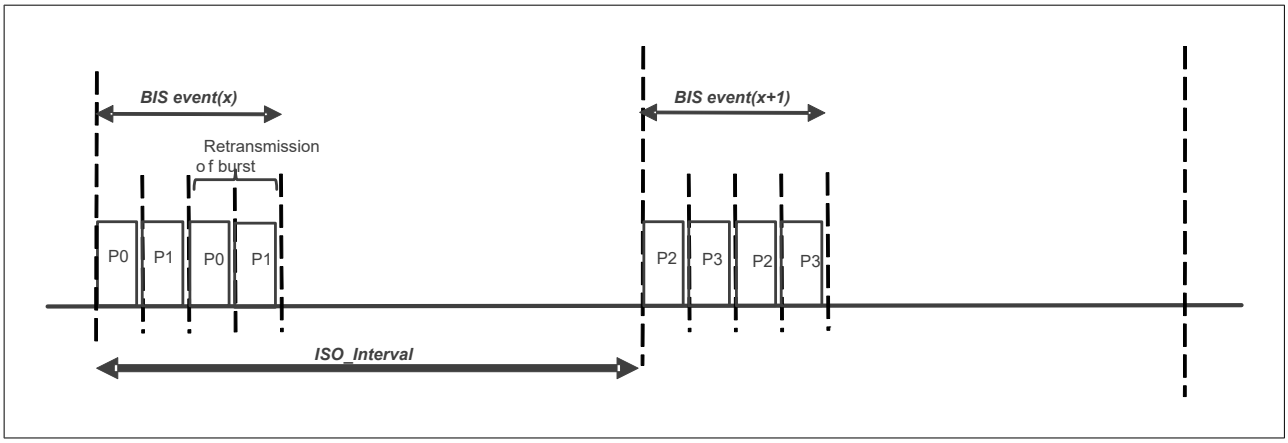



*Figure 4.36: Allocations of payloads within a BIS with BN=2, IRC=2, PTO=0, and NSE=4*

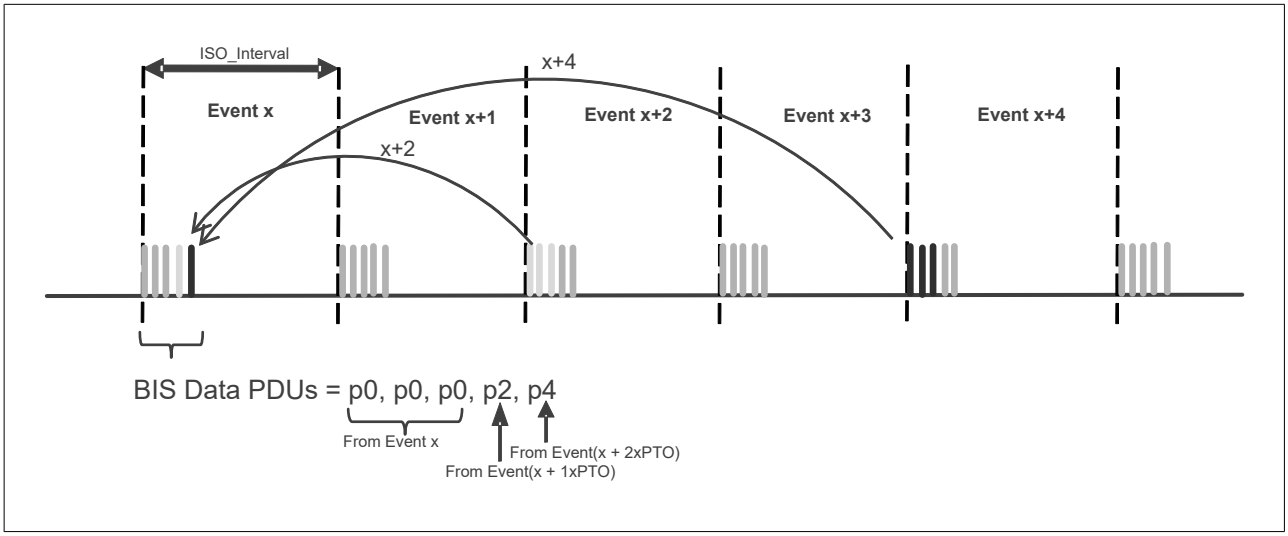



*Figure 4.37: Allocations of payloads within a BIS with BN=1, IRC=3, PTO=2, and NSE=5*

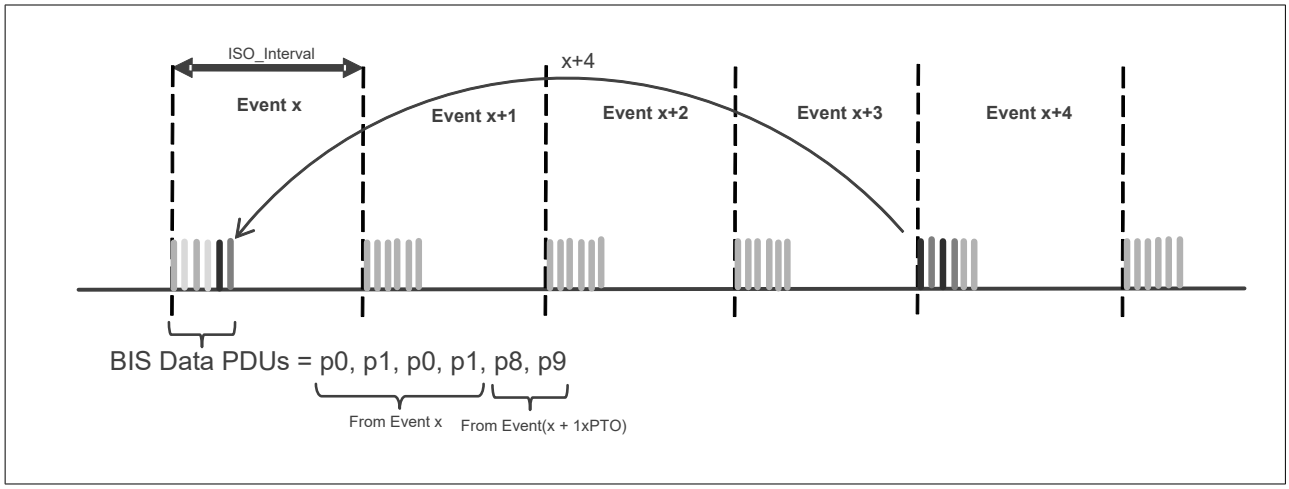



*Figure 4.38: Allocations of payloads within a BIS with BN=2, IRC=2, PTO=4, and NSE=6*

#### 4.4.6.7 Control subevents

Each BIG event may contain a control subevent. If so, the Link Layer shall transmit a single BIG Control PDU at the start of the control subevent to send control information about the BIG (see Section 5.6). The Link Layer shall not transmit a BIG Control PDU at any other time.

The time from the BIG anchor point to the start of the control subevent, designated BIG_Control_Offset, shall be:

BIG_Control_Offset = Num_BIS × BIS_Spacing for sequential arrangement
BIG_Control_Offset = NSE × Sub_Interval for interleaved arrangement

Note: See Section 4.4.6.4 for sequential and interleaved arrangements of the BISes in a BIG.

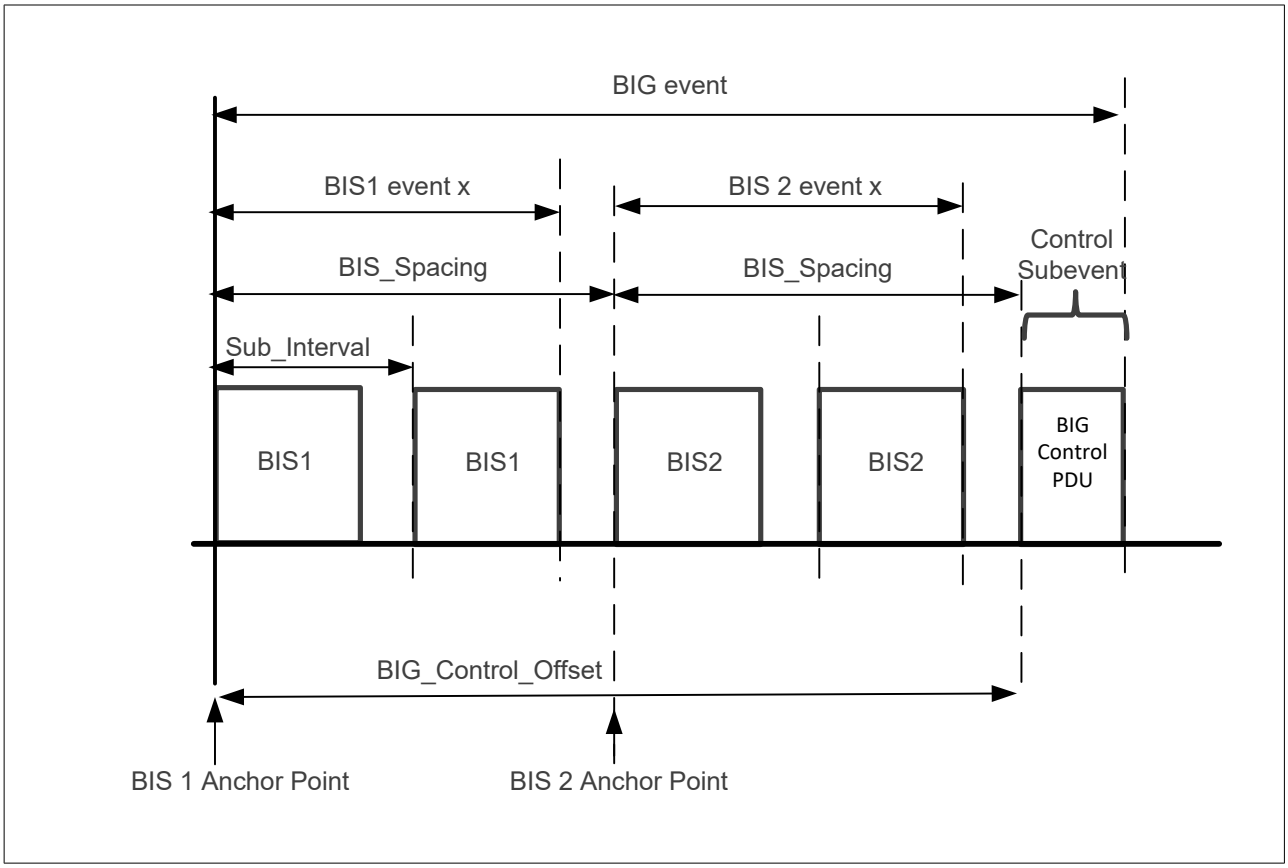



*Figure 4.39: Example of a Control subevent in a BIG with 2 BISes in sequential arrangement*

If the Link Layer schedules a BIG Control PDU to be transmitted in a BIG event, it shall set the CSTF bit to 1 in the Header field of every BIS Data PDU sent in that same BIG event; otherwise it shall set the bit to 0. It shall set the CSTF bit to 0 in the Header field of all BIG Control PDUs.

The value of the CSSN of every BIS PDU in a BIG event shall be the same. The Link Layer shall increment CSSN by 1 (with 7 wrapping to 0) at the start of a BIG event that contains the first transmission of a new BIG Control PDU and shall leave the CSSN unchanged otherwise (i.e. when a BIG Control PDU is being retransmitted or is not scheduled to be transmitted).

Note: A Synchronized Receiver can use the CSSN to determine that a BIG Control PDU is a retransmission of one that it has already received.

#### 4.4.6.8 Channel indices

Each packet containing a BIS PDU shall be transmitted on the channel index specified by Channel Selection Algorithm #2 (see Section 4.5.8.3). The subevent number *se_n* shall be set to the values 1 to NSE, in order, for the subevents on a given BIS – the same values shall be used for all the BISes in a BIG.

The channel map used in the BIG shall be included in the BIGInfo. When the channel map changes, it shall be transmitted in the BIG Control logical link using the BIG Channel Map Update procedure (see Section 5.6.1).

#### 4.4.6.9 Associated periodic advertising train

Every BIG shall have an associated periodic advertising train. A periodic advertising train shall not be associated with more than one BIG at the same time. The train and BIG may be enabled and disabled independently. The ACAD field of the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU in the periodic advertising train is used to carry the BIGInfo of a BIG. The BIGInfo shall not be transmitted when the BIG is disabled. Whenever the BIG and associated train are both active, the BIGInfo shall be transmitted in the periodic advertising train whenever the ACAD of the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU has sufficient space for carrying it.

It is vendor-specific whether the associated periodic advertising train may have responses.

The ACAD can be used for other information as well, such as a change in channel map. Even if it is not possible to fit both in the same PDU, the Link Layer will still need to schedule transmissions of each information so as to meet any relevant requirements.

The transmission of the AUX_SYNC_IND or AUX_SYNC_SUBEVENT_IND PDU of the associated periodic advertising train should not be scheduled within a BIG event.

#### 4.4.6.10 Encryption

The Link Layer of an Isochronous Broadcaster or Synchronized Receiver shall support unencrypted BIGs.

A BIG may be encrypted, in which case all BIS PDUs (except those with an empty Payload field) of all BISes in that BIG shall be encrypted. The Link Layer shall determine if a BIG is encrypted by examining the length of the BIGInfo (see Section 4.4.6.11). The rest of this section only applies to encrypted BISes.

The following parameters shall be used in the process of encrypting or decrypting all Broadcast Isochronous PDUs in BIGs:

- Broadcast_Code – a 16-octet parameter provided by the Host.The Broadcast_Code applies to all the BISes in a single BIG and different BIGs broadcast by the same device may use different Broadcast_Codes.
- GIV – a 64-bit parameter generated by the Controller.
- GSKD – a 128-bit parameter generated by the Controller.

For each encrypted BIG, the Controller of an Isochronous Broadcaster shall generate a new GIV and GSKD using the requirements for random number generation as defined

in [Vol 3] Part H, Section 2 and shall transmit them in the BIGInfo. Each Broadcast Isochronous PDU in the encrypted BIG shall be encrypted using the CCM algorithm (see [Vol 6] Part E, Section 2).

#### 4.4.6.11 BIGInfo

The length of the BIGInfo is 33 octets for an unencrypted BIG and 57 octets for an encrypted BIG. The format of the BIGInfo is shown in Figure 4.40.

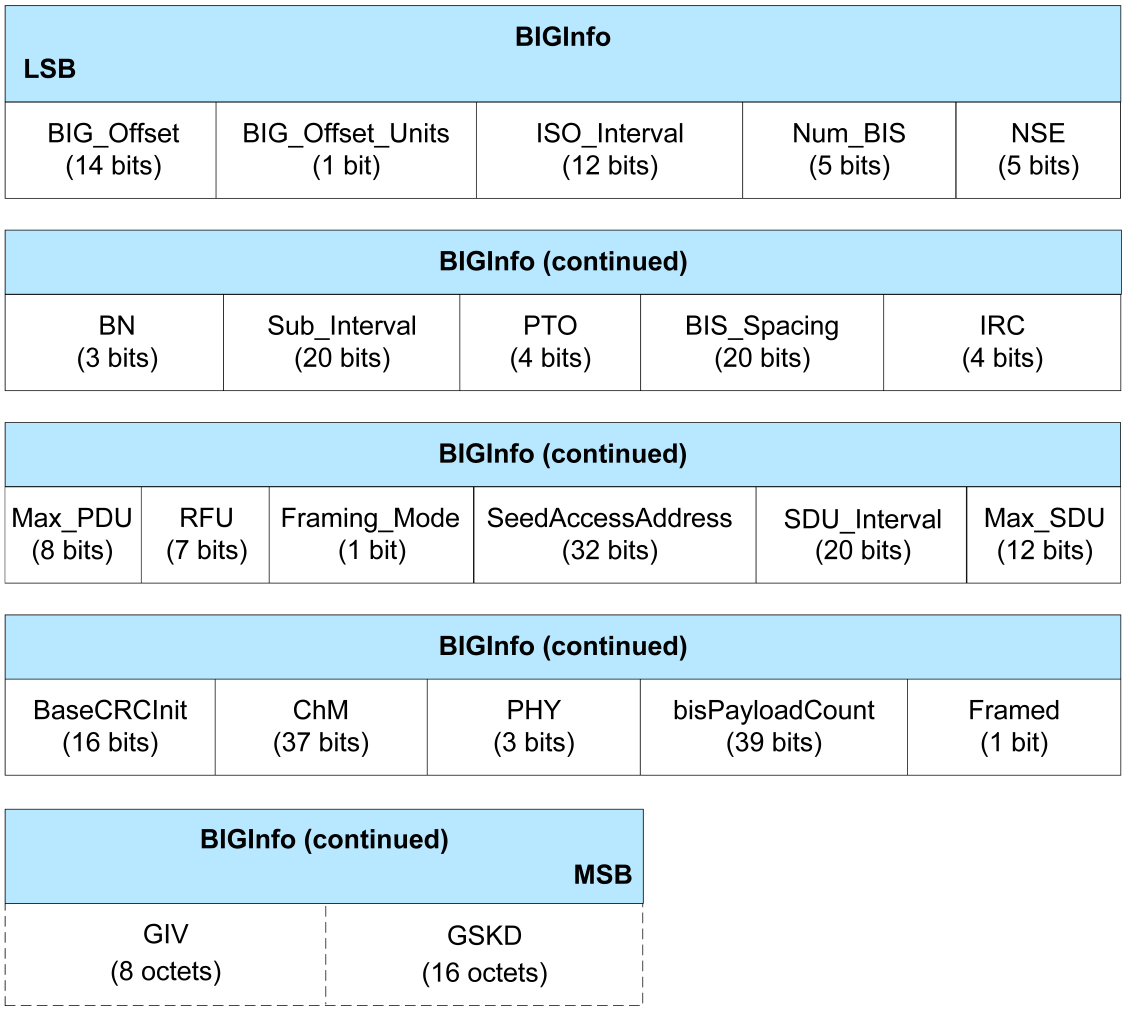



*Figure 4.40: Format of BIGinfo*

The BIG_Offset field contains the time from the start of the packet containing the BIGInfo to the BIG anchor point that this BIGInfo describes. The value of the BIG_Offset field is in the unit of time indicated by the BIG_Offset_Units bit; the actual time offset is determined by multiplying the value of BIG_Offset by the unit. The offset shall be greater than 600 µs.

If the BIG_Offset_Units bit is set then the unit is 300 µs; otherwise it is 30 µs. The BIG_Offset_Units bit shall not be set if the offset is less than 491,460 µs.

The BIG anchor point shall be no earlier than the offset and no later than the offset plus one unit after the start of the relevant packet, as shown in Figure 4.41.

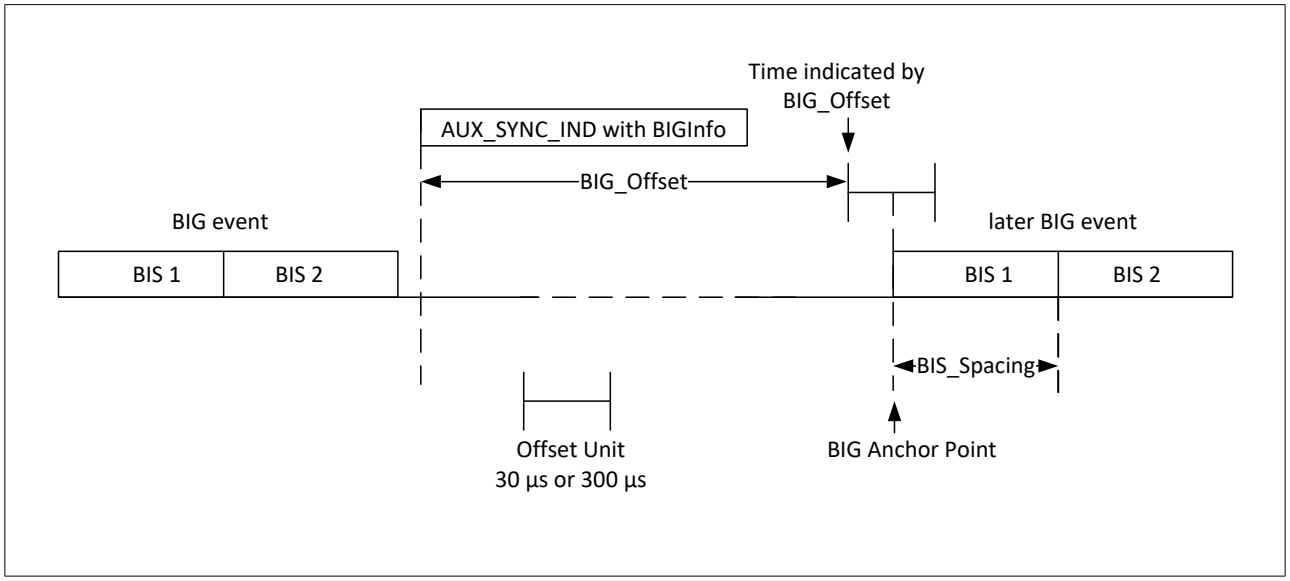



*Figure 4.41: Time reference of a BIG event from a periodic advertising event.*

The ISO_Interval, NSE, BN, Sub_Interval, PTO, BIS_Spacing, and IRC fields shall contain the values described in Section 4.4.6.3. Sub_Interval and BIS_Spacing shall be in units of microseconds.

The Num_BIS field shall contain the number of BISes in the BIG.

The Max_PDU field shall contain the value described in Section 4.4.6.3.

The SeedAccessAddress field shall contain the Seed Access Address for the BIG (see Section 2.1.2).

The SDU_Interval field shall contain the interval described in [Vol 6] Part G, Section 2.

The Max_SDU field shall contain the value specified in Section 4.4.6.3.

The BaseCRCInit field shall contain the value described in Section 3.1.1.

The ChM field shall have the same meaning as the corresponding field in the CONNECT_IND PDU (see Section 2.3.3.1).

The PHY field shall be set to indicate the PHY used by the BIG. The values for the PHYs are specified in Table 4.5.

| Value | Meaning |
|---|---|
| 0 | LE 1M PHY |
| 1 | LE 2M PHY |
| 2 | LE Coded PHY |
| All other values | Reserved for future use |

*Table 4.5: PHY Types*

The bisPayloadCount field shall contain the value specified in Section 4.4.6.5. The value shall be for the first subevent of the BIG event referred to by the BIG_Offset field.

The Framed bit shall be set if the BIG carries framed data.

The Framing_Mode bit shall be set if the BIG carries framed data and Unsegmented mode is in use, and is zero for Segmentable mode. If the BIG carries unframed data, then Framing_Mode is RFU.

The GIV and GSKD fields shall contain the values described in Section 4.4.6.10 if the BIG is encrypted.

## 4.5 Connection state

The Link Layer enters the Connection state when an initiator sends a CONNECT_IND PDU on the primary advertising physical channel to an advertiser, an advertiser receives a CONNECT_IND PDU on the primary advertising physical channel from an initiator, an advertiser sends an AUX_CONNECT_RSP PDU on the secondary advertising physical channel to an initiator, or an initiator receives an AUX_CONNECT_RSP PDU on the secondary advertising physical channel from an advertiser.

After entering the Connection State, the connection is considered to be created. The connection is not considered to be established at this point. A connection is only considered to be established once a data physical channel packet has been received (regardless of a valid CRC match) from the peer device. The only difference between a connection that is created and a connection that is established is the Link Layer connection supervision timeout value that is used (see Section 4.5.2).

If the connection is first created using the CONNECT_IND PDU on the primary advertising physical channel, it shall use the LE 1M PHY in both directions. If the connection is first created on the secondary channel using the AUX_CONNECT_REQ and AUX_CONNECT_RSP PDUs, it shall use the same PHY in both directions as was used for the AUX_CONNECT_REQ and AUX_CONNECT_RSP PDU. Either PHY may be changed subsequently using the PHY Update procedure (Section 5.1.10). When the LE Coded PHY is in use, the coding of each packet is determined by the transmitting

Controller. The coding is indicated by the CI as defined in Section 2.2.3 and may be different in each direction and in adjacent packets in a given direction.

When two devices are in a connection, the two devices act in different roles. A Link Layer in the Central Role is called a Central. A Link Layer in the Peripheral Role is called a Peripheral. The Central controls the timing of a connection event. A connection event is a point of synchronization between the Central and the Peripheral. There shall be only one ACL connection, whether or not established, between two LE device addresses (including two different Resolvable Private Addresses that resolve to the same IRK). An initiator shall not send a connection request to an advertiser it is already connected to. A periodic advertiser shall not send a connection request to a synchronized device that it is already connected to.

If an advertiser receives a connection request from an initiator it is already connected to, then it shall ignore that request. If a synchronized device receives a connection request from a periodic advertiser it is already connected to, then it shall ignore that request.

If the initiator sent a CONNECT_IND PDU in response to an ADV_IND or ADV_DIRECT_IND PDU and either or both devices' PDU had the ChSel field set to 0, then Channel Selection Algorithm #1 (see Section 4.5.8.2) shall be used on the connection. Otherwise, Channel Selection Algorithm #2 (see Section 4.5.8.3) shall be used.

The Central, when directed by the Host, may create a Connected Isochronous Stream (CIS) with the peer device using the Connected Isochronous Stream Creation procedure (see Section 5.1.15). The CIS shall be associated with the ACL used to create it with the same device being Central on both connections. The Link Layer may create multiple CISes with the same Peripheral.

Note: A Peripheral cannot initiate a request to create a CIS at the Link Layer level but it can do so at a higher layer.

When the ACL connection between the Central and Peripheral is terminated, all associated CISes shall be terminated at the same time.

### 4.5.1 Connection events

The Link Layer in the Connection state shall only transmit Data Physical Channel PDUs (see Section 2.4) in connection events. The Central and Peripheral shall determine the data channel index for each connection event as defined in Section 4.5.8. The same data channel index shall be used for all packets in the connection event. Each connection event normally contains at least one packet sent by the Central. The Central can, however, completely fail to transmit in a connection event due to scheduling conflicts or the effect of the subrate factor (see below), but shall transmit at least one Data Channel PDU within each supervision timeout period (see Section 4.5.2).

During a connection event, the Central and Peripheral alternate sending and receiving packets spaced T_IFS_ACL_CP between a packet transmitted by the Central and one transmitted by the Peripheral and spaced T_IFS_ACL_PC between a packet transmitted by the Peripheral and one transmitted by the Central, as shown in Figure 4.42.

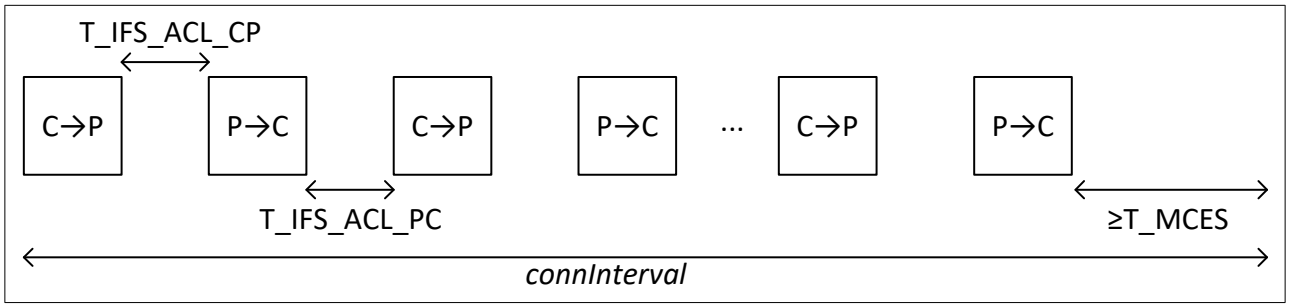



*Figure 4.42: The different frame space values in a connection event*

The connection event is considered open while both devices continue to send packets. The Peripheral shall always send a packet if it receives a packet from the Central regardless of a valid CRC match, except after multiple consecutive invalid CRC matches as specified in Section 4.5.6. The Central may send a packet if it receives a packet from the Peripheral regardless of a valid CRC match, except after multiple consecutive invalid CRC matches as specified in Section 4.5.6. When determining the end of the received packet, the Length and CP fields of the Header field, and the CTETime field of the CTEInfo field (if present), are assumed to be correct even if the CRC match was invalid; however, if the receiving device can determine the correct length and CTETime in some other way, it may use those values instead of those in the Header field.

The connection event can be closed by either device, as defined in Section 4.5.6.

Both the Central and the Peripheral shall have a 16-bit connection event counter (*connEventCounter*), containing the value connEventCount, for each ACL connection. Each counter shall be set to zero on the first connection event and shall be incremented by one for each new connection event; the *connEventCounter* shall wrap from 0xFFFF to 0x0000. This counter is used to synchronize Link Layer control procedures.

Both devices shall increment *connEventCounter* for all connection events, even if the Peripheral is not listening to the Central in those events or the Central did not transmit during the event (e.g., because of subrating or Peripheral latency).

The timing of connection events is determined by the following parameters: connection interval (*connInterval*), subrate base event (*connSubrateBaseEvent*), subrate factor (*connSubrateFactor*), continuation number (*connContinuationNumber*), and Peripheral latency (*connPeripheralLatency*).

The start of a connection event is called an anchor point. If the Central transmits in a connection event, it shall start to transmit a Data Physical Channel PDU to the Peripheral at the anchor point. The start of connection events are spaced regularly with an interval of *connInterval* and shall not overlap. The Central shall ensure that a connection event closes at least T_MCES before the anchor point of the next connection event. The Peripheral should listen for the packet sent by its Central at the anchor point.

The *connInterval* shall be a multiple of 125 µs in resolution between 375 µs and 4 s. The extended *connInterval* values (ECV) is the set of all such values. The rounded *connInterval* values (RCV) is the subset of ECV where all values are multiples of 1250 µs. The baseline *connInterval* values (BCV) is the subset of RCV where all values are greater than or equal to 7.5 ms.

If the Controller supports the Shorter Connection Intervals feature (see Section 4.6.50), then it shall support all values of RCV and may support any other value from ECV. Otherwise, the Controller shall support all values of BCV and shall not support any other value.

The *connInterval* is set by the Initiator's Link Layer in the CONNECT_IND or AUX_CONNECT_REQ PDU from the range given by the Host. The anchor point and *connInterval* can be changed using the Connection Update procedure (see Section 5.1.1), the Connection Parameters Request procedure (see Section 5.1.7), the Connection Rate Update procedure (see Section 5.1.32), or the Connection Rate Request procedure (see Section 5.1.33). When the connection is created and when the *connInterval* is changed using the Connection Update procedure or the Connection Parameters Request procedure, *connInterval* shall be set to a value from BCV.

The subrate factor allows the Central and Peripheral to use a reduced number of connection events. The Central shall only transmit on subrated connection events, the events specified in Section 5.1.1, and, if the continuation number is non-zero, continuation events.

A subrated connection event is a connection event where (*connEventCount* - *connSubrateBaseEvent*) is an integer multiple of *connSubrateFactor*; *connSubrateBaseEvent* is a *connEventCount* that is used to determine the phase of the subrated events. The connection event where *connEventCount* equals *connSubrateBaseEvent* will always be a subrated connection event. Adding an integer multiple of *connSubrateFactor* to *connSubrateBaseEvent* (including a negative multiple) results in a *connEventCount* value that will also always be a subrated connection event. For example, if *connSubrateFactor* equals 100, then *connSubrateBaseEvent* values of 42, 6942, and 65442 are equivalent. In each case, connection event number 24242 is a subrated event because 24242 - 42 = 24200, 24242 - 6942 = 17300, and 24242 - 65442 = -41200 are all integer multiples of 100.

A continuation event is a connection event where, in at least one of the previous *connContinuationNumber* connection events (ignoring any before the last subrated connection event), at least one packet was transmitted or validly received containing a Link Layer PDU with a non-zero Length field. Continuation events are determined by activity in a subrated connection event and any subsequent continuation events. Some connection events between two consecutive subrated connection events might not be continuation events.

The value of *connSubrateFactor* shall be in the range 1 to 500 and shall be set to 1 for a new connection. A Controller shall either support only a *connSubrateFactor* of 1 (in which case the value of *connSubrateBaseEvent* will not be used) or shall support all valid subrate factors. The value of *connContinuationNumber* shall be in the range 0 to *connSubrateFactor* - 1 and shall be set to zero for a new connection. A Controller shall support all valid continuation numbers.

Figure 4.43 and Figure 4.44 show how the subrate base event affects which events are subrated connection events.

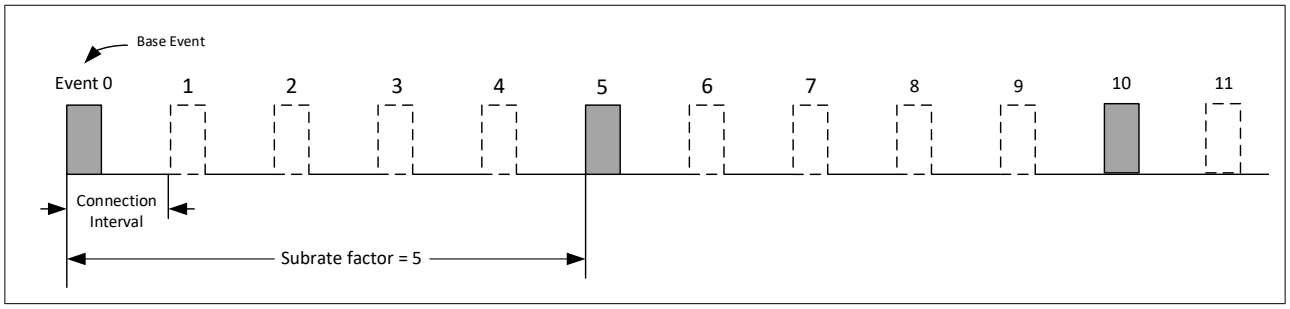



*Figure 4.43: Connection events used when connSubrateFactor = 5 and connSubrateBaseEvent = 0*

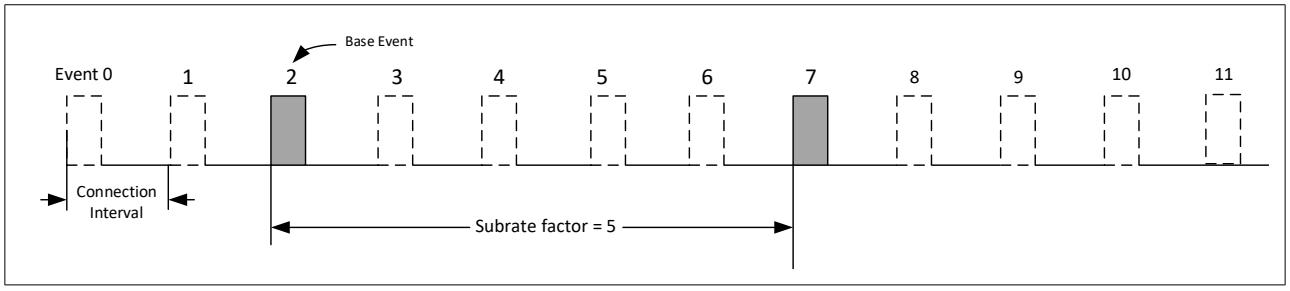



*Figure 4.44: Connection events used when connSubrateFactor = 5 and connSubrateBaseEvent = 2*

Peripheral latency also allows a Peripheral to use a reduced number of connection events. The *connPeripheralLatency* parameter defines the number of consecutive subrated connection events that the Peripheral is not required to listen for the Central. For example, if *connSubrateFactor* is 3, *connContinuationNumber* is 0, and *connPeripheralLatency* is 6, then a Peripheral implementation can choose to only listen to every 21$^{st}$ connection event (i.e., every 7$^{th}$ subrated connection event).

*connPeripheralLatency* shall be an integer such that *connSubrateFactor* × (*connPeripheralLatency* + 1) is less than or equal to 500 and *connInterval* × *connSubrateFactor* × (*connPeripheralLatency* + 1) is less than half *connSupervisionTimeout*. When *connPeripheralLatency* is set to zero the Peripheral should listen at the anchor point of every subrated connection event and continuation event. Irrespective of Peripheral latency, if the Peripheral receives a valid packet from the Central at a subrated connection event, it shall listen at the anchor point of every continuation event before the next subrated connection event. If the Peripheral does not receive a valid packet from the Central after applying Peripheral latency, it should listen at each of the subrated anchor points and not apply Peripheral latency until it receives a packet from the Central. Irrespective of the value of *connPeripheralLatency* or any scheduling conflicts, the Peripheral shall listen for the Central at least once within each connection supervision timeout duration (see Section 4.5.2).

Figure 4.45 to Figure 4.47 show how the connection events are used for various values of *connSubrateFactor*, *connPeripheralLatency*, and *connContinuationNumber*. In the figures, S indicates the subrated events, C indicates continuation events, and L indicates subrated events where Peripheral latency is applied. In Figure 4.46, the data is being transmitted by the Peripheral, not the Central.

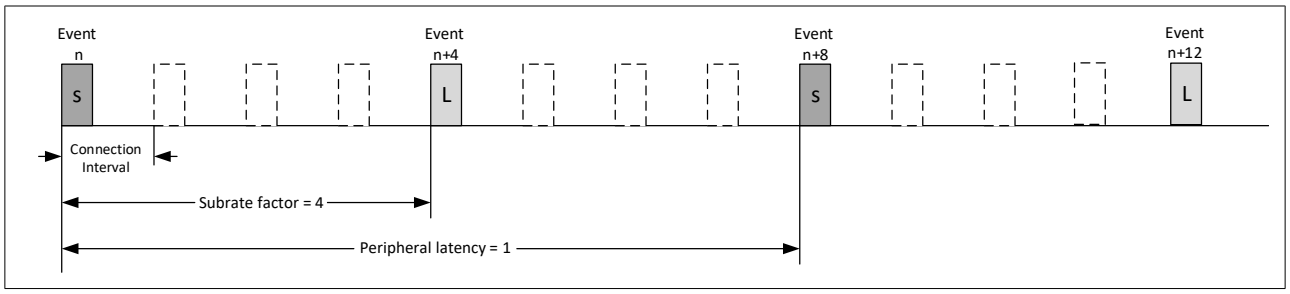



*Figure 4.45: Connection events used when connSubrateFactor = 4, connPeripheralLatency = 1, and connContinuationNumber = 0*

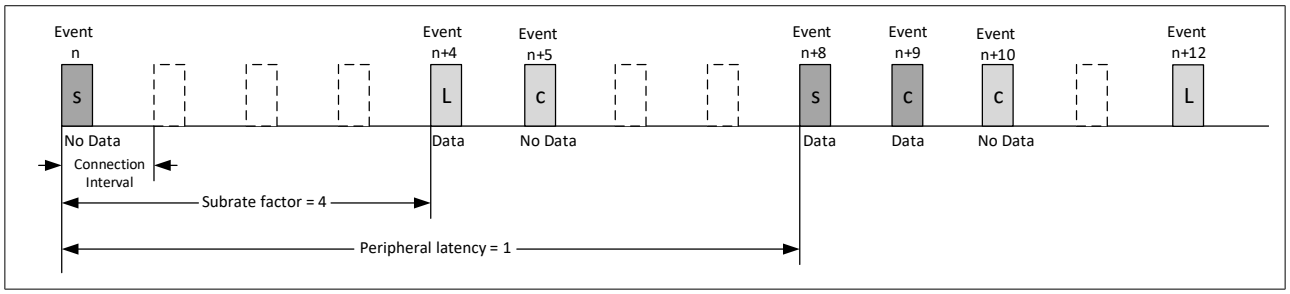



*Figure 4.46: Connection events used when connSubrateFactor = 4, connPeripheralLatency = 1, and connContinuationNumber = 1*

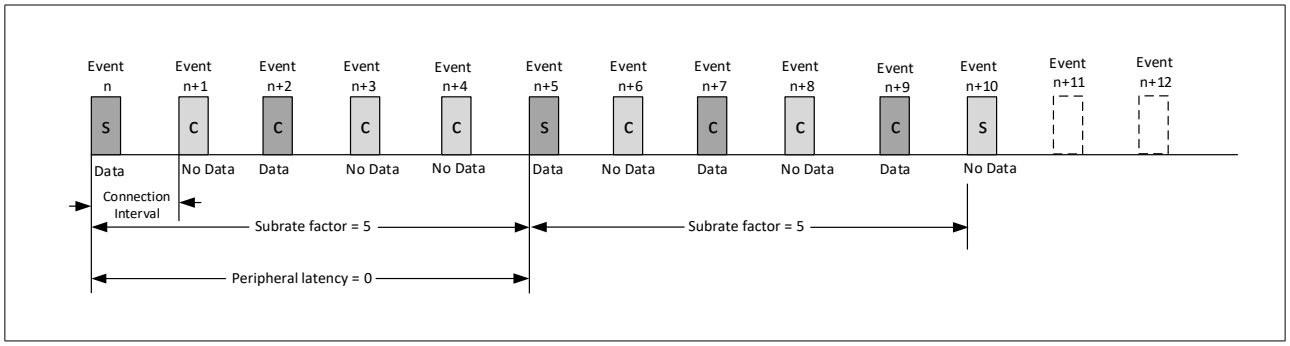



*Figure 4.47: Connection events used when connSubrateFactor = 5, connPeripheralLatency = 0, and connContinuationNumber = 2*

If the Connection Subrating feature is supported then, when *connEventCounter* wraps, the Link Layer shall set the value of *connSubrateBaseEvent* to (*connSubrateBaseEvent* + *K* × *connSubrateFactor* - 65536), where *K* is any integer that will cause the new value to be in the range 0 to 65535, so that the subrated connection events will remain equally spaced.

### 4.5.2 Supervision timeout

A connection can break down due to various reasons such as a device moving out of range, encountering severe interference or a power failure condition. Since this may happen without any prior warning, it is important for both the Central and the Peripheral to monitor the status of the connection.

To be able to detect link loss in an ACL connection, both the Central and the Peripheral shall use an ACL connection supervision timer, $T_{LLconnSupervision}$. Upon reception of a valid packet on the ACL, the timer shall be reset. To be able to detect link loss in a CIS, both the Central and the Peripheral shall use a CIS supervision timer, $T_{CISSupervision}$. Upon reception of a valid packet on the CIS, the timer $T_{CISSupervision}$ shall be reset.

If the ACL connection supervision timer reaches 6 × *connInterval* before the connection is established (see Section 4.5), the ACL connection may, but should not, be considered lost. If the ACL connection is not established after 6 connection events, it shall be considered lost. This enables fast termination of ACL connections that fail to establish.

Depending on the connection interval and whether a CONNECT_IND or AUX_CONNECT_REQ PDU was used, this timer can expire after 4, 5, or 6 connection events.

Because a packet with a CRC error is sufficient to establish the connection, the connection can become established without the timer $T_{LLconnSupervision}$ being reset.

When establishing a CIS, the Central shall start the CIS supervision timer at the start of the next CIS event after receiving the acknowledgment for the LL_CIS_IND. If the CIS

supervision timer reaches 6 × ISO_Interval before the CIS is established, the CIS shall be considered lost.

When establishing a CIS, the Peripheral shall start the CIS supervision timer at the start of the next CIS event after receiving the LL_CIS_IND. If the CIS supervision timer reaches 6 × ISO_Interval before the CIS is established, the CIS shall be considered lost.

Connection supervision timeout (*connSupervisionTimeout*) is a parameter that defines the maximum time between two received Data Channel PDUs or Connected Isochronous PDUs before the connection is considered lost. The *connSupervisionTimeout* shall be a multiple of 10 ms in the range 100 ms to 32.0 s and it shall be larger than

$$(1 + connPeripheralLatency) \times connSubrateFactor \times connInterval \times 2.$$

If, at any time in Connection State outside a connection event after the connection has been established, the timer $T_{LLconnSupervision}$ reaches the *connSupervisionTimeout* value, the connection shall be considered lost (see Section 4.5.12).

If, at any time in Connection State outside a CIS event after the CIS has been established, the timer $T_{CISSupervision}$ reaches the *connSupervisionTimeout* value, the CIS shall be considered lost.

In either case, the Controller may send the notification of the loss earlier provided that the most recent event has closed and the timer will reach the *connSupervisionTimeout* value before the next ACL or CIS anchor point.

In either case the Controller may, but should not, consider the connection lost at any time within an event after the timer reaches *connSupervisionTimeout*.

The supervision timeout can be changed using the Connection Update procedure (see Section 5.1.1), the Connection Parameters Request procedure (see Section 5.1.7), the Connection Subrate Update procedure (see Section 5.1.19), or the Connection Rate Update procedure (see Section 5.1.32). If any of these procedures are used, then the new value shall be greater than twice the ISO_Interval of any associated CIS.

Each *supervision timeout period* starts with the event following the reception of a valid packet and ends with the last event before the timer reaches the *connSupervisionTimeout* value.

### 4.5.3 Connection setup - connection event transmit window

To allow the Central to efficiently schedule connection events for multiple connections or other activities it may be involved in, the Central has the flexibility to schedule the

first connection event anchor point at a time of its choosing. The CONNECT_IND and AUX_CONNECT_REQ PDUs include parameters to determine when the Central sends its first packet in the Connection State to set the anchor point and when the Peripheral listens.

The CONNECT_IND and AUX_CONNECT_REQ PDUs include three parameters used to determine the transmit window. The transmit window starts at *transmitWindowDelay* + *transmitWindowOffset* after the end of the packet containing the CONNECT_IND PDU or AUX_CONNECT_REQ PDU, and the *transmitWindowSize* parameter shall define the size of the transmit window. The *connInterval* is used in the calculation of the maximum offset and size of the transmit window. The *transmitWindowOffset* and *transmitWindowSize* parameters are determined by the Link Layer.

The *transmitWindowOffset* shall be a multiple of 1.25 ms in the range 0 ms to *connInterval*. The *transmitWindowSize* shall be a multiple of 1.25 ms in the range 1.25 ms to the lesser of 10 ms and (*connInterval* - 1.25 ms).

Therefore the start of the first packet will be no earlier than *transmitWindowDelay* + *transmitWindowOffset* and no later than *transmitWindowDelay* + *transmitWindowOffset* + *transmitWindowSize* after the end of the packet containing the CONNECT_IND PDU or AUX_CONNECT_REQ PDU.

The value of *transmitWindowDelay* shall be 1.25 ms when a CONNECT_IND PDU is used, 2.5 ms when an AUX_CONNECT_REQ PDU is used on an LE Uncoded PHY, and 3.75 ms when an AUX_CONNECT_REQ PDU is used on the LE Coded PHY.

### 4.5.4 Connection setup – Central Role

After the initiator sends a CONNECT_IND PDU on the primary advertising physical channel or receives an AUX_CONNECT_RSP PDU on the secondary advertising physical channel, the Link Layer is in the Connection state in the Central Role.
The Central shall reset the Link Layer connection supervision timer $T_{LLconnSupervision}$.
The Link Layer shall notify the Host that the connection has been created. The first connection event shall use the data channel index specified in Section 4.5.8.

The Central shall start to send the first packet within the transmit window as defined in Section 4.5.3. The Central's first packet can extend beyond the transmit window.

The first packet sent in the Connection State by the Central determines the anchor point for the first connection event, and therefore the timings of all future connection events in this connection.

The second connection event anchor point shall be *connInterval* after the first connection event anchor point. All the normal connection event transmission rules specified in Section 4.5.1 shall apply.

Two examples of the LL connection setup procedure timing from the Central's perspective are shown in Figure 4.48 and in Figure 4.49.

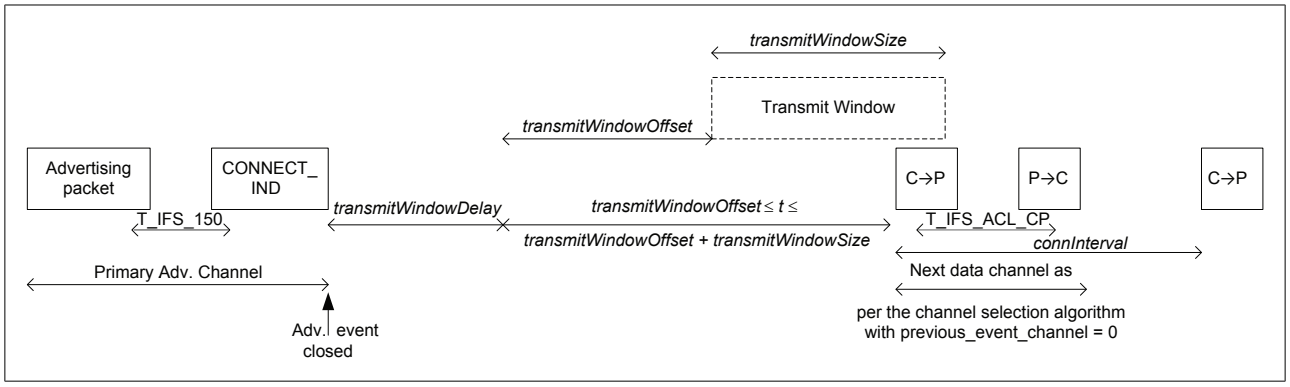



*Figure 4.48: Central's view of LL connection setup with CONNECT_IND*

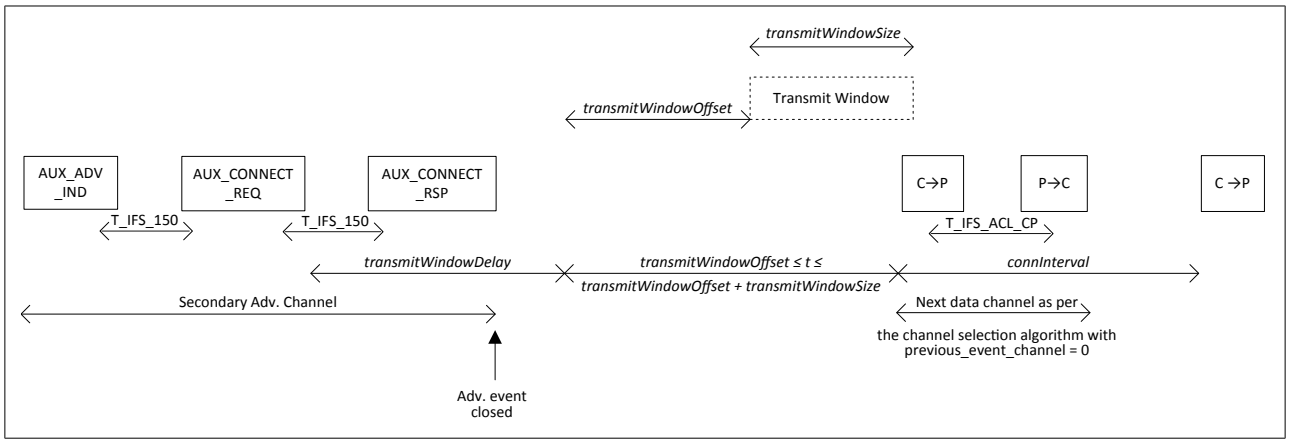



*Figure 4.49: Central's view of LL connection setup with AUX_CONNECT_REQ*

### 4.5.5 Connection setup – Peripheral Role

After the advertiser receives a CONNECT_IND PDU on the primary advertising physical channel or sends an AUX_CONNECT_RSP PDU on the secondary advertising physical channel, the Link Layer is in the Connection state in the Peripheral Role. The Peripheral shall reset the Link Layer connection supervision timer $T_{LLconnSupervision}$ . The Link Layer shall notify the Host that the connection has been created. The first connection event shall use the data channel index specified in Section 4.5.8.

The Peripheral shall start to listen for the first packet within the transmit window as defined in Section 4.5.3; while doing so, it shall perform the window widening specified in Section 4.2.4. The Central's first packet can extend beyond the transmit window, and therefore the Peripheral must take this into account.

The first packet received, regardless of a valid CRC match (i.e., only the access address needs to match), in the Connection State by the Peripheral determines the

anchor point for the first connection event, and therefore the timings of all future connection events in this connection.

If a packet is not received in a transmit window, the Peripheral shall attempt to receive a packet in a subsequent transmit window. A subsequent transmit window shall start *connInterval* after the start of the previous transmit window, with the same *transmitWindowSize*. The data channel index shall be the next data channel index as specified in Section 4.5.8. The *connEventCount* shall also be incremented by one.

Two examples of the procedure from the Peripheral's perspective are shown in Figure 4.50 and in Figure 4.51. In these examples the Peripheral fails to receive any part of the first packet (i.e., connEventCount = 0) from the Central and acquires anchor point timing from the second packet (i.e., connEventCount = 1).

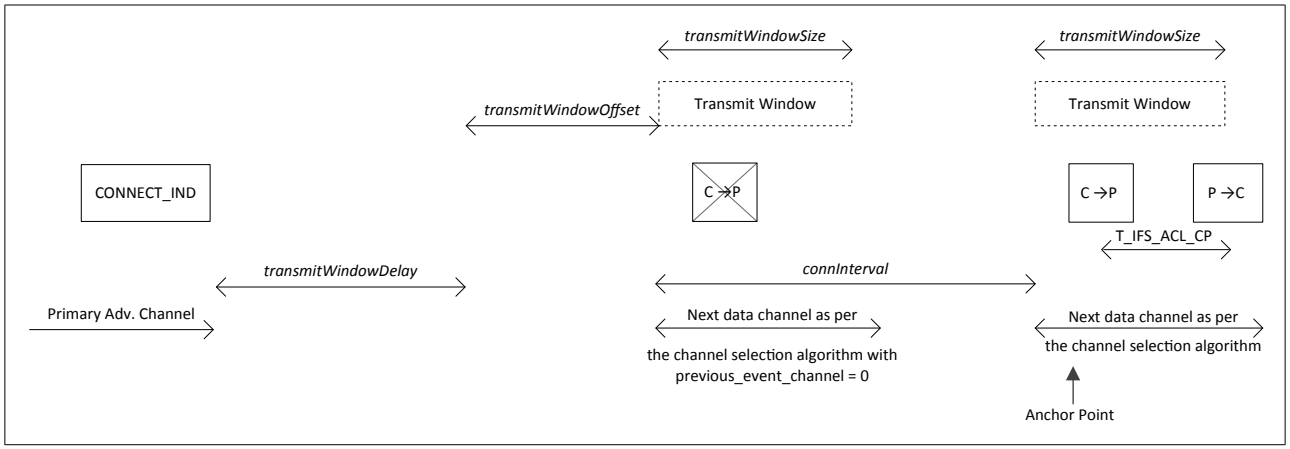



*Figure 4.50: Peripheral closing LL connection setup in the second LL connection event with CONNECT_IND*

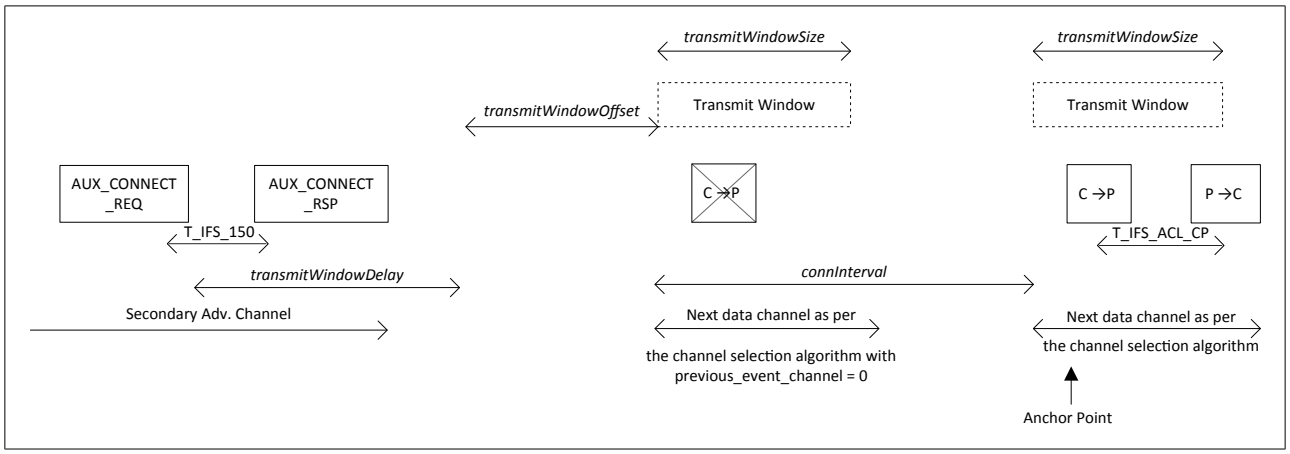



*Figure 4.51: Peripheral closing LL connection setup in the second LL connection event with AUX_CONNECT_REQ*

The Peripheral shall be active in every connection event until it receives a packet from the Central with the NESN set to one. From then on it may use Peripheral latency as defined in Section 4.5.1.

### 4.5.6 Closing connection events

The MD bit of the Header field of the Data Physical Channel PDU is used to indicate that the device has more data to send. If neither device has set the MD bit in their packets, the packet from the Peripheral closes the connection event. If either or both of the devices have set the MD bit, the Central may continue the connection event by sending another packet, and the Peripheral should listen after sending its packet.

Failure to receive a packet, or two consecutive packets received with an invalid CRC match within a connection event shall close the event.

MD bit usage is summarized in Table 4.6.

| | | **Central** | |
|---|---|---|---|
| | | **MD = 0** | **MD = 1** |
| **Peripheral** | **MD = 0** | Central shall not send another packet, closing the connection event.<br>Peripheral does not need to listen after sending its packet. | Central may continue the connection event.<br>Peripheral should listen after sending its packet. |
| | **MD = 1** | Central may continue the connection event.<br>Peripheral should listen after sending its packet. | Central may continue the connection event.<br>Peripheral should listen after sending its packet. |

*Table 4.6: MD bit usage for closing connection events*

### 4.5.7 Sleep clock accuracy

Because of sleep clock accuracies (see Section 4.2.2), there is uncertainty in the Peripheral of the exact timing of the Central's anchor point. Therefore the Peripheral shall re-synchronize to the Central's anchor point at each connection event where it listens for the Central. If the Peripheral receives a packet from the Central at the anchor point, regardless of a CRC match, the Peripheral shall update its timing of the anchor point.

If it listens for a packet, the Peripheral shall perform the window widening described in Section 4.2.4 at each anchor point and during the Connection Update procedure (see Section 5.1.1) and Connection Parameters Request procedure (see Section 5.1.7). In doing so, and in the absence of more accurate information about the Central's clock, the Peripheral shall use the Central's sleep clock accuracy (*centralSCA*) from the

most recent CONNECT_IND, AUX_CONNECT_REQ, LL_CLOCK_ACCURACY_REQ, or LL_CLOCK_ACCURACY_RSP PDU on the connection.

#### 4.5.7.1 Sleep clock accuracy for Channel Sounding

Sleep clock accuracies (see Section 4.2.2) result in uncertainty in the device's exact timing of the start of a Channel Sounding event (see Section 4.5.18.1). If the device is a reflector then it shall re-synchronize to the start of each subevent. The Central and Peripheral may either assume the initiator or reflector roles. In the case that the Central is the initiator, then the reflector shall use the Central's sleep clock accuracy (centralSCA). Alternatively, if the Peripheral is the initiator, then the reflector shall use the Peripheral's sleep clock accuracy (peripheralSCA).

The connection events that offset Channel Sounding events within that Channel Sounding procedure shall be subrated connection events.

A device that is a Peripheral and the initiator shall listen to the connection event from which a Channel Sounding event is directly offset even if Peripheral latency means it would not normally do so (see Section 4.5.1).

At the beginning of each Channel Sounding subevent which includes the beginning of a Channel Sounding event, in the case that the device is the reflector, it shall perform window widening described in Section 4.2.4. In doing so, and in the absence of more accurate information about the initiator's clock, the reflector shall use the sleep clock accuracy from the most recent LL_CLOCK_ACCURACY_REQ or LL_CLOCK_ACCURACY_RSP on the connection.

### 4.5.8 General-purpose channel group index selection

This section specifies the requirements for channel classification and several channel selection algorithms used for connection events, advertising events, and isochronous events. Additional channel selection algorithms for Channel Sounding are described in [Vol 6] Part H, Section 4.

#### 4.5.8.1 Channel classification

Support for Adaptive Frequency Hopping (AFH) is mandatory on the general-purpose channels. The Link Layer can classify each general-purpose channel as being *unknown*, *bad*, or *good*. These classifications are determined individually by the Link Layer based on local information (e.g., from active or passive channel assessment methods or from the Host). Information received from other devices (e.g., via an LL_CHANNEL_MAP_IND) shall not be included in the channel classification. The Host may provide channel classification information to the Link Layer. The Link Layer may use the information provided by the Host.

The three possible channel classifications are defined in Table 4.7.

| Classification | Definition |
|---|---|
| *unknown* | A channel shall be classified as *unknown* if the channel assessment measurements are insufficient to reliably classify the channel. |
| *bad* | A channel may be classified as *bad*, for example, when it is marked as *bad* in the most recent HCI_LE_Set_Host_Channel_Classification command or when the assessment of failure rate or interference with other systems exceeds some threshold. |
| *good* | A channel shall be classified as *good* if it is neither *unknown* nor *bad*. |

*Table 4.7: Channel classification descriptions*

The Central's, periodic advertiser's, and isochronous broadcaster's Link Layer shall classify the RF channels in the general-purpose group into *used channels* (used for transmitting data) and *unused channels* (not used for transmitting data). This is called the channel map. The minimum number of used channels shall be 2.

A Central shall determine a channel map for the connection based on any combination of the following information:

- Channel classification from local measurements (e.g., active or passive channel assessment in the Controller or input from other collocated technologies)
- Channel classification information from the Host
- Channel classification reports received from the Peripheral in LL_CHANNEL_STATUS_IND PDUs (see Section 2.4.2.39)

The algorithm used by the Central to combine these information sources and generate the channel map is not defined in the specification and is vendor-specific.

For a connection, the Peripheral shall receive the channel map from the Central in the CONNECT_IND PDU or the AUX_CONNECT_REQ PDU. If the Central changes the channel map it shall notify the Peripheral as specified in Section 5.1.2. If the periodic advertiser changes the channel map then it shall notify any scanning devices using the Channel Map Update Indication (see Section 1.20 of [1]). On periodic advertising with responses, the advertiser shall transmit the Channel Map Update Indication data type in each subevent. If the isochronous broadcaster changes the channel map it shall notify any Synchronized Receivers as specified in Section 5.6.1.

#### 4.5.8.2 Channel Selection algorithm #1

Channel Selection Algorithm #1 only supports channel selection for connection events.

Channel Selection Algorithm #1 consists of two stages: calculation of the unmapped channel index followed by mapping this index to a data channel index from the set of *used channels*.

The *unmappedChannel* and *lastUnmappedChannel* are the unmapped channel indices of two consecutive connection events. The *unmappedChannel* is the unmapped channel index for the current connection event. The *lastUnmappedChannel* is the unmapped channel index of the previous connection event. The *lastUnmappedChannel* shall be 0 for the first connection event of a connection.

At the start of a connection event, *unmappedChannel* shall be calculated using the following basic algorithm:

*unmappedChannel* = (*lastUnmappedChannel* + *hopIncrement*) *mod* 37

When a connection event closes, the *lastUnmappedChannel* shall be set to the value of the *unmappedChannel*.

If the *unmappedChannel* is a *used channel* according to the channel map, Channel Selection Algorithm #1 shall use the *unmappedChannel* as the data channel index for the connection event.

If the *unmappedChannel* is an *unused channel* according to the channel map, the *unmappedChannel* shall be re-mapped to one of the *used channels* in the channel map using the following algorithm:

*remappingIndex* = *unmappedChannel mod numUsedChannels*

where *numUsedChannels* is the number of *used channels* in the channel map.

A remapping table is built that contains all the *used channels* in ascending order, indexed from zero. The *remappingIndex* is then used to select the data channel index for the connection event from the remapping table.

The complete procedure is as shown in Figure 4.52.

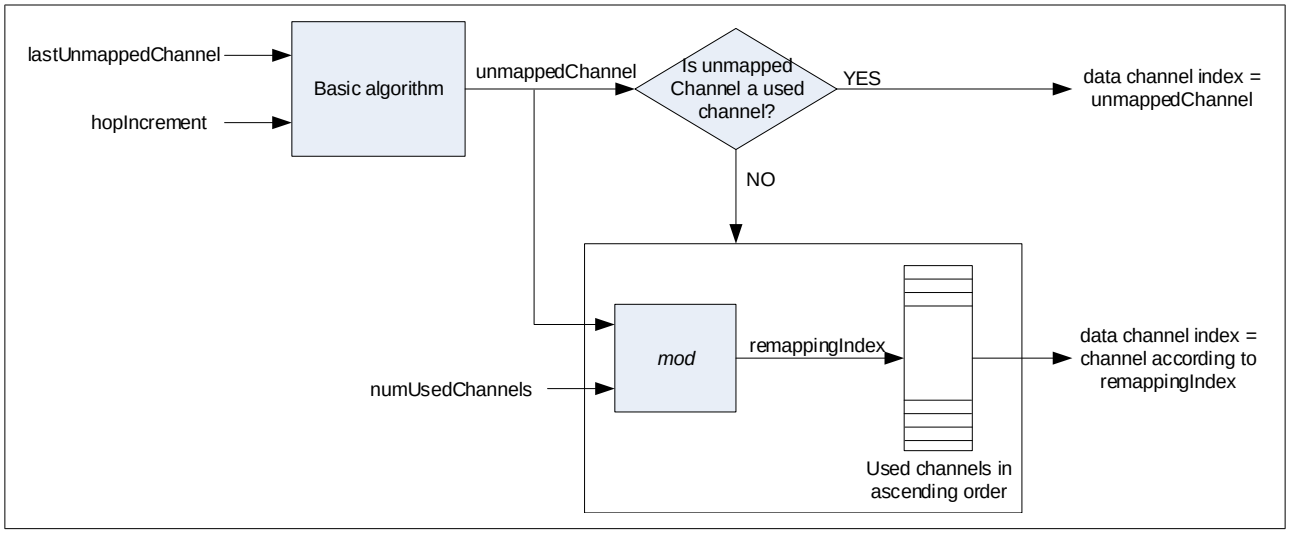



*Figure 4.52: Block diagram of Channel Selection algorithm #1*

### 4.5.8.3 Channel Selection algorithm #2

#### 4.5.8.3.1 Overview

Channel Selection Algorithm #2 supports channel selection for both events and subevents.

For each connection event, isochronous subevent (which can be a BIS or CIS subevent), periodic advertising without responses event, or periodic advertising with responses subevent, the algorithm described here generates an event or subevent channel index (which is a general purpose channel index).

Note: In a given isochronous event, Channel Selection Algorithm #2 results in two consecutive subevents using different channel indices.

A block diagram of the overall algorithm is shown in Figure 4.53. The upper part generates "event" channel indices and the lower part "subevent" channel indices; in some cases event channel indices are used for subevents. Connection events, periodic advertising events, the first subevent of each isochronous event, and BIG control subevents shall use the event channel index as defined in Section 4.5.8.3.3 and Section 4.5.8.3.4; all subsequent subevent(s) in the same isochronous event shall use the subevent channel index as defined in Section 4.5.8.3.5 and Section 4.5.8.3.6.

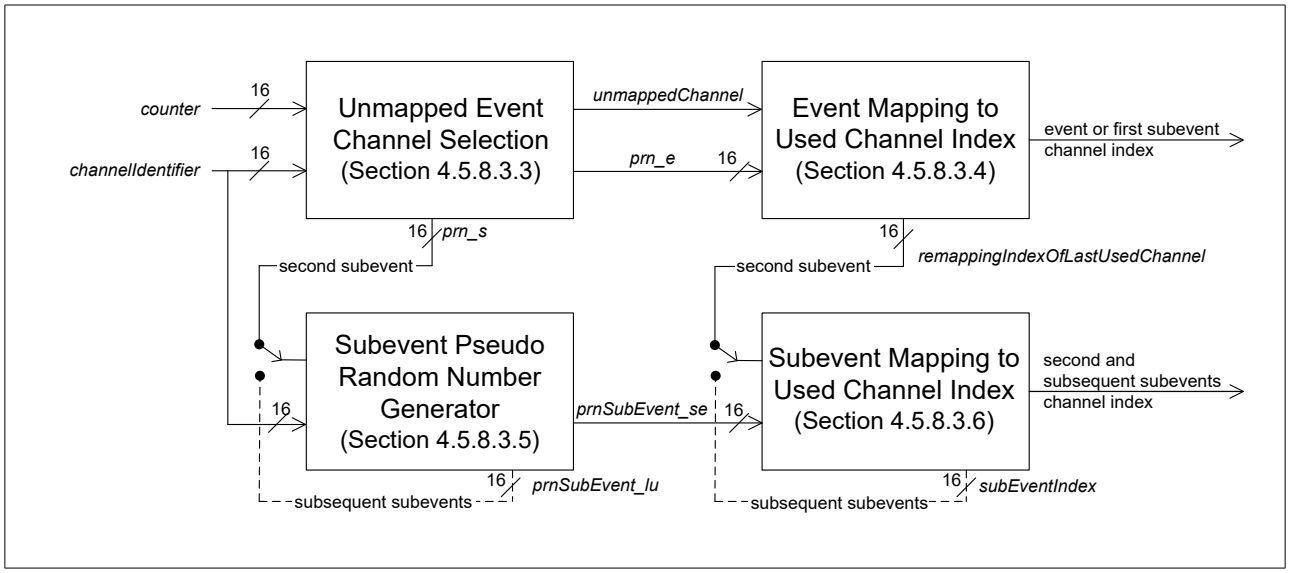



*Figure 4.53: General block diagram of Channel Selection algorithm #2*

#### 4.5.8.3.2 Inputs and basic components

The algorithm makes use of several inputs and basic components.

The 6-bit input *N* is the number of channels classified as *Used* channels.

The 16-bit input *channelIdentifier* is fixed for any given connection, BIS, or periodic advertising train; it is calculated from the Access Address by: *channelIdentifier* = (Access Address$_{31\text{-}16}$) ⊕ (Access Address$_{15\text{-}0}$)

The 16-bit input *counter* depends on the event or sub-event type.

- For ACL connections, it is the connection event counter *connEventCounter* defined in Section 4.5.1.
- For periodic advertising without responses, it is the event counter *paEventCounter* defined in Section 4.4.2.1.
- For periodic advertising with responses, it is the XOR of the two event counters *paEventCounter* and *paSubEventCounter* defined in Section 4.4.2.1 (so is different for each subevent of an event).
- For isochronous logical transports, it is bits 0 to 15 of the event counter *bigEventCounter* defined in Section 4.4.6.3 or *cisEventCounter* defined in Section 4.5.13.1 (so does not change during an event).

For isochronous events, the input *se_n* is defined in Section 4.4.6.8 for BIS events and Section 4.5.13.6 for CIS events.

The permutation operation consists of separately bit-reversing the lower 8 input bits and upper 8 input bits, as illustrated in Figure 4.54.

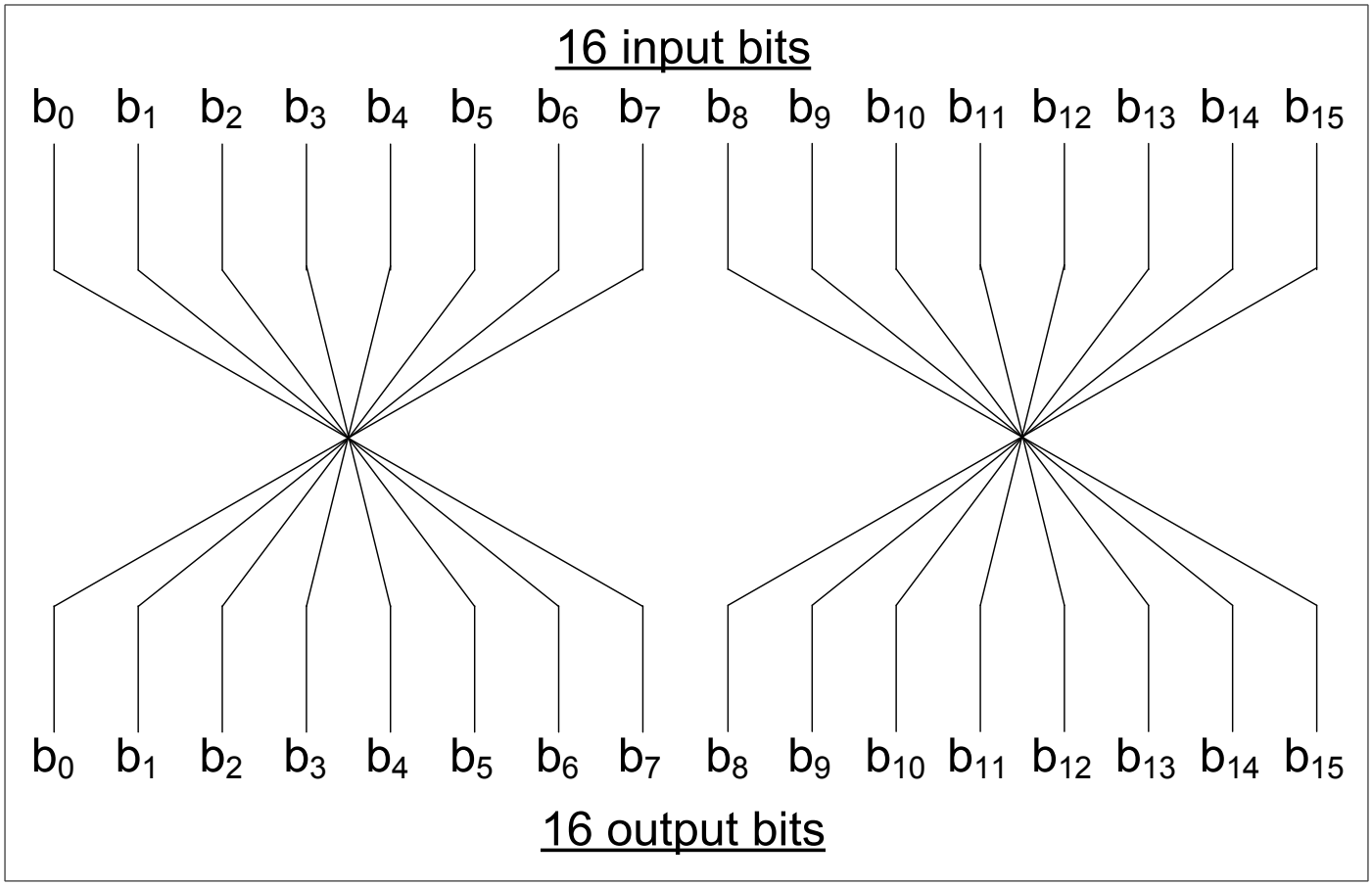



*Figure 4.54: Permutation operation*

The Multiply, Add, and Modulo (MAM) block performs a multiplication operation, an addition operation, and a *mod* operation, as illustrated in Figure 4.55.

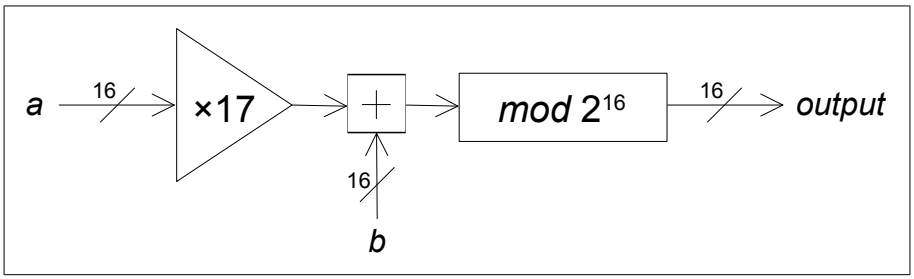



*Figure 4.55: Multiply, Add, and Modulo block operation*

The output of the MAM operation, given inputs *a* and *b*, is:

$$output = (17 \times a + b) \bmod 2^{16}$$

A *remapping table* is built that contains all the *used channels* in ascending order, indexed from zero.

#### 4.5.8.3.3 Unmapped event channel selection

The unmapped event channel selection process consists of two stages. First, the two unsigned pseudo-random numbers *prn_e* and *prn_s* are generated (*prn_s* is only needed for subevent channel selection), after which the unmapped channel index *unmappedChannel* is derived from *prn_e*.

The first stage shall be as shown in Figure 4.56.

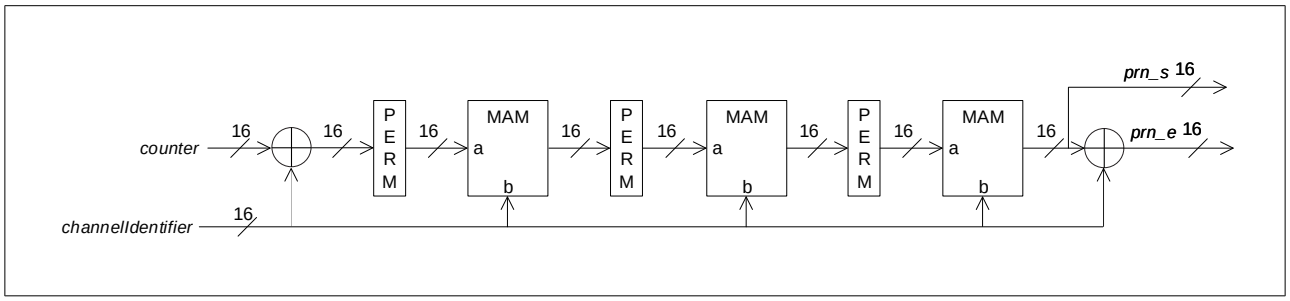



*Figure 4.56: Event pseudo-random number generation*

*unmappedChannel* is then calculated as *prn_e mod* 37. A block diagram of the overall process is shown in Figure 4.57.

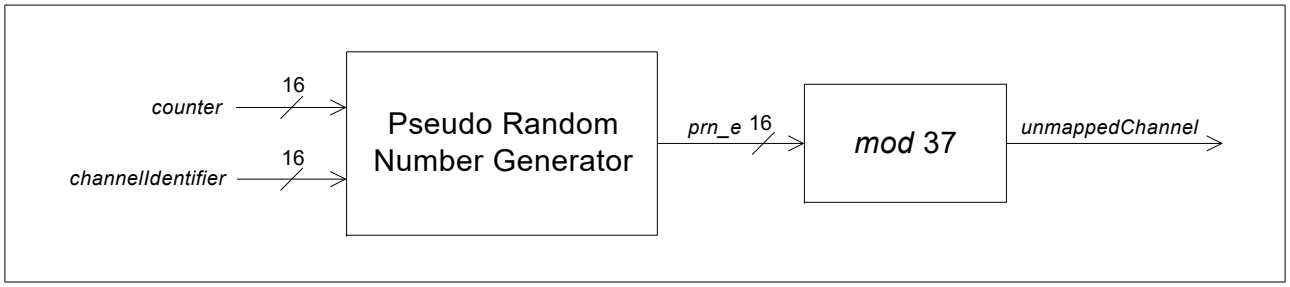



*Figure 4.57: Unmapped channel selection process*

#### 4.5.8.3.4 Event mapping to used channel index

If *unmappedChannel* is the channel index of a *used channel* according to the channel map, it is used as the channel index for the event. If *unmappedChannel* is the index of an *unused channel* according to the channel map, then the channel index for the event is calculated from *prn_e* and *N* (the number of *used channels*) by first calculating the value *remappingIndex* as:

$$remappingindex = \left\lfloor \left( \frac{N \times prn_e}{2^{16}} \right) \right\rfloor$$

and then using *remappingIndex* as an index into the remapping table to obtain the channel index for the event.

In either case, the value *remappingIndexOfLastUsedChannel* is the index in the remapping table of the channel index for the event. This value is only needed for subevent channel selection.

The overall process is illustrated in Figure 4.58.

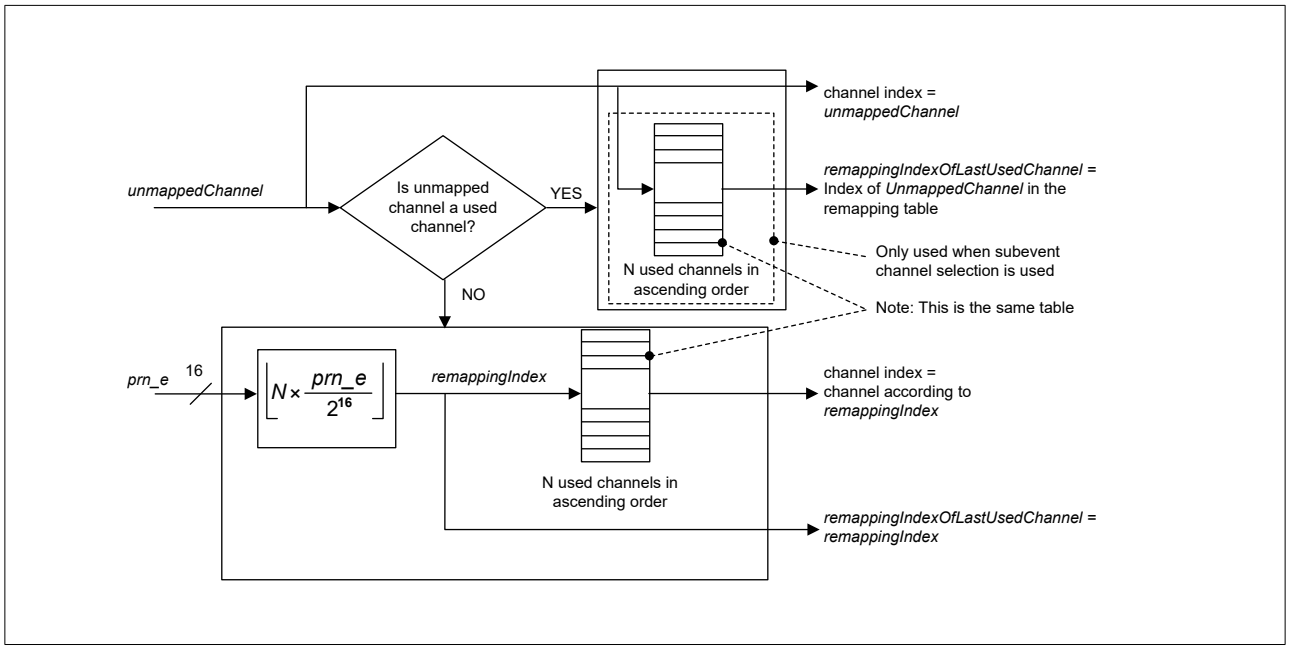



*Figure 4.58: Event mapping to used channel index process*

#### 4.5.8.3.5 Subevent pseudo-random number generation

Subevent pseudo-random number generation involves generating two more pseudo-random numbers *prnSubEvent_se* and *prnSubEvent_lu* for each subevent except the first. The process shall be as shown in Figure 4.59.

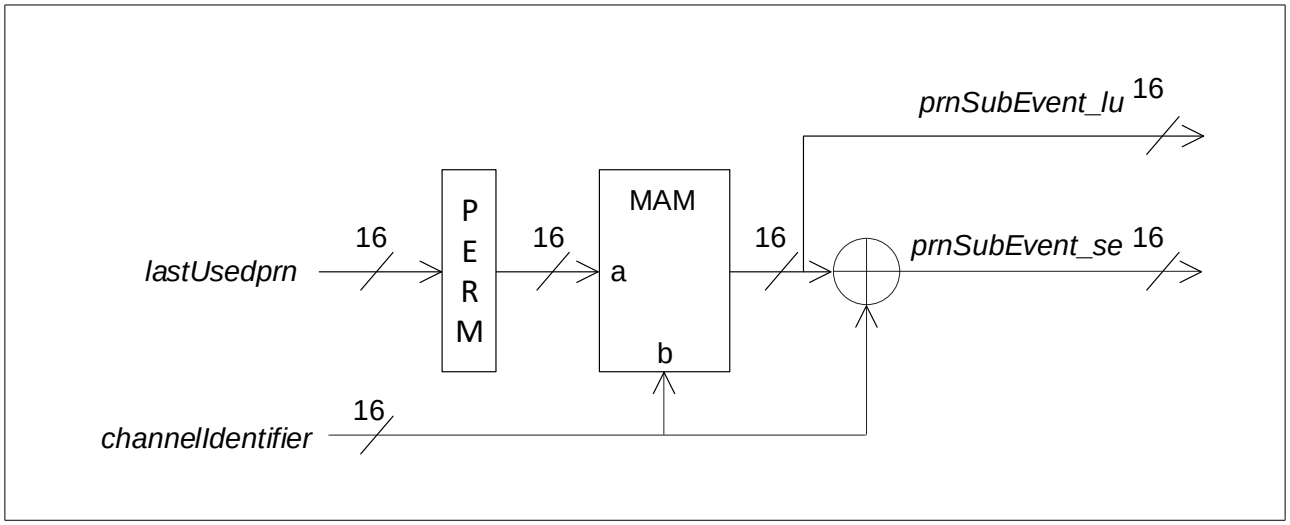



*Figure 4.59: Subevent pseudo-random number generation process*

Where:

$$lastUsedprn_{se_n} = \begin{cases} prn_s & for\ se_n = 2 \\ prnSubEvent_lu_{se_n-1} & for\ se_n > 2 \end{cases}$$

#### 4.5.8.3.6 Subevent mapping to used channel index

The channel index for a subevent is determined in two stages: calculating the subevent index *subEventIndex* and then indexing into the remapping table. The value *subEventIndex* is calculated as:

$$subEventIndex_{se_n} = \left(indexOfLastUsedChannel_{se_n} + d + \left\lfloor prnSubEvent_se_{se_n} \times \frac{N - 2d + 1}{2^{16}} \right\rfloor\right) \bmod N$$

where *indexOfLastUsedChannel* is:

$$indexOfLastUsedChannel_{se_n} = \begin{cases} remappingIndexOfLastUsedChannel & for\ se_n = 2 \\ subEventIndex_{se_n-1} & for\ se_n > 2 \end{cases}$$

and *d* is calculated as:

$$d = max\left(1, max\left(min(3, N - 5), min\left(11, \left\lfloor \frac{N - 10}{2} \right\rfloor\right)\right)\right)$$

where *d* is the minimum distance between the channel indices used in consecutive subevents.

The value of *subEventIndex* is then used as an index into the remapping table. The overall process is illustrated in Figure 4.60.

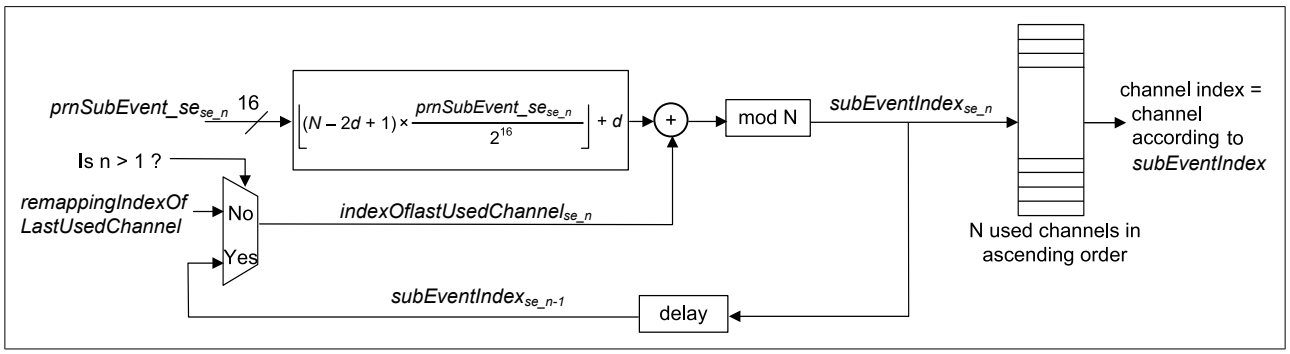



*Figure 4.60: Subevent mapping to used channel index process*

### 4.5.9 Acknowledgment and flow control

The Link Layer acknowledgment and flow control scheme shall be used in all ACL connections and CISes. Unless specified otherwise, the unqualified use of “PDU” in this section means either a Data Physical Channel PDU or a CIS Data PDU.

For each connection the Link Layer has two parameters, *transmitSeqNum* and *nextExpectedSeqNum*, each one bit in size. The *transmitSeqNum* parameter is used to identify packets sent by the Link Layer. The *nextExpectedSeqNum* parameter is used by the Link Layer to either acknowledge the last PDU sent by the peer, or to request the peer to resend the last PDU sent.

The *transmitSeqNum* and *nextExpectedSeqNum* parameters for an ACL or CIS shall be set to zero upon entering the Connection State or when the CIS is created.

If the last Data Physical Channel PDU was sent on the LE Coded PHY, the coding scheme (see Section 2.2.3) used when resending may be the same as or different from that used in the last Data Physical Channel PDU. If the instant of a PHY Update procedure (see Section 5.1.10) occurs while a Data Physical Channel PDU is waiting to be resent, the new PHY shall be used when resending.

A new PDU is a PDU sent for the first time by the Link Layer. A last PDU is a PDU that is resent by the Link Layer. When resending a Data Physical Channel PDU, the LLID, SN, and CP fields, the CTEInfo field (if present), and the Payload field of the sent Data Physical Channel PDU shall be equal to those of the last Data Physical Channel PDU sent by the Link Layer. When resending a CIS Data PDU, the LLID, SN, NPI fields, and the Payload field of the sent CIS Data PDU shall be equal to those of the last CIS Data PDU sent by the Link Layer.

For each new PDU that is sent, the SN bit of the Header field shall be set to *transmitSeqNum*. If a PDU is resent, then the SN bit shall not be changed.

Upon reception of a PDU, the SN bit shall be compared to *nextExpectedSeqNum*. If the bits are different, then this is a resent PDU, and *nextExpectedSeqNum* shall not be changed. If the bits are the same, then this is a new PDU, and *nextExpectedSeqNum* may be incremented by one (see Section 4.5.9.1).

When a PDU is sent, the NESN bit of the Header field shall be set to *nextExpectedSeqNum*.

Upon receiving a PDU (including a CIS Null PDU), if the NESN bit of that PDU is the same as *transmitSeqNum*, then the last sent PDU has not been acknowledged and shall be resent except as specified below. If the NESN bit of the PDU is different from *transmitSeqNum*, then the last sent PDU has been acknowledged, *transmitSeqNum* shall be incremented by one, and a new PDU may be sent.

The above process is illustrated in Figure 4.61 (tSqNo means *transmitSeqNum* and nExSqNo means *nextExpectedSeqNum*).

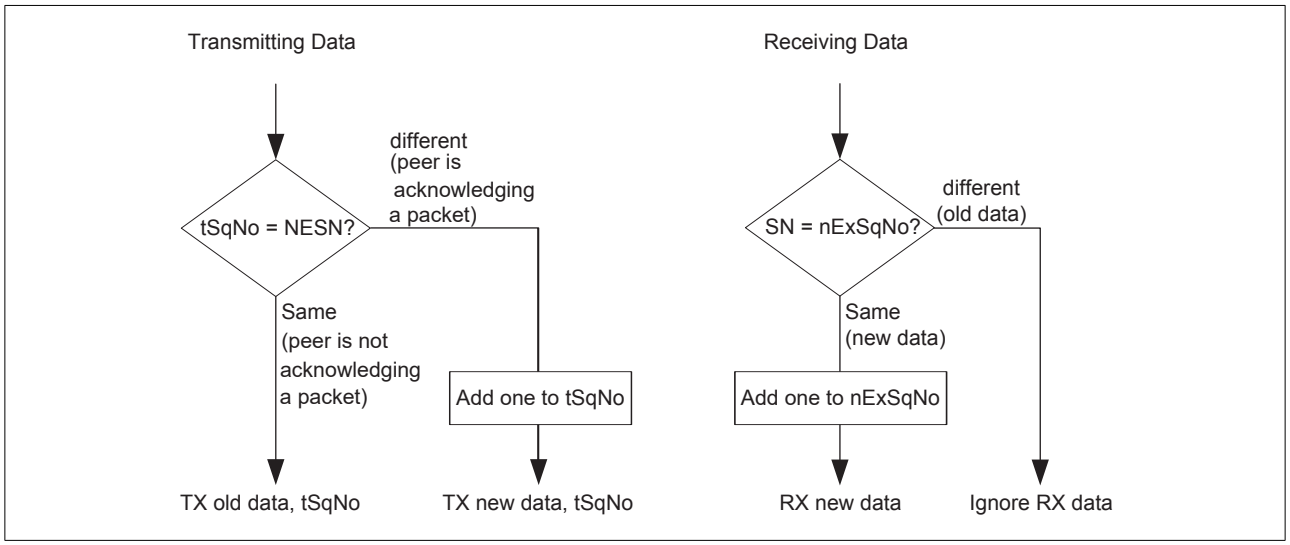



*Figure 4.61: Transmit and receive SN and NESN flow diagram*

If a PDU is received with an invalid CRC match, *nextExpectedSeqNum* shall not be changed except as specified below; this means that the PDU will not be acknowledged, causing the peer to resend the PDU. Since the received PDU has been rejected, the *nextExpectedSeqNum* from the peer device cannot be trusted, and therefore the last sent PDU from this device was not acknowledged and, as this section requires, must be retransmitted; *transmitSeqNum* shall not be changed.

The SN, NESN and MD bits shall be used from every received Data Physical Channel PDU which has passed the CRC check. The SN, NESN, CIE and NPI bits shall be used from every received CIS Data PDU that has passed the CRC check. The NESN, CIE and NPI bits shall be used from every received CIS Null PDU that has passed the CRC check. The PDU Payload field shall be ignored on every received PDU that has the same SN value as the previously received PDU.

When the transmitting Link Layer either does not send a CIS Data PDU for a given payload number or sends one but does not receive an acknowledgment for that PDU by the time the flush point occurs (see Section 4.5.13.5), the *transmitSeqNum* shall be incremented by one and the PDU shall not be retransmitted after the flush point.

When the Link Layer expecting to receive a CIS Data PDU either does not receive that PDU or fails to acknowledge the PDU by the time the flush point occurs, the *nextExpectedSeqNum* shall be incremented by one and the Link Layer shall not expect the PDU to be retransmitted after the flush point.

The increments to *transmitSeqNum* and *nextExpectedSeqNum* shall not happen before the end of the CIS subevent that marks the flush point of the PDU in question.

These rules mean that both devices act, for the purposes of this section, as if, for every payload number, a CIS Data PDU was received and acknowledged at the flush point of that payload number if not before. For example, this means that two consecutive transmitted or successfully received CIS Data PDUs can have the same sequence number but different contents because the intermediate PDU was not transmitted. For any CIS Data PDU, *transmitSeqNum* will equal $cisPayloadCounter_0$.

#### 4.5.9.1 Flow control

A Link Layer may fail to update *nextExpectedSeqNum* for reasons, including, but not limited to, lack of receive buffer space. This will cause the peer to resend the Data Physical Channel PDU at a later time, thus enabling flow control.

### 4.5.10 Data PDU length management

The Controller shall maintain the following global parameters.

Note: All parameters with "Octets" in the name represent the length of the Payload field of an LL Data PDU. All parameters with "Time" in the name represent the time taken to transmit a packet in microseconds.

- *connInitialMaxTxOctets* - the value of *connMaxTxOctets* that the Controller will use for a new connection.
- *connInitialMaxTxTime* - a value that the Controller will use to determine the value of *connMaxTxTime* that it will use for a new connection.
- *connInitialMaxTxTimeUncoded* - the value of *connMaxTxTime* that the Controller will use for a new connection on an LE Uncoded PHY. The value of *connInitialMaxTxTimeUncoded* shall be the greater of 328 and the value of *connInitialMaxTxTime*.
- *connInitialMaxTxTimeCoded* - the value of *connMaxTxTime* that the Controller will use for a new connection on the LE Coded PHY. The value of *connInitialMaxTxTimeCoded* shall be the greater of 2704 and the value of *connInitialMaxTxTime*.
- *supportedMaxTxOctets* - the maximum value of *connMaxTxOctets* that the Controller supports.
- *supportedMaxTxTime* - the maximum value of *connMaxTxTime* that the Controller supports.
- *supportedMaxRxOctets* - the maximum value of *connMaxRxOctets* that the Controller supports.
- *supportedMaxRxTime* - the maximum value of *connMaxRxTime* that the Controller supports.

Note: 2704 µs is derived from the duration of a packet with a 27 octet Payload field when sent on the LE Coded PHY using S=8 coding.

The Controller shall maintain the following parameters for each connection:

- *connMaxTxOctets* - the maximum number of octets in the Payload field that the local device will send to the remote device.
- *connMaxRxOctets* - the maximum number of octets in the Payload field that the local device is able to receive from the remote device.
- *connRemoteMaxTxOctets* - the maximum number of octets in the Payload field that the remote device will send to the local device.
- *connRemoteMaxRxOctets* - the maximum number of octets in the Payload field that the remote device is able to receive from the local device.
- *connMaxTxTime* - the maximum number of microseconds that the local device will take to transmit a packet to the remote device.
- *connMaxRxTime* - the maximum number of microseconds that the local device can take to receive a packet from the remote device.
- *connRemoteMaxTxTime* - the maximum number of microseconds that the remote device will take to transmit a packet to the local device.
- *connRemoteMaxRxTime* - the maximum number of microseconds that the remote device can take to receive a packet from the local device.

The values of the above parameters (both global and per-connection) shall each be within the range given in Table 4.8:

| LE Data Packet Length Extension feature supported | LE Coded PHY feature supported | CTEs supported on Data Physical Channel PDUs | Parameters with names containing "Octets" | | Parameters with names containing "Time" | |
|---|---|---|---|---|---|---|
| | | | Minimum | Maximum | Minimum | Maximum |
| No | No | No | 27 | 27 | 328 | 328 |
| Yes | No | No | 27 | 251 | 328 | 2120 |
| No | No | Yes | 27 | 27 | 328 | 336 |
| Yes | No | Yes | 27 | 251 | 328 | 2128 |
| No | Yes | Don’t care | 27 | 27 | 328 | 2704 |
| Yes | Yes | Don’t care | 27 | 251 | 328 | 17040 |

*Table 4.8: Valid ranges for data PDU length management parameters*

The following values are derived from the parameters maintained by the Controller:

- *connEffectiveMaxTxOctets* - the lesser of *connMaxTxOctets* and *connRemoteMaxRxOctets*.
- *connEffectiveMaxRxOctets* - the lesser of *connMaxRxOctets* and *connRemoteMaxTxOctets*.
- *connEffectiveMaxTxTimeUncoded* - the lesser of *connMaxTxTime* and *connRemoteMaxRxTime*.
- *connIntervalRequired* - the value T_IFS_ACL_CP + T_MCES + min (*connEffectiveMaxRxTime*, ((*connEffectiveMaxRxOctets* × 64) + 976)).
- *connIntervalUncodedMin* - the value *connIntervalRequired* + 328.
- *connIntervalCodedMin* - the value *connIntervalRequired* + 2704.

  Note: 976 µs and 2704 µs are the durations of packets with a zero octet and 27 octet Payload field when sent on the LE Coded PHY using S=8 coding.
- *connIntervalPortionAvailable* - the current *connInterval* for the connection minus *connIntervalRequired*.
- *connEffectiveMaxTxTimeAvailable* - the lesser of *connEffectiveMaxTxTimeUncoded* and *connIntervalPortionAvailable*.
- *connEffectiveMaxTxTimeCoded* - the greater of 2704 and *connEffectiveMaxTxTimeAvailable*.
- *connEffectiveMaxTxTime* - equal to *connEffectiveMaxTxTimeUncoded* while the connection is on an LE Uncoded PHY and equal to *connEffectiveMaxTxTimeCoded* while the connection is on the LE Coded PHY.
- *connEffectiveMaxRxTimeUncoded* - the lesser of *connMaxRxTime* and *connRemoteMaxTxTime*.
- *connEffectiveMaxRxTimeCoded* - the greater of 2704 and *connEffectiveMaxRxTimeUncoded*.
- *connEffectiveMaxRxTime* - equal to *connEffectiveMaxRxTimeUncoded* while the connection is on an LE Uncoded PHY and equal to *connEffectiveMaxRxTimeCoded* while the connection is on the LE Coded PHY.

Note: Corresponding octet and time parameters do not have to be mutually consistent. For example, it is permissible for a time parameter to be 2120 µs even though, on some PHYs, the maximum possible time is less.

The Controller shall not change the values of *supportedMaxTxOctets*, *supportedMaxTxTime*, *supportedMaxRxOctets*, and *supportedMaxRxTime*.

For a new connection:

- *connMaxTxOctets* shall be set to *connInitialMaxTxOctets* and *connMaxRxOctets* shall be chosen by the Controller. If either value is not 27 then the Controller should initiate the Data Length Update procedure (Section 5.1.9) at the earliest practical opportunity.
- *connRemoteMaxTxOctets* and *connRemoteMaxRxOctets* shall be 27.

For a new connection on an LE Uncoded PHY:

- *connMaxTxTime* shall be set to *connInitialMaxTxTimeUncoded* and *connMaxRxTime* shall be chosen by the Controller. If either value is not 328, then the Controller should initiate the Data Length Update procedure (Section 5.1.9) at the earliest practical opportunity.
- *connRemoteMaxTxTime* and *connRemoteMaxRxTime* shall be 328.

For a new connection on the LE Coded PHY:

- *connMaxTxTime* shall be set to *connInitialMaxTxTimeCoded* and *connMaxRxTime* shall be chosen by the Controller. If either value is not 2704, then the Controller should initiate the Data Length Update procedure (Section 5.1.9) at the earliest practical opportunity.
- *connRemoteMaxTxTime* and *connRemoteMaxRxTime* shall be 2704.

The Controller may change the values of *connMaxTxOctets, connMaxRxOctets, connMaxTxTime,* and *connMaxRxTime* at any time after entering the Connection State. Whenever it does so, it shall communicate these values to the peer device using the Data Length Update procedure. The values shall not exceed the values of *supportedMaxTxOctets*, *supportedMaxTxTime*, *supportedMaxRxOctets*, and *supportedMaxRxTime* respectively.

The values of *connMaxTxOctets*, *connMaxRxOctets*, *connMaxTxTime*, and *connMaxRxTime* can be used to represent limitations in the implementation; for example, *connMaxRxOctets* can be set to the size of the receiver's data buffer or *connMaxTxTime* can be set so that the transmitter frequency will not have enough time to drift outside permitted limits. Their values are only restricted by Table 4.8 and there is no requirement to change them because, for example, the current value is greater than the largest possible value on a new PHY.

The Controller shall not transmit packets containing LL Data PDUs that have a maximum Payload field length greater than *connEffectiveMaxTxOctets* or that take more than *connEffectiveMaxTxTime* microseconds to transmit (omitting any Constant Tone Extension) except during the period where the values of either *connEffectiveMaxTxOctets* or *connEffectiveMaxTxTime* are being modified. During that period, the Controller may still have LL Data PDUs queued for transmission that

conformed to the old parameters but violate the new ones. These PDUs remain valid; only PDUs queued after the Data Length Update procedure has completed are required to conform to the changed parameters. However, a Controller should ensure that it has no LL Data PDUs queued for transmission when it transmits an LL_LENGTH_REQ or LL_LENGTH_RSP PDU.

Note: These requirements do not apply to LL Control PDUs (see Section 4.5.11).

If the Controller decreases the value of *connMaxRxOctets* or *connMaxRxTime*, it shall not apply the new values until a Data Length Update procedure (Section 5.1.9) that sends the new value has completed.

The Controller shall notify its Host if any of the parameters *connEffectiveMaxTxOctets*, *connEffectiveMaxRxOctets*, *connEffectiveMaxTxTime*, or *connEffectiveMaxRxTime* have changed.

Note: These parameters can change as part of a Data Length Update procedure (see Section 5.1.9), a PHY Update procedure (see Section 5.1.10), a Connection Update procedure (see Section 5.1.1), a Connection Parameters Request procedure (see Section 5.1.7), a Frame Space Update procedure (see Section 5.1.30), a Connection Rate Update procedure (see Section 5.1.32), or a Connection Rate Request procedure (see Section 5.1.33.

### 4.5.11 Control PDU length management

The Link Layer shall not transmit a packet containing an LL Control PDU with a CtrData field longer than 26 octets until it has successfully completed a Feature Exchange procedure (see Section 5.1.4) on the same connection.

If the Link Layer in the Central role supports receiving LL Control PDUs with a CtrData field longer than 26 octets, it should initiate the Feature Exchange procedure on each connection.

Note: As specified in Section 4.6, once the feature exchange has completed, the Link Layer must not use a procedure that the peer did not mark as supported. Therefore the Link Layer will never transmit an LL Control PDU with a CtrData field longer than 26 octets to a device that does not support it.

### 4.5.12 Connection termination and loss

An ACL or CIS connection can be terminated by either Link Layer using the ACL Termination procedure (see Section 5.1.6) or the CIS termination procedure (see Section 5.1.16) respectively. A connection can also be considered lost for various reasons. The Host shall be notified when the termination procedure completes, irrespective of whether the Central or Peripheral initiated it.

If an ACL connection is considered lost, the Link Layer shall not send or receive any further packets on the Data Physical Channel for the ACL connection or on the Isochronous Physical Channel for any associated CIS. Also in this case, any active CS Procedures and CS Procedure repeat instances shall be considered lost as described in [Vol 6] Part H, Section 4.4.5. The Link Layer shall exit the Connection State, shall transition to the Standby State, and shall notify the Host of the loss of the ACL and of any associated CIS.

If a CIS is considered lost, the Link Layer shall not send or receive any further packets on the Isochronous Physical Channel and shall notify the Host of the loss of the CIS. The associated ACL connection shall not be affected, except that the Link Layer shall not send any further PDUs related to that CIS on the ACL connection.

### 4.5.13 Connected Isochronous Stream (CIS)

A CIS is a logical transport that enables connected devices to transfer isochronous data in either direction. The data may be fixed or variable size and may be framed or unframed. The isochronous data can be transferred either in an LE-S or LE-F logical link using the CIS logical transport. Each CIS shall be associated with an ACL.

A CIS supports variable size packets and transmission of one or more packets in each CIS event, allowing a range of data rates to be supported. Data traffic is unidirectional or bidirectional between the devices. There is an acknowledgment protocol to improve the reliability of packet delivery in a CIS.

#### 4.5.13.1 CIS parameters

Each CIS shall have an identifier, denoted as CIS_ID, that is assigned by the Host. The CIS_ID shall be sent to the Peripheral's Host via the two Link Layers as part of creation of the CIS, but is not otherwise used by the Link Layer.

Each CIS is defined by the following parameters:

- ISO_Interval is the time between the CIS anchor points of adjacent CIS events.
- Sub_Interval is the time between start of two consecutive subevents of a CIS.
- SE_Length is the time that needs to be reserved for a subevent.
- Max_PDU is the maximum number of data octets that can be carried in each CIS Data PDU; the value may be different in each direction.
- Max_SDU is the maximum size of an SDU on this CIS (see [Vol 6] Part G, Section 1); the value may be different in each direction.
- MPT_C and MPT_P shall equal the time taken by the Central and Peripheral respectively to transmit a packet containing a CIS PDU with a Payload field of Max_PDU octets (for that direction) on the PHY being used for the CIS; on the LE

Coded PHY, the S=8 coding shall be assumed. These values should include the MIC if it is possible that the CIS will be encrypted.

- NSE is the maximum number of subevents in each CIS event.
- BN and FT control which data is transmitted in each CIS event; the values may be different in each direction.
- Framed indicates whether the CIS carries framed or unframed data; the value shall be the same in both directions.
- Framing mode indicates whether Segmentable or Unsegmented mode is being used when the CIS carries framed data. The value shall be the same in both directions.

These parameters shall not change during the lifetime of the CIS. They are discussed further in the following subsections. The mandatory range for each parameter is the entire range of valid values except for the following, where only the listed values are mandatory:

- BN: 0 and 1
- NSE: all supported values of BN except 0
- FT: 1

Note: The encryption status of a CIS follows the encryption status of the associated ACL.

Both the Central and Peripheral shall have a 39-bit counter *cisEventCounter*. It shall be set to 0 for the first CIS event of a CIS and incremented by 1 for each CIS event whether or not the Central transmits any Connected Isochronous PDUs during the event.

Each CIS shall have a 39-bit *cisPayloadCounter* associated with it, described further in Section 4.5.13.3. The Link Layer of the Central and Peripheral shall terminate the CIS no later than when *cisPayloadCounter* equals $2^{39} – 1$.

#### 4.5.13.2 CIS events and subevents

A CIS event is an opportunity for the Central and Peripheral to exchange CIS PDUs; CIS events occur at regular intervals. Each CIS event in turn contains NSE subevents. Each subevent can be used to transmit a CIS PDU from the Central to the Peripheral followed by a response from the Peripheral to the Central. As described in Section 4.5.13.4, not all subevents are always used in an event.

Each CIS event starts at a moment called the CIS anchor point and ends when closed as specified in Section 4.5.13.4. The CIS anchor points shall be spaced regularly, ISO_Interval apart.

The first subevent of a CIS event starts at the CIS anchor point. The start of two consecutive subevents of a CIS shall be Sub_Interval apart. A subevent ends at the end of the Peripheral's packet, if any, and at the end of the Central's packet if not.

Each CIS event normally contains at least one CIS PDU sent by the Central. The Central can, however, completely fail to transmit in a CIS event due to scheduling conflicts but shall transmit at least one CIS PDU within each CIS supervision timeout.

The length of a particular CIS event is at most (NSE − 1) × Sub_Interval + MPT_C + T_IFS_CIS + MPT_P.

The Link Layer shall transmit CIS PDUs only in CIS events. The Link Layer shall transmit only CIS PDUs as part of a CIS event.

Figure 4.62 shows a CIS with two subevents.

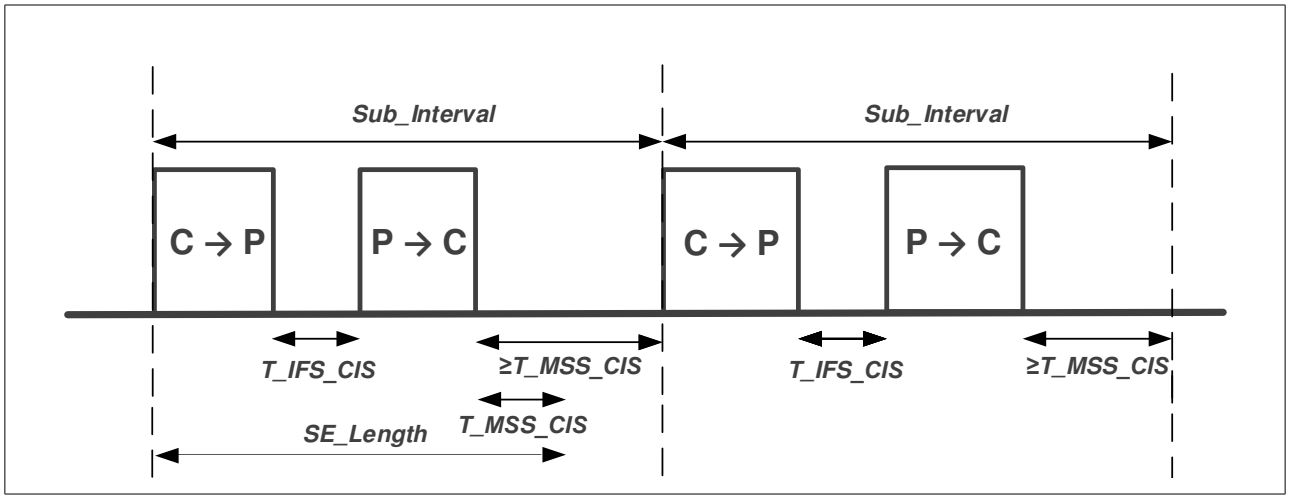



*Figure 4.62: Example of two CIS subevents*

Figure 4.63 shows an example of a CIS event where not all the subevents are used.

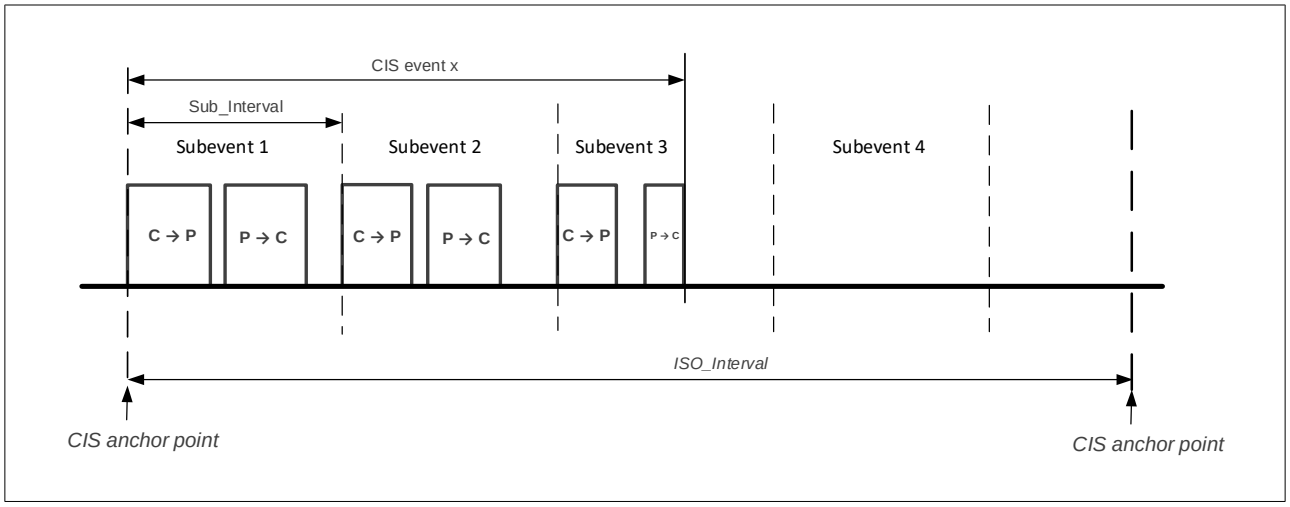



*Figure 4.63: Example of a CIS event in a CIS with NSE = 4 and 3 actual subevents*

The Central should transmit a packet at the start of each subevent until the event is closed. If the Peripheral receives a packet from the Central, regardless of whether the CRC is valid, it may transmit a response T_IFS_CIS after the end of the Central's packet. The Peripheral shall not transmit if it does not receive a packet from the Central in the same subevent. Where either device does not transmit during a subevent, the Link Layer shall behave for all other purposes (e.g. the timing of packets and the choice of payload) as if it had done so.

ISO_Interval shall be a multiple of 1.25 ms in the range of 5 ms to 4 s, shall be at least NSE × Sub_Interval, and shall be less than half the *connSupervisionTimeout* for the associated ACL.

SE_Length shall be MPT_C + T_IFS_CIS + MPT_P + T_MSS_CIS.

Sub_Interval shall be greater than or equal to SE_Length (also see Section 4.5.14.2).

BN shall be in the range 0 to 15. For a bidirectional link the value shall be non-zero for both directions. For a unidirectional link it shall be non-zero in the direction that data is being sent and zero in the other direction.

NSE shall be in the range max (BN_C_To_P, BN_P_To_C) to 31.

#### 4.5.13.3 Connected Isochronous Data

A CIS carries a single stream of isochronous data in each direction or in a single direction. The data is divided into payloads of at most Max_PDU octets, each of which is transmitted as the Payload field of a single CIS Data PDU; the payloads need not all be the same size and can be zero length.

Note: Max_PDU is the size of the data and omits the MIC in the CIS Data PDU, if any. Therefore, it has a value in the range 0 to 251.

The Framed parameter of a CIS shall indicate whether the CIS is framed or unframed. Framed CISes shall only use framed CIS Data PDUs to carry data; unframed CISes shall only use unframed CIS Data PDUs to carry data.

For each CIS event the source of the data shall supply, via ISOAL, a burst of data consisting of up to BN payloads, each of which in turn shall hold either a single fragment or one or more SDU segments, or shall be empty. This burst is associated with the corresponding CIS event but the payloads may be transmitted in later events as well; they shall not be transmitted in earlier events. These payloads shall be numbered in order (though not necessarily consecutively); this number shall be used as the value of *cisPayloadCounter* for the PDU containing that payload. The burst of payloads associated with the CIS event where *cisEventCounter* has the value E shall consist of payloads with *cisPayloadCounter* between E × BN and (E + 1) × BN - 1.

CIS Null PDUs do not have a payload and so do not have a *cisPayloadCounter*.

The payloads shall be transmitted in the order provided. In those subevents that the Link Layer transmits on, it shall continue to retransmit the same CIS Data PDU until either it is acknowledged or the data within it has reached its flush point. The Link Layer shall not transmit the CIS Data PDU with *cisPayloadCounter* N until either payload number N-1 has reached its flush point or the CIS Data PDU with *cisPayloadCounter* N-1 (if any) has been acknowledged (therefore if payload number N-1 is not provided, payload number N will be delayed until that flush point; the source of the data can provide an empty payload to avoid or reduce this delay). The Link Layer shall not transmit a payload after its flush point. If the source of the data fails to provide a payload in time for a CIS subevent, then the Link Layer shall either transmit a CIS Null PDU instead or not transmit on that subevent.

Note: It is not specified how the payload numbers assigned by ISOAL are communicated to the Link Layer or how the receiving Link Layer communicates the payload numbers to ISOAL.

Note: If BN is zero, then no payloads are provided and therefore the Link Layer will only transmit CIS Null PDUs.

#### 4.5.13.4 Closing CIS events

The Link Layer shall close a CIS event at the end of its last subevent.

The Central or Peripheral may close a CIS event early and may indicate this to the peer device by setting the Close Isochronous Event (CIE) bit. A device that sends a CIS PDU with the CIE bit set to 1 shall not transmit in the remaining subevents in the current CIS event.

Note: Link Layer implementations will normally end a CIS event early when all the scheduled payloads in both directions have been transmitted and acknowledged.

### 4.5.13.5 Flush Timeout and Flush Points

The Flush Timeout (FT) parameter is the maximum number of CIS events that may be used to transmit (and retransmit) a given payload. FT shall be between 1 and 255. Each payload number shall have a flush point: a point in time at which the corresponding payload and associated CIS Data PDU, if any, shall be discarded by the Link Layer. The flush point of a payload number occurs immediately after U subevents in the CIS event with *cisEventCounter* equal to (E + FT – 1), where:

- E = floor (*cisPayloadCounter* ÷ BN)
- U = NSE – floor (NSE ÷ BN) × (BN – 1 – *cisPayloadCounter mod* BN)

Figure 4.64 and Figure 4.65 show examples of data transmissions and payloads reaching their flush points. In these figures, "ACK" and "NAK" have the meaning given in Figure 4.61.

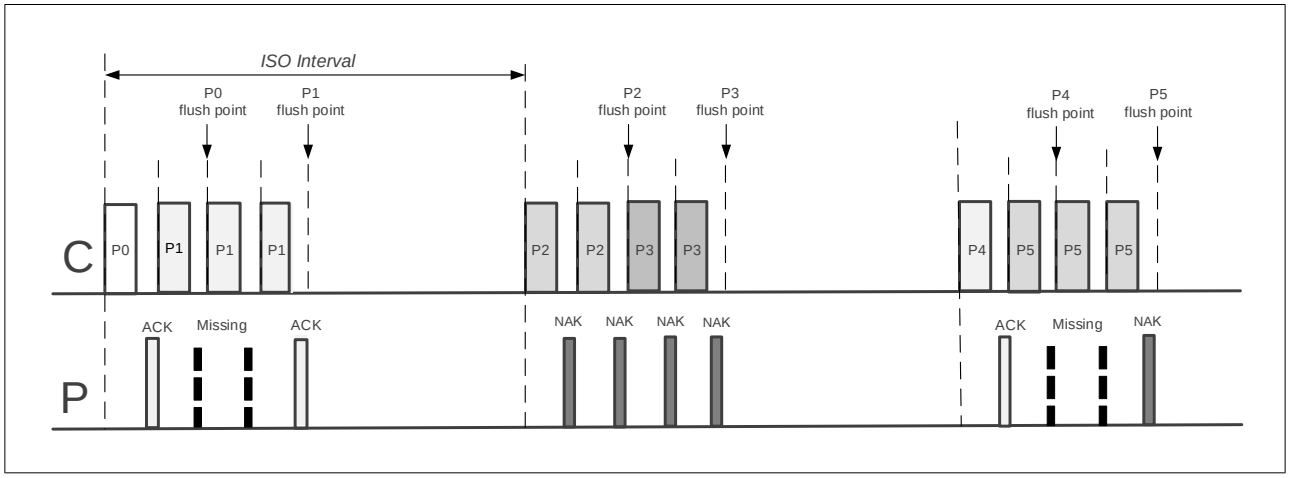



*Figure 4.64: Example of flush points with BN=2, FT=1, and NSE=4*

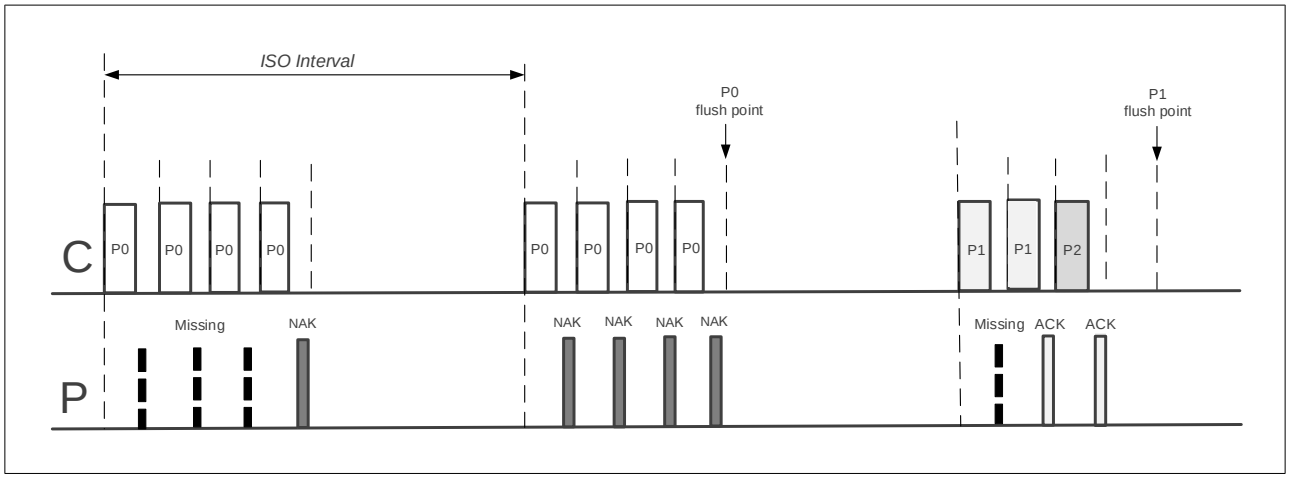



*Figure 4.65: Example of flush points with BN=1, FT=2, and NSE=4*

#### 4.5.13.6 Channel indices

Each packet containing a CIS PDU shall be transmitted on the channel index specified by Channel Selection Algorithm #2 (see Section 4.5.8.3). The subevent number *se_n* shall be set to the values 1 to NSE, in order, for the subevents of each CIS event. The Peripheral shall transmit on the same channel index as the Central. The channel map used by the CIS shall be the same as the channel map of the associated ACL.

#### 4.5.13.7 CIS Encryption

If an ACL is not encrypted, any associated CIS shall not be encrypted. If an ACL is encrypted, all associated CISes shall be encrypted, in which case all CIS Data PDUs (except those with an empty Payload field) on those CISes shall be encrypted using the same session key as that used by the associated ACL.

### 4.5.14 Connected Isochronous Group (CIG)

A CIG consists of either two or more CISes that have the same ISO_Interval and are expected to have a time relationship at the application layer, or of a single CIS. The maximum number of CISes in a CIG shall be 31. An implementation in the Central role is not required to support CIGs with more than 1 CIS.

The Central's Host assigns an identifier to each CIG, denoted by the parameter CIG_ID. The CIG_ID shall be sent to the Peripheral's Host via the two Link Layers as part of creation of each CIS in the CIG but is not otherwise used by the Link Layer.

All CISes in a CIG shall have the same Central but may have different Peripherals.

All CISes in a CIG shall have the same value of FT applying from the Central to the Peripheral and the same value of FT applying from the Peripheral to the Central (these two values may be different).

All CISes in a CIG shall have the same Framed configuration and the same Framing_Mode.

All CISes in a CIG shall have different CIS_IDs, but if a CIS is terminated or considered lost its configuration remains stored within the CIG so that another CIS may then be created in the same CIG with the same CIS_ID and configuration. The configuration is deleted when the CIG is removed.

The Link Layer may use different parameters for a CIS each time that it is created provided that the parameters meet the configuration provided by the Host.

On the Central, a CIG represents a data structure within the Link Layer and does not involve any connection separate from the CISes that make it up. On the Peripheral(s), it represents those CISes with the same CIG_ID.

#### 4.5.14.1 CIG event

A CIG event consists of the corresponding CIS events of the CISes currently making up that CIG. Each CIG event starts at the anchor point of the earliest (in transmission order) CIS of the CIG and ends at the end of the last subevent of the latest CIS of the same CIG event. Two CIG events on the same CIG shall not overlap (that is, the last CIS event of a given CIG event shall end before the first CIS anchor point of the next CIG event).

The Central's Link Layer shall provide timing parameters (CIS_Sync_Delay and CIG_Sync_Delay) to the Peripherals' Link Layers which enable synchronization of isochronous data at the application layer.

Each CIG event shall have a CIG reference point and a CIG synchronization point; these shall be CIG_Sync_Delay apart. Each CIG event shall start no earlier than the CIG reference point and shall end no later than the CIG Synchronization point. For a given CIS, the CIS anchor point shall be a fixed offset (which may be zero) after the CIG reference point; therefore CIG reference points are spaced ISO_Interval apart and CIG synchronization points are also spaced ISO_Interval apart. For each CIS, CIS_Sync_Delay shall equal the time from the CIS anchor point to the CIG synchronization point.

Figure 4.66 shows the various elements of a CIG event.

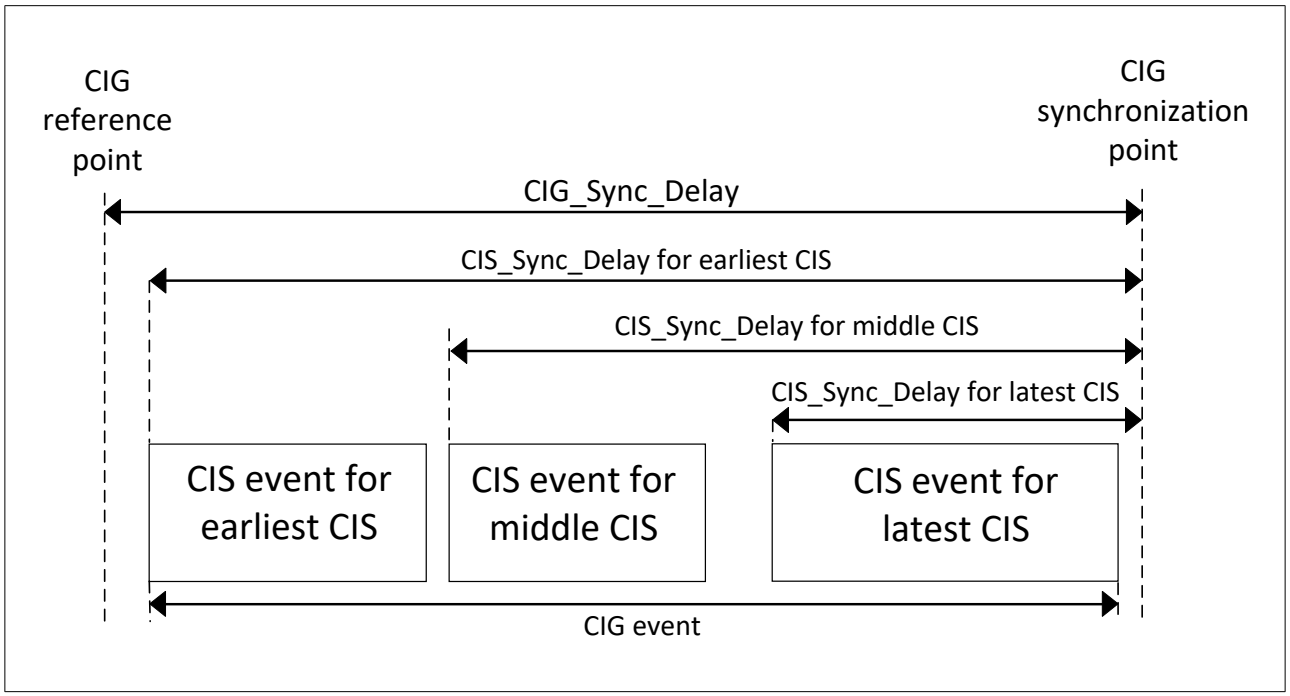



*Figure 4.66: Layout of a CIG event with three CISes*

Note: CIG_Sync_Delay will be no less than the maximum possible time for a CIG event; i.e. the time from the CIG reference point to the end of the Peripheral's packet in the last subevent when both Central and Peripheral transmit packets containing Max_PDU octets. It will have the same value for all CISes in the same CIG. CIS_Sync_Delay for each CIS will equal CIG_Sync_Delay minus the offset from the CIG reference point to the CIS anchor point. The actual maximum possible time for a CIG event cannot be determined until all the CISes in the CIG have been created. Therefore the value that the Central sends is an upper bound.

Note: The maximum possible time for a CIS event equals (NSE − 1) × Sub_Interval + MPT_C + T_IFS_CIS + MPT_P for that CIS.

Note: The CIS events making up a CIG event need not have the same values of *cisEventCounter*, but the difference between the counters will be the same for the lifetime of the CIG.

#### 4.5.14.2 Arrangement of multiple CISes

The CISes in a CIG shall be arranged either sequentially or interleaved by setting the values of the Sub_Interval and the spacing between the CIS anchor points appropriately.

If they are sequential, the CIS events of the different CISes do not overlap and so all the subevents of a CIS event occur together. For each adjacent pair of CISes, the interval between their CIS anchor points shall be at least NSE × Sub_Interval, using the values for the lower numbered CIS.

If they are interleaved, all the first subevents of the CISes are adjacent, followed by the second subevents, and so on. For each CIS, its value of Sub_Interval shall be at least the sum of the values of SE_Length for all the CISes in the CIG. For each adjacent pair of CISes, the interval between their CIS anchor points shall be at least SE_Length of the lower numbered CIS.

Figure 4.67 shows each arrangement for a CIG with two CISes and NSE = 2.

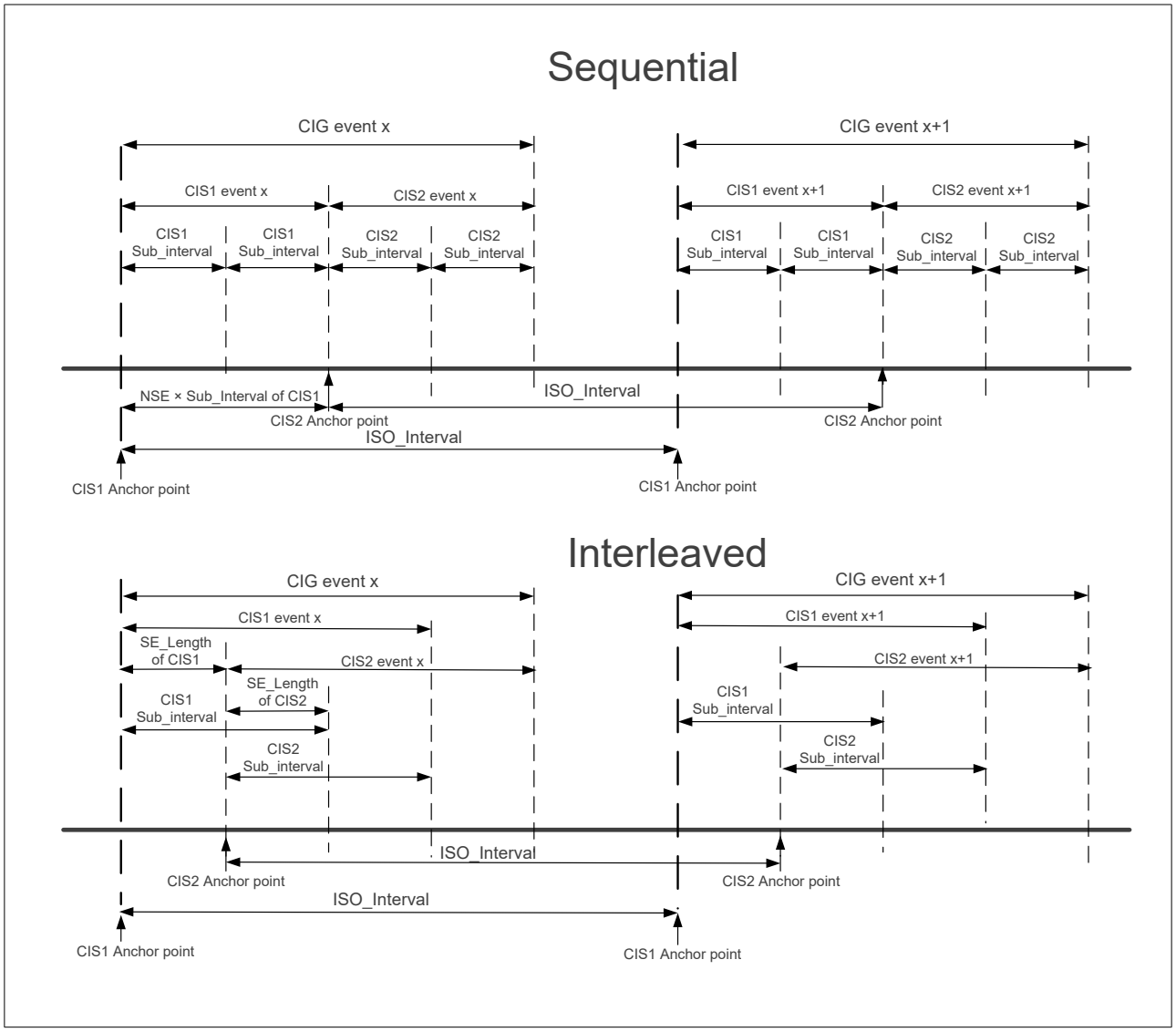



*Figure 4.67: CIG event with events for two CISes in sequential and interleaved arrangement*

### 4.5.14.3 States of a CIG

On the Central, a CIG has three states: configurable, active, and inactive. When the Host creates the CIG, it shall be in the configurable state. When the Host creates a CIS in the CIG, if the CIG is not already in the active state, then the Controller shall change the state of the CIG to active. When all CISes in a CIG are terminated or considered lost, the Controller shall change the state of the CIG to inactive. If the Host sends an explicit request to remove the CIG and the CIG is not in the active state, then the Controller shall remove the CIG.

The Host configures the CIG by notifying the Link Layer of the individual configurations of CIS(es) to be stored within the CIG. The Link Layer shall only change the configuration when requested by the Host and while the CIG is in the configurable state.

If the Link Layer is unable to schedule a CIG of the requested configuration, it shall notify the Host and not create any of the CISes in the CIG.

Note: The Controller can use the configuration information to determine whether it is able to schedule all the CISes in the CIG without conflict with each other or other activities. If it is unable to do so, it can notify the Host when requested to create the first CIS. Once it has successfully created a schedule, subsequent CISes can then be created without any risk of conflict. Figure 4.68 illustrates the states of a CIG and the CIS configurations stored within it.

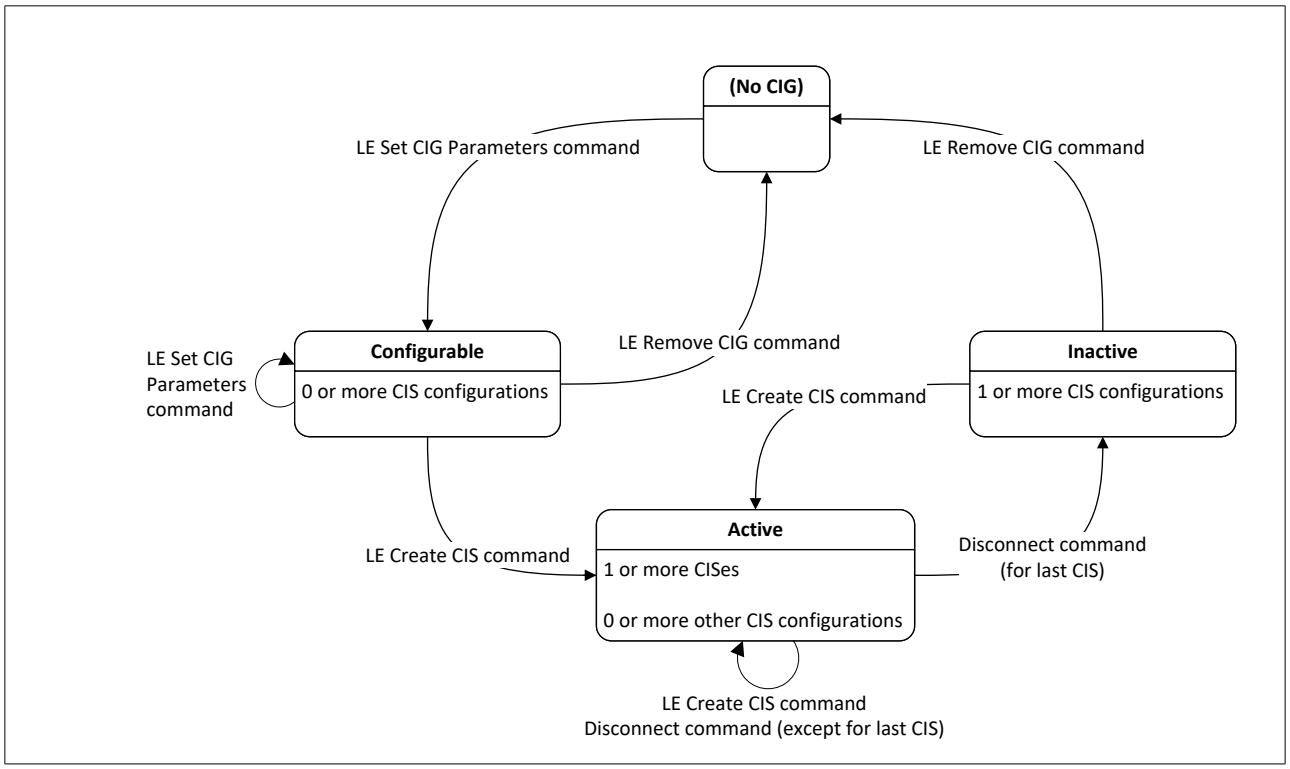



*Figure 4.68: States of a CIG and its CIS configurations*

### 4.5.15 Power level management

A Link Layer that supports power control shall manage power levels on all active PHYs for a given peer and may manage power levels for some or all non-active PHYs that it supports. An active PHY for a peer is the current transmit PHY for the ACL connection to that peer or the transmit PHY for an associated CIS. This implies that, when a CIS is created on a new PHY or the ACL connection changes to a new transmit PHY, the Link Layer must start managing that PHY if it doesn't already do so.

When a device is managing the power level for a PHY, it shall store the power level for that PHY and shall transmit all packets to the peer on that PHY using that power level. The device shall change the power level when requested by the peer and may change it autonomously.

If a device starts to manage the power level for a PHY (e.g. because it has become an active PHY) then the implementation shall choose an initial power level.

Where a PHY has more than one power level, then the different power levels shall be treated as separate PHYs for the purpose of this section, the Power Control Request procedure (see Section 5.1.17), and the Power Change Indication procedure (see Section 5.1.18). In the case of the LE Coded PHY, the two power levels are "packets transmitted with an S=8 payload" and "packets transmitted with an S=2 payload"; the power level shall not change during a packet. Nevertheless, both shall be active or inactive at the same time.

### 4.5.16 Path loss monitoring

The Host may request the Controller to perform path loss monitoring on an ACL connection. There shall be two path loss zones, called the Low and Middle zones, and optionally a third High zone. The Controller shall notify the Host whenever the path loss changes from one zone to another. The path loss is defined as the difference between the remote transmit power level and the average local RSSI measurement for the connection; how measurements are averaged is not specified. The path loss shall be deemed to have entered a new zone when it becomes greater than or equal to the upwards boundary when moving to a higher zone or becomes less than or equal to the downwards boundary when moving to a lower zone, as shown in Figure 4.69, and, in each case, has spent at least the minimum time specified in the new zone if one is specified. For each pair of zones, the upwards boundary shall be greater than or equal to the downwards boundary and should not be equal so as to provide some hysteresis.

The zone boundaries in each direction and, optionally, the minimum time to spend in a new zone are specified by the Host.

The Controller may notify the Host when the path loss becomes unavailable. If so, it shall notify the Host when the path loss becomes available again as if it had just changed zones.

Two consecutive reports shall not indicate entering the same zone.

Notes:

- Path loss can be measured by making use of the Power Control Request procedure or the Power Change Indication procedure to obtain the remote transmit power level, and by making local RSSI measurements.
- Path loss is often correlated with distance from the peer device and, therefore, moving to a higher zone might indicate that the peer has moved further away.
- A Controller may use path loss measurements from associated physical links, such as CISes, when determining path loss threshold events on the ACL connection.

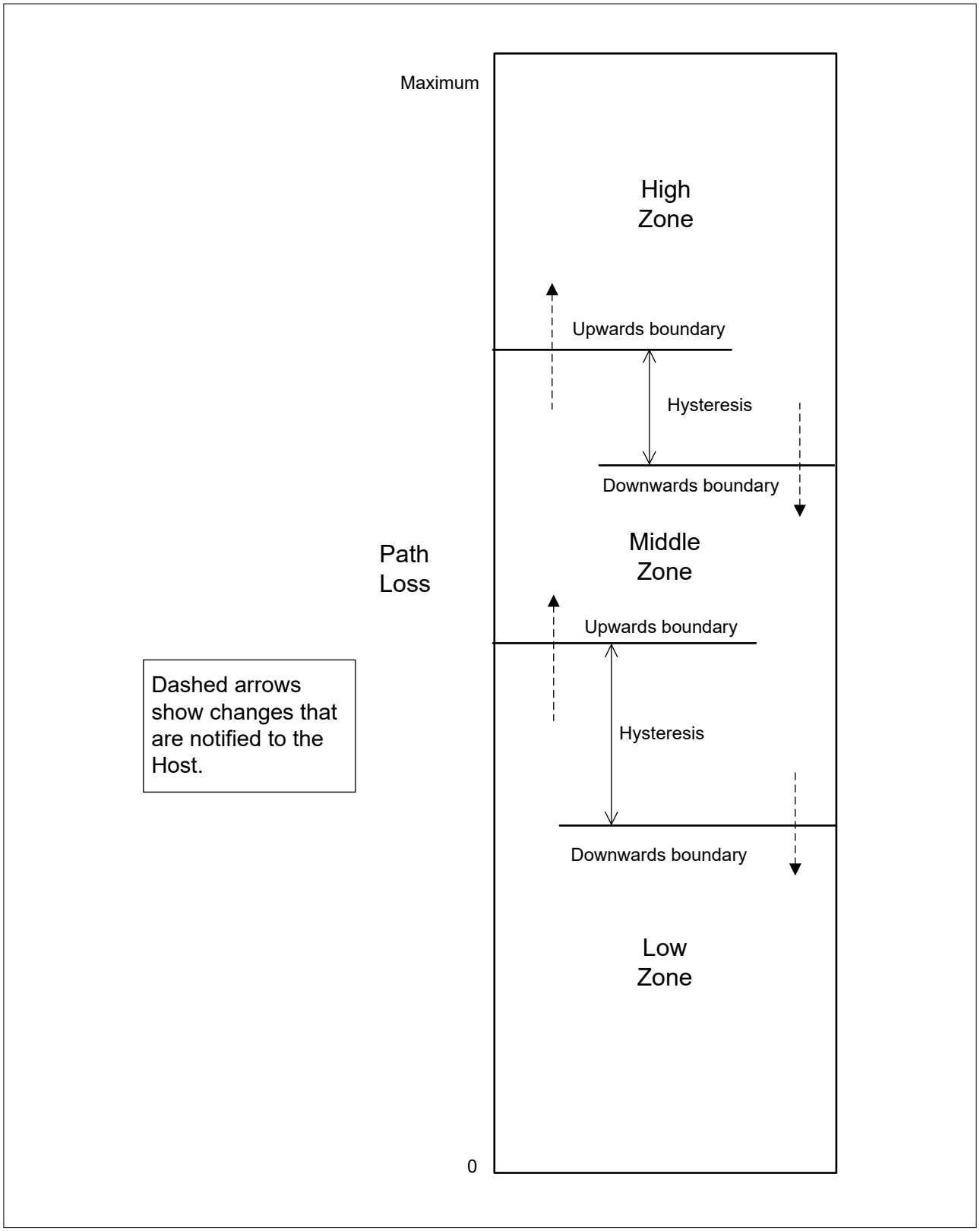



*Figure 4.69: Path loss notification zones*

### 4.5.17 ACL data Host transport

For each logical link, the Controller shall transmit data over the air in the same order that it is received from the Host. The boundaries between packets over the air for a specific L2CAP PDU may be different from the boundaries in the data provided by the Host. Each new L2CAP PDU shall start a new packet over the air. (See [Vol 3] Part A, Section 7.2.1 for related requirements in L2CAP.)

For each logical link, the Controller shall transmit the data received over the air to the Host (whether over HCI or otherwise) in the same order that it was received. The boundaries in the data sent to the Host for a specific L2CAP PDU may be different from the boundaries between packets received over the air. The Controller shall retain boundaries between L2CAP PDUs. Data from different logical links may be interleaved. (See [Vol 3] Part A, Section 7.2.2 for related requirements in L2CAP.)

#### 4.5.17.1 Flushing of ACL data from Host to Controller

The Host may mark the start fragments of L2CAP PDUs as flushable data. In HCI, flushable data is identified via the Packet_Boundary_Flag (see [Vol 4] Part E, Section 5.4.2). The ACL Flush Timeout value can be set by HCI (see [Vol 4] Part E, Section 6.19). Flushing applies to the entire L2CAP PDU, not to the individual fragments sent over HCI or over the air. The default value of the ACL Flush Timeout is infinite, in which case all L2CAP PDUs are transmitted by the Link Layer over the air via the ACL logical link. If the ACL Flush Timeout value for the logical link is set to a non-infinite value, then the Link Layer shall start a new ACL flush timer for each new L2CAP PDU received from the Host that is marked as flushable data and set it to the current value of ACL Flush Timeout. The Link Layer shall cancel the ACL flush timer when any fragment of the L2CAP PDU is first transmitted over the air; if the timer is cancelled, then the Link Layer shall not flush any fragment of the L2CAP PDU. If the timer expires for a given L2CAP PDU before the Link Layer transmits any fragment of the L2CAP PDU over the air, then the Link Layer shall flush all fragments of the L2CAP PDU.

### 4.5.18 Channel Sounding

A CS procedure is subdivided into one or more CS events which each contain one or more CS subevents, which in turn may contain two or more CS steps. The duration of a CS subevent is selected during the CS Start procedure as described in Section 5.1.26. The maximum duration of a CS event and subevent may extend to the processing time for the entire CS procedure and is negotiated during the CS Configuration procedure.

Within an LE connection, each device shall maintain a unique instance of a 16-bit CS procedure counter (*CSProcCount*). The usage of *CSProcCount* is described in Section 5.1.26.

#### 4.5.18.1 Channel Sounding procedures and subevents

During the CS Start procedure described in Section 5.1.26, CS procedures may be set up to run one or more procedure instances in a sequential manner. The selection of the LE connection event anchor point used for the scheduling of the first CS procedure is also described in Section 5.1.26, as is the selection of the following parameters which are used in the scheduling of additional CS procedure instances:

- T_PROCEDURE_INTERVAL – the number of LE connection intervals between the LE connection event anchor points from which subsequent CS procedure instances are offset.
- N_PROCEDURE_COUNT – the number of consecutive CS procedures to execute.

Figure 4.70 shows an example with T_PROCEDURE_INTERVAL = 4 and N_PROCEDURE_COUNT = 4. In Figure 4.70, the length shown for each CS procedure is not intended to match the entire span of T_PROCEDURE_INTERVAL.

A CS procedure or CS procedure repeat instance is in progress from the beginning of the associated CS Start Procedure as described in Section 5.1.26. A CS procedure or CS procedure repeat instance is complete when either the associated CS Start procedure completes with an error, or the last procedure associated with the CS Start procedure is complete, or the CS Procedure Repeat Termination procedure issued on this series of CS procedure repeat instances has completed, as described in Section 5.1.27.

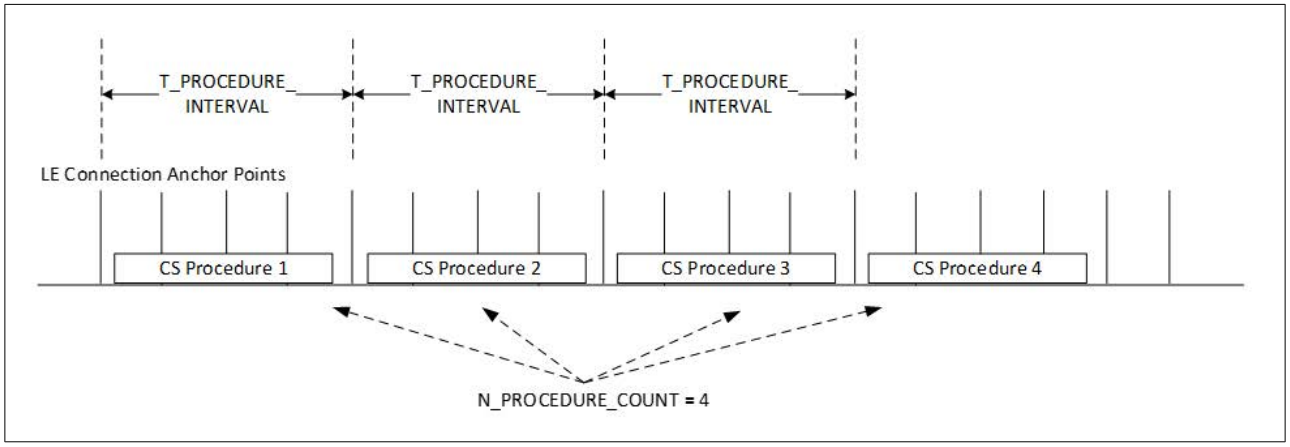



*Figure 4.70: CS procedure reoccurrence with N_PROCEDURE_COUNT = 4*

Within a CS procedure, there shall be 1 to N_MAX_SUBEVENTS_PER_PROCEDURE CS subevents, where N_MAX_SUBEVENTS_PER_PROCEDURE is equal to 32. CS subevents are composed of a structured sequence of CS steps. CS steps are described in [Vol 6] Part H, Section 4.3. The minimum number of CS steps in a CS subevent shall be N_MIN_STEPS_PER_SUBEVENT, where N_MIN_STEPS_PER_SUBEVENT is equal to 2. The maximum number of CS steps within any CS subevent shall be

N_MAX_STEPS_PER_SUBEVENT, where N_MAX_STEPS_PER_SUBEVENT is equal to 160. The maximum number of CS steps in a CS procedure, N_STEPS_MAX, shall be 256. The first CS step for each CS subevent shall be a mode-0 step. Multiple mode-0 steps may be selected to begin each CS subevent within a CS procedure.

The following CS event and subevent parameters are exchanged during the CS configuration and start procedures (see Section 5.1.25 and Section 5.1.26).

- T_SUBEVENT_LEN – the maximum duration of a CS subevent, in microseconds.
- T_EVENT_OFFSET – the time in microseconds between the LE connection event anchor point and the beginning of the CS event.
- T_EVENT_INTERVAL – the time in units of LE connection intervals between the LE connection event anchor points from where the start of two consecutive CS events are offset.
- T_SUBEVENT_INTERVAL – the time in units of 625 microseconds between the beginning of a CS subevent and the beginning of the next CS subevent within the same CS event.
- N_SUBEVENTS_PER_EVENT – the number of CS subevents in a CS event.

CS events are scheduled at regular intervals based on the timing of the LE connection event anchor points of the underlying LE connection, as shown in Figure 4.71. CS events are scheduled with an offset from an ACL anchor point. This offset is specified in microseconds. The extent of a CS event may exceed that of the underlying LE connection interval. In Figure 4.71, connInterval is the LE connection interval as defined in Section 4.5.1.

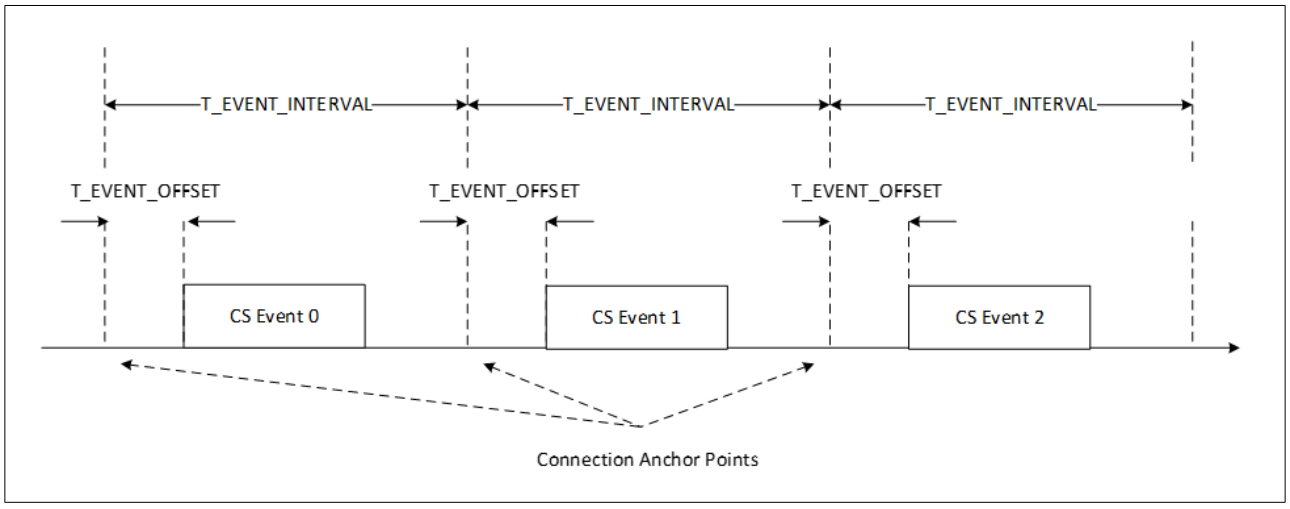



*Figure 4.71: CS event scheduling offset from ACL anchor points*

To accommodate various coexistence scenarios, a Controller may schedule more than one CS subevent within a CS event, as shown in Figure 4.72. The first

CS subevent within a CS event shall always occur T_EVENT_OFFSET from the connection event anchor point. Subsequent CS subevents within that CS event shall occur T_SUBEVENT_INTERVAL from the start of the prior CS subevent. The value of T_SUBEVENT_INTERVAL shall be greater than the selected duration of CS subevents, T_SUBEVENT_LEN, plus a minimum subevent spacing value of T_MES (see Section 4.1.4). The extent of a CS subevent may exceed that of the underlying LE connection interval.

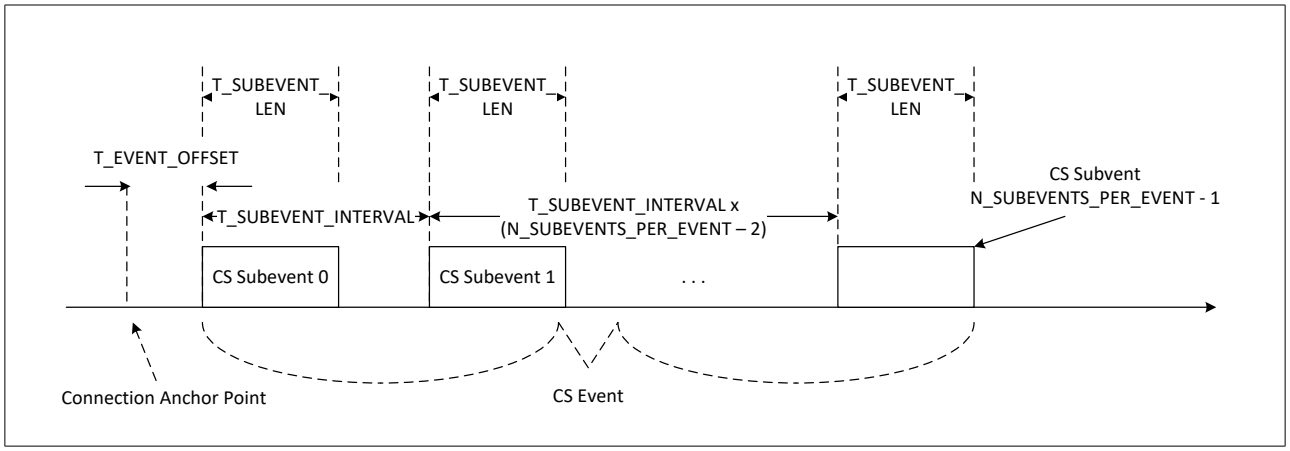



*Figure 4.72: Multiple CS subevents of one CS event scheduled from the same LE connection event anchor point*

Within a CS subevent, the duration of each CS step will vary based on the mode selected for that step, the duration of the CS tone selected if mode-2 or mode-3 is being used, the type of the selected RTT exchange (with or without the optional sounding sequence or random sequence) if mode-1 or mode-3 is being used, or other selectable factors. As a result, the number of CS steps included in each CS subevent may also vary. After a CS step is processed within a CS subevent, there may be a remaining duration in that subevent. A Link Layer shall not begin the next CS step within a CS subevent unless the remaining duration within that subevent is greater than or equal to the time needed to fully include that CS step.

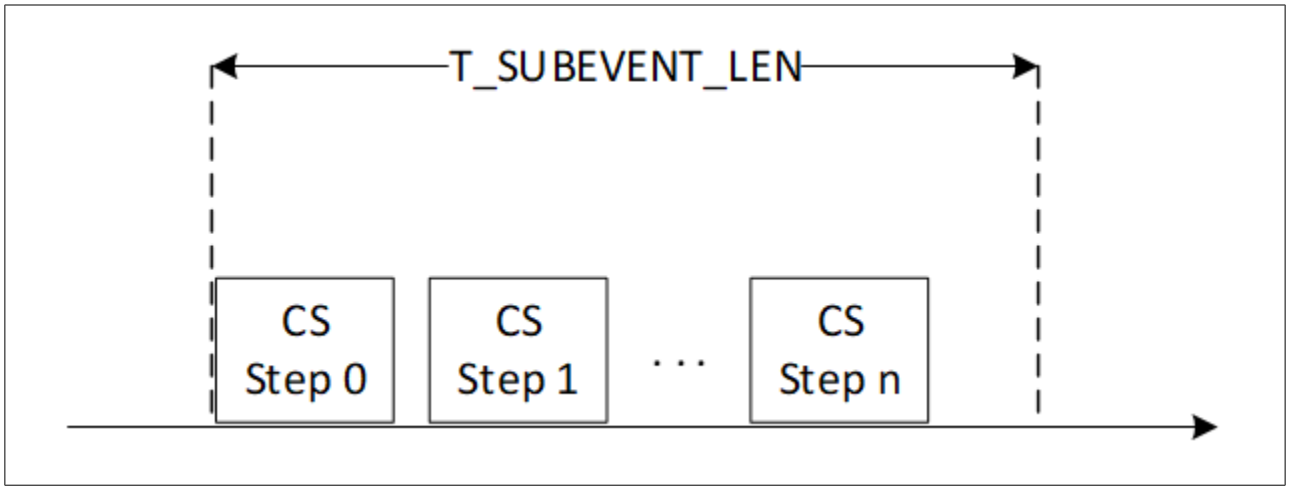



*Figure 4.73: CS subevent maximum extent*

A Controller that supports CS should be capable of reporting all result data generated from that procedure. When starting a CS procedure as described in Section 5.1.26, a Controller shall report an error to the Host if the Controller determines that it has insufficient resources to collect and report all procedure results for that procedure.

If the Controller has insufficient resources to hold result information for a specific CS subevent, the Controller shall discard the result information for that subevent and report this action to the Host.

In addition, at the start of any subsequent CS procedure instance a Controller shall discard any remaining CS procedure result data that it may be holding for a previous CS procedure instance, under the following circumstances:

- The Controller has been instructed to run multiple CS procedure instances, and
- The Controller has already discarded CS subevent result information for the previous CS procedure instance.

The remaining CS procedure result data is discarded so that enough resources are available for the newly started procedure instance. The Controller shall also report this action to the Host.

#### 4.5.18.2 Channel Sounding security

The ACL shall be encrypted before the CS feature is used, as well as before any of the following LL control PDU exchanges:

- LL_CS_SEC_REQ
- LL_CS_SEC_RSP
- LL_CS_CAPABILITIES_REQ

- LL_CS_CAPABILITIES_RSP
- LL_CS_CONFIG_REQ
- LL_CS_CONFIG_RSP
- LL_CS_REQ
- LL_CS_RSP
- LL_CS_IND
- LL_CS_TERMINATE_REQ
- LL_CS_TERMINATE_RSP
- LL_CS_FAE_REQ
- LL_CS_FAE_RSP
- LL_CS_CHANNEL_MAP_IND

## 4.6 Feature support

The set of features supported by a Link Layer is represented by a bit mask called FeatureSet. This mask consists of 1984 bits divided into page 0 containing bits 0 to 63 (octets 0 to 7) and 10 pages of 192 bits (24 octets) each, numbered starting from 1 (i.e., page 1 is octets 8 to 31, page 2 is octets 32 to 55, etc.).

The value of FeatureSet shall not change while the Controller has a connection to another device. A peer device may cache information about features that the device supports. The Link Layer may cache information about features that a peer supports during a connection.

Within FeatureSet, a bit set to 0 indicates that the Link Layer feature is not supported in this Controller; a bit set to 1 indicates that the Link Layer feature is supported in this Controller.

Except where explicitly stated elsewhere in this specification, if the peer Link Layer has indicated either during a feature exchange procedure or by responding with an LL_UNKNOWN_RSP PDU that it does not support a procedure, then the Link Layer shall not initiate that procedure. A Link Layer shall not transmit a PDU listed in the following subsections unless it supports at least one of the features that requires support for that PDU.

Unless explicitly stated otherwise, when a Link Layer supports a feature it shall support it on all PHYs that the Controller supports.

The bit positions for each Link Layer feature shall be as shown in Table 4.9, which also shows various properties of the feature bits. In the "Send to Peer" column:

- "Y" indicates that the bit shall be set correctly when sent to the peer.
- "O" indicates that the bit shall either be zero or set correctly when sent to the peer. The peer device shall ignore the value of this bit.
- "N" indicates that the bit shall be set to zero when sent to the peer.

When sent to the Host, the bit shall always be set correctly. If a bit is shown as Host Controlled, the value may be set by the Host and shall default to zero.

If a bit does not have "Y" for "Send to Peer", it shall not be used to determine whether a peer device supports any associated procedure.

| Bit position | Link Layer Feature | Send to Peer | Host Controlled |
|---|---|---|---|
| 0 | LE Encryption | Y | N |
| 1 | Connection Parameters Request procedure | Y | N |
| 2 | Extended Reject Indication | Y | N |
| 3 | Peripheral-initiated Features Exchange | Y | N |
| 4 | LE Ping | O | N |
| 5 | LE Data Packet Length Extension | Y | N |
| 6 | LL Privacy | O | N |
| 7 | Extended Scanning Filter Policies | O | N |
| 8 | LE 2M PHY | Y | N |
| 9 | Stable Modulation Index - Transmitter | Y | N |
| 10 | Stable Modulation Index - Receiver | Y | N |
| 11 | LE Coded PHY | Y | N |
| 12 | LE Extended Advertising | O | N |
| 13 | LE Periodic Advertising | O | N |
| 14 | Channel Selection Algorithm #2 | Y | N |
| 15 | LE Power Class 1 (see [Vol 6] Part A, Section 3) | Y | N |
| 16 | Minimum Number of Used Channels procedure | Y | N |
| 17 | Connection CTE Request | Y | N |
| 18 | Connection CTE Response | Y | N |
| 19 | Connectionless CTE Transmitter | O | N |
| 20 | Connectionless CTE Receiver | O | N |

| Bit position | Link Layer Feature | Send to Peer | Host Controlled |
|---|---|---|---|
| 21 | Antenna Switching During CTE Transmission (AoD) | O | N |
| 22 | Antenna Switching During CTE Reception (AoA) | O | N |
| 23 | Receiving Constant Tone Extensions | Y | N |
| 24 | Periodic Advertising Sync Transfer - Sender | Y | N |
| 25 | Periodic Advertising Sync Transfer - Recipient | Y | N |
| 26 | Sleep Clock Accuracy Updates | Y | N |
| 27 | Remote Public Key Validation | N | N |
| 28 | Connected Isochronous Stream – Central | Y | N |
| 29 | Connected Isochronous Stream – Peripheral | Y | N |
| 30 | Isochronous Broadcaster | Y | N |
| 31 | Synchronized Receiver | Y | N |
| 32 | Connected Isochronous Stream (Host Support) | Y | Y |
| 33 | LE Power Control Request | Y | N |
| 34 | LE Power Control Request | Y | N |
| 35 | LE Path Loss Monitoring | Y | N |
| 36 | Periodic Advertising ADI support | O | N |
| 37 | Connection Subrating | Y | N |
| 38 | Connection Subrating (Host Support) | Y | Y |
| 39 | Channel Classification | Y | N |
| 40 | Advertising Coding Selection | Y | N |
| 41 | Advertising Coding Selection (Host Support) | Y | Y |
| 42 | Decision-Based Advertising Filtering | N | N |
| 43 | Periodic Advertising with Responses - Advertiser | Y | N |
| 44 | Periodic Advertising with Responses - Scanner | Y | N |
| 45 | Unsegmented Framed Mode | Y | N |
| 46 | Channel Sounding | Y | N |
| 47 | Channel Sounding (Host Support) | Y | Y |
| 48 | Channel Sounding Tone Quality Indication | N | N |
| 56 to 62 | Reserved for specification development purposes | | |
| 63 | LL Extended Feature Set | Y | N |
| 64 | Monitoring Advertisers | N | N |
| 65 | Frame Space Update | Y | N |

| Bit position | Link Layer Feature | Send to Peer | Host Controlled |
|---|---|---|---|
| 66 | UTP OTA mode | Y | N |
| 67 | UTP HCI mode | N | N |
| 68 and 69 | LL_OTA_UTP_IND maximum length | Y | N |
| 72 | Shorter Connection Intervals | Y | N |
| 73 | Shorter Connection Intervals (Host Support) | Y | Y |
| 74 | LE Flushable ACL Data | N | N |
| 75 | Channel Sounding Enhancement #1 | Y | N |
| 1952 to 1983 | Reserved for specification development purposes | | |
| All other bits | Reserved for future use | | |

*Table 4.9: FeatureSet field's bit mapping to Controller features*

Bits 33 and 34 shall always have the same value.

If the Link Layer supports any feature with bit number 64 or greater, then it shall also support the LL Extended Feature Set feature.

### 4.6.1 LE Encryption

A Controller that supports LE Encryption shall support encryption on all logical transports that it supports and the following sections of this document:

- LL_ENC_REQ (Section 2.4.2.4)
- LL_ENC_RSP (Section 2.4.2.5)
- LL_START_ENC_REQ (Section 2.4.2.6)
- LL_START_ENC_RSP (Section 2.4.2.7)
- LL_PAUSE_ENC_REQ (Section 2.4.2.11)
- LL_PAUSE_ENC_RSP (Section 2.4.2.12)
- Encryption Start procedure (Section 5.1.3.1)
- Encryption Pause procedure (Section 5.1.3.2)

### 4.6.2 Connection Parameters Request procedure

A Controller that supports Connection Parameters Request procedure shall support the Extended Reject Indication feature and the following sections of this document:

- LL_CONNECTION_PARAM_REQ (Section 2.4.2.16)
- LL_CONNECTION_PARAM_RSP (Section 2.4.2.17)

- Connection Parameters Request procedure (Section 5.1.7)

### 4.6.3 Extended Reject Indication

A Controller that supports Extended Reject Indication shall support the following sections of this document:

- LL_REJECT_EXT_IND (Section 2.4.2.18)

### 4.6.4 Peripheral-initiated Features Exchange

A Controller that supports Peripheral-initiated Features Exchange shall support the following sections of this document:

- LL_PERIPHERAL_FEATURE_REQ (Section 2.4.2.15)
- Receiving LL_FEATURE_RSP (see Section 2.4.2.10)

### 4.6.5 LE Ping

A Controller that supports LE Ping shall support the following sections of this document:

- LL_PING_REQ (Section 2.4.2.19)
- LL_PING_RSP (Section 2.4.2.20)
- LE Ping procedure (Section 5.1.8)
- LE Authenticated Payload Timeout (Section 5.4)

### 4.6.6 LE Data Packet Length Extension

A Controller that supports LE Data Packet Length Extension shall support the following sections of this document:

- LL_LENGTH_REQ and LL_LENGTH_RSP (Section 2.4.2.21)
- Data Length Update procedure (Section 5.1.9)

### 4.6.7 LL Privacy

A Controller that supports LL Privacy shall support the following sections of this document:

- Resolving List (Section 4.7)
- LL Privacy (Section 6)

### 4.6.8 Extended Scanning Filter Policies

A Controller that supports Extended Scanning Filter Policies shall support the following sections of this document:

- Extended Scanning Filter Policies (Section 4.3.3.1)

### 4.6.9 Multiple PHYs

A Controller that supports any PHY other than LE 1M PHY shall support the Extended Reject Indication feature and the following sections of this document:

- Transmission and reception using the supported modulation schemes ([Vol 6] Part A, Section 1)
- Longer preamble when supporting the LE 2M PHY (see Section 2.2.1)
- LL_PHY_REQ (Section 2.4.2.22)
- LL_PHY_RSP (Section 2.4.2.22)
- LL_PHY_UPDATE_IND (Section 2.4.2.23)
- PHY Update procedure (Section 5.1.10)

and, when supporting the LE Coded PHY:

- Packet format for the LE Coded PHY (Section 2.2)
- Coding (Section 3.3)

#### 4.6.9.1 Symmetric and asymmetric connections

A Controller shall support connections using the same PHY in each direction (“symmetric connections”) and may support connections using different PHYs in each direction (“asymmetric connections”).

If a Controller cannot support asymmetric connections then:

- Any LL_PHY_REQ, LL_PHY_RSP, or LL_CIS_REQ PDUs sent shall indicate a symmetric connection.
- Any LL_PHY_UPDATE_IND PDU sent shall not specify an asymmetric connection.

### 4.6.10 Stable Modulation Index - Transmitter

A Controller that supports Stable Modulation Index - Transmitter shall support the following section of this document:

- Stable Modulation Index ([Vol 6] Part A, Section 3.1.1)

### 4.6.11 Stable Modulation Index - Receiver

A Controller that supports Stable Modulation Index - Receiver shall support the following section of this document:

- Stable Modulation Index ([Vol 6] Part A, Section 4.7)

### 4.6.12 LE Extended Advertising

A Controller that supports LE Extended Advertising shall support reception of an Advertising Physical Channel PDU Payload field of 255 octets, shall support the following sections of this document:

- ADV_EXT_IND (Section 2.3.1.5)
- AUX_ADV_IND (Section 2.3.1.6)
- AUX_CHAIN_IND (Section 2.3.1.8)
- AUX_SCAN_REQ (see Section 2.3.2.1)
- AUX_SCAN_RSP (Section 2.3.2.3)
- AUX_CONNECT_REQ (see Section 2.3.3.1)
- AUX_CONNECT_RSP (Section 2.3.3.2)
- Common Extended Advertising Payload Format (Section 2.3.4)
- Advertising Sets (Section 4.4.2.10)
- Using AdvDataInfo (ADI) (Section 4.4.2.11)
- Advertising Sets (Section 4.4.3.3)
- Connect Requests on the Secondary Advertising Physical Channel (Section 4.4.4.2)

and shall support the following sections of this document in accordance with the requirements in Section 4.4.2.13, Section 4.4.3.7, and Section 4.4.4.3:

- Connectable Directed event type using ADV_EXT_IND (Section 4.4.2.4.4)
- Scannable Undirected event type using ADV_EXT_IND (Section 4.4.2.5.2)
- Connectable Undirected event type (Section 4.4.2.7)
- Scannable Directed event type (Section 4.4.2.8)
- Non-Connectable and Non-Scannable Directed event type (Section 4.4.2.9)

A Controller that supports connections shall also support the Channel Selection Algorithm #2 feature.

### 4.6.13 LE Periodic Advertising

A Controller that supports LE Periodic Advertising shall support the LE Extended Advertising feature, Channel Selection Algorithm #2 feature, and the following sections of this document:

- AUX_SYNC_IND (Section 2.3.1.7)
- Periodic Advertising (Section 4.4.2.12)

A Controller that supports Scanning state shall also support the following sections of this document:

- Scanning for periodic advertisements (Section 4.4.3.4)
- Synchronization state (Section 4.4.5) for periodic advertising trains

### 4.6.14 Channel Selection Algorithm #2

A Controller that supports Channel Selection Algorithm #2 shall support the following sections of this document:

- ChSel bit set to 1 (see Section 2.3, Section 2.3.1.1, Section 2.3.1.2, and Section 2.3.3.1)
- Channel Selection Algorithm #2 (Section 4.5.8.3).

### 4.6.15 Minimum Number of Used Channels procedure

A Controller that supports the Minimum Number of Used Channels procedure shall support the following sections of this document:

- LL_MIN_USED_CHANNELS_IND (Section 2.4.2.24)
- Minimum Number Of Used Channels procedure (Section 5.1.11)

### 4.6.16 Connection CTE Request

A Controller that supports Connection CTE Request shall support the Receiving Constant Tone Extensions feature, the Extended Reject Indication feature, and the following sections of this document on all supported PHYs that allow Constant Tone Extensions:

- LL_CTE_REQ (Section 2.4.2.25)
- LL_CTE_RSP (Section 2.4.2.26)
- Constant Tone Extension Request procedure (Section 5.1.12) - as initiator

### 4.6.17 Connection CTE Response

A Controller that supports Connection CTE Response shall support the Extended Reject Indication feature and the following sections of this document on all supported PHYs that allow Constant Tone Extensions:

- LL_CTE_REQ (Section 2.4.2.25)
- LL_CTE_RSP (Section 2.4.2.26)
- Transmitting Constant Tone Extensions (Section 2.5.3)
- Constant Tone Extension Request procedure (Section 5.1.12) - as responder

### 4.6.18 Connectionless CTE Transmitter

A Controller that supports Connectionless CTE Transmitter shall support the LE Periodic Advertising feature in Advertising state and the following section of this document on all supported PHYs that allow Constant Tone Extensions:

- Transmitting Constant Tone Extensions (Section 2.5.3)

### 4.6.19 Connectionless CTE Receiver

A Controller that supports Connectionless CTE Receiver shall support the LE Periodic Advertising feature in Synchronization state and the following sections of this document on all supported PHYs that allow Constant Tone Extensions:

- Receiving Advertising Physical Channel PDUs containing a CTEInfo field in the Extended Header field (see Section 2.3.4)
- IQ Sampling (Section 2.5.4)

### 4.6.20 Antenna Switching During CTE Transmission (AoD)

A Controller that supports Antenna Switching During CTE Transmission (AoD) shall support the following sections of this document on all supported PHYs that allow Constant Tone Extensions:

- Transmitting Constant Tone Extensions (Section 2.5.3)
- Antenna Switching ([Vol 6] Part A, Section 5)

### 4.6.21 Antenna Switching During CTE Reception (AoA)

A Controller that supports Antenna Switching During CTE Reception (AoA) shall support the Receiving Constant Tone Extensions feature and the following section of this document on all supported PHYs that allow Constant Tone Extensions:

- Antenna Switching ([Vol 6] Part A, Section 5)

### 4.6.22 Receiving Constant Tone Extensions

A Controller that supports Receiving Constant Tone Extensions shall support the following sections of this document on all supported PHYs that allow Constant Tone Extensions:

- Receiving Data Channel PDUs with the CP bit set to 1 and containing a CTEInfo field (see Section 2.4)
- IQ Sampling (Section 2.5.4)

### 4.6.23 Periodic Advertising Sync Transfer - Sender

A Controller that supports Periodic Advertising Sync Transfer - Sender shall support the LE Periodic Advertising feature and the following sections of this document:

- LL_PERIODIC_SYNC_IND (Section 2.4.2.27)
- Periodic Advertising Sync Transfer procedure (Section 5.1.13) - as initiator

### 4.6.24 Periodic Advertising Sync Transfer - Recipient

A Controller that supports Periodic Advertising Sync Transfer - Recipient shall support the LE Periodic Advertising feature and the following sections of this document:

- LL_PERIODIC_SYNC_IND (Section 2.4.2.27)
- Periodic Advertising Sync Transfer procedure (Section 5.1.13) - as recipient

### 4.6.25 Sleep Clock Accuracy Updates

A Controller that supports Sleep Clock Accuracy Updates shall support the following sections of this document:

- LL_CLOCK_ACCURACY_REQ and LL_CLOCK_ACCURACY_RSP (Section 2.4.2.28)
- Sleep Clock Accuracy Update procedure (Section 5.1.14)

### 4.6.26 Remote Public Key Validation

A Controller that supports Remote Public Key Validation shall validate the remote public key (see [Vol 3] Part H, Section 2.3.5.6.1) sent by the Host (e.g., in the HCI_LE_Generate_DHKey command; see [Vol 4] Part E, Section 7.8.37).

### 4.6.27 Connected Isochronous Stream - Central and Connected Isochronous Stream - Peripheral

A Controller that supports the Connected Isochronous Stream - Central feature or the Connected Isochronous Stream - Peripheral feature shall support the Channel Selection Algorithm #2 feature, the Sleep Clock Accuracy Updates feature, the Extended Reject Indication feature, and the following sections of this document:

- LL_CIS_REQ (Section 2.4.2.29)
- LL_CIS_RSP (Section 2.4.2.30)
- LL_CIS_IND (Section 2.4.2.31)
- LL_CIS_TERMINATE_IND (Section 2.4.2.32)
- Connected Isochronous PDU (Section 2.6.1)
- Connected Isochronous Stream (Section 4.5.13)
- Connected Isochronous Group (Section 4.5.14)
- Connected Isochronous Stream Creation procedure (Section 5.1.15)
- Connected Isochronous Stream Termination procedure (Section 5.1.16)
- ISO Transmit Test Mode (Section 7.1)
- ISO Receive Test Mode (Section 7.2)
- Isochronous Adaptation Layer (ISOAL) ([Vol 6] Part G)

### 4.6.28 Isochronous Broadcaster

A Controller that supports the Isochronous Broadcaster feature shall support the LE Periodic Advertising feature and the following sections of this document:

- Broadcast Isochronous PDU (Section 2.6.2)
- BIG Control PDU (Section 2.6.3)
- Isochronous Broadcasting State (Section 4.4.6)
- BIG control procedures (Section 5.6)
- ISO Transmit Test Mode (Section 7.1)
- Isochronous Adaptation Layer (ISOAL) ([Vol 6] Part G)

### 4.6.29 Synchronized Receiver

A Controller that supports the Synchronized Receiver feature shall support the LE Periodic Advertising feature and the following sections of this document:

- Broadcast Isochronous PDU (Section 2.6.2)
- BIG Control PDU (Section 2.6.3)
- Synchronization state (see Section 4.4.5)
- BIG control procedures (Section 5.6)
- ISO Receive Test Mode (Section 7.2)
- Isochronous Adaptation Layer (ISOAL) ([Vol 6] Part G)

### 4.6.30 [This section is no longer used]

### 4.6.31 LE Power Control Request

A Controller that supports LE Power Control Request shall support the Extended Reject Indication feature and the following sections of this document:

- LL_POWER_CONTROL_REQ (Section 2.4.2.33)
- LL_POWER_CONTROL_RSP (Section 2.4.2.34)
- LL_POWER_CHANGE_IND (Section 2.4.2.35)
- Power level management (Section 4.5.15)
- Power Control Request procedure (Section 5.1.17)
- Power Change Indication procedure (Section 5.1.18)

### 4.6.32 LE Path Loss Monitoring

A Controller that supports LE Path Loss Monitoring shall support the following sections of this document:

- LE Path Loss Monitoring (Section 4.5.16)

### 4.6.33 Host-set feature bits

The Controller shall only set these bits on request from the Host. The Controller shall reject a request to set a bit if the conditions in this section are not met.

#### 4.6.33.1 Connected Isochronous Stream (Host Support)

This feature bit indicates that the Host supports creating CISes.

The Controller shall only set this feature bit if it supports the Connected Isochronous Stream - Central or Connected Isochronous Stream - Peripheral feature.

#### 4.6.33.2 Connection Subrating (Host Support)

This feature bit indicates that the Host supports Connection Subrating.

The Controller shall only set this feature bit if it supports the LE Connection Subrating feature.

#### 4.6.33.3 Advertising Coding Selection (Host Support)

This feature bit indicates that the Host supports Advertising Coding Selection.

The Controller shall only set this feature bit if it supports the Advertising Coding Selection feature.

#### 4.6.33.4 Channel Sounding (Host Support)

This feature bit indicates that the Host supports the Channel Sounding feature.

The Controller shall only set this feature bit if the Controller supports the Channel Sounding feature.

#### 4.6.33.5 Shorter Connection Intervals (Host Support)

This feature bit indicates that the Host supports the Shorter Connection Intervals feature.

The Controller shall only set this feature bit if it supports the Shorter Connection Intervals feature.

### 4.6.34 Periodic Advertising ADI Support

A Controller that supports the Periodic Advertising ADI Support feature shall support the LE Periodic Advertising feature and the ability to transmit and interpret the ADI field in the AUX_SYNC_IND PDU as described in the following sections of this document:

- AUX_SYNC_IND PDU (Section 2.3.1.7)
- Periodic Advertising (Section 4.4.2.12)
- Periodic Advertising Trains (Section 4.4.5.1)

A Controller that does not support the Periodic Advertising ADI Support feature shall not transmit or interpret the ADI field in the AUX_SYNC_IND PDU.

### 4.6.35 Connection Subrating

A Controller that supports the Connection Subrating feature shall support all valid values for *connSubrateFactor* (see Section 4.5.1) and the following sections of this document:

- LL_SUBRATE_REQ (Section 2.4.2.36)
- LL_SUBRATE_IND (Section 2.4.2.37)
- Connection Subrate Update procedure (Section 5.1.19)
- Connection Subrate Request procedure (Section 5.1.20)

A Controller that does not support the Connection Subrating feature shall only support a *connSubrateFactor* of 1.

### 4.6.36 Channel Classification

A Controller that supports the Channel Classification feature shall support the following sections of this document:

- LL_CHANNEL_REPORTING_IND (Section 2.4.2.38)
- LL_CHANNEL_STATUS_IND (Section 2.4.2.39)
- Channel Classification Enable procedure (Section 5.1.21)
- Channel Classification Reporting procedure (Section 5.1.22)

### 4.6.37 Advertising Coding Selection

A Controller that supports Advertising Coding Selection shall support the LE Extended Advertising and LE Coded PHY features and the following section of this document:

- Host selection of the coding scheme used in advertising (see Section 4.4)

and, if the Controller supports HCI:

- advertising reports specifying the coding scheme used (see [Vol 4] Part E, Section 7.7.65.13).

### 4.6.38 Periodic Advertising with Responses - Advertiser

A Controller that supports Periodic Advertising with Responses - Advertiser shall support the LE Periodic Advertising feature in the Advertising state, the Periodic Advertising Sync Transfer - Sender feature, and the following sections of this document:

- AUX_SYNC_SUBEVENT_IND (Section 2.3.1.9)
- AUX_SYNC_SUBEVENT_RSP (Section 2.3.1.10)
- LL_PERIODIC_SYNC_WR_IND (Section 2.4.2.40)
- Trains with responses (Section 4.4.2.12.2)

### 4.6.39 Periodic Advertising with Responses - Scanner

A Controller that supports Periodic Advertising with Responses - Scanner shall support the LE Periodic Advertising feature in the Scanning state, the Periodic Advertising Sync Transfer - Recipient feature, and the following sections of this document:

- AUX_SYNC_SUBEVENT_IND (Section 2.3.1.9)
- AUX_SYNC_SUBEVENT_RSP (Section 2.3.1.10)
- LL_PERIODIC_SYNC_WR_IND (Section 2.4.2.40)
- Scanning for periodic advertisements (Section 4.4.3.4)
- Trains with responses (Section 4.4.5.1.2)

### 4.6.40 LL Extended Feature Set

A Controller that supports the LL Extended Feature Set feature shall support the Peripheral-initiated Features Exchange feature and the following sections of this document:

- LL_FEATURE_EXT_REQ and LL_FEATURE_EXT_RSP (Section 2.4.2.41)
- Feature Page Exchange procedure (Section 5.1.4.3)

### 4.6.41 Channel Sounding

A Controller that supports the Channel Sounding feature shall support the Extended Reject Indication feature, the Peripheral-initiated Features Exchange feature, and the following sections of this document:

- Channel Sounding RF frequencies (see [Vol 6] Part A, Section 2)
- Stable phase ([Vol 6] Part A, Section 3.4)
- Antenna switching for Channel Sounding ([Vol 6] Part A, Section 5.3)

- Phase measurements ([Vol 6] Part A, Section 6)
- LL_CS_SEC_REQ (Section 2.4.2.42)
- LL_CS_SEC_RSP (Section 2.4.2.43)
- LL_CS_CAPABILITIES_REQ [v1] (Section 2.4.2.44)
- LL_CS_CAPABILITES_RSP [v1] (Section 2.4.2.44)
- LL_CS_CONFIG_REQ (Section 2.4.2.45)
- LL_CS_CONFIG_RSP (Section 2.4.2.46)
- LL_CS_REQ (Section 2.4.2.47)
- LL_CS_RSP (Section 2.4.2.48)
- LL_CS_IND (Section 2.4.2.49)
- LL_CS_TERMINATE_REQ (Section 2.4.2.50)
- LL_CS_TERMINATE_RSP (Section 2.4.2.50)
- LL_CS_FAE_REQ (Section 2.4.2.51)
- LL_CS_FAE_RSP (Section 2.4.2.52)
- LL_CS_CHANNEL_MAP_IND (Section 2.4.2.53)
- Minimum Channel Sounding subevent space (Section 4.1.4)
- Channel Sounding (Section 4.5.18)
- Channel Sounding Security Start procedure (Section 5.1.23)
- Channel Sounding Capabilities Exchange procedure (Section 5.1.24)
- Channel Sounding Configuration procedure (Section 5.1.25)
- Channel Sounding Start procedure (Section 5.1.26)
- Channel Sounding Procedure Repeat Termination procedure (Section 5.1.27)
- Channel Sounding Channel Map Update procedure (Section 5.1.28)
- Channel Sounding Mode-0 FAE Table Request procedure (Section 5.1.29)
- Channel Sounding ([Vol 6] Part H)

If the Controller supports the LE 2M PHY, it may support either, both, or neither of the LE 2M PHY and LE 2M 2BT PHY for Channel Sounding.

### 4.6.42 Channel Sounding Tone Quality Indication

A Controller that supports the Channel Sounding Tone Quality Indication feature shall support the Channel Sounding feature and the following section of this document:

- Phase measurements during T_PM ([Vol 6] Part H, Section 4.6)

### 4.6.43 Decision-Based Advertising Filtering

A Controller that supports the Decision-Based Advertising Filtering feature shall support the LE Extended Advertising feature and the following sections of this document:

- ADV_DECISION_IND (Section 2.3.1.11)
- Decision scanning filter policies (Section 4.3.3.2)
- Decision PDU scanning (Section 4.4.3.6)

### 4.6.44 ISOAL Unsegmented Framed Mode

A Controller that supports the ISOAL Unsegmented Framed Mode feature shall support at least one of the following features:

- Connected Isochronous Stream - Central
- Connected Isochronous Stream - Peripheral
- Isochronous Broadcaster
- Synchronized Receiver

and shall support the following sections of this document:

- Unsegmented mode for framed PDUs (see [Vol 6] Part G, Section 2.2)
- Unsegmented mode in SDU synchronization reference and transport latency using framed PDUs (see [Vol 6] Part G, Section 3.2.1)

### 4.6.45 Monitoring Advertisers

A Controller that supports the Monitoring Advertisers feature shall support the following section of this document:

- Monitoring Advertisers (Section 4.4.3.8)

### 4.6.46 Frame Space Update

A Controller that supports the Frame Space Update feature shall support a minimum frame space value that is less than or equal to 145 µs or support a maximum frame

space value that is greater than or equal to 155 µs (or both), support the Extended Reject Indication feature, and support the following sections of this document:

- LL_FRAME_SPACE_REQ (Section 2.4.2.54)
- LL_FRAME_SPACE_RSP (Section 2.4.2.55)
- Frame Space Update procedure (Section 5.1.30)

### 4.6.47 UTP OTA mode

A Controller that supports the UTP OTA mode feature shall support the LE Encryption and Extended Reject Indication features and the following sections of this document:

- LL_OTA_UTP_IND (Section 2.4.2.56)
- UTP OTA mode procedure (Section 5.1.31)
- Unified Test Protocol ([Vol 6] Part F, Section 5)

### 4.6.48 UTP HCI mode

A Controller that supports the UTP HCI mode shall support the following section of this document:

- Unified Test Protocol ([Vol 6] Part F, Section 5)

### 4.6.49 LL_OTA_UTP_IND maximum length

A Controller that supports the LL_OTA_UTP_IND maximum length feature shall support the UTP OTA mode feature and shall support receiving LL_OTA_UTP_IND PDUs with any length of CtrData less than or equal to the maximum specified in Table 4.10.

| Feature bit 68 | Feature bit 69 | Maximum CtrData length |
|---|---|---|
| 0 | 0 | 26 or UTP OTA mode not supported |
| 0 | 1 | 80 |
| 1 | 0 | 180 |
| 1 | 1 | 250 |

*Table 4.10: Maximum CtrData lengths for LL_OTA_UTP_IND*

### 4.6.50 Shorter Connection Intervals

A Controller that supports the Shorter Connection Intervals feature shall support the Connection Subrating feature (see Section 4.6.35) and the following sections of this document:

- LL_CONNECTION_RATE_REQ (Section 2.4.2.57)
- LL_CONNECTION_RATE_IND (Section 2.4.2.58)
- Connection Rate Update procedure (Section 5.1.32)
- Connection Rate Request procedure (Section 5.1.33)

### 4.6.51 LE Flushable ACL Data

A Controller that supports the LE Flushable ACL Data feature shall support a non-zero value for the ACL Flush Timeout value (see [Vol 4] Part E, Section 6.19) and shall support the following section of this document:

- Flushing of ACL data from Host to Controller (Section 4.5.17.1)

### 4.6.52 Channel Sounding Enhancement #1

A Controller that supports the Channel Sounding Enhancement #1 feature shall support the Channel Sounding feature and the following sections of this document:

- LL_CS_CAPABILITIES_REQ [v2] (see Section 2.4.2.44)
- LL_CS_CAPABILITIES_RSP [v2] (see Section 2.4.2.44)

## 4.7 Resolving List

All Link Layers supporting Link Layer privacy (see Section 6) shall contain a set of records for local and peer IRK value pairs. These values are known as the Local IRK and the Peer IRK. The Resolving List IRK pairs shall be associated with a public or static device address known as the Identity Address. The Identity Address may be in the Filter Accept List. All Link Layers supporting Link Layer privacy shall support a Resolving List capable of storing at least one Resolving List Record.

On reset, the Resolving List shall be empty.

The Resolving List is configured by the Host and is used by the Link Layer to resolve Resolvable Private Addresses used by advertisers, scanners, or initiators and to determine the privacy mode of devices on the list (whether or not address resolution is enabled). This allows the Host to configure the Link Layer to act on a request without awakening the Host.

The Filter Accept List and filter policies set by the Host are applied to the associated Identity Address once the Resolvable Private Address has been resolved.

The Identity Address of an associated advertiser in the Monitored Advertisers List of devices set by the Host is applied once the Resolvable Private Address has been resolved.

If the Host, when populating the resolving list, sets a peer IRK to all zeros, then the peer address used within an advertising physical channel PDU shall use the peer's Identity Address, which is provided by the Host.

The Host specifies the privacy mode to be used with each peer identity on the resolving list. If it specifies that device privacy mode is to be used, then the Controller shall accept both the peer's device Identity Address and a resolvable private address generated by the peer device using its distributed IRK. Otherwise, network privacy mode is used: the Controller shall only accept resolvable private addresses generated by the peer device using its distributed IRK. If the Host has added the peer device to the resolving list with an all-zero peer IRK, the Controller shall only accept the peer's Identity Address, as defined in Section 6.5.

If the Host, when populating the resolving list, sets a local IRK to all zeros, then any local address used within an advertising physical channel PDU shall use the local Identity Address, which is provided by the Host.

If the Link Layer is using the Resolving List and the peer device has been resolved, the Address returned to the Host is the peer device's Identity Address.

If the Link Layer is using the Resolving List and the peer device has been resolved but the encryption fails then the current Resolvable Private Address(es) shall be immediately discarded and new Resolvable Private Address(es) shall be generated.

Note: Encryption can fail when the address was resolved successfully using an incorrect IRK and, therefore, encryption keys on both sides did not match.

When the Controller address resolution is enabled, both peer and local RPAs received by the Link Layer shall be resolved using the same Resolving List Record.

Note: The Controller can generate Resolvable Private Addresses even when address resolution is disabled.

# 5 LINK LAYER CONTROL

The Link Layer Control protocol is used to control and negotiate aspects of the operation of a connection between two Link Layers. This includes procedures for control of the connection, starting and pausing encryption and other link procedures.

Procedures have specific timeout rules as defined in Section 5.2. The ACL Termination procedure may be initiated at any time, even if any other Link Layer control procedure is currently active. For all other Link Layer control procedures, only one Link Layer control procedure shall be initiated in the Link Layer at a time per connection per device. A new Link Layer control procedure shall not be initiated until any previous Link Layer control procedure initiated by the same device on the same connection has completed. However, except where forbidden elsewhere in this section, a Link Layer may initiate an LL control procedure while responding to a procedure initiated by its peer device.

Except where stated otherwise in this section, there are no restrictions on the order that Link Layer control procedures are carried out except that no procedure shall be started until after entering the Connection state.

The prioritization of LL Control PDUs and LL Data PDUs is implementation specific. For example, a Host cannot assume that pending data will be sent when a termination of the link is requested without waiting for those data PDUs to have completed and be indicated to the Host.

If the remote device does not support a procedure, the Link Layer will normally receive an LL_UNKNOWN_RSP with the UnknownType field set to the opcode of the initiating PDU. In this case, the procedure is terminated when the LL_UNKNOWN_RSP PDU is received.

## 5.1 Link Layer ACL control procedures

Except for any wording describing the behavior of a Link Layer that does not support a feature, the requirements in each subsection below only apply if the Link Layer supports the relevant feature (see Section 4.6).

### 5.1.1 Connection Update procedure

The Central or Peripheral may update the Link Layer parameters for a connection (*connInterval*, *connPeripheralLatency,* and *connSupervisionTimeout*) by applying the following rules.

If both the Central and Peripheral support the Connection Parameters Request procedure (see Section 5.1.7) then either device should use that procedure. However,

if the Peripheral rejects the proposed connection parameters, the Central may update them using the Connection Update procedure.

If the Central, the Peripheral, or both do not support the Connection Parameters Request procedure, then the Central shall send an LL_CONNECTION_UPDATE_IND PDU (the Peripheral shall not send this PDU) while the Peripheral shall use the L2CAP LE Signaling channel (see [Vol 3] Part A, Section 4.20 and Section 4.21).

If a device supports the Connection Parameters Request procedure but does not know whether its peer does because, in the current connection, neither device has previously attempted that procedure or performed a Feature Exchange procedure, then it shall initiate the Connection Parameters Request procedure and, if the peer responds with an LL_UNKNOWN_RSP PDU, by then using the method described in the previous paragraph.

A Central shall not issue the LL_CONNECTION_UPDATE_IND PDU when a CS procedure or CS procedure repeat instances, as described in Section 4.5.18.1, a Connection Rate Request procedure, or a Connection Rate Update procedure is in progress.

The Link Layer of the Central shall determine the *connInterval* from the interval range given by the Host (Connection_Interval_Min to Connection_Interval_Max); the value chosen shall be at least *connIntervalUncodedMin* µs. However, if the current PHY in either direction is the LE Coded PHY and the Controller supports the LE Data Packet Length Extension feature, then the new connection interval shall be at least *connIntervalCodedMin* µs.

The Link Layer shall indicate to the Host the selected interval value.

Section 5.5 shall apply to the LL_CONNECTION_UPDATE_IND PDU. The Central should transmit on the connection event where *connEventCount* equals Instant and the connection event before that event, irrespective of subrating. When the Peripheral receives such a PDU with the instant in the future, it shall listen to the connection event where *connEventCount* equals Instant and the connection event before that event, even if subrating or Peripheral latency means it would not normally do so.

The connection interval used before the instant is known as $connInterval_{OLD}$. The connection interval contained in the LL_CONNECTION_UPDATE_IND PDU and used at the instant and after, is known as $connInterval_{NEW}$.

The connection Peripheral latency used before the instant is known as $connPeripheralLatency_{OLD}$. The connection Peripheral latency contained in the LL_CONNECTION_UPDATE_IND PDU and used at the instant and after, is known as $connPeripheralLatency_{NEW}$.

The connection supervision timeout used before the instant is known as *connSupervisionTimeout*$_{OLD}$. The connection supervision timeout contained in the LL_CONNECTION_UPDATE_IND PDU and used at the instant and after, is known as *connSupervisionTimeout*$_{NEW}$. The connection supervision timer shall be reset at the instant.

If the connection interval is changed, the subrate factor shall be set to 1 and the continuation number shall be set to 0 at the instant.

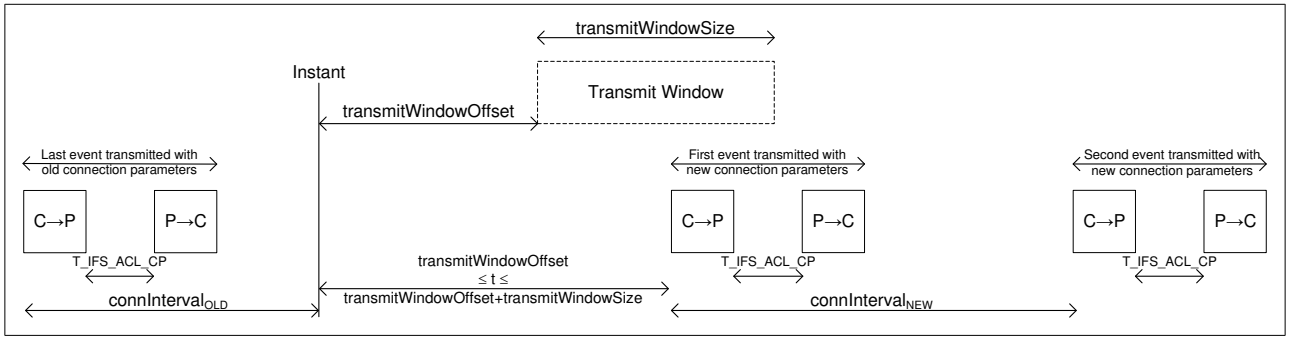



*Figure 5.1: Connection event timing in the case of connection parameter update*

For example, the interval between the preceding connection event before the instant and the instant will be *connInterval*$_{OLD}$. The interval between the connection event after the instant and the following connection event will be *connInterval*$_{NEW}$.

The Central may adjust the anchor point when deciding the timing of the first packet transmitted with new connection parameters. A transmit window is used, as defined in Section 4.5.3. The transmit window starts at *connInterval*$_{OLD}$ + *transmitWindowOffset* after the anchor point of the connection event before the instant. The *transmitWindowOffset* shall be a multiple of 1.25 ms in the range 0 ms to *connInterval*$_{NEW}$.The *transmitWindowSize* shall be a multiple of 1.25 ms in the range 1.25 ms to the lesser of 10 ms and (*connInterval*$_{NEW}$ - 1.25 ms).

Note: If the Peripheral first receives the LL_CONNECTION_UPDATE_IND PDU on the instant, it can immediately use that packet as the new anchor point and does not apply the *transmitWindowOffset* and *transmitWindowSize*.

The Central shall start to send the first packet within the transmit window as defined in Section 4.5.3. The Central's first packet may extend beyond the transmit window.

The first packet sent after the instant by the Central determines the new anchor point for the connection events, and therefore the timings of all future connection events in this connection.

The instant occurs after *connInterval*$_{OLD}$ and before *transmitWindowOffset*. All the normal connection event transmission rules specified in Section 4.5.1 shall apply.

At the start of the transmit window, the Link Layer shall reset $T_{LLconnSupervision}$.

If the Link Layer of the Central transmits an LL_CONNECTION_UPDATE_IND PDU autonomously, for example without being requested to by the Host, the Latency and Timeout parameters shall not be changed and shall remain the same as in the last LL_CONNECTION_UPDATE_IND, CONNECT_IND, or AUX_CONNECT_REQ PDU. Any of the other parameters (*transmitWindowSize*, *transmitWindowOffset*, *connInterval*, Instant) may be changed within the restrictions given above.

Note: Autonomous updates can be used to change the anchor points to allow the Central to change the scheduling of the connection due to other activities.

The Link Layer shall notify its Host if any of the three connection parameters have changed. If no connection parameters are changed, the Host would not be notified; this is called an anchor point move.

The procedure has completed when the instant has passed, and the new connection event parameters have been applied.

### 5.1.2 Channel Map Update procedure

The Central may update the Link Layer parameter for channel map (*channelMap*) by sending an LL_CHANNEL_MAP_IND PDU. The Peripheral shall not send this PDU. The Central's Controller may update the channel map without being requested to by the Host.

Section 5.5 shall apply to the LL_CHANNEL_MAP_IND PDU.

The channel map used before the instant is known as $channelMap_{OLD}$. The channel map contained in the LL_CHANNEL_MAP_IND PDU and used at the instant and after, is known as $channelMap_{NEW}$.

When *connEventCount* is equal to the Instant field, the $channelMap_{NEW}$ shall be the current *channelMap*. The *lastUnmappedChannel* shall not be reset. If the *unmappedChannel* is an unused channel, then the $channelMap_{NEW}$ will be used when remapping. The only parameter that changes is the *channelMap*.

For example:

At connection set-up:

- initial $channelMap_{OLD}$: 0x1FFFFFFFFF (i.e., all channels enabled)
- initial *hopIncrement*: 10 (decimal)

An LL_CHANNEL_MAP_IND PDU with the following parameters is then issued:

- Instant: 100 (decimal). Assume that no connection event count wrap-around occurred since the start of the connection.
- $channelMap_{NEW}$: 0x1FFFFFF7FF (i.e. all channels enabled except channel 11)

Channels used:

- *connEventCount* 99 --> data channel index 1 ($channelMap_{OLD}$)
- *connEventCount* 100 --> data channel index 12 (remapped from 11) ($channelMap_{NEW}$)
- *connEventCount* 101 --> data channel index 21 ($channelMap_{NEW}$)

The procedure has completed when the instant has passed and the new channel map has been applied to the ACL. If the ACL has any associated CISes, the new channel map shall be used on each CIS starting with the next CIS event after the instant.

### 5.1.3 Encryption procedure

The Link Layer of the Central or Peripheral, upon request from the Host, may enable the encryption of packets using the encryption start procedure.

Once the connection is encrypted, the Link Layer may change the encryption key by using the encryption pause procedure, which encapsulates the encryption start procedure.

The Link Layer shall not initiate the encryption start procedure or pause procedure while there is an established CIS associated with the ACL.

#### 5.1.3.1 Encryption Start procedure

To enable encryption, two parameters have to be exchanged, IV and SKD. Both are composed of two parts, a Central part and a Peripheral part, and exchanged in LL_ENC_REQ and LL_ENC_RSP PDUs. After these are exchanged, and the Peripheral's Host has notified its Link Layer of the Long Term Key to be used on this connection, encryption is started using a three way handshake, using LL_START_ENC_REQ and LL_START_ENC_RSP PDUs.

To start encryption, the Link Layer of the Central shall generate the Central’s part of the initialization vector (IV_C) and the Central’s part of the session key diversifier (SKD_C). IV_C shall be a 32 bit random number generated by the Link Layer of the Central. SKD_C shall be a 64 bit random number generated by the Link Layer of the Central. Both IV_C and SKD_C shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

Before transmitting the LL_ENC_REQ PDU, the Link Layer of the Central shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure has completed or specifies otherwise, the Link Layer of the Central shall only send Empty PDUs, LL_TERMINATE_IND PDUs, and PDUs required by this procedure.

The Link Layer of the Central shall then send an LL_ENC_REQ PDU; the Rand and EDIV fields are provided by the Host. After the Central receives the LL_ENC_RSP PDU in response, only PDUs required by this procedure are expected.

If encryption is not supported by the Link Layer of the Peripheral, the Link Layer of the Peripheral shall send an LL_REJECT_IND or LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported Remote Feature* (0x1A). The Link Layer of the Central receiving the LL_REJECT_IND or LL_REJECT_EXT_IND PDU shall notify the Host. The Link Layer of the Central may now send LL Data PDUs and LL Control PDUs; these packets will not be encrypted. This procedure has completed in the Central when the Central receives the LL_REJECT_IND or LL_REJECT_EXT_IND PDU from the Peripheral. The Central should acknowledge this PDU using an Empty PDU. The procedure has completed in the Peripheral when it sends the LL_REJECT_IND or LL_REJECT_EXT_IND PDU to the Central.

Otherwise, when the Link Layer of the Peripheral receives an LL_ENC_REQ PDU it shall generate the Peripheral's part of the initialization vector (IV_P) and the Peripheral's part of the session key diversifier (SKD_P). IV_P shall be a 32 bit random number generated by the Link Layer of the Peripheral. SKD_P shall be a 64 bit random number generated by the Link Layer of the Peripheral. Both IV_P and SKD_P shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

The Link Layer of the Peripheral shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure has completed or specifies otherwise, the Link Layer of the Peripheral shall only send Empty PDUs, LL_TERMINATE_IND PDUs, and PDUs required by this procedure.

If any of the Data Physical Channel PDUs sent by the Peripheral is an LL Control PDU, the Link Layers shall resume any outstanding procedure(s) after the Encryption Start procedure has completed.

The Link Layer of the Peripheral shall then send an LL_ENC_RSP PDU. The Link Layer of the Peripheral shall then notify the Host with the Rand and EDIV fields. After having sent the LL_ENC_RSP PDU, the Link Layer of the Peripheral can receive an

LL_UNKNOWN_RSP PDU corresponding to an LL Control PDU sent by the Peripheral. The Peripheral should not disconnect the link in this case.

Each Link Layer shall combine the initialization vector parts and session key diversifier parts in the following manner:

SKD = SKD_P || SKD_C
IV = IV_P || IV_C

The SKD_C is concatenated with the SKD_P. The least significant octet of SKD_C becomes the least significant octet of SKD. The most significant octet of SKD_P becomes the most significant octet of SKD.

The IV_C is concatenated with the IV_P. The least significant octet of IV_C becomes the least significant octet of IV. The most significant octet of IV_P becomes the most significant octet of IV.

The Long Term Key is always provided by the Host to the Link Layer in the Central and may be provided by the Host to the Link Layer in the Peripheral. One of the following three actions shall occur:

- If this procedure is being performed after a Pause Encryption procedure, and the Peripheral's Host does not provide a Long Term Key, the Peripheral shall perform the ACL Termination procedure with the error code *PIN or Key Missing* (0x06).
- If the Peripheral's Host does not provide a Long Term Key, either because the event to the Host was masked out or if the Host indicates that a key is not available, the Peripheral shall either send an LL_REJECT_IND with the ErrorCode set to *PIN or Key Missing* (0x06) or an LL_REJECT_EXT_IND PDU with the RejectOpcode set to "LL_ENC_REQ" and the ErrorCode set to *PIN or Key Missing* (0x06). Upon receiving an LL_REJECT_IND or LL_REJECT_EXT_IND PDU, the Central's Link Layer shall notify its Host. Both Link Layers may now send LL Data PDUs and LL Control PDUs; these packets will not be encrypted. This procedure has completed in the Central when the Central receives the LL_REJECT_IND or LL_REJECT_EXT_IND PDU from the Peripheral. The procedure has completed in the Peripheral when the acknowledgment has been received for the LL_REJECT_IND or LL_REJECT_EXT_IND PDU from the Central.
- If the Peripheral's Host does provide a Long Term Key, both Link Layers shall calculate the session key as e(LTK, SKD), where e is defined in [Vol 3] Part H, Section 2.2.1.

After the Peripheral's Link Layer has calculated the session key, it shall send an LL_START_ENC_REQ PDU. This packet shall be sent unencrypted and the Link Layer shall be set up to receive an encrypted packet in response.

When the Link Layer of the Central receives an LL_START_ENC_REQ PDU it shall send an LL_START_ENC_RSP PDU. This PDU shall be sent encrypted and the Link Layer shall be set up to receive an encrypted packet in response.

When the Link Layer of the Peripheral receives an LL_START_ENC_RSP PDU it shall transmit an LL_START_ENC_RSP PDU. This packet shall be sent encrypted.

When the Link Layer of the Central receives the LL_START_ENC_RSP PDU, the connection is encrypted. The Link Layer may now send LL Data PDUs and LL Control PDUs; these PDUs will be encrypted.

The Link Layers shall notify the Hosts that the connection is encrypted.

The procedure has completed in the Central when the Central receives the LL_START_ENC_RSP PDU from the Peripheral. The procedure has completed in the Peripheral when the Peripheral receives the LL_START_ENC_RSP PDU from the Central.

If, at any time during the encryption start procedure after the Peripheral has received the LL_ENC_REQ PDU or the Central has received the LL_ENC_RSP PDU, the Link Layer of the Central or the Peripheral receives an unexpected Data Physical Channel PDU from the peer Link Layer, it shall immediately exit the Connection state, and shall transition to the Standby state. The Host shall be notified that the link has been disconnected with the error code *Connection Terminated due to MIC Failure* (0x3D).

#### 5.1.3.2 Encryption Pause procedure

To enable a new encryption key to be used without disconnecting the link, encryption is disabled and then enabled again. During the pause, data PDUs shall not be sent unencrypted to protect the data.

The Link Layer of the Central shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure has completed, the Link Layer of the Central shall only send Empty PDUs, LL_TERMINATE_IND PDUs, and PDUs required by this procedure.

The Link Layer of the Central shall then send an LL_PAUSE_ENC_REQ PDU. After the Central receives the LL_PAUSE_ENC_RSP PDU in response, only PDUs required by this procedure are expected.

When the Link Layer of the Peripheral receives an LL_PAUSE_ENC_REQ PDU it shall finalize the sending of the current Data Physical Channel PDU and may finalize the sending of additional Data Physical Channel PDUs queued in the Controller. After these Data Physical Channel PDUs are acknowledged, until this procedure has completed,

the Link Layer of the Peripheral shall only send Empty PDUs, LL_TERMINATE_IND PDUs, and PDUs required by this procedure.

If any of the Data Physical Channel PDUs sent by the Peripheral is an LL Control PDU, the Link Layers shall resume any outstanding procedure(s) after the Encryption Start procedure has completed.

The Link Layer of the Peripheral shall then send an LL_PAUSE_ENC_RSP PDU. This packet shall be sent encrypted and Link Layer shall be set up to receive an unencrypted packet in response.

When the Link Layer of the Central receives an LL_PAUSE_ENC_RSP PDU it shall be set up to send and receive unencrypted. It shall then send an LL_PAUSE_ENC_RSP PDU to the Peripheral unencrypted.

When the Link Layer of the Peripheral receives an LL_PAUSE_ENC_RSP PDU it shall be set up to also send unencrypted.

The two Link Layers shall then carry out the steps of the encryption start procedure to re-enable encryption using a new session key. The encryption pause procedure has completed when this encapsulated encryption start procedure has completed.

If, at any time during the encryption pause procedure after the Peripheral has received the LL_PAUSE_ENC_REQ PDU or the Central has received the LL_PAUSE_ENC_RSP PDU, the Link Layer of the Central or the Peripheral receives an unexpected Data Physical Channel PDU from the peer Link Layer, it shall immediately exit the Connection state, and shall transition to the Standby state. The Host shall be notified that the link has been disconnected with the error code *Connection Terminated due to MIC Failure* (0x3D).

### 5.1.4 Feature Exchange procedure

The Central or Peripheral may initiate the Feature Exchange procedure to exchange the Link Layer parameter for the current supported feature set (FeatureSet).

The FeatureSet information may be cached either during a connection or between connections. A Link Layer should not request this information on every connection if the information has been cached for this device. Cached information for a device from a previous connection is not authoritative and, therefore, an implementation must be able to accept the LL_UNKNOWN_RSP PDU if use of a feature is attempted that is not currently supported or used by the peer (see Section 2.4.2).

The FeatureSet_C parameter is the feature capabilities of the Link Layer of the Central with certain bits set to zero as specified in Section 4.6.

The FeatureSet_P parameter is the feature capabilities of the Link Layer of the Peripheral with certain bits set to zero as specified in Section 4.6.

The FeatureSet_USED parameter is one octet long and is the logical AND of the least significant octets of FeatureSet_C and FeatureSet_P.

#### 5.1.4.1 Central-initiated Feature Exchange procedure

The Link Layer of the Central initiates this procedure by sending an LL_FEATURE_REQ PDU to the Peripheral. This may be done on request from the Host or autonomously. When the Link Layer of the Peripheral receives this, it shall respond by sending an LL_FEATURE_RSP PDU.

When the Link Layer of the Central sends an LL_FEATURE_REQ PDU, the FeatureSet field shall be set to the first 8 octets of FeatureSet_C.

When the Link Layer of the Peripheral sends an LL_FEATURE_RSP PDU, octet 0 of the FeatureSet field shall be set to FeatureSet_USED and the remaining octets shall be set to octets 1 to 7 of FeatureSet_P.

The procedure has completed when the Central receives the LL_FEATURE_RSP PDU from the Peripheral.

#### 5.1.4.2 Peripheral-initiated Feature Exchange procedure

The Link Layer of the Peripheral initiates this procedure by sending an LL_PERIPHERAL_FEATURE_REQ PDU to the Central. This may be done on request from the Host or autonomously. When the Link Layer of the Central receives this, it shall respond by sending an LL_FEATURE_RSP PDU.

When the Link Layer of the Peripheral sends an LL_PERIPHERAL_FEATURE_REQ PDU, the FeatureSet field shall be set to the first 8 octets of FeatureSet_P.

When the Link Layer of the Central sends an LL_FEATURE_RSP PDU, octet 0 of the FeatureSet field shall be set to FeatureSet_USED and the remaining octets shall be set to octets 1 to 7 of FeatureSet_C.

If the LL_PERIPHERAL_FEATURE_REQ PDU was issued as a result of a Host-initiated read remote features procedure (see [Vol 4] Part E, Section 7.8.21) and the Central does not support this procedure, then the Host shall be notified that the read remote features procedure has completed with the ErrorCode set to *Unsupported Remote Feature* (0x1A).

The procedure has completed when the Peripheral receives the LL_FEATURE_RSP PDU from the Central.

#### 5.1.4.3 Feature Page Exchange procedure

The Link Layer of the Central or Peripheral initiates this procedure by sending an LL_FEATURE_EXT_REQ PDU to the peer device. This may be done on request from the Host or autonomously. When the peer device receives this PDU, it shall respond by sending an LL_FEATURE_EXT_RSP PDU to the initiating device.

The responding device shall set the PageNumber field of the LL_FEATURE_EXT_RSP PDU to the same value as received in the PageNumber field of the LL_FEATURE_EXT_REQ PDU. Each device shall set FeaturePage to the corresponding page of FeatureSet_C if the device is the Central or of FeatureSet_P if the device is the Peripheral.

Each device shall set MaxPage to the number of the highest-numbered page of FeatureSet_C (if the device is the Central) or FeatureSet_P (if the device is the Peripheral) containing at least one bit set to 1.

The procedure has completed when the initiating device receives the LL_FEATURE_EXT_RSP PDU from the responding device.

### 5.1.5 Version Exchange procedure

The Central or Peripheral may initiate the Version Exchange procedure to exchange the Link Layer parameters for version information (*companyID*, *subVerNum*, *linkLayerVer*, as defined in Section 2.4.2.13) by sending an LL_VERSION_IND PDU. This procedure should be used when requested by the Host. This procedure may be initiated autonomously by the Link Layer.

The Link Layer shall only queue for transmission a maximum of one LL_VERSION_IND PDU during a connection.

If the Link Layer receives an LL_VERSION_IND PDU and has not already sent an LL_VERSION_IND then the Link Layer shall send an LL_VERSION_IND PDU to the peer device.

If the Link Layer receives an LL_VERSION_IND PDU and has already sent an LL_VERSION_IND PDU then the Link Layer shall not send another LL_VERSION_IND PDU to the peer device.

The procedure has completed when an LL_VERSION_IND PDU has been received from the peer device.

### 5.1.6 ACL Termination procedure

This procedure is used for voluntary termination of an ACL connection while in the Connection state. Voluntary termination occurs when the Host requests the Link Layer

to terminate the connection. Either the Link Layer of the Central or Peripheral may initiate this procedure by sending an LL_TERMINATE_IND PDU. The ACL Termination procedure is not used in the event of the loss of the connection, for example after link supervision timeout or after a procedure timeout.

The Link Layer shall start a timer, $T_{terminate}$, when the LL_TERMINATE_IND PDU has been queued for transmission. The initiating Link Layer shall send LL_TERMINATE_IND PDUs until an acknowledgment is received or until the timer, $T_{terminate}$, expires, after which it shall exit the Connection State and transition to the Standby State. The initial value for $T_{terminate}$ shall be set to the value of the *connSupervisionTimeout*.

When the Link Layer receives an LL_TERMINATE_IND PDU it shall send the acknowledgment, exit the Connection State and shall transition to the Standby State.

As soon as the Link Layer has received or queued for transmission an LL_TERMINATE_IND PDU all associated CISes shall be considered lost (see Section 4.5.12). The Link Layer shall not send separate LL_CIS_TERMINATE_IND PDUs when the Host requests termination.

The procedure has completed when the acknowledgment has been received or the timer, $T_{terminate}$, expires.

### 5.1.7 Connection Parameters Request procedure

The Central or Peripheral may initiate a Connection Parameters Request procedure to request the remote device to have the Link Layer parameters for the connection (*connInterval*, *connPeripheralLatency* and *connSupervisionTimeout*) updated any time after entering the Connection State.

A device shall only initiate this procedure when permitted by Section 5.1.1. A device shall not initiate this procedure while a Connection Subrate Update procedure is in progress. A device shall not initiate this procedure while a Connection Rate Request procedure or a Connection Rate Update procedure is in progress. A device shall not initiate this procedure while any CS procedures are in progress (as described in Section 4.5.18.1).

#### 5.1.7.1 Issuing an LL_CONNECTION_PARAM_REQ PDU

The Connection Parameters Request procedure is initiated by issuing an LL_CONNECTION_PARAM_REQ PDU. The procedure may be initiated as a result of a Host initiated connection update procedure (see [Vol 4] Part E, Section 7.8.18) or autonomously by the Link Layer (that is, without being requested by the Host).

If the LL_CONNECTION_PARAM_REQ PDU was issued by the Link Layer of the Peripheral as a result of a Host initiated connection update procedure and the Central

does not support this procedure, then the Host shall be notified that the connection update procedure has completed with the ErrorCode set to *Unsupported Remote Feature* (0x1A).

If the Link Layer initiates this procedure as a result of a Host-initiated connection update procedure, then the Link Layer:

- Should set the Interval_Min, Interval_Max, Timeout, and Latency fields to the values received from the Host.

  The Link Layer may modify the values of these fields, for example, because the values received from the Host would prevent the Link Layer from meeting commitments in another piconet.
- May indicate the preferred periodicity by setting the PreferredPeriodicity field to a value other than zero, as described in Section 2.4.2.16.
- May set the Offset0 to Offset5 fields to a value other than 0xFFFF as described in Section 2.4.2.16. If all of the Offset0 to Offset5 fields have been set to 0xFFFF, then the Link Layer has no preference about the offset to be used. If one or more of the Offset0 to Offset5 fields have been set to a value other than 0xFFFF, then:
  - The ReferenceConnEventCount field shall be set to indicate that at least one of the Offset0 to Offset5 fields is valid. If the ReferenceConnEventCount field is set, then it shall always be set to the connEventCount of a connection event that is less than 32767 connection events in the future from the first transmission of the PDU.

    Note: Retransmissions of the PDU can result in the ReferenceConnEventCount to be up to 32767 events in the past when the PDU is successfully received by the remote device. See Section 5.1.7.3.2 for examples on how to set the ReferenceConnEventCount field.
  - If Interval_Min is not equal to Interval_Max, then the PreferredPeriodicity field shall be set to a value other than zero. If Interval_Min is equal to Interval_Max, then the PreferredPeriodicity field may be set to any value and shall be ignored by the recipient.

If the Link Layer initiates this procedure autonomously, then the Latency field shall be set to the current value of *connPeripheralLatency* and the Timeout field (in milliseconds) shall be set to the current value of *connSupervisionTimeout*. Any of the other fields (Interval_Min, Interval_Max, PreferredPeriodicity, ReferenceConnEventCount and Offset0 to Offset5) may be changed within the restrictions given above.

The Link Layer shall ensure that the parameters in the LL_CONNECTION_PARAM_REQ shall not cause supervision timeout. That is, the Link Layer shall ensure that the Timeout (in milliseconds) is greater than 2 × Interval_Max × (Latency + 1).

#### 5.1.7.2 Responding to LL_CONNECTION_PARAM_REQ and LL_CONNECTION_PARAM_RSP PDUs

Upon receiving an LL_CONNECTION_PARAM_REQ PDU:

- The Peripheral shall respond with either an LL_CONNECTION_PARAM_RSP PDU or an LL_REJECT_EXT_IND PDU.
- The Central shall respond with either an LL_CONNECTION_UPDATE_IND PDU or an LL_REJECT_EXT_IND PDU.

If an LL_CONNECTION_PARAM_REQ PDU is received while a CS procedure or CS procedure repeat instances, as described in Section 4.5.18.1, are in progress, then the receiving Link Layer shall respond with an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Controller Busy* (0x3A).

Upon receiving an LL_CONNECTION_PARAM_RSP PDU, the Central shall respond with either an LL_CONNECTION_UPDATE_IND PDU or an LL_REJECT_EXT_IND PDU.

The Central shall not send the LL_CONNECTION_PARAM_RSP PDU. The Peripheral shall send an LL_CONNECTION_PARAM_RSP PDU only in response to an LL_CONNECTION_PARAM_REQ PDU.

If the received LL_CONNECTION_PARAM_REQ PDU contains parameters that are not acceptable to the Link Layer, then the Link Layer of the device shall respond to the LL_CONNECTION_PARAM_REQ PDU with one of the following:

- An LL_CONNECTION_PARAM_RSP PDU (if the Link Layer is the Peripheral of the connection) or an LL_CONNECTION_UPDATE_IND PDU (if the Link Layer is the Central of the connection), in each case containing alternative parameters.
- An LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported LL Parameter Value* (0x20).

If the received LL_CONNECTION_PARAM_REQ PDU contains any fields that are out of valid range, then the Link Layer shall reject the LL_CONNECTION_PARAM_REQ PDU by issuing an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Invalid LL Parameters* (0x1E).

If an LL_REJECT_EXT_IND PDU is sent during the Connection Parameters Request procedure, then the procedure has completed on a device when it receives the LL_REJECT_EXT_IND PDU, and has completed on the device that issued the LL_REJECT_EXT_IND PDU when it receives the acknowledgment for the LL_REJECT_EXT_IND PDU.

If the received LL_CONNECTION_PARAM_REQ PDU requests only a change in the anchor points of the LE connection, then the Link Layer shall not indicate this request to its Host.

If the received LL_CONNECTION_PARAM_REQ PDU requests a change to one or more of *connInterval*, *connPeripheralLatency*, and *connSupervisionTimeout* and if the values selected by the Link Layer are, respectively, within the range of the *connInterval*, the value of *connPeripheralLatency* and the value of *connSupervisionTimeout* provided by the local Host, then the Link Layer may choose to not indicate this request to its Host and proceed as if the Host has accepted the remote device's request. Otherwise, if the event to the Host is not masked, then the Link Layer shall first indicate this request to its Host.

If the local Host has not provided the range of *connInterval*, the value of *connPeripheralLatency* and the value of *connSupervisionTimeout* to the Link Layer of the Peripheral, then the Link Layer of the Peripheral may indicate the received request to its Host if the event to the Host is not masked.

If the request is being indicated to the Host and the event to the Host is masked, then the Link Layer shall issue an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported Remote Feature* (0x1A). The initiating device may retry in case this is a temporary situation. If the request is not being indicated to the Host, then the event mask shall be ignored.

If the Host is indicated of the request, it shall either accept or reject this request. If the Host rejects this request, then the device shall issue an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unacceptable Connection Parameters* (0x3B).

If the Host accepts this request or if the request was not indicated to the Host, then:

- The Peripheral shall respond to an LL_CONNECTION_PARAM_REQ PDU with an LL_CONNECTION_PARAM_RSP PDU. The rules for filling in various fields of the LL_CONNECTION_PARAM_RSP PDU are the same as those for filling in various fields of the LL_CONNECTION_PARAM_REQ PDU, as described in Section 5.1.7.1. The rules for handling a received LL_CONNECTION_PARAM_RSP PDU on the Link Layer of the Central are identical to the rules for handling a received LL_CONNECTION_PARAM_REQ PDU that are described earlier in this section.
- The Central shall respond to an LL_CONNECTION_PARAM_REQ PDU or an LL_CONNECTION_PARAM_RSP PDU with an LL_CONNECTION_UPDATE_IND PDU. The Central should try to choose a value of Interval that is a multiple of PreferredPeriodicity if the Peripheral has set the PreferredPeriodicity field of the LL_CONNECTION_PARAM_REQ or LL_CONNECTION_PARAM_RSP PDU. The chosen value shall be at least *connIntervalUncodedMin* µs. However, if the current PHY in either direction is the LE Coded PHY and the Controller supports the LE

Data Packet Length Extension feature, then the new connection interval shall be at least *connIntervalCodedMin* µs. The Central should try to pick the values of WinOffset and WinSize such that the timing of the new connection events matches one of the Offset0 to Offset5 fields of the LL_CONNECTION_PARAM_REQ PDU or the LL_CONNECTION_PARAM_RSP PDU sent by the Peripheral. The Instant field of the LL_CONNECTION_UPDATE_IND PDU is set as described in Section 5.1.1.

Once the Central issues the LL_CONNECTION_UPDATE_IND PDU, the connection parameters get updated as described in Section 5.1.1.

If the connection interval is changed, the subrate factor shall be set to 1 and the continuation number shall be set to 0 at the instant of the procedure.

The procedure has completed when the instant has passed and the new connection event parameters have been applied.

#### 5.1.7.3 Examples

##### 5.1.7.3.1 Peripheral initiated anchor point move

The following example shows the Link Layer of the Peripheral requesting a change in the anchor points of the LE connection by 3.75 ms.

The Link Layer of the Peripheral issues an LL_CONNECTION_PARAM_REQ PDU with the following parameters:

- Interval_Min: *connInterval*
- Interval_Max: *connInterval*
- Latency: *connPeripheralLatency*
- Timeout: *connSupervisionTimeout*
- PreferredPeriodicity: 0
- ReferenceConnEventCount: <any value that is less than 32767 connection events in the future>
- Offset0: 0x0003
- Offset1: 0xFFFF
- Offset2: 0xFFFF
- Offset3: 0xFFFF
- Offset4: 0xFFFF
- Offset5: 0xFFFF

If the Link Layer of the Central accepts the Peripheral's request, then it could respond with an LL_CONNECTION_UPDATE_IND PDU that contains any one of the following

set of parameters. In all the sets, Interval is set to *connInterval*, Latency is set to *connPeripheralLatency*, Timeout is set to *connSupervisionTimeout* and Instant is set to any value that is less than 32767 connection events in the future.

- Option 1: the first packet sent after the instant by the Central is inside the Transmit Window and 3.75 ms from the beginning of the Transmit Window.
  - 3 ≤ WinSize ≤ 8
  - WinOffset: 0
- Option 2: the first packet sent after the instant by the Central is inside the Transmit Window and 2.5 ms from the beginning of the Transmit Window.
  - 2 ≤ WinSize ≤ 8
  - WinOffset: 1
- Option 3: the first packet sent after the instant by the Central is inside the Transmit Window and 1.25 ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 2
- Option 4: the first packet sent after the instant by the Central is inside the Transmit Window and 0 ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 3

#### 5.1.7.3.2 ReferenceConnEventCount

Figure 5.2 and Figure 5.3 show examples of how the ReferenceConnEventCount and the Offset0 to Offset5 fields of the LL_CONNECTION_PARAM_REQ and the LL_CONNECTION_PARAM_RSP PDU can be used to indicate the possible position of the anchor points of the connection with the new connection parameters relative to the anchor points of the connection with the old connection parameters. This figure only shows Offset0 (and not Offset1 to Offset5) for simplicity. The figure also shows the Instant where the updated connection parameters are applied. The actual Instant occurs $connInterval_{OLD}$ after the last connection event transmitted with the old connection parameters whereas the Instant field in the LL_CONNECTION_UPDATE_IND PDU is set to the *connEventCount* of the connection event transmitted with the old connection parameters.

The ReferenceConnEventCount is set to the *connEventCount* of the connection event on the old connection parameters such that the start of the very next connection event on the new connection parameters is Offset0 (in milliseconds) away from the start of the ReferenceConnEventCount connection event.

Figure 5.2 shows the case where the Instant is before the ReferenceConnEventCount. Figure 5.3 shows the case where the Instant is after the ReferenceConnEventCount. Imaginary connection events transmitted with the old connection parameters have been shown beyond the Instant and imaginary connection events transmitted with the new connection parameters have been shown before the Instant.

In Figure 5.2 and Figure 5.3, the time interval, Δt, between the Instant and the start of the first connection event transmitted with the new connection parameters can be calculated using the following equation:

$$\Delta t = (connInterval_{NEW} - ((Instant - ReferenceConnEventCount) \times connInterval_{OLD}) \bmod connInterval_{NEW} + offset0) \bmod connInterval_{NEW}$$

Note: The case where the ReferenceConnEventCount and Instant are on different sides of the eventCount wraparound point is not shown in the equations above.

Based on the calculated Δt, the WinOffset and WinSize fields in the LL_CONNECTION_UPDATE_IND PDU could be set accordingly. See Section 5.1.7.3.3 for an example.

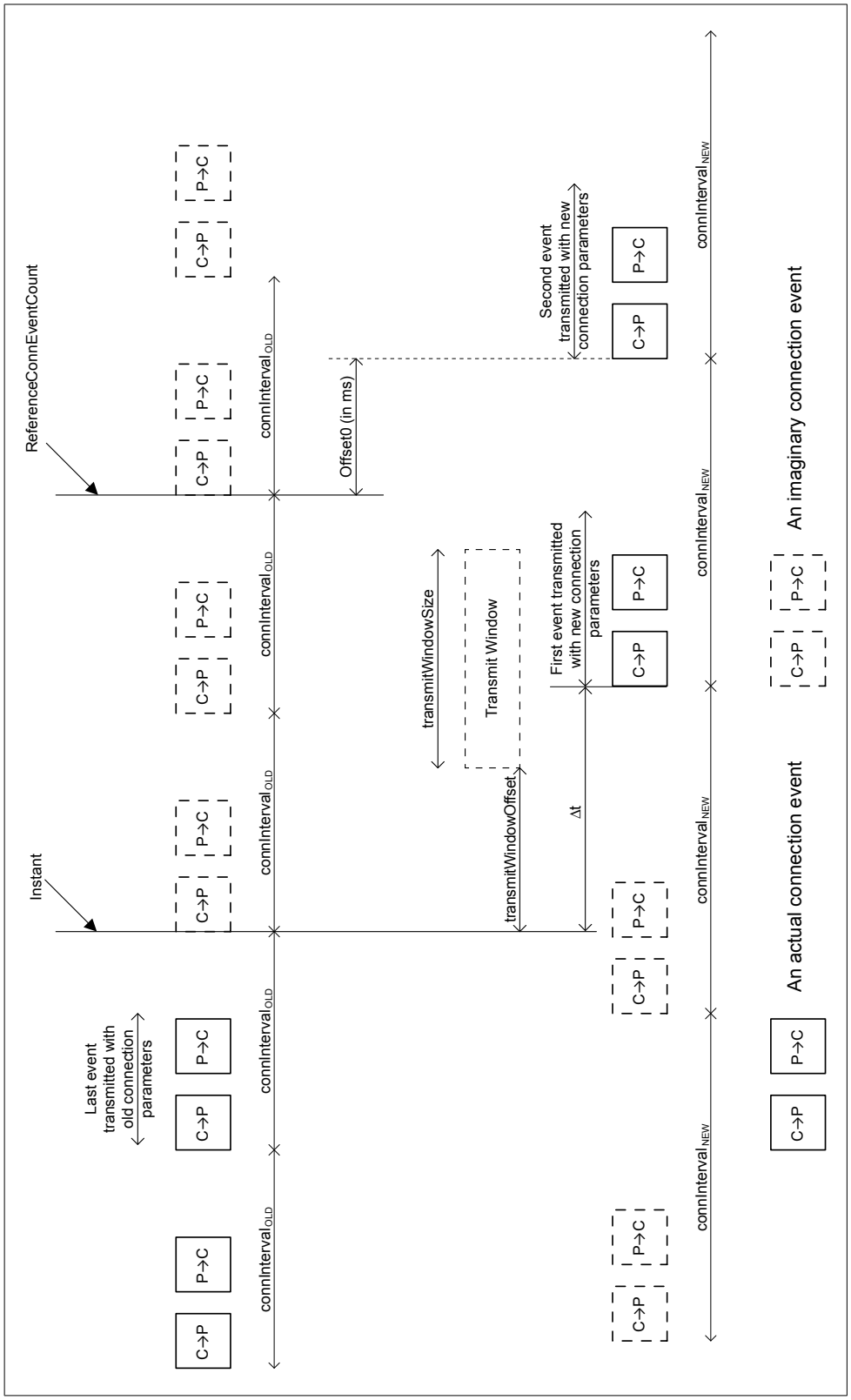



*Figure 5.2: Using the ReferenceConnEventCount and Offset0 fields to indicate position of the new anchor points (Instant is before the ReferenceConnEventCount)*

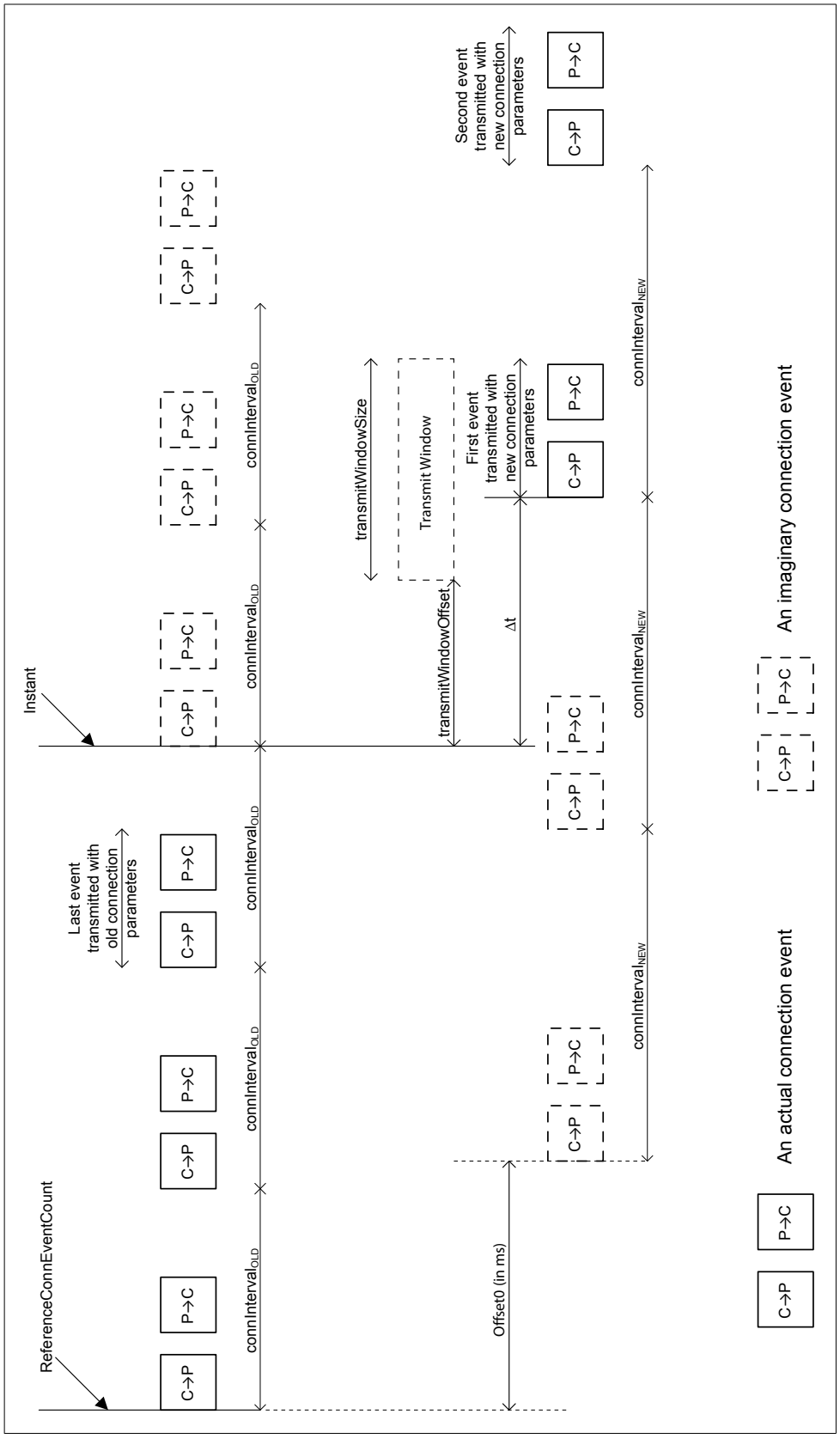



*Figure 5.3: Using the ReferenceConnEventCount and Offset0 fields to indicate position of the new anchor points (Instant is after the ReferenceConnEventCount)*

#### 5.1.7.3.3 Peripheral initiated interval and anchor point move

The following example shows the Link Layer of the Peripheral requesting a change in both the connection interval (by indicating a PreferredPeriodicity such that PreferredPeriodicity and $connInterval_{OLD}$ are not integer multiples of one another) and a change in anchor points of the LE connection by 3.75 ms with respect to the ReferenceConnEventCount.

In this example, $connInterval_{OLD}$ is 0x0C (15 ms). The Link Layer of the Peripheral issues an LL_CONNECTION_PARAM_REQ PDU with the following parameters:

- Interval_Min: 0x16
- Interval_Max: 0x20
- Latency: *connPeripheralLatency*
- Timeout: *connSupervisionTimeout*
- PreferredPeriodicity: 0x0A
- ReferenceConnEventCount: 0x1F00
- Offset0: 0x0003
- Offset1: 0xFFFF
- Offset2: 0xFFFF
- Offset3: 0xFFFF
- Offset4: 0xFFFF
- Offset5: 0xFFFF

If the Link Layer of the Central accepts the Peripheral's request, then it could respond with an LL_CONNECTION_UPDATE_IND PDU that contains any one of the following set of parameters. In all the sets, the new connection interval $connInterval_{NEW}$ is set to 0x1E (37.5 ms), Latency is set to *connPeripheralLatency*, Timeout is set to *connSupervisionTimeout* and Instant is set to 0x1F06.

Δt, as described in Section 5.1.7.3.2 is calculated as 21 (26.25 ms).

The WinSize and WinOffset fields in the LL_CONNECTION_UPDATE_IND PDU can contain any of the following example set of parameters:

- Option 1: the first packet sent after the instant by the Central is inside the Transmit Window and 3.75 ms from the beginning of the Transmit Window.
  - 3 ≤ WinSize ≤ 8
  - WinOffset: 18

- Option 2: the first packet sent after the instant by the Central is inside the Transmit Window and 2.5 ms from the beginning of the Transmit Window.
  - 2 ≤ WinSize ≤ 8
  - WinOffset: 19
- Option 3: the first packet sent after the instant by the Central is inside the Transmit Window and 1.25 ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 20
- Option 4: the first packet sent after the instant by the Central is inside the Transmit Window and 0 ms from the beginning of the Transmit Window.
  - 1 ≤ WinSize ≤ 8
  - WinOffset: 21

#### 5.1.7.4 Packet transmit time restrictions

This section only applies if the current PHY in either direction is the LE Coded PHY and the Controller supports the LE Data Packet Length Extension feature.

After having sent or received an LL_CONNECTION_UPDATE_IND PDU that decreases the connection interval, and until the instant has been reached, the Link Layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds (see Section 4.5.10) to transmit, calculated using the connection interval that will apply after the instant.

After a Peripheral sends an LL_CONNECTION_PARAM_REQ or LL_CONNECTION_PARAM_RSP PDU where *Interval_Min* indicates an interval less than the current connection interval, and until it receives an LL_CONNECTION_UPDATE_IND, LL_UNKNOWN_RSP, or LL_REJECT_EXT_IND PDU in response, its Link Layer shall not transmit a packet that would take longer than *connEffectiveMaxTxTime* microseconds to transmit, calculated using a connection interval corresponding to the *Interval_Min* value in the transmitted PDU.

If the value of *connEffectiveMaxTxTime* changes during the procedure, the above requirements apply to the value at the moment the LL Data PDU is queued for transmission.

Note: The requirements of this section are in addition to, and do not override, those in Section 4.5.10.

Note: If a Link Layer has any LL Data PDUs queued for transmission at the start of the procedure or queues any during the procedure, it might need to re-fragment those PDUs in order to meet these requirements.

### 5.1.8 LE Ping procedure

The Link Layer may use the LE Ping procedure, when supported, to verify the presence of the remote Link Layer, or to verify message integrity on the LE ACL logical transport, by forcing the remote device to send an LE ACL packet that contains a valid MIC.

This procedure may be used even if it is not supported by the peer's Link Layer.

Either the Central or the Peripheral Link Layer may initiate this procedure at any time after entering the Connection state by sending an LL_PING_REQ PDU. The responding Link Layer responds with the LL_PING_RSP PDU.

The Link Layer supporting this feature shall send an LL_PING_REQ PDU when the remote device has not sent a packet containing a Payload field protected by a MIC within the authenticated payload timeout set by the Host ([Vol 4] Part E, Section 7.3.94). The Link Layer should send an LL_PING_REQ PDU in advance enough of the expiration of the authenticated payload timeout to allow the remote device reasonable time to respond with an LL_PING_RSP PDU before the timeout expires.

The procedure has completed when an LL_PING_RSP is received.

### 5.1.9 Data Length Update procedure

A Controller uses the Data Length Update procedure to transmit the latest values of the current maximum Receive LL Data PDU Payload length and PDU Time (*connMaxRxOctets* and *connMaxRxTime*) and the current maximum Transmit LL Data PDU Payload length and PDU Time (*connMaxTxOctets* and *connMaxTxTime*) to the peer device.

Both the Central and Peripheral may initiate this procedure by sending an LL_LENGTH_REQ PDU. This procedure shall be initiated by the Link Layer whenever any of these parameters change, whether requested by the Host or autonomously by the Link Layer. However, if this procedure has already been initiated by the remote Controller and the local Controller has not yet responded, it shall use the response to communicate the changes instead of initiating a new procedure.

If the Link Layer receives an LL_LENGTH_REQ, or an LL_LENGTH_RSP PDU that was a response to an LL_LENGTH_REQ PDU, then it shall update its *connRemoteMaxTxOctets*, *connRemoteMaxRxOctets*, *connRemoteMaxTxTime*, and *connRemoteMaxRxTime* parameters for the connection with the values in the PDU. It shall immediately start using the updated values for all new LL Data PDUs queued for transmission (including any response as specified in the next paragraph). The lengths of any LL Data PDUs that have already been queued for transmission or transmitted at least once shall not be changed.

Note: Because Link Layer PDUs are not required to be processed in real time, it is possible for the local Controller to have queued but not yet transmitted an LL_LENGTH_REQ PDU when it receives an LL_LENGTH_REQ PDU from the peer device. In this situation each device responds as normal; the resulting collision is harmless.

Upon receiving an LL_LENGTH_REQ PDU, the Link Layer shall respond with an LL_LENGTH_RSP PDU containing its own *connMaxTxOctets*, *connMaxRxOctets*, *connMaxTxTime*, and *connMaxRxTime* values for the connection (which it may have updated based on the values received, for example so as to allow the remote device to transmit longer packets).

If the peer device does not support the LE Coded PHY feature, then the MaxRxTime and MaxTxTime fields in the LL_LENGTH_REQ and LL_LENGTH_RSP PDUs shall be set to a value less than or equal to 2128 microseconds.

The procedure has completed when the initiating Controller receives an LL_LENGTH_RSP PDU.

### 5.1.10 PHY Update procedure

The Central or Peripheral may use the PHY Update procedure, when supported, to change the transmit or receive PHYs, or both, of an ACL connection; it does not affect the transmit or receive PHY of any associated Connected Isochronous Streams. The procedure may be initiated either on a request by the Host or autonomously by the Link Layer. Link Layer PHY preferences may change during a connection or between connections and, therefore, they should not be cached by the peer device.

When this procedure is initiated by the Central, it sends an LL_PHY_REQ PDU. The Peripheral responds with an LL_PHY_RSP PDU. The Central then responds to this with an LL_PHY_UPDATE_IND PDU.

When this procedure is initiated by the Peripheral, it sends an LL_PHY_REQ PDU. The Central responds with an LL_PHY_UPDATE_IND PDU.

The TX_PHYS and RX_PHYS fields of the LL_PHY_REQ and LL_PHY_RSP PDUs shall be used to indicate the PHYs that the sending Link Layer prefers to use. These shall represent PHYs that the sending Link Layer supports. The sender can request a symmetric connection (one where the two PHYs are the same) by making both fields the same, only specifying a single PHY.

The PHY_C_TO_P and PHY_P_TO_C fields of the LL_PHY_UPDATE_IND PDU shall indicate the PHYs that shall be used after the instant.

If the Central initiated the procedure, it shall determine the PHY to use in each direction based on the contents of the LL_PHY_REQ and LL_PHY_RSP PDUs using the following rules:

- the PHY_C_TO_P field of the LL_PHY_UPDATE_IND PDU shall be determined from the Central's TX_PHYS field and the Peripheral's RX_PHYS field;
- the PHY_P_TO_C field of the LL_PHY_UPDATE_IND PDU shall be determined from the Central's RX_PHYS field and the Peripheral's TX_PHYS field.

In each of those cases the following rules apply:

- if, for at least one PHY, the corresponding bit is set to 1 in both the TX_PHYS and RX_PHYS fields, the Central shall select any one of those PHYs for that direction;
- if there is no PHY for which the corresponding bit is set to 1 in both the TX_PHYS and RX_PHYS fields, the Central shall not change the PHY for that direction.

If the Peripheral initiated the procedure, the Central shall determine the PHY to use in each direction based on the contents of the LL_PHY_REQ PDU sent by the Peripheral using the following rules:

- the PHY_C_TO_P field of the LL_PHY_UPDATE_IND PDU shall be determined from the RX_PHYS field of the Peripheral’s PDU;
- the PHY_P_TO_C field of the LL_PHY_UPDATE_IND PDU shall be determined from the TX_PHYS field of the Peripheral’s PDU.

In each of those cases the following rules apply:

- if, for at least one PHY, the PHY is one that the Central prefers to use and the corresponding bit is set to 1 in the relevant field of the Peripheral’s PDU, the Central shall select any one of those PHYs for that direction;
- if there is no PHY which the Central prefers to use and for which the corresponding bit is set to 1 in the relevant field of the Peripheral’s PDU, the Central shall not change the PHY for that direction.

The remainder of this section shall apply irrespective of which device initiated the procedure.

Irrespective of the above rules, the Central may leave both directions unchanged. If the Peripheral specified a single PHY in both the TX_PHYS and RX_PHYS fields and both fields are the same, the Central shall either select the PHY specified by the Peripheral for both directions or shall leave both directions unchanged.

If either PHY will change, Section 5.5 shall apply to the LL_PHY_UPDATE_IND PDU. Both devices shall use the new PHYs starting at the instant.

The procedure has completed when:

- an LL_REJECT_EXT_IND PDU has been sent or received;
- an LL_PHY_UPDATE_IND PDU indicating that neither PHY will change has been sent or received; or
- the Central sends an LL_PHY_UPDATE_IND PDU indicating that at least one PHY will change and the instant has been reached. In this case, the procedure response timeout shall be stopped on the Central when it sends that PDU and on the Peripheral when it receives that PDU.

If the Peripheral receives an LL_PHY_UPDATE_IND where either PHY field specifies a PHY that the Peripheral does not support, has a bit set that is reserved for future use, or has more than one bit set, the Peripheral shall not change the PHY in that direction.

The Controller shall notify the Host of the PHYs now in effect when the PHY Update procedure completes if either it has resulted in a change of one or both PHYs or if the procedure was initiated by a request from the Host. Otherwise, it shall not notify the Host that the procedure took place.

The Link Layer can reject a proposed change to either PHY (by not setting the corresponding bit in its response) because, for example, a PHY with a lower bit rate could not be scheduled among other activities or because the requested PHY does not support Constant Tone Extensions. Such a rejection could, however, result in link loss if the change was requested to improve reliability.

#### 5.1.10.1 Packet transmit restrictions

For each row of Table 5.1, on the Link Layer(s) in the role(s) listed in the first column and during the period starting with the event described in the second column and ending with the event described in the third column, the Link Layer shall not transmit either of:

- any packet that would take longer than *connEffectiveMaxTxTime* microseconds (see Section 4.5.10) to transmit on any relevant PHY
- any packet with a Constant Tone Extension if any relevant PHY does not allow Constant Tone Extensions

where a “relevant PHY” is a PHY described in the fourth column.

| Role(s) | Starting event | Ending event | Relevant PHY(s) |
|---|---|---|---|
| Either | the Link Layer sends or receives an LL_PHY_UPDATE_-IND PDU that changes either PHY | the instant | the PHY that will apply after the instant |
| Peripheral | the Link Layer sends an LL_PHY_REQ PDU | the Link Layer receives an LL_PHY_UPDATE_IND, LL_UNKNOWN_RSP, or LL_REJECT_EXT_IND PDU in response | any PHY that appears in the TX_PHYS field of that LL_PHY_REQ PDU |
| Peripheral | the Link Layer sends an LL_PHY_RSP PDU | the Link Layer receives the LL_PHY_UPDATE_IND PDU in response | any PHY that appears in both the TX_PHYS field of the Peripheral's LL_PHY_RSP PDU and the RX_PHYS field of the Central's LL_PHY_REQ PDU |

*Table 5.1: PHY update packet transmit restrictions*

If the value of *connEffectiveMaxTxTime* changes during the procedure (for example, if the Data Length Update procedure is performed before the Instant is reached), the above requirements apply to the value at the moment the LL Data PDU is queued for transmission.

Note: The requirements of this section are in addition to, and do not override, those in Section 4.5.10.

If a Link Layer has any LL Data PDUs queued for transmission at the start of the procedure or queues any during the procedure, it might need to re-fragment those PDUs in order to obey the requirements in this section. This can be necessary if, for example, the transmit PHY changes from the LE 2M PHY to the LE 1M PHY and the value of *connEffectiveMaxTxTime* does not increase enough to compensate for the lower bit rate. If a Link Layer has any packets with a Constant Tone Extension queued for transmission at the start of the procedure or queues any during the procedure, then it might need to delay them, cancel them, or (if the packets contain an LL_CTE_RSP PDU) replace them by LL_REJECT_EXT_IND PDUs in order to obey the requirements in this section.

### 5.1.11 Minimum Number Of Used Channels procedure

A Peripheral's Controller may use the Minimum Number Of Used Channels procedure to request that the peer device uses a minimum number of channels on a given PHY.

The Peripheral initiates this procedure by sending an LL_MIN_USED_CHANNELS_IND PDU. The Central shall not send this PDU.

If the Link Layer receives an LL_MIN_USED_CHANNELS_IND PDU, it should ensure that, whenever the Peripheral-to-Central PHY is one of those specified, the connection uses at least the number of channels given in the MinUsedChannels field of the PDU.

The procedure has completed when the Link Layer acknowledgment of the LL_MIN_USED_CHANNELS_IND PDU is sent or received.

If the channel map does not include the minimum number of channels the Peripheral requires for regulatory compliance, the Peripheral must take steps to remain regulatory compliant, which can include disconnecting the link or reducing the output power.

### 5.1.12 Constant Tone Extension Request procedure

The Central or Peripheral may use the Constant Tone Extension Request procedure, when supported, to request the remote Link Layer to send a packet containing an LL_CTE_RSP PDU and a Constant Tone Extension (see Section 2.5.3).

Either the Central or the Peripheral Link Layer initiates this procedure by sending an LL_CTE_REQ PDU.

If:

- Constant Tone Extension responses are enabled;
- the current transmitter PHY is one that allows Constant Tone Extensions;
- Section 5.1.10.1 would not prohibit the response from being transmitted;
- the length of Constant Tone Extension requested in the LL_CTE_REQ PDU is not greater than the maximum length the responding device supports; and
- the responding device is currently configured to respond with the type of Constant Tone Extension requested in the LL_CTE_REQ PDU;

then the remote Link Layer shall respond with an LL_CTE_RSP PDU that includes a Constant Tone Extension of the requested type and whose length is greater than or equal to that requested. Otherwise the remote Link Layer shall respond with an LL_REJECT_EXT_IND PDU. If Constant Tone Extension responses are disabled or the Link Layer is not currently configured to respond with the type and length of the requested Constant Tone Extension, the ErrorCode shall be set to *Unsupported LMP Parameter Value/Unsupported LL Parameter Value* (0x20). If Constant Tone Extensions are not allowed on the current transmitter PHY, the ErrorCode shall be set to *Invalid LMP Parameters/Invalid LL Parameters* (0x1E). If Section 5.1.10.1 would prohibit the response from being transmitted, the ErrorCode shall be set to *Different Transaction Collision* (0x2A).

IQ sampling (see Section 2.5.4) shall be performed while receiving the Constant Tone Extension of the LL_CTE_RSP PDU and the sampling results shall be reported to the Host.

The procedure has completed when either an LL_CTE_RSP PDU has been sent or received or an LL_REJECT_EXT_IND PDU is sent or received with the RejectOpcode set to LL_CTE_REQ.

### 5.1.13 Periodic Advertising Sync Transfer procedure

A Controller may use the Periodic Advertising Sync Transfer procedure to transfer, to a connected peer device, the synchronization information necessary to synchronize to a periodic advertising train (see Section 4.4.3.4).

Either the Central or the Peripheral Link Layer initiates this procedure by sending an LL_PERIODIC_SYNC_IND PDU or an LL_PERIODIC_SYNC_WR_IND PDU. The PDU used is determined by the type of the periodic advertising used. For PAwR, the LL_PERIODIC_SYNC_WR_IND PDU shall be used, otherwise the LL_PERIODIC_SYNC_IND PDU shall be used. If the LL_PERIODIC_SYNC_WR_IND PDU is used, then the RspAA shall be set to an Access Address value as specified in Section 2.1.2.

If the Host has enabled receipt of transfers and the Link Layer receives an LL_PERIODIC_SYNC_IND PDU or an LL_PERIODIC_SYNC_WR_IND PDU that describes a periodic advertising train that the Link Layer is neither already synchronized with nor in the process of synchronizing with, the Link Layer shall synchronize with the described periodic advertising train and then notify the Host; it shall also notify the Host if synchronization fails. However, if the PHY field of the LL_PERIODIC_SYNC_IND PDU or the LL_PERIODIC_SYNC_WR_IND PDU has no bits or more than one bit set, or the bit set corresponds to a PHY that the recipient does not support or is reserved for future use, the recipient shall ignore the PDU.

The procedure has completed when an LL_PERIODIC_SYNC_IND PDU or an LL_PERIODIC_SYNC_WR_IND PDU has been sent or received.

#### 5.1.13.1 Timing considerations

This section explains the issues in handling the relative drifts of three separate device clocks and does not create any requirements. This section does not apply when devices A and B are the same because there is then no clock drift between the two.

In general there are three devices involved in the Periodic Advertising Sync Transfer procedure: the periodic advertiser A, the initiating device B, and the receiving device C. Each of these can have a different sleep clock accuracy. Therefore device C needs to carry out various steps to determine the timing and required receive window for synchronizing to the periodic advertising.

In the following formulae and in Figure 5.4:

- *PEa* is the value of *paEventCounter* stored in the SyncInfo within the LL_PERIODIC_SYNC_IND PDU.
- *PEb* is the value of *paEventCounter* for a recent AUX_SYNC_IND PDU that device B has received; this value is stored in the *lastPaEventCounter* field of the LL_PERIODIC_SYNC_IND PDU.
- *PEc* is the value of *paEventCounter* for the AUX_SYNC_IND PDU that device C is attempting to receive.

  Note: *PEc* can be before, after, or the same as *PEa*.
- *PAI* is the periodic advertising interval as represented by the *Interval* field of the SyncInfo within the LL_PERIODIC_SYNC_IND PDU.
- *CEs* is the value of *connEventCounter* for the connection event when devices B and C synchronized their anchor points and that device B used to determine the contents of the LL_PERIODIC_SYNC_IND PDU; this value is stored in the *syncConnEventCount* field of the PDU.
- *CEt* is the value of *connEventCounter* for the connection event when the LL_PERIODIC_SYNC_IND PDU was (re)transmitted by device B and received by device C.
- *CEref* is the value of *connEventCount* in the LL_PERIODIC_SYNC_IND PDU.
- *CEc* is a connection event before the AUX_SYNC_IND PDU that device C is attempting to receive.
- *CI* is the connection interval for the connection between devices B and C.
- *Offset* is the value represented by the syncPacketWindowOffset value within the LL_PERIODIC_SYNC_IND PDU.
- *Target* is the time from the anchor point of connection event *CEc* to the AUX_SYNC_IND PDU that device C is attempting to receive.
- *CAa*, *CAb*, and *CAc* are the clock accuracies of devices A, B, and C respectively. *CAa* is stored in the *SCA* field of the SyncInfo within the LL_PERIODIC_SYNC_IND PDU and *CAb* is stored in the *SCA* field of the LL_PERIODIC_SYNC_IND PDU.

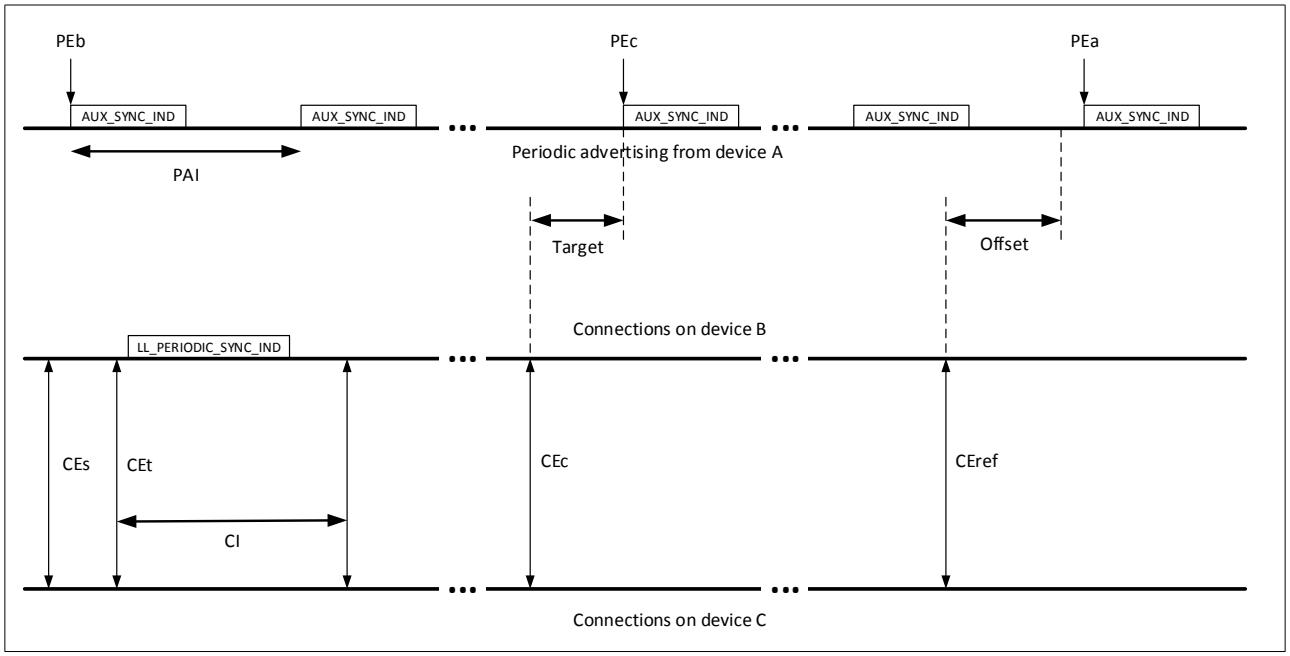



*Figure 5.4: Periodic Advertising Sync Transfer procedure timings*

If all three devices had a perfectly accurate clock, then:

$$Tnominal \leq Target < Tnominal + U$$

where:

$$Tnominal = (CEref - CEc) \times CI + Offset - (PEa - PEc) \times PAI$$

$U$ = 30 µs or 300 µs depending on the value of the Offset Units field of the SyncInfo

Because of clock drift and jitter, this becomes:

$$Tnominal - D - 16\ \mu s \leq Target < Tnominal + U + D + 16\ \mu s$$

where:

$$D = (Da + Db) \times (1 + CAa + CAb + CAc)$$

*Da* represents the drift of the periodic advertising and *Db* the drift of B's clock between *CEs* and *PEb*:

$$Da = |PEc - PEb| \times PAI \times (CAa + CAc)$$

$$Db = |CEt - CEs| \times CI \times (CAb + CAc)$$

D should be rounded to the next whole microsecond above.

These values are derived from the assumption that device B will have calculated *Offset* without making any allowance for drift in its own clock. So if *PEb* occurred at time *TPEb* and *CEs* at time *TCEs*, both according to its own clock, device B will have chosen values for *PEa* and *CEref* and then calculated:

$$Offset = (TPEb + (PEa - PEb) \times PAI) - (TCEs + (CEref - CEs) \times CI)$$

When device C calculates *Tnominal*, it again does so without allowing for drift. So, by substituting the above expression for *Offset* in that for *Tnominal* above, we find that:

***Tnominal***

$= (CEref - CEc) \times CI + (TPEb + (PEa - PEb) \times PAI) - (TCEs + (CEref - CEs) \times CI) - (PEa - PEc) \times PAI$

$= TPEb + (PEa - PEb) \times PAI - (PEa - PEc) \times PAI - TCEs - (CEref - CEs) \times CI + (CEref - CEc) \times CI$

$= TPEb + (PEc - PEb) \times PAI - TCEs - (CEc - CEs) \times CI$

$= (PEc - PEb) \times \mathrm{PAI} + (TPEb - TCEs) - (CEc - CEs) \times CI$

Each of these three terms is based on a different clock:

- The first term depends on *PAI*, which is measured using device A's clock. However, since device C is calculating *Tnominal*, device C's clock must be allowed for as well. The drift in this term is represented by *Da*.
- The expression (*TPEb* – *TCEs*) is the change in B's clock between the two events and so is measured using that clock; again, it is necessary to allow for device C's clock as well. The drift in this term is represented by *Db;* the term in parentheses in the expression for *Db* is an upper bound for the difference.
- The last term is on device C's clock and therefore does not require any adjustment.

The second factor in the formula for *D* allows for second-order effects: if the relevant periodic events are 12 seconds apart, a 500 ppm drift will result in a $20 \times 0.0005^2 = 3$ µs residual error in the required window.

If device C allows for clock drift when making these calculations, it may be able to reduce the window it uses accordingly. If device B allows for clock drift (e.g. by measuring the drift in the periodic advertising timing), the actual drift will be less than that calculated above but device C will not be aware of this. A more detailed analysis based on the exact techniques used in the implementation may allow device C to compute a narrower window.

### 5.1.14 Sleep Clock Accuracy Update procedure

The Link Layer of the Central or Peripheral may inform its peer of a change to its sleep clock accuracy (*centralSCA* or *peripheralSCA* – see Section 4.2.2), or may query the

peer's sleep clock accuracy, by sending an LL_CLOCK_ACCURACY_REQ PDU. The procedure may be initiated either on a request by the Host or autonomously by the Link Layer. A device should not initiate this procedure more than once per second.

Where the Link Layer will specify a more accurate clock than that currently in use, it shall switch to that clock before initiating or responding to this procedure.

When the Link Layer receives an LL_CLOCK_ACCURACY_REQ PDU, it shall send an LL_CLOCK_ACCURACY_RSP PDU in reply. The sleep clock accuracy specified in the reply shall not be less than the value in use when the request was received on the Link Layer sending that reply.

Note: The Link Layer can delay its response in order to make any necessary internal adjustments to the accuracy change, but not for so long as to trigger a timeout. To change to a less accurate clock, the responding device, as this section requires, must initiate this procedure separately.

The procedure has completed when the LL_CLOCK_ACCURACY_RSP is sent or received. Where the initiating Link Layer has specified a less accurate clock than that currently in use, it shall not switch to that clock until it has received the LL_CLOCK_ACCURACY_RSP PDU in reply. If it receives an LL_UNKNOWN_RSP PDU, it shall maintain at least the accuracy specified in the CONNECT_IND or AUX_CONNECT_REQ PDU that created the connection.

### 5.1.15 Connected Isochronous Stream Creation procedure

The Central Link Layer may use the Connected Isochronous Stream Creation procedure to create a CIS between a Central and a Peripheral. The Central Link Layer initiates this procedure by sending an LL_CIS_REQ PDU. The Peripheral Link Layer shall not initiate this procedure. The Central's Link Layer shall only create a CIS when requested by the Host, only using a CIS_ID that the Host has already stored a configuration for in this CIG, and not using a CIS_ID that corresponds to an existing CIS in the CIG (see Section 4.5.14.3). The Central shall not initiate this procedure if the Connected Isochronous Stream (Host Support) feature bit is not set in its Controller.

When the Peripheral's Link Layer receives the LL_CIS_REQ PDU, it shall either reject the proposed CIS immediately or notify the Host. In the latter case, the Host requests the Link Layer to either accept or reject the proposed CIS. If the Connected Isochronous Stream (Host Support) feature bit is not set in the local Link Layer, then the Peripheral shall reject the proposed CIS with the error code *Unsupported Remote Feature* (0x1A). If either PHY field of the LL_CIS_REQ PDU has no bits or more than one bit set, or if the bit set corresponds to a PHY that the recipient does not support or is reserved for future use, then the Peripheral shall reject the proposed CIS. If the Peripheral does not support the BN, FT, and NSE values in the LL_CIS_REQ PDU, it shall reject the proposed CIS with the error code *Parameter Out of Mandatory Range* (0x30). If the

Framing_Mode field of the LL_CIS_REQ PDU specifies a mode that the Peripheral does not support, then the Peripheral shall reject the proposed CIS. If the Peripheral rejects the proposed CIS, then it shall send an LL_REJECT_EXT_IND PDU with the appropriate reason code. If it accepts the CIS, then it shall send an LL_CIS_RSP PDU.

When the Central's Link Layer receives an LL_CIS_RSP PDU, it shall either create the CIS by replying with an LL_CIS_IND PDU or shall cancel it by replying with an LL_REJECT_EXT_IND PDU with the appropriate reason code. The Central shall not cancel the CIS if the proposed timings are within those specified in the LL_CIS_REQ PDU unless the Host requested that the CIS be terminated.

When the Central sends and the Peripheral receives the LL_CIS_IND PDU, both devices shall stop the procedure response timeout timer, create the CIS, and start transmitting and receiving CIS PDUs. The first anchor point of the CIS, with *cisEventCounter* equal to zero, shall be at the moment specified in the LL_CIS_IND PDU. The CIS is considered established by each Link Layer when that Link Layer has received a CIS PDU from the peer that is part of the CIS.

The Central's Link Layer shall calculate CIG_Sync_Delay and CIS_Sync_Delay for each CIS and send them to the Peripheral in the LL_CIS_IND PDU.

If either Link Layer sends or receives an LL_REJECT_EXT_IND PDU, it shall terminate the procedure immediately and not create the CIS. The CIS configuration nevertheless remains stored within the CIG and the corresponding CIS can be created later. The procedure has completed on each Link Layer when the CIS is established or when an LL_REJECT_EXT_IND PDU has been sent or received. Each Link Layer shall notify its Host when the procedure has completed.

The values of the CIS_Offset_Min, CIS_Offset_Max, and connEventCount fields of the LL_CIS_REQ and LL_CIS_RSP PDUs each specify a window for the CIS anchor point. The window specified by the LL_CIS_RSP PDU shall lie entirely within a window equivalent to that specified by the LL_CIS_REQ PDU. The first CIS anchor point shall lie within a window equivalent to that specified by the LL_CIS_RSP PDU. For this purpose, two windows are equivalent if they have the same width and the difference between their start times is an integer multiple of ISO_Interval for the CIS. The edges of each window are part of the window so, for example, the two windows can be identical.

Section 5.5 shall apply to the LL_CIS_IND PDU as if the *connEventCount* field was an Instant.

The Link Layer should schedule the CIS so that the CIS events do not overlap with the connection events on the associated ACL.

### 5.1.16 Connected Isochronous Stream Termination procedure

The Central or Peripheral may use this procedure for voluntary termination of a CIS while in the Connection state. Voluntary termination occurs when the Host requests the Link Layer to terminate the connection. The Link Layer initiates this procedure by sending an LL_CIS_TERMINATE_IND PDU.

The Link Layer shall not transmit or receive on the CIS after it has received or queued for transmission the LL_CIS_TERMINATE_IND PDU.

The procedure has completed when the Link Layer acknowledgment has been sent or received.

Note: Terminating a CIS does not affect the associated ACL.

### 5.1.17 Power Control Request procedure

The Link Layer of the Central or Peripheral may use the Power Control Request procedure, when supported, to request a remote Controller to adjust its transmit power level on a specified PHY by a given amount. Power Control requests carried over an LE-ACL logical link only affect the power level used on that link and any associated physical link(s) such as isochronous physical links and CS physical links; they do not affect the power level used on the physical links to other connected and unconnected devices.

The Link Layer initiates this procedure by sending an LL_POWER_CONTROL_REQ PDU.

The Link Layer can query the current transmit power level and acceptable power reduction of the remote Controller by sending an LL_POWER_CONTROL_REQ PDU with Delta set to zero.

The responding Link Layer shall make the requested change to its transmit power level unless that would take it above the maximum or below the minimum supported power levels or unless it is not currently managing power levels on the requested PHY. If the requested change would take it above the maximum, it shall change the power level to the maximum supported. Otherwise, if it is unable to make exactly the requested change, it shall change the power level to the lowest available level greater than the requested level. In any case, it shall then reply with an LL_POWER_CONTROL_RSP PDU whose contents indicate the actual change made, if any, or that it is not managing the power level for the requested PHY (by setting the power level to 126).

Note: A request to make a small decrease in the power level can result in the power level not changing.

Note: A request with delta equal to zero on a PHY that is not an active PHY can indicate that the sender is about to make that PHY an active PHY.

The Link Layer shall not initiate the Power Control Request procedure until any outstanding CS Start procedure, as described in Section 5.1.26, has completed. If the LL_POWER_CONTROL_REQ PDU is received in this case, then the receiving Link Layer shall respond with an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Controller Busy* (0x3A).

If the Link Layer has sent an LL_POWER_CONTROL_REQ PDU and not yet received a response, or has an LL_POWER_CONTROL_REQ PDU queued for transmission, and then receives an LL_POWER_CONTROL_REQ PDU from the same peer device, it shall set the APR field to 0xFF in its response to that PDU. A responding Link Layer may also set the APR field to 0xFF when it is not managing the power level of the requested PHY, if it does not have a valid value to report, or if it does not support this field. Otherwise the responding Link Layer shall set the APR field as described in Section 5.1.17.1.

The new power level for the PHY shall take effect before the response is sent.

Note: The Link Layer can request the remote Link Layer to change the preferred transmit power level for a different PHY, for example before initiating a PHY Update procedure or creating an associated CIS on a PHY different from the one used for the ACL. To do this, it queries the remote Link Layer for the transmit power level that it would use for that PHY and then requests a change to the transmit power level.

If the PHY in the LL_POWER_CONTROL_REQ PDU is not supported by the remote Link Layer in its transmit direction or the PHY field contains no set bits, more than one set bit, or a bit set that is reserved for future use, the remote Link Layer shall respond with an LL_REJECT_EXT_IND PDU with the ErrorCode set to *Unsupported LMP Parameter Value/Unsupported LL Parameter Value* (0x20).

If the TxPower in the LL_POWER_CONTROL_REQ PDU is set to 126, the remote Link Layer shall respond with an LL_REJECT_EXT_IND_PDU with the ErrorCode set to *Invalid LL Parameters* (0x1E).

The procedure has completed when either an LL_POWER_CONTROL_RSP PDU has been sent or received or an LL_REJECT_EXT_IND PDU has been sent or received with the RejectOpcode field set to LL_POWER_CONTROL_REQ.

#### 5.1.17.1 Acceptable power reduction

A radio receiver may have a “golden range” of RSSI that it prefers the incoming signal to remain within. A device with such a receiver can use the Power Control Request procedure to bring the current RSSI ($RSSI_{curr}$) of the incoming signal to a preferred

value within its golden range. Nevertheless, it may still be able to receive the signal at a level that is equal to or above a minimum acceptable RSSI ($RSSI_{min}$) that is lower than the current RSSI. A device can use the Power Control Request procedure to check whether its peer can accept such a reduction in power and, if so, adjust its transmit power based on the response.

When a device sends an LL_POWER_CONTROL_RSP PDU, it should set the APR field to the value given by the following equation:

$$APR = \begin{cases} 0, & if\ RSSI_{curr} \leq RSSI_{min} \\ RSSI_{curr} - RSSI_{min}, & if\ RSSI_{curr} > RSSI_{min} \end{cases}$$

When a device reports an APR value to the peer device other than zero or 0xFF, it should wait at least two connection intervals to see if the peer device has made use of it to change its local transmit power before initiating a new Power Control Request procedure to request a remote transmit power level change. A device can determine if the peer device has changed its transmit power by sending an LL_POWER_CONTROL_REQ PDU with Delta set to zero, by looking for changes in the RSSI of incoming packets, or by the receipt of an LL_POWER_CHANGE_IND PDU.

When a device receives an APR value from the peer device other than 0xFF, the time period for which that value is considered to remain valid is implementation-specific; for example, the Link Layer can examine changes in received RSSI and remote transmit power level since the APR value was received. When a device receives an APR value other than 0xFF from the peer device, it shall not reduce its power level more than the value specified. When the device receives an APR value of 0xFF, it may choose to ignore any previous APR values.

### 5.1.18 Power Change Indication procedure

The Link Layer uses the Power Change Indication procedure, when supported, to notify the remote Link Layer of transmit power changes.

After the peer has sent at least one LL_POWER_CONTROL_REQ PDU, a Link Layer shall send an autonomous notification consisting of an LL_POWER_CHANGE_IND PDU each time that any of the following happens:

- It changes the power level autonomously on any PHY that it is managing power levels for.
- It changes the maximum power level on its current transmit PHY to the current power level.
- It starts managing the power level for a PHY.
- It stops managing the power level for a PHY.

A Link Layer shall not send the LL_POWER_CHANGE_IND PDU in any other circumstance. If two or more of the above situations occur for the same PHY at the same time or within a connection interval, it may combine the reports into a single LL_POWER_CHANGE_IND PDU. A Link Layer should not perform autonomous power level updates more than once per second to avoid sending too many LL_POWER_CHANGE_IND PDUs to the remote device.

If the notification is for the current ACL PHY, it shall be sent at the power level specified in the notification.

The procedure has completed when the LL_POWER_CHANGE_IND PDU has been sent or received.

The recipient shall ignore all bits of the PHY field of the LL_POWER_CHANGE_IND PDU that correspond to PHYs that it does not support or are reserved for future use. If the PHY field has no bits set or if every bit set corresponds to a PHY that the recipient does not support or is reserved for future use, the recipient shall ignore the PDU.

### 5.1.19 Connection Subrate Update procedure

The Central's Link Layer may update the subrate parameters (*connSubrateBaseEvent*, *connSubrateFactor*, and *connContinuationNumber*), *connPeripheralLatency*, and *connSupervisionTimeout* of a connection by sending an LL_SUBRATE_IND PDU. The Peripheral shall not send this PDU. The Central shall only initiate this procedure when requested by the Host, when requested by the Peripheral via the Connection Subrate Request procedure, or as recommended in Section 5.5. The Central shall not initiate this procedure while a Connection Parameters Request procedure is in progress. The Central shall not initiate this procedure until it has performed a Feature Exchange procedure (see Section 5.1.4) to determine that the Connection Subrating (Host Support) bit is set in the Peripheral's FeatureSet. In addition, the Central shall not initiate this procedure while any CS procedures are in progress (as described in Section 4.5.18.1). The Central shall not initiate this procedure while a Connection Rate Request procedure or a Connection Rate Update procedure is in progress.

After the Central transmits this PDU for the first time during a Connection Subrate Update procedure, it shall enter subrate transition mode. The Central leaves subrate transition mode when it receives the Link Layer acknowledgment for the PDU and then uses the new subrate base event, subrate factor, continuation number, Peripheral latency, and supervision timeout.

During subrate transition mode, the Central shall retransmit the PDU on all connection events which are subrated connection events based on the old subrate base event and subrate factor or are subrated connection events based on the new subrate base event and subrate factor (ignoring Peripheral latency). It shall continue to use the old supervision timeout. If the new continuation number is non-zero then the Central

may also transmit on other connection events. The Central's Link Layer may take the Peripheral's opportunities for reception into account when determining which connection events it chooses to transmit on while in subrate transition mode.

When the Peripheral receives this PDU it shall immediately switch to the new subrate base event, subrate factor, continuation number, Peripheral latency, and supervision timeout.

For example, as shown in Figure 5.5, if the old subrate base event is 32, the old subrate factor is 5, the new subrate base event is 38, the new subrate factor is 3, and the Central transmitted the LL_SUBRATE_IND PDU on connection event 42, then the old parameters will result in the devices using connection events 37, 42, 47, 52, 57 etc. while the new parameters will result in the devices using connection events 44, 47, 50, 53, 56, etc. Therefore, while it is in subrate transition mode, the Central will use connection events 42, 44, 47, 50, 52, 53, 56, 57, and so on until it receives the acknowledgment.

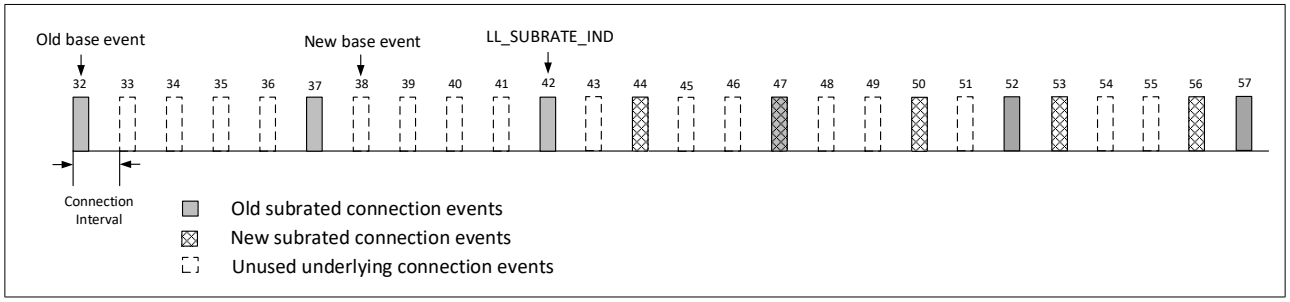



*Figure 5.5: Connection events used during subrate transition mode*

The Central shall set the value of SubrateBaseEvent ("S") in the PDU so that $S_{15\text{-}14} = E_{15\text{-}14}$, where E is the value of *connEventCount* for the connection event when the PDU is first queued for transmission. If the Peripheral receives an LL_SUBRATE_IND PDU with $S_{15\text{-}14}$ = 0b11 in a connection event with $connEventCounter_{15\text{-}14}$ = 0b00 then, immediately after updating the parameters, it shall change *connSubrateBaseEvent* as described in Section 4.5.1 as if the value of *connEventCount* had just wrapped.

The requirement on the value of $S_{15\text{-}14}$ means the Peripheral can determine whether the Central intended to send the PDU before or after *connEventCount* wrapped. For example, consider the situation where the current *connSubrateFactor* is 10 and the current *connSubrateBaseEvent* is 12002. As the point of wrap approaches, these parameters mean that the connection events used are 65522, 65532, 6, 16, 26, etc. If the LL_SUBRATE_IND PDU contained those values for these two parameters then, if the PDU was queued, sent, and received on event 65532, it indicated that the same connection events are to be used; on the other hand, if the PDU was queued, sent, and received on event 6, then it implies a change of phase so that connection events 12, 22, 32, etc. are to be used instead. Since internal queues and retransmissions can

mean that a packet queued for event 65532 could be received in event 6, the Peripheral needs to be able to distinguish these two cases, which it does by making the test described.

The procedure has completed when the Link Layer acknowledgment for the LL_SUBRATE_IND PDU has been sent or received.

The Peripheral shall accept an LL_SUBRATE_IND PDU. However, if the Peripheral's Host would prefer a different subrate factor it may, after this procedure has completed, initiate the Connection Subrate Request procedure or the Connection Parameters Request procedure to change the connection parameters.

### 5.1.20 Connection Subrate Request procedure

The Peripheral's Link Layer may request the Central to update the subrate factor, Peripheral latency, continuation number, and supervision timeout by sending an LL_SUBRATE_REQ PDU. The Central shall not send this PDU. The Peripheral shall only initiate this procedure when requested by the Host.

A device shall not initiate this procedure while any CS procedures are in progress (as described in Section 4.5.18.1). A device shall not initiate this procedure while a Connection Rate Request procedure or a Connection Rate Update procedure is in progress.

When the Central receives this PDU it shall either initiate the Connection Subrate Update procedure or shall respond with an LL_REJECT_EXT_IND PDU to reject the request. If the Central's Host has not set the Connection Subrating (Host Support) bit in the FeatureSet, the Central's Link Layer shall reject the request. If the Central's Host has provided acceptable subrate parameters for requests from the Peripheral, then the Central's Link Layer shall initiate the Connection Subrate Update procedure if and only if all of the following are true ("acceptable" indicates values provided by the Host, see [Vol 4] Part E, Section 7.8.123 and [Vol 4] Part E, Section 7.8.124; "requested" indicates values in the LL_SUBRATE_REQ PDU):

- $Max_Latency_{requested} \leq Max_Latency_{acceptable}$,
- $Timeout_{requested} \leq Supervision_Timeout_{acceptable}$,
- $SubrateFactorMax_{requested} \geq Subrate_Min_{acceptable}$,
- $SubrateFactorMin_{requested} \leq Subrate_Max_{acceptable}$,
- $(connInterval_{current} \times SubrateFactorMin_{requested} \times (Max_Latency_{requested} + 1)) \times 2 < Timeout_{requested}$

If the Central accepts the Peripheral's request, then:

- the new *connSubrateFactor* shall be between $Subrate_Min_{acceptable}$ and $Subrate_Max_{acceptable}$ and shall also be between $SubrateFactorMin_{requested}$ and $SubrateFactorMax_{requested}$,
- the new *connContinuationNumber* shall equal $min(max(Continuation_Number_{acceptable}, ContinuationNumber_{requested}), (new\ connSubrateFactor) - 1)$,
- the new *connPeripheralLatency* shall be less than or equal to $min(Max_Latency_{requested}, Max_Latency_{acceptable})$,
- the new *connSupervisionTimeout* shall equal $min(Timeout_{requested}, Supervision_Timeout_{acceptable})$.

The procedure has completed when the resulting Connection Subrate Update procedure has completed or an LL_REJECT_EXT_IND PDU has been sent or received.

### 5.1.21 Channel Classification Enable procedure

A Controller uses the Channel Classification Enable procedure to enable or disable reporting of channel classification information on the peer device. Until the Central initiates this procedure, reporting shall be disabled.

The Central can initiate this procedure at any time after entering the Connection state by sending an LL_CHANNEL_REPORTING_IND PDU. The Peripheral shall not send this PDU.

When a Peripheral that supports the Channel Classification feature receives this PDU, it shall enable or disable reporting of channel classification information to the Central as specified in the PDU. When reporting is enabled, the Peripheral should send channel classification information by initiating the Channel Classification Reporting procedure (see Section 5.1.22). When reporting is disabled, the Peripheral shall not send channel classification information.

The procedure has completed when the Link Layer acknowledgment of the LL_CHANNEL_REPORTING_IND PDU is sent or received.

### 5.1.22 Channel Classification Reporting procedure

A Controller uses the Channel Classification Reporting procedure to report channel classification information to the peer device.

The Peripheral may initiate this procedure by sending an LL_CHANNEL_STATUS_IND PDU after channel classification reporting has been enabled by the Central. The Central shall not send this PDU.

If channel classification information has not changed since the last time the Peripheral reported the information to the Central, then the Peripheral shall not initiate this procedure. Otherwise, the Peripheral shall initiate this procedure within, or in the first subrated connection event after, the maximum reporting delay from determining that a change in channel classification has occurred or from channel classification reporting being enabled, whichever is later. Two consecutive channel classification reports shall be spaced apart by a duration that is greater than or equal to the minimum reporting spacing.

The procedure has completed when the Link Layer acknowledgment of the LL_CHANNEL_STATUS_IND PDU is sent or received.

### 5.1.23 Channel Sounding Security Start procedure

The Link Layer, upon request from the Host, may enable ciphered bit stream generation for CS after the Encryption Start procedure has successfully completed as specified in Section 5.1.3.1, using the CS Security Start procedure.

To start or restart CS security, three parameters are exchanged: the CS_IV, CS_IN, and the CS_PV. Each value is composed of two parts: a Central part and a Peripheral part. Both parts are exchanged in the LL_CS_SEC_REQ and LL_CS_SEC_RSP PDUs. After these parameters are exchanged, CS security is also started.

To start CS security, the Link Layer of the Central shall generate the Central's part of the CS initialization vector (CS_IV_C), instantiation nonce (CS_IN_C), and personalization vector (CS_PV_C). CS_IV_C shall be a 64-bit random number and CS_IN_C shall be a 32-bit random number. Both shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2.

The CS_PV_C shall be a 64-bit value. There are no requirements on the content of the personalization vector, and it is not considered a critical security parameter. The intent of this value is to introduce additional input into the DRBG instantiation function, as described in [Vol 6] Part E, Section 3.1.5. The personalization vector may be generated from a cryptographic module or from other pseudo-random sources.

The Link Layer of the Central initiates the CS Security Start procedure by sending an LL_CS_SEC_REQ PDU to the Peripheral. Before transmitting the LL_CS_SEC_REQ PDU, the Link Layer of the Central shall have completed all outstanding CS procedures, including CS procedure repeats, associated with the ACL. While the CS Security Start procedure is in progress, the local Link Layer shall reject the CS Start Procedure invoked from either the local or remote Host until the CS Security Start procedure has completed.

The Central or Peripheral shall not enable the CS Security Start procedure if the Channel Sounding (Host Support) feature bit is not set in the Controller. If the remote

Link Layer sends an LL_CS_SEC_REQ PDU when the Channel Sounding (Host Support) feature bit is not set in the local Link Layer, then the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

The Central or Peripheral shall not enable a CS Security Start procedure if the Encryption Start procedure has not successfully completed, as specified in Section 5.1.3.1. If the remote Link Layer sends an LL_CS_SEC_REQ PDU without the Encryption Start procedure having successfully completed, the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Insufficient Security* (0x2F).

If the Encryption Start procedure has successfully completed, then when the Link Layer of the Peripheral receives an LL_CS_SEC_REQ PDU, it shall generate the Peripheral's part of the CS initialization vector (CS_IV_P), instantiation nonce (CS_IN_P), and personalization vector (CS_PV_P). CS_IV_P shall be a 64-bit random number and CS_IN_P shall be a 32-bit random number. Both shall be generated using the same requirements used for the generation of CS_IV_C and CS_IN_C. The CS_PV_P shall be a 64-bit value generated using the same requirements used for the generation of the CS_PV_C.

The Link Layer of the Peripheral shall then transmit these values back to the Central by sending an LL_CS_SEC_RSP PDU.

Each Link Layer shall combine the initialization vector, instantiation nonce, and personalization vector parts in the following manner:

CS_IV = CS_IV_P || CS_IV_C
CS_IN = CS_IN_P || CS_IN_C
CS_PV = CS_PV_P || CS_PV_C

The CS_IV_C is concatenated with the CS_IV_P. The least significant octet of CS_IV_C becomes the least significant octet of CS_IV. The most significant octet of CS_IV_P becomes the most significant octet of CS_IV.

The CS_IN_C is concatenated with the CS_IN_P. The least significant octet of CS_IN_C becomes the least significant octet of CS_IN. The most significant octet of CS_IN_P becomes the most significant octet of CS_IN.

The CS_PV_C is concatenated with the CS_PV_P. The least significant octet of CS_PV_C becomes the least significant octet of CS_PV. The most significant octet of CS_PV_P becomes the most significant octet of CS_PV.

The procedure has completed when the LL_CS_SEC_RSP PDU has been sent or received. On completion, the CSProcCount shall be initialized to 0.

### 5.1.24 Channel Sounding Capabilities Exchange procedure

The Link Layer parameters for CS capabilities information are exchanged before executing the CS Configuration procedure described in Section 5.1.25. The Link Layer of either the initiator or reflector can initiate the Channel Sounding Capabilities Exchange procedure by sending an LL_CS_CAPABILITIES_REQ PDU. This procedure should be used when requested by the Host. The procedure may also be initiated autonomously by the Link Layer. The Link Layer shall not allow the CS Capability Exchange procedure if the Channel Sounding (Host Support) feature bit is not set in the Controller. If the remote Link Layer sends an LL_CS_CAPABILITIES_REQ PDU when the Channel Sounding (Host Support) feature bit is not set in the local Link Layer, the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

CS capabilities of the peer device may be supplied by the Host if previously known, or cached by the Controller, including between connections. A Link Layer should not send an LL_CS_CAPABILITIES_REQ PDU on every connection if the information has been cached for this device. A Link Layer, however, may send an LL_CS_CAPABILITIES_REQ PDU to refresh this cached information. Cached information for a device from a previous connection may have changed and an implementation shall be able to accept an error response from a subsequent CS Link Layer control exchange if a capability is not supported or not used by the peer.

The Link Layer that receives an LL_CS_CAPABILITIES_REQ PDU shall respond with an LL_CS_CAPABILITIES_RSP PDU.

The peer device's capabilities shall be reported to the local Host on successful completion of this exchange.

The procedure has completed when either an LL_CS_CAPABILITIES_RSP PDU or the LL_REJECT_EXT_IND PDU has been sent or received.

### 5.1.25 Channel Sounding Configuration procedure

The Host may supply CS procedure configuration parameters and a previously associated configuration ID parameter if previously cached from a prior connection by that Host. Otherwise, the CS procedure configuration parameters shall be exchanged before starting a CS procedure. This exchange shall only occur after the peer device's CS capabilities are known, as described in Section 5.1.24. A configuration ID is selected by the Host of the Link Layer issuing the LL_CS_CONFIG_REQ PDU and is assigned to each CS parameter group. The ID assigned shall be unique among all created configuration parameter sets between two devices. The parameter exchange may be started by either side. CS configuration parameters may be exchanged by sending an LL_CS_CONFIG_REQ PDU. A Link Layer shall only begin the exchange of CS configuration parameters when requested by the Host. The Central or Peripheral

device shall not allow an exchange of CS configuration parameters if the Channel Sounding (Host Support) feature bit is not set in the Controller. If the remote Link Layer sends an LL_CS_CONFIG_REQ PDU when the Channel Sounding (Host Support) feature bit is not set in the local Link Layer, then the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

If the parameters received in an LL_CS_CONFIG_REQ PDU are not acceptable to that Link Layer, then it shall immediately reject the configuration parameter set with an LL_REJECT_EXT_IND PDU with the error code *Unsupported LL Parameter Value* (0x20). If the receiving Link Layer accepts the LL_CS_CONFIG_REQ PDU parameters, then it shall send an LL_CS_CONFIG_RSP PDU.

Several CS procedure configuration parameter sets may be supported concurrently between Link Layers and shall be identified by the Config_ID parameter. The value of the Config_ID parameter returned in the LL_CS_CONFIG_RSP PDU shall be the same as the value received in the LL_CS_CONFIG_REQ PDU.

Configuration parameter sets may be changed using the CS Configuration procedure, with the same configuration ID used to set up the configuration parameter set. A device shall not initiate this procedure while a CS procedure or CS procedure repeat instances, as described in Section 4.5.18.1, with the same configuration ID is in progress. A Link Layer receiving an LL_CS_CONFIG_REQ PDU while a CS procedure or CS procedure repeat instances using the same configuration ID is in progress shall immediately respond with an LL_REJECT_EXT_IND with the error code *Command Disallowed* (0x0C).

If an attempt to change or remove a configuration parameter set is rejected, then that configuration set and the associated configuration ID shall not be updated by both Link Layers. Otherwise, if an LL_CS_CONFIG_REQ Action field is set to removed, then that configuration set shall no longer be used for subsequent CS procedures. Either device may remove a configuration ID even if the peer device originally created that configuration ID. A previously removed configuration ID may be reused to establish a new configuration parameter set.

The procedure collision rules described in Section 5.3 apply so that the Central and the Peripheral cannot simultaneously create or update a configuration parameter set. These procedure collision rules may result in the Peripheral Link Layer receiving an LL_REJECT_EXT_IND PDU to allow the Central initiated procedure to complete.

The Link Layer transmitting the LL_CS_CONFIG_REQ PDU shall select either the initiator or reflector role for that configuration, as requested by the Host. The Link Layer responding with the LL_CS_CONFIG_RSP PDU is then in the other role for the duration of that configuration.

Devices may support various parameter values or ranges of values that are discovered during the CS Capabilities Exchange procedure, as described in Section 5.1.24. If the Link Layer that receives the LL_CS_CONFIG_REQ PDU does not support a suggested parameter selection, then it shall respond with an LL_REJECT_EXT_IND PDU with the error code *Unsupported LL Parameter Value* (0x20).

Table 5.2 describes how the values suggested in the LL_CS_CONFIG_REQ PDU shall take into account the various capabilities of the peer device. Values suggested in Table 5.2 shall also take into account the CS capabilities supported by the local device. Parameters with mandatory settings are always included in both the local and peer device settings as if they were included in either the LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU.

| Parameter | Content of the LL_CS_CONFIG_REQ PDU |
|---|---|
| Main_Mode | Shall be selected from one of the valid Main_Mode and Sub_Mode combinations described in [Vol 6] Part H, Table 4.11 that also uses one of the Main_Mode types included in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| Sub_Mode | Shall be selected from one of the valid Main_Mode and Sub_Mode combinations described in [Vol 6] Part H, Table 4.11 that also uses one of the Sub_Mode types included in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| CS_SYNC_PHY_Capability | Shall be selected from one of the CS_SYNC_PHY_Capability values included in both the local and peer device's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| RTT_Type | Shall be selected based on the supported RTT capabilities indicated in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| Role | Shall be selected to be compatible with what was included in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. Specifically, if the peer indicated support for the initiator role, then the reflector role may be selected; if the peer indicated support for the reflector role, then the initiator role may be selected. |
| IPT | Shall be set by default to not enabled and may be set to enabled if support for IPT was indicated in the peer's LL_CS_CAPABILITIES_REQ [v2] or LL_CS_CAPABILITIES_RSP [v2] PDU. |
| ChSel | Shall be set by default to Channel Selection Algorithm #3b and may be set to Channel Selection Algorithm #3c if support for this parameter was indicated in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| T_IP1 | Shall be selected from one of the valid values for T_IP1_Capability described in [Vol 6] Part H, Section 4.3.1 that is greater than or equal to the value contained in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |

| Parameter | Content of the LL_CS_CONFIG_REQ PDU |
|---|---|
| T_IP2 | Shall be selected from one of the valid values for T_IP2_Capability described in [Vol 6] Part H, Section 4.3.3 that is greater than or equal to the value contained in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| T_FCS | Shall be selected from one of the valid values for T_FCS_Capability described in Section 4.5.18.1 that is greater than or equal to the value contained in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |
| T_PM | Shall be selected from one of the valid values for T_PM_Capability described in [Vol 6] Part H, Section 4.3.3 that is greater than or equal to the value contained in the peer's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. |

*Table 5.2: Content of the LL_CS_CONFIG_REQ PDU as limited by the peer's LL_CS_CAPABILITIES content*

After the LL_CS_CONFIG_RSP PDU has been received, then the CS configuration associated with the configuration ID shall be considered available for use within the CS Start procedure, as described in Section 5.1.26.

The CS Start procedure has completed when an LL_CS_CONFIG_RSP PDU or the LL_REJECT_EXT_IND PDU has been sent or received. Each Link Layer shall notify its Host when the Configuration procedure has completed.

### 5.1.26 Channel Sounding Start procedure

CS procedures may be initiated by either the initiator or reflector. Initiators and reflectors may be in either the Central or Peripheral role. A CS Start procedure shall only be started by sending an LL_CS_REQ PDU, which may be sent any time after entering the Connection state, but only after the CS Security Start procedure has completed, the CS Capability Exchange procedure has completed or the capabilities are previously known, the CS Configuration procedure has completed or the configuration content is previously known, and either the mode-0 FAE table is previously known or the Channel Sounding Mode-0 FAE Table Request procedure has completed. A Link Layer shall only begin a CS Start procedure when configured to do so by the Host. The Central or Peripheral shall not allow a CS procedure to start if the Channel Sounding (Host Support) feature bit is not set in the Controller. If the remote Link Layer sends an LL_CS_REQ PDU when the Channel Sounding (Host Support) feature bit is not set in the local Link Layer, the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

The Link Layer shall not initiate the CS Start procedure until any outstanding CS Procedure Repeat Termination procedure, as described in Section 5.1.27, or Power Control Request procedure, as described in Section 5.1.17, or Connection Update procedure, as described in Section 5.1.1, or Connection Parameters Request

procedure, as described in Section 5.1.7, or Connection Subrate Update procedure, as described in Section 5.1.19, or Connection Subrate Request procedure, as described in Section 5.1.20, has completed. CS procedures are dependent on parameters selected during a previous Power Control Request procedure. After a CS procedure starts, including when an individual instance of a CS procedure starts within procedure repeat activity, the selected transmit power values relative to the values selected in a prior Power Control Request procedure shall remain unchanged for the duration of that procedure instance, regardless of later changes to the respective ACL power control parameters.

CS procedures are also dependent on parameters selected during the CS Configuration procedure, as described in Section 5.1.25, which are identified by the Config_ID identifier. If the CS configuration ID received during the CS Start procedure is not properly created, then the receiving Link Layer shall immediately respond with an LL_REJECT_EXT_IND PDU with the error code *Invalid LL Parameters* (0x1E).

If the Host has asked the Controller to enforce CS security requirements and the security requirements are not being met during the CS Start procedure, then the receiving Link Layer shall immediately respond with an LL_REJECT_EXT_IND PDU with the error code *Rejected due to Security Reasons* (0x0E).

If the receiving Link Layer is in the Peripheral role and accepts the parameters received in the LL_CS_REQ PDU or chooses to select alternative parameters, then it shall send an LL_CS_RSP PDU. If the parameters received in an LL_CS_REQ PDU are not acceptable to the receiving Link Layer (Central or Peripheral), then that Link Layer shall immediately reject the procedure by sending an LL_REJECT_EXT_IND PDU with the appropriate error code.

When a Link Layer in the Central role receives either an LL_CS_REQ PDU or an LL_CS_RSP PDU, it shall either prepare to start the CS procedure by replying with an LL_CS_IND PDU or it shall cancel the CS Start procedure by replying with an LL_REJECT_EXT_IND PDU with the appropriate error code.

When an LL_CS_IND PDU is sent by the Link Layer of the Central and received by the Link Layer of the Peripheral, both devices shall stop the procedure response timeout timer and start the CS procedure. The first CS subevent shall be anchored at the connection event specified in the LL_CS_IND PDU.

The LL_REJECT_EXT_IND PDU shall never be sent in response to an LL_CS_IND PDU. If either Link Layer sends or receives an LL_REJECT_EXT_IND PDU, that Link Layer shall terminate the CS Start procedure. The CS Start procedure has completed on each Link Layer when the LL_REJECT_EXT_IND PDU has been transmitted or received.

If a Link Layer agrees with suggested parameters received in the LL_CS_REQ PDU, then it shall reply with the same Config_ID, and with the same parameter or set of parameters that are within the suggested range in the LL_CS_RSP or LL_CS_IND PDU. Alternatively, the values of connEventCount, Offset_Min, Offset_Max, Event_Interval, Subevents_Per_Event, Subevent_Interval, Subevent_Len, and ACI may be re-suggested to better suit the internal scheduling and resource availability of the Link Layer replying with the LL_CS_RSP PDU. Because some of these values are related to Link Layer scheduling, changes to the values might not coincide with the scheduling constraints of the Link Layer that transmitted the LL_CS_REQ PDU and might cause that Link Layer to reject the suggested values.

The value of the connEventCount field specifies the LE connection event anchor point from which the first CS event within a CS procedure is offset. The connEventCount value supplied in either the LL_CS_RSP or LL_CS_IND PDU shall be no sooner in time than the value received in the LL_CS_REQ or LL_CS_RSP PDU that is being responded to.

Section 5.3 shall apply to the LL_CS_IND PDU as if the connEventCount field were an instant. The selection of the connEventCount field supplied in the LL_CS_RSP or LL_CS_IND PDU shall follow the requirements specified in Section 5.5. The instant passed requirements specified in Section 5.5.1 do not apply to this connEventCount field and instead the following requirements apply.

Let currConnEventCount represent the current *connEventCount* value as described in Section 4.5.1. The connEventCount field is determined to be in the past when (connEventCount – currConnEventCount) *mod* 65535 is greater than or equal to 32767. In this case, the first few CS subevents of the CS procedure may be lost and if so, the requirements described in [Vol 6] Part H, Section 4.4.5 shall apply.

The values of the Offset_Min and Offset_Max fields of the LL_CS_REQ and LL_CS_RSP PDUs each specify a window for the start of the first CS event within a CS procedure. The window specified in the LL_CS_RSP PDU shall lie entirely within that the window specified by the LL_CS_REQ PDU. The start of the first CS event within a CS procedure shall lie within the window specified by the LL_CS_REQ PDU and the LL_CS_RSP PDU.

The value of Subevent_Interval supplied in either the LL_CS_RSP PDU or the LL_CS_IND_PDU shall be greater than or equal to the value received in the LL_CS_REQ or LL_CS_RSP PDU that is being responded to.

The Subevent_Interval shall be greater than or equal to the sum of the Subevent_Len selected plus T_MES. A Controller shall be capable of supporting a minimum Subevent_Len of 2.5 ms. The value of Subevent_Len supplied in either the LL_CS_RSP or LL_CS_IND PDU shall be less than or equal to the value received in the LL_CS_REQ or LL_CS_RSP PDU that is being responded to.

Suggesting different values for Event_Interval, Subevents_Per_Event, Subevent_Interval, or Subevent_Len in the LL_CS_RSP or the LL_CS_IND PDU should result in an overall CS procedure duration that is less than or equal to the value of Max_Procedure_Len specified in the LL_CS_REQ PDU. If the Link Layer previously transmitted an LL_CS_REQ PDU, then received an LL_CS_RSP PDU without any changes to the Event_Interval, Subevents_Per_Event, Subevent_Interval, or Subevent_Len fields, then the Link Layer shall not alter those values when sending the LL_CS_IND PDU.

The ACI parameter is set according to the Num_Ant and Max_Ant_Path parameters from the CS Capabilities Exchange procedure described in Section 5.1.24. The Num_Ant value reflected in the peer's LL_CS_CAPABILITIES_REQ PDU or the LL_CS_CAPABILITIES_RSP PDU indicates the maximum number of antenna elements supported by the sending Link Layer. The Max_Ant_Path value reflected in the peer's LL_CS_CAPABILITIES_REQ PDU or the LL_CS_CAPABILITIES_RSP PDU indicates the maximum number of antenna paths supported by the sending Link Layer. The ACI parameter selected for the LL_CS_REQ PDU and finalized in the LL_CS_IND PDU shall yield an antenna configuration setting with an antenna element count that is equal to or less than the Num_Ant limit indicated for both Link Layers. The ACI parameter shall also contain an antenna path count that is equal to or less than the value for Max_Ant_Path indicated by both Link Layers. The ACI parameter shall be selected according to the ACI value described in [Vol 6] Part A, Section 5.3. Re-suggested ACI values shall only propose modification for the local device antenna selection and shall use a setting that contains antenna element numbers that are equal to or less than those suggested.

**Example 1:**

Device A supports a value of two for Num_Ant and a value of four for Max_Ant_Path. Device B supports a value of four Num_Ant and a value of four Max_Ant_Path. Device A initially suggests a 1:4 ACI configuration. Device B can then re-suggest a 1:3 ACI configuration but cannot re-suggest a 2:2 ACI configuration.

**Example 2:**

Device A supports a value of three for Num_Ant and a value of four for Max_Ant_Path. Device B supports a value of one Num_Ant and a value of two Max_Ant_Path. Device A can initially suggest a 2:1 ACI configuration but cannot initially suggest a 3:1 ACI configuration.

The Preferred_Peer_Ant field indicates the preferred ordered antenna elements that should be used, if possible, by the device that receives the LL_CS_REQ PDU. This ordering is described in Section 2.4.2.44 in the description for the Num_Ant field. The number of bits set in this field shall not exceed the Num_Ant parameter from the CS

Capabilities Exchange procedure described in Section 5.1.24. The number of bits set in this field shall be greater than or equal to the number of antenna elements denoted by the ACI field. If the number of bits set in this field exceeds the number of antenna elements denoted by the ACI field, then the ordered antenna elements specified should be used, with the lowest ordered antenna elements denoted by this field preferred. If the local device is not concerned with the peer's ordered antenna selection, then it shall set the lowest ordered Num_Ant (received from the CS Capabilities Exchange) bits within this field.

Each Link Layer may use the PHY and Pwr_Delta fields to request the peer Controller to adjust the transmit power level it uses during the CS procedure. The power control adjustment is specified relative to the current power level of the ACL connection PHY specified by the PHY parameter. The Link Layer receiving the LL Control PDU shall adjust its transmit power level as requested during all transmissions within the CS procedure. If the requested change would take the Link Layer receiving the LL Control PDU above the maximum power level the device supports, it shall change the power level to the maximum supported. If it is unable to make the requested change for any other reason, the Link Layer receiving the LL Control PDU shall change the power level to the lowest available level greater than the requested level. If the power level of the local transmit PHY indicated by the PHY parameter received during the CS Start procedure is not the PHY in use by the current ACL connection and is not currently being managed (as described in Section 5.1.17 and Section 5.1.18) by the receiving Link Layer, then that Link Layer shall immediately respond with an LL_REJECT_EXT_IND PDU with the error code *Unsupported LMP Parameter Value / Unsupported LL Parameter Value* (0x20). Refer to Section 5.1.17 for information on managing power levels for specific PHYs.

If the local CS role is that of the initiator, then the value of the TX_SNR_I field shall be selected from one of the TX_SNR_Capabilities included in the local device's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. Similarly, the value of the TX_SNR_R field shall be selected from one of the TX_SNR_Capabilities included in the peer device's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU.

Alternatively, if the local CS role is that of the reflector, then the value of the TX_SNR_R field shall be selected from one of the TX_SNR_Capabilities included in the local device's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU. Similarly, the value of the TX_SNR_I field shall be selected from one of the TX_SNR_Capabilities included in the peer device's LL_CS_CAPABILITIES_REQ or LL_CS_CAPABILITIES_RSP PDU.

If the Link Layer that transmits the LL_CS_IND PDU also previously transmitted the LL_CS_REQ PDU within the same CS Start procedure, then it shall use the same values for the PHY and Pwr_Delta fields in both transmissions.

CSChM shall be set to the ChM parameter value of the CS configuration used for the procedure.

CSNumRepetitions shall be set to the ChM_Repetition parameter value of the CS configuration used for the procedure.

CSMode0Steps shall be set to the Mode_0_Steps parameter value of the CS configuration used for the procedure.

CSShapeSelection shall be set to Ch3cShape parameter value of the CS configuration used for the procedure.

CSChannelJump shall be set to the Ch3cJump parameter value of the CS configuration used for the procedure.

Parameter values exchanged during the CS Start procedure are used to select the values for CS event and subevent scheduling, as described in Section 4.5.18.1. Table 5.3 shows this mapping.

| LL_CS_REQ, LL_CS_RSP, LL_CS_IND Parameter | CS Event/Subevent Parameter |
|---|---|
| Offset_Min, Offset_Max, Offset | T_EVENT_OFFSET |
| Event_Interval | T_EVENT_INTERVAL |
| Subevents_Per_Event | N_SUBEVENTS_PER_EVENT |
| Subevent_Interval | T_SUBEVENT_INTERVAL |
| Subevent_Len | T_SUBEVENT_LEN |
| Procedure_Interval | T_PROCEDURE_INTERVAL |
| Procedure_Count | N_PROCEDURE_COUNT |
| Max_Procedure_Len | T_MAX_PROCEDURE_LEN |

*Table 5.3: CS LL PDU parameter versus CS Event/Subevent parameter*

The procedure has completed when the LL_CS_IND PDU or an LL_REJECT_EXT_IND PDU has either been transmitted or received. The Controller shall notify its Host when the CS Start procedure completes either if it has completed successfully or if the procedure was initiated by a request from the Host. Otherwise, it shall not notify the Host that the procedure took place.

The CS procedure counter CSProcCount shall be incremented by one before the start of each CS procedure occurrence, except for the occurrence that immediately follows the completion of the CS Security Start procedure (see Section 5.1.23). The CSProcCount value shall wrap from 0xFFFF to 0x0000. Between any initiator and reflector pair, the CS Start procedure may be invoked again before the completion of an ongoing CS procedure. However, the timing and execution of two CS procedures

between that pair, which includes the timing of any procedure repeat activity, shall not overlap.

### 5.1.27 Channel Sounding Procedure Repeat Termination procedure

The initiator or reflector may only use the Channel Sounding Procedure Repeat Termination procedure to terminate repetitions of CS procedure instances if N_PROCEDURE_COUNT has been set to 0 or a value greater than 1 (see Section 4.5.18.1). The CS Procedure Repeat Termination procedure is started by sending an LL_CS_TERMINATE_REQ PDU.

The procedure collision rules described in Section 5.3 apply so that the Central and Peripheral cannot simultaneously execute the CS Procedure Repeat Termination procedure. These procedure collision rules may result in the Peripheral Link Layer receiving an LL_REJECT_EXT_IND PDU to allow the Central initiated procedure to complete.

If N_PROCEDURE_COUNT (as described in Section 5.1.26) is greater than 1, then the CS procedure repeat instance series is bounded by a maximum procedure count value. For this purpose, let StartCSProcCount be defined as the starting CSProcCount value used for the first instance of the CS procedure series that is being terminated. The Link Layer receiving the LL_CS_TERMINATE_REQ PDU shall respond by sending an LL_REJECT_EXT_IND PDU with the error code *Command Disallowed* (0x0C) if the ProcCount value received in the LL_CS_TERMINATE_REQ PDU satisfies the following condition:

(ProcCount - StartCSProcCount + 1) *mod* 65536 > N_PROCEDURE_COUNT

Additionally, the Link Layer receiving the LL_CS_TERMINATE_REQ PDU shall respond by sending an LL_REJECT_EXT_IND PDU with error code *Command Disallowed* (0x0C) if the Config_ID value received is not associated with the CS procedure repeat series associated with the received ProcCount value.

Otherwise, the Link Layer receiving the LL_CS_TERMINATE_REQ PDU shall respond by transmitting an LL_CS_TERMINATE_RSP PDU.

This procedure shall complete at most once per CS procedure repeat instance series.

Termination of all subsequent procedure instances shall occur when the LL_CS_TERMINATE_REQ PDU is sent or received. This termination shall occur before the start of the next procedure instance and shall not be applied to a CS procedure that is in progress.

Based on implementation delays, the ProcCount value received in either the LL_CS_TERMINATE_REQ PDU or the LL_CS_TERMINATE_RSP PDU may differ from

the value transmitted. In this case, the higher of the two values shall be used by the two Link Layers to keep their respective DRBGs synchronized as it relates to the DRBG backtracking resistance as described in [Vol 6] Part E, Section 3.1.7. The higher of the two values is determined by subtracting the transmitted ProcCount value from the received ProcCount value modulo 65536. If this result is greater than or equal to 32767, then the transmitted ProCount value is higher, otherwise the received ProcCount value is higher.

The CS Procedure Repeat Termination procedure has completed when an LL_CS_TERMINATE_RSP PDU or the LL_REJECT_EXT_IND PDU has been sent or received.

### 5.1.28 Channel Sounding Channel Map Update procedure

The channel map used in any CS procedure may be updated before initiating the start of that procedure (see Section 5.1.26) or before the start of any procedure instance (see Section 4.5.18.1). The Link Layer initiating the start of the CS procedure can update the channel map by sending the LL_CS_CHANNEL_MAP_IND PDU.

Because either the initiator or reflector may initiate the start of a CS procedure by transmitting an LL_CS_REQ PDU, either side may issue the LL_CS_CHANNEL_MAP_IND PDU. However, only the channel map update issued by the Link Layer initiating a subsequent CS procedure shall be applied toward that CS procedure.

An LL_CS_CHANNEL_MAP_IND PDU sent or received while any CS procedure is in progress shall not take effect for that CS procedure. These updates shall apply to all CS procedures starting after the specified instant, until the next occurrence of the CS Channel Map Update procedure.

An LL_CS_CHANNEL_MAP_IND PDU sent or received while both no CS procedure is in progress and no CS procedure instances are pending shall take effect immediately and the Instant parameter shall be processed as if that field was RFU.

The value of the instant parameter supplied in the LL_CS_CHANNEL_MAP_IND PDU shall follow the requirements specified in Section 5.5. The instant passed requirements specified in Section 5.5.1 do not apply to the instant parameter of the LL_CS_CHANNEL_MAP_IND PDU, and instead the following requirements apply.

Any pending CS procedure repeat instances shall be terminated using the rules defined in Section 5.1.27 with error code *Instant Passed* (0x28) if the instant is determined to be in the past. An instant is determined to be in the past when (Instant – *connEventCount*) *mod* 65535 is greater than or equal to 32767. In this case, updates include in the LL_CS_CHANNEL_MAP_IND PDU shall still be applied to all subsequent CS procedures, until the next occurrence of the CS Channel Map Update procedure.

The default channel map shall include all allowed CS channels as defined in [Vol 6] Part H, Section 1 and is equivalent to the ChM field of the LL_CS_CHANNEL_MAP_IND PDU with all valid channel bits set to 1.

The channel map derived from the CS Channel Map Update procedure shall be combined with the CSChM parameter selected during the CS Start procedure (see Section 5.1.26) with a logical AND operation to generate the CSFilteredChM channel map value. This value has the same format as the ChM parameter used in this update procedure and represents the filtered channel map to be used in the next CS procedure.

CSFilteredChM = CSChM & (CS Channel Map Update procedure parameter ChM)

The CSFilteredChM shall be used in the Channel Selection Algorithm #3 procedure as described in [Vol 6] Part H, Section 4.1.

The minimum number of channels in CSFilteredChM shall be 15. If, at the start of any procedure instance and after applying a channel map update, the number of channels is less than 15, then that CS procedure shall not start, and the Host shall be notified of this. If a series of procedure instances are in progress (see Section 4.5.18.1), and at the start of a procedure instance the number of channels in CSFilteredChM is less than 15, then that specific procedure instance shall not start, the procedure instance series shall terminate, and the Host shall be notified that the procedure instance series has been aborted.

The procedure has completed when the instant has passed and the new channel map has been applied.

### 5.1.29 Channel Sounding Mode-0 FAE Table Request procedure

This Table Request procedure shall only be used when the peer's No_FAE bit is set to 0 in its CS capabilities, as described in Section 2.4.2.44.

A reflector's mode-0 FAE table shall be known by the initiator before starting a CS procedure as described in Section 5.1.26. The reflector's mode-0 FAE table may be provided by the initiator's Host to the local Controller if known previously by that Host. If not previously known, then a potential initiator shall issue the LL_CS_FAE_REQ PDU to a prospective reflector to request the table. A Link Layer shall only begin a request for a peer device's mode-0 FAE table when requested by the Host. A Controller shall not allow a local Host to request a peer's FAE table if the Channel Sounding (Host Support) feature bit is not set in the Controller. Likewise, a Link Layer shall not allow the exchange of its mode-0 FAE table if the Channel Sounding (Host Support) feature bit is not set in that Controller. If a remote Link Layer sends an LL_CS_FAE_REQ PDU when the Channel Sounding (Host Support) feature bit is not set in the local Link Layer, then the local Link Layer shall send an LL_REJECT_EXT_IND PDU with the error code *Unsupported Remote Feature / Unsupported LMP Feature* (0x1A).

The Link Layer shall not transmit the LL_CS_FAE_REQ PDU if that peer device has indicated support for zero FAE in its CS capabilities (see Section 5.1.24).

Because either side may be the initiator of a CS procedure, either side may issue the LL_CS_FAE_REQ PDU.

A Link Layer receiving an LL_CS_FAE_REQ PDU after having set the No_FAE bit in its CS capabilities shall immediately respond with an LL_REJECT_EXT_IND PDU with the error code *Unsupported Feature or Parameter Value* (0x11). Otherwise, the Link Layer that receives the LL_CS_FAE_REQ PDU shall respond with the LL_CS_FAE_RSP PDU and its local per-channel mode-0 FAE table.

The procedure has completed when an LL_CS_FAE_RSP PDU or the LL_REJECT_EXT_IND PDU has been sent or received.

### 5.1.30 Frame Space Update procedure

The Central or Peripheral may initiate the Frame Space Update procedure, when supported, to change one or more of the following frame space parameters:

- T_IFS_ACL_CP
- T_IFS_ACL_PC
- T_MCES
- T_IFS_CIS
- T_MSS_CIS

The Frame Space Update procedure may be initiated when requested by the Host, or autonomously by the Link Layer.

The Spacing_Types and PHYS fields shall be used to indicate the parameters and PHYs to be changed.

The procedure shall affect the ACL and any CIS associated with the ACL created after the procedure has completed; it does not affect any other ACL connections or any existing CISes. Subject to Section 5.1.30.1, the affected values shall change on the initiating device no later than the 6th connection anchor point after it receives the LL_FRAME_SPACE_RSP PDU and on the responding device before it first sends the LL_FRAME_SPACE_RSP PDU.

When the Controller receives an LL_FRAME_SPACE_REQ PDU, it shall respond with an LL_FRAME_SPACE_RSP PDU to accept the request or an LL_REJECT_EXT_IND PDU to reject the request. The Controller may reject the request for any reason.

FS_Max shall be greater than or equal to the frame space value in use. If FS_Min and FS_Max in the LL_FRAME_SPACE_REQ PDU are both less than the frame space

value in use for any of the selected frame space types and PHYs, then the responding device may reject the request by sending an LL_REJECT_EXT_IND PDU with the error code set to *Unsupported Feature or Parameter Value* (0x11).

If the resulting frame space value causes *connIntervalUncodedMin* (if the current PHY in each direction is an LE Uncoded PHY) or *connIntervalCodedMin* (if the current PHY in either direction is the LE Coded PHY) to exceed the connection interval and the change is being done on the existing ACL connection on the current PHY in either direction, then the responding device shall reject the request by sending an LL_REJECT_EXT_IND PDU with the error code set to *Unsupported LMP Parameter Value / Unsupported LL Parameter Value* (0x20).

The responding device shall set the FS field of the LL_FRAME_SPACE_RSP PDU to a value between the FS_Min and FS_Max of the LL_FRAME_SPACE_REQ PDU, and should set it to the lowest value the responding device supports within that range.

All bits set to 0 in the PHYS and Spacing_Types fields of an LL_FRAME_SPACE_REQ PDU shall be set to 0 in the LL_FRAME_SPACE_RSP PDU sent in response. If a bit is set to 1 in the PHYS field or the Spacing_Types field of the LL_FRAME_SPACE_RSP PDU and the corresponding bit(s) are not set in the LL_FRAME_SPACE_REQ PDU, then this is considered invalid behavior (see [Vol 1] Part E, Section 2.7).

If either the PHYS or the Spacing_Types field of an LL_FRAME_SPACE_REQ PDU is set to 0, then the responding device shall reject the request by sending an LL_REJECT_EXT_IND PDU with the error code set to *Invalid LL Parameters* (0x1E).

The procedure has completed when an LL_FRAME_SPACE_RSP PDU or LL_REJECT_EXT_IND PDU has been sent or received.

If the procedure results in a change to one or more frame space values regardless of whether the PHYs or Frame_Space changed is in use, then each Controller shall notify the Host about the change.

#### 5.1.30.1 Adjacent packets in the same connection event

For the purposes of this section:

- TIA_C is the T_IFS_ACL_CP for the PHY(s) in use on the ACL.
- TIA_P is the T_IFS_ACL_PC for the PHY(s) in use on the ACL.
- If the initiating device is the Central, then TIA_I is TIA_C and TIA_R is TIA_P.
- If the initiating device is the Peripheral, then TIA_I is TIA_P and TIA_R is TIA_C.

If the Frame Space Update procedure affects TIA_I, then the initiating device shall use the following parameters instead of those in Table 4.1 during the period

between transmitting the first packet containing the LL_FRAME_SPACE_REQ PDU and receiving an LL_FRAME_SPACE_RSP PDU or LL_REJECT_EXT_IND PDU:

- *receiveWindowStart* shall be the smaller of FS_Min and TIA_I after the end of the previous packet
- *receiveWindowEnd* shall be the greater of FS_Max and TIA_I after the end of the previous packet

where FS_Min and FS_Max are the corresponding values in the LL_FRAME_SPACE_REQ PDU. In addition, during the period between transmitting the first packet containing the LL_FRAME_SPACE_RSP PDU and receiving either an ACK or a new PDU:

- If the responding device is the Central and the response is sent as a continuation of a connection event, then the LL_FRAME_SPACE_RSP PDU shall be transmitted after an Inter Frame Space specified by the FS value in the LL_FRAME_SPACE_RSP PDU.
- If the responding device is the Peripheral, then the LL_FRAME_SPACE_RSP PDU shall be transmitted after an Inter Frame Space specified by the FS value in the LL_FRAME_SPACE_RSP PDU.

If the Frame Space Update procedure affects TIA_R, then the responding device shall use the following parameters instead of those in Table 4.1 during the period between transmitting the first packet containing the LL_FRAME_SPACE_RSP PDU and receiving a packet using the new frame space value:

- *receiveWindowStart* shall be the smaller of FS and TIA_R after the end of the previous packet
- *receiveWindowEnd* shall be the greater of FS and TIA_R after the end of the previous packet

where FS is the value in the LL_FRAME_SPACE_RSP PDU.

### 5.1.31 UTP OTA mode procedure

A Controller uses the UTP OTA mode procedure to enter the RFPHY test mode. The Central or Peripheral can initiate this procedure at any time after entering the connection state by sending an LL_OTA_UTP_IND PDU.

The procedure has completed when the Link Layer acknowledgment to the LL_OTA_UTP_IND PDU is sent or received.

The Unified Test Protocol described in [Vol 6] Part F, Section 5 is transported using LL_OTA_UTP_IND PDUs. How the Controller uses these PDUs is described in [Vol 6] Part F, Section 6.4.

UTP PDUs shall be processed if and only if the ACL is encrypted, the UTP OTA mode feature is supported, and UTP OTA mode is enabled at the IUT.

If UTP OTA mode is enabled but the ACL is not encrypted when one of the UTP PDUs is received, then the Controller shall immediately reject the PDU by sending an LL_REJECT_EXT_IND PDU with the error code *Insufficient Security* (0x2F).

If UTP OTA mode is not enabled when one of the UTP PDUs is received, then the Controller shall immediately reject the PDU by sending an LL_REJECT_EXT_IND PDU with the error code *Command Disallowed* (0x0C).

### 5.1.32 Connection Rate Update procedure

The Central may update the *connInterval*, *connSubrateBaseEvent*, *connSubrateFactor*, *connPeripheralLatency*, *connContinuationNumber*, and *connSupervisionTimeout* by sending an LL_CONNECTION_RATE_IND PDU. The Peripheral shall not send this PDU. The Central shall only initiate the Connection Rate Update Procedure when requested by the Host or when requested by the Peripheral via the Connection Rate Request procedure. The Central shall not initiate the Connection Rate Update procedure while a Connection Parameters Request procedure or a Connection Subrate Request procedure is in progress. The Central shall not initiate this procedure until the Central has performed a Feature Exchange procedure (see Section 5.1.4) and has determined that the Connection Rate (Host Support) bit is set in the Peripheral's FeatureSet. The Central shall not initiate this procedure while any CS procedures are in progress (see Section 4.5.18.1).

The Link Layer of the Central shall select the *connInterval* from the interval range given by the Host (i.e., $connInterval_{min}$ and $connInterval_{max}$). The selected value shall be at least *connIntervalUncodedMin* on the LE Uncoded PHY and *connIntervalCodedMin* on the LE Coded PHY.

The requirements of Section 5.5 shall apply to the LL_CONNECTION_RATE_IND PDU. The Central shall transmit on the connection event where *connEventCount* equals Instant and the connection event before that event, irrespective of subrating. When the Peripheral receives such a PDU with the instant in the future, it shall listen to the connection event where *connEventCount* equals Instant and the connection event before that event, even if subrating or Peripheral latency means it would not normally do so.

The values of *connInterval*, *connSubrateFactor*, *connPeripheralLatency*, *connContinuationNumber*, and *connSupervisionTimeout* specified in the LL_CONNECTION_RATE_IND PDU shall be used starting at the instant. At the instant, the value of *connSubrateBaseEvent* shall be set to the instant specified in the LL_CONNECTION_RATE_IND PDU. The connection interval used before the instant is

known as $connInterval_{OLD}$; the connection interval used at the instant and after is known as $connInterval_{NEW}$.

Figure 5.6 shows the connection event timing around the instant.

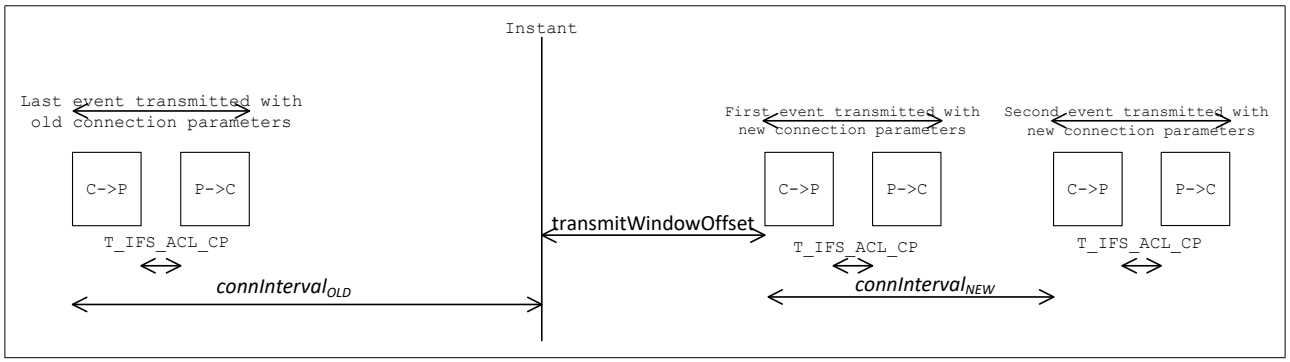



*Figure 5.6: Connection event timing in the case of Connection Rate Update*

The Central may adjust the anchor point when deciding the timing of the first packet transmitted with new connection parameters. The first connection event at or after the instant shall start at $connInterval_{OLD}$ + transmitWindowOffset after the anchor point of the last connection event before the instant. The transmitWindowOffset shall be a multiple of 125 µs in the range 0 µs to $connInterval_{NEW}$. The Central and the Peripheral shall follow the timing requirements in Section 4.2 when switching between the last event transmitted with old connection parameters and the first event transmitted with new connection parameters.

The instant occurs after $connInterval_{OLD}$ and before transmitWindowOffset. All the connection event transmission rules and subrating rules specified in Section 4.5.1 shall apply.

The connection supervision timer $T_{LLconnSupervision}$ shall be reset at the anchor point of the first connection event at or after the instant.

If the received LL_CONNECTION_RATE_IND PDU contains any fields that are out of valid range or unsupported by the Peripheral's Link Layer, then the Peripheral's Link Layer shall respond with LL_UNKNOWN_RSP PDU to signal back to the Central's Link Layer that it does not continue the procedure.

The Link Layer shall notify the Host of the new connection event parameters upon completion of the procedure.

The procedure is complete when the instant has passed and the new connection event parameters have been applied, or an LL_UNKNOWN_RSP PDU has been sent or received.

Note: Depending on the connection parameters in effect after this procedure completes, there can be LL Control PDUs that are longer than can fit in a single connection event together with an empty PDU sent in the other direction. If so, then the Link Layer cannot use a procedure that includes such an LL Control PDU without first increasing the connection interval.

### 5.1.33 Connection Rate Request procedure

The Peripheral may initiate a Connection Rate Request procedure to request the Central to update the *connInterval*, *connSubrateBaseEvent*, *connSubrateFactor*, *connPeripheralLatency*, *connContinuationNumber*, and *connSupervisionTimeout* any valid time after entering the Connection state.

The Connection Rate Request procedure is initiated by issuing an LL_CONNECTION_RATE_REQ PDU. The Link Layer shall only initiate this procedure when requested by the Host. The Link Layer shall not initiate this procedure if the Connection Rate (Host Support) feature bit is not set in both the local and the remote Controller.

A Peripheral shall not initiate the Connection Rate Request procedure while a Connection Parameters Request procedure or a Connection Subrate Request procedure is in progress. The Peripheral shall not initiate this procedure while any CS procedures are in progress (see Section 4.5.18.1).

The Peripheral's Link Layer shall select the values of the LL_CONNECTION_RATE_REQ PDU for Interval_Min, Interval_Max, Timeout, PreferredPeriodicity, ReferenceConnEventCount, Offset0, Offset1, Offset2, and Offset3 according to the rules specified in the Connection Parameters Request procedure (see Section 5.1.7).

The Peripheral's Link Layer shall select the values of the LL_CONNECTION_RATE_REQ PDU for SubrateFactorMin, SubrateFactorMax, Max_Latency, and Continuation_Number according to the rules specified in the Connection Subrate Request procedure (see Section 5.1.20).

Upon receiving an LL_CONNECTION_RATE_REQ PDU, the Central shall respond either by initiating the Connection Rate Update procedure (see Section 5.1.32) or by sending an LL_REJECT_EXT_IND PDU. The Central shall follow the rules specified in Section 5.1.7 when accepting the Interval_Min, Interval_Max, Timeout, PreferredPeriodicity, ReferenceConnEventCount, Offset0, Offset1, Offset2, and Offset3 values. The Central shall follow the rules specified in Section 5.1.20 when accepting the SubrateFactorMin, SubrateFactorMax, Max_Latency, and Continuation_Number values.

If an LL_CONNECTION_RATE_REQ PDU is received while a CS procedure or CS procedure instance repeats, as described in Section 4.5.18.1, are in progress, then the

receiving Link Layer shall respond with an LL_REJECT_EXT_IND PDU with the error code *Controller Busy* (0x3A).

If the received LL_CONNECTION_RATE_REQ PDU contains parameters that are not acceptable to the Link Layer, then the Link Layer of the device shall respond to the LL_CONNECTION_RATE_REQ PDU with an LL_REJECT_EXT_IND PDU with the error code *Unsupported LL Parameter Value* (0x20).

If the received LL_CONNECTION_RATE_REQ PDU contains any fields that are out of valid range, then the Link Layer shall reject the LL_CONNECTION_RATE_REQ PDU by issuing an LL_REJECT_EXT_IND PDU with the error code *Invalid LL Parameters* (0x1E).

If the received LL_CONNECTION_RATE_REQ PDU has Interval_Max set to a value lower than *connIntervalRequired* (see Section 4.5.10), then the Link Layer shall reject the LL_CONNECTION_RATE_REQ PDU by issuing an LL_REJECT_EXT_IND PDU with the error code *Unsupported Feature or Parameter Value* (0x11).

Note: The Central's reason for any rejection could have been temporary (e.g., due to other activities) and, therefore, the initiating device can retry.

The procedure has completed when the resulting Connection Rate Update procedure has completed or an LL_REJECT_EXT_IND PDU has been sent or received.

## 5.2 Procedure response timeout

This section specifies procedure timeout rules that shall be applied to all the Link Layer control procedures specified in Section 5.1, except for the Connection Update and Channel Map Update procedures for which there are no timeout rules.

To be able to detect a non-responsive Link Layer Control procedure, both the Central and the Peripheral shall use a procedure response timeout timer, $T_{PRT}$ . Upon the initiation of a procedure, the procedure response timeout timer shall be reset and started.

Each LL Control PDU that is queued for transmission resets the procedure response timeout timer.

When the procedure has completed, the procedure response timeout timer shall be stopped.

If the procedure response timeout timer reaches 40 seconds, the ACL connection is considered lost (see Section 4.5.12). The Link Layer exits the Connection state and shall transition to the Standby state. The Host shall be notified of the loss of connection.

## 5.3 Procedure collisions

Since LL Control PDUs are not interpreted in real time, collisions can occur where the Link Layer of the Central and the Link Layer of the Peripheral initiate incompatible procedures. Two procedures are incompatible in the following cases:

- The two procedures both involve an instant.
- The two procedures are both the Channel Sounding Configuration procedure (see Section 5.1.25).
- The two procedures are both the Channel Sounding Procedure Repeat Termination procedure (see Section 5.1.27).
- One procedure is the Connection Subrate Request procedure (see Section 5.1.20) and the other is the Channel Sounding Start procedure (see Section 5.1.26).
- One procedure is the Frame Space Update procedure (see Section 5.1.30); the other is either the Frame Space Update procedure or the Connected Isochronous Stream Creation procedure (see Section 5.1.15).
- One procedure is either the Connection Subrate Update procedure (see Section 5.1.19) or the Connection Subrate Request procedure (see Section 5.1.20) and the other is either the Connection Rate Update procedure (see Section 5.1.32) or the Connection Rate Request procedure (see Section 5.1.33).

In these cases, the rules in this section shall be followed:

A device shall not initiate a procedure after responding to a PDU that had initiated an incompatible procedure until that procedure has completed.

If device initiates a procedure A and, while that procedure has not completed, receives a PDU from its peer that initiates an incompatible procedure B, then:

- If the peer has already sent at least one PDU as part of procedure A, the device should immediately exit the Connection State and transition to the Standby State.
- Otherwise, if the device is the Central, it shall reject the PDU received from the Peripheral by issuing an LL_REJECT_EXT_IND (if supported by both devices) or LL_REJECT_IND (otherwise) PDU. It shall then proceed with procedure A.
- Otherwise (the device is the Peripheral) it shall proceed to handle the Central-initiated procedure B and take no further action in the Peripheral-initiated procedure A except processing the rejection from the Central.

The Host shall be notified that the link has been disconnected with, or the rejection PDU shall use (as appropriate):

- the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23) if procedures A and B are the same procedure;
- the error code *LMP Error Transaction Collision / LL Procedure Collision* (0x23) if procedure A is the Connection Update procedure and procedure B is the Connection Parameters Request procedure;
- the error code *Different Transaction Collision* (0x2A) otherwise.

## 5.4 LE Authenticated Payload Timeout

LE Authenticated Payload Timeout (*authenticatedPayloadTO*) is a parameter that defines the maximum amount of time, in milliseconds, allowed between receiving ACL packets containing a valid MIC. The Host can change the value of *authenticatedPayloadTO* using the HCI_Write_Authenticated_Payload_Timeout command ([Vol 4] Part E, Section 7.3.94). The default value for *authenticatedPayloadTO* is 30 seconds.

When the connection is encrypted, a device supporting LE Ping feature shall start the LE Authenticated Payload timer $T_{LE_Authenticated_Payload}$ to monitor the time since the last reception of a packet containing a valid MIC from the remote device. Each device shall reset the timer $T_{LE_Authenticated_Payload}$ upon reception of a packet with a valid MIC. The timer shall not be reset upon the reception of a resent packet.

If at any time in the CONNECTION state the timer $T_{LE_Authenticated_Payload}$ reaches the *authenticatedPayloadTO* value, the Host shall be notified (using the HCI_Authenticated_Payload_Timeout_Expired event if the Controller supports HCI; see [Vol 4] Part E, Section 7.7.75). The $T_{LE_Authenticated_Payload}$ Timer restarts after it is expired.

The timer $T_{LE_Authenticated_Payload}$ shall continue to run during encryption pause procedure.

Whenever the Host sets the *authenticatedPayloadTO* while the timer $T_{LE_Authenticated_Payload}$ is running, the timer shall be reset.

## 5.5 Procedures with Instants

Where a procedure involves a PDU with an Instant field, then the following rules shall apply.

The Instant field shall be used to indicate the *connEventCount* or *bigEventCounter* when the relevant change shall be applied; this is known as the instant for the

procedure. In the case of the LL_CS_CHANNEL_MAP_IND PDU, either Link Layer may select the instant value. In all other cases, only the Link Layer of the Central shall select the instant value.

If the Link Layer of the Central is selecting the instant value, then the Central should allow a minimum of 6 connection events when it intends to transmit and that the Peripheral will be listening for before the instant occurs, considering that the Peripheral may only be listening once every *connSubrateFactor* × (*connPeripheralLatency* + 1) events. If the Link Layer of the Peripheral is selecting the instant value, then the Peripheral should allow a minimum of 6 connection events that it intends to schedule before the instant occurs, considering that the Central may only transmit once every *connSubrateFactor* events. The event shall be the next one with the specified value of *connEventCount* or of $bigEventCounter_{15\text{-}0}$.

Note: Comparisons of the *connEventCount* or *bigEventCounter* and the Instant field are performed using *mod* 65536 math (only values from 0 to 65535 are allowed).

When performing a Link Layer procedure that has an instant that is selected by the Central's Link Layer, it may (particularly for large values of the subrate factor) use the Connection Subrate Update procedure to set the subrate factor to 1 and the Peripheral latency to 0 before performing the procedure and then use it again to restore the previous values afterwards. If the Link Layer does so, it should not notify the Host of these changes. However, the Link Layer shall not restore the settings autonomously after a Connection Update procedure that changed the connection interval and shall not restore *connPeripheralLatency* or *connSupervisionTimeout* after a Connection Update procedure that did not change the connection interval.

### 5.5.1 ACL control procedures

When a Peripheral receives such a PDU where (Instant – *connEventCount*) *mod* 65536 is less than 32767 and Instant is not equal to *connEventCount*, the Peripheral shall listen to all the connection events until it has confirmation that the Central has received its acknowledgment of the PDU or *connEventCount* equals Instant.

When a Peripheral receives such a PDU where (Instant – *connEventCount*) *mod* 65536 is greater than or equal to 32767 (because the instant is in the past), it shall take the following actions:

- If the PDU is an LL_CONNECTION_UPDATE_IND, the Link Layer of the Peripheral shall consider the connection to be lost.
- Otherwise, the Link Layer of the Peripheral may consider the connection to be lost.

If the connection is considered to be lost, the Link Layer of the Peripheral shall exit the Connection state and transition to the Standby state, and shall notify the Host using the error code *Instant Passed* (0x28).

### 5.5.2 BIG control procedures

The Isochronous Broadcaster shall set the instant at least 6 BIG events after the first BIG event where the BIG Control PDU is transmitted.

When a Synchronized Receiver receives such a PDU where (Instant – *bigEventCounter*) *mod* 65536 is greater than or equal to 32767 (because the instant is in the past), the Link Layer may stop synchronization with the BIG.

## 5.6 BIG control procedures

BIG control procedures are used to send control information concerning a BIG from the Isochronous Broadcaster to the Synchronized Receivers.

Each BIG Control procedure involves transmitting a single BIG Control PDU during the control subevent of BIG events. Each such PDU shall be transmitted in six consecutive BIG events and may be transmitted in other BIG events (not necessarily consecutive) after these six; the procedure ends when the BIG Control PDU has been retransmitted for the last time. Only one BIG control procedure shall be in progress at a time for a given BIG.

### 5.6.1 BIG Channel Map Update procedure

The BIG Channel Map Update procedure is used to send a new channel map for all BISes in a BIG.

When instructed by the Host or autonomously, the Link Layer of an Isochronous Broadcaster shall initiate this procedure by transmitting a BIG_CHANNEL_MAP_IND PDU.

The Link Layer of the Isochronous Broadcaster shall not initiate a subsequent instance of this procedure until the instant has passed.

The Link Layer of both Isochronous Broadcaster and Synchronized Receiver shall use the new channel map starting with the BIG event identified by the instant. The Link Layer shall update the ChM field in the BIGInfo and send the updated BIGInfo in the associated periodic advertising train (if enabled) at the nearest future periodic advertising event to the instant.

### 5.6.2 BIG Termination procedure

The BIG Termination procedure is used to notify all Synchronized Receivers of a BIG that the transmission of that BIG is about to be terminated.

When instructed by the Host, the Link Layer shall initiate this procedure by transmitting a BIG_TERMINATE_IND PDU. The Link Layer shall stop transmitting BIG events at the

instant and shall return to the Standby state. If the Link Layer is still transmitting the associated BIGInfo, it shall stop doing so no later than the instant.

The Link Layer shall terminate the BIG no later than when the *bisPayloadCounter* equals $2^{39} – 1$.

When the Link Layer receives a BIG_TERMINATE_IND PDU, it shall stop synchronization with the BIG and, unless it is still synchronized to the periodic advertising train, shall return to the Standby state.

# 6 PRIVACY

The Link Layer provides Privacy by using Private Addresses (see Section 1.3.2).

If a device is using Resolvable Private Addresses Section 1.3.2.2, it shall also have an Identity Address that is either a Public or Random Static address type.

## 6.1 Resolvable Private address generation interval

A resolvable private address shall be generated using the Resolvable Private Address Generation (see Section 1.3.2.2).

The Link Layer shall set a timer determined by the Host. A new resolvable private address shall be generated when the timer expires and then the Link Layer shall set the timer to a new value selected randomly within the range specified by the Host. If the Link Layer is reset, then a new resolvable private address shall be generated and the timer started with any value in the specified range.

If the resolvable private address is generated frequently, connection establishment times can be affected. The timer should be set to the value of 15 minutes recommended in [Vol 3] Part C, Appendix A.

If requested by the Host, the Controller shall generate a new resolvable private address each time the advertising data or scan response data changes.

## 6.2 Privacy in the Advertising state

Privacy in the advertising state determines how the Link Layer processes Resolvable Private Addresses for advertising events.

The requirements in the following subsections apply in addition to those in Section 4.4.2 and Section 4.7.

### 6.2.1 Connectable and scannable undirected event type

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using connectable and scannable undirected events.

The AdvA field of the connectable and scannable undirected advertising event PDU is generated using the resolving list's Local IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2). If the Host has not provided any Resolving List IRK pairs for the peer to the Link Layer, then the AdvA field shall use a Host-provided address.

When an advertiser receives a connection request that contains a resolvable private address for the initiator's address (InitA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy (see Section 4.3.2) shall then determine if the advertiser establishes a connection.

When an advertiser receives a connection request that contains a device Identity Address for the initiator's address field (InitA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy (see Section 4.3.2) shall determine if the advertiser establishes a connection.

When an advertiser receives a scan request that contains a resolvable private address for the scanner's device address (ScanA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy (see Section 4.3.2) shall then determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a device Identity Address for the scanner's device address (ScanA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan request.

When an advertiser receives a scan or connection request that contains a non-resolvable private address, the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan or connection request.

If the advertiser processes the scan request, the advertiser's device address (AdvA field) in the SCAN_RSP PDU shall be the same as the advertiser's device address (AdvA field) in the SCAN_REQ PDU to which it is responding.

### 6.2.2 Connectable directed event type

The Link Layer shall use resolvable private addresses for the advertiser's device address (AdvA field). If an IRK is available in the Link Layer Resolving List for the peer device, then the target's device address (TargetA field) shall use a resolvable private address. If an IRK is not available in the Link Layer Resolving List or the IRK is set to zero for the peer device, then the target's device address (TargetA field) shall use the Identity Address when entering the Advertising State and using connectable directed events.

The AdvA field of the connectable directed advertising event PDU is generated using the resolving list's Local IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2).

The TargetA field of the connectable directed advertising event PDU is generated using the Peer IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2). The TargetA field uses the public or static device address of the peer device if no peer IRK is available.

When an advertiser receives a connection request that contains a resolvable private address for the initiator's address (InitA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3).

When an advertiser receives a connection request that contains a device Identity Address for the initiator's address field (InitA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertiser shall establish a connection. When responding with an AUX_CONNECT_RSP, the Link Layer should not set the TargetA field to the same value as the InitA field in the received PDU.

### 6.2.3 Non-connectable and non-scannable undirected and scannable undirected event types

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using the following event types:

- non-connectable and non-scannable undirected event
- scannable undirected event

The AdvA field of the non-connectable and non-scannable undirected advertising event PDU and scannable undirected event PDU are generated using the resolving list's Local IRK value and the Resolvable Private Address Generation procedure or Non-Resolvable Private Address Generation procedure (see Section 1.3.2.2).

When an advertiser receives a scan request that contains a resolvable private address for the scanner's device address (ScanA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy (see Section 4.3.2) shall then determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a device Identity Address for the scanner's device address (ScanA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan request.

When an advertiser receives a scan request that contains a non-resolvable private address, the advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the scan request.

If the advertiser processes the scan request, the advertiser's device address (AdvA field) in the scan response PDU shall be the same as the advertiser's device address (AdvA field) in the scan request PDU to which it is responding.

### 6.2.4 Connectable undirected event type

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using connectable undirected events.

The AdvA field of the connectable undirected advertising event PDU is generated using the resolving list's Local IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2). If the Host has not provided any Resolving List IRK pairs for the peer to the Link Layer, then the AdvA field shall use a Host-provided address.

When an advertiser receives a connection request that contains a resolvable private address for the initiator's address (InitA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The advertising filter policy (see Section 4.3.2) shall then determine if the advertiser establishes a connection. If the advertiser was able to resolve the InitA field, then it should set the TargetA field of the AUX_CONNECT_RSP PDU to a new resolvable private address using the Resolvable Private Address Generation procedure. Otherwise the advertiser shall set the TargetA field to the same value as the InitA field in the AUX_CONNECT_REQ PDU.

The advertising filter policy (see Section 4.3.2) shall determine if the advertiser processes the connect request if an advertiser receives a connect request that contains a non-resolvable private address.

### 6.2.5 Non-connectable and non-scannable directed and scannable directed event types

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the advertiser's device address (AdvA field) when entering the Advertising State and using the following event types:

- non-connectable and non-scannable directed event
- scannable directed event

If an IRK is available in the Link Layer Resolving List for the peer device, then the target's device address (TargetA field) shall use a resolvable private address. If an IRK

is not available in the Link Layer Resolving List or the IRK is set to zero for the peer device, then the target's device address (TargetA field) shall use the Identity Address when entering the Advertising State and using non-connectable and non-scannable directed and scannable directed events.

The AdvA field of the non-connectable and non-scannable directed advertising event PDU and scannable directed event PDU is generated using the resolving list's Local IRK value and the Resolvable Private Address Generation procedure or Non-Resolvable Private Address Generation procedure (see Section 1.3.2.2).

The TargetA field of the scannable directed advertising event PDU is generated using the Peer IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2). The TargetA field uses the public or static device address of the peer device if no peer IRK is available.

When an advertiser receives a scan request that contains a resolvable private address for the scanner's device address (ScanA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3).

If the advertiser processes the scan request, the advertiser's device address (AdvA field) in the AUX_SCAN_RSP PDU shall be the same as the advertiser's device address (AdvA field) in the AUX_SCAN_REQ PDU to which it is responding.

## 6.3 Privacy in the Scanning state

The requirements in this section apply in addition to those in Section 4.4.3 and Section 4.7.

The Link Layer may use resolvable private addresses or non-resolvable private addresses for the scanner's device address (ScanA field) when entering the Scanning State.

The ScanA field of the scanning PDU is generated using the Resolving List's Local IRK value and the Resolvable Private Address Generation procedure (see Section 1.3.2.2), or the address is provided by the Host.

The advertiser's device address (AdvA field) in the scan request PDU shall be the same as the advertiser's device address (AdvA field) received in the advertising PDU to which the scanner is responding.

When a scanner receives an advertising packet that contains a resolvable private address for the advertiser's device address (AdvA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The scanning filter policy (see Section 4.3.3) shall then determine if the scanner responds with a scan request.

When a scanner receives an advertising packet that contains a device Identity Address for the advertiser's device address (AdvA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the scanning filter policy (see Section 4.3.3) shall determine if the scanner responds with a scan request. The Link Layer should not set the ScanA field to the same value as the TargetA field in the received advertising PDU.

When a scanner receives a directed scannable advertising packet that contains a resolvable private address for the target's address (TargetA field) and address resolution is enabled, the Link Layer shall resolve the private address using the Local IRK values (see Section 1.3.2.3). If the Link Layer is unable to resolve the address, the scanning filter policy shall determine if the Host is notified about this advertisement.

A scanner that has been instructed by the Host to use Resolvable Private Addresses (e.g., using the HCI_LE_Set_Scan_Parameters command; see [Vol 4] Part E, Section 7.8.10) shall not respond to directed scannable advertising events that contain public or static addresses for the target's address (TargetA field).

When a scanner receives an advertising packet that contains a non-resolvable private address, the scanning filter policy (see Section 4.3.3) shall determine if the scanner processes the advertising packet.

## 6.4 Privacy in the Initiating state

The requirements in this section apply in addition to those in Section 4.4.4 and Section 4.7.

The Link Layer may use resolvable private addresses, the public address, or an address provided by the Host for the initiator's device address (InitA field) when in the Initiating state.

When an initiator receives a connectable advertising event that contains a resolvable private address for the advertiser's address (AdvA field) and address resolution is enabled, the Link Layer shall resolve the private address using the resolving list's Peer IRK values (see Section 1.3.2.3). The initiator filter policy (see Section 4.3.4) shall determine if the initiator establishes a connection.

The advertiser's device address (AdvA field) in the initiating PDU shall be the same as the advertiser's device address (AdvA field) received in the advertising event PDU to which the initiator is responding.

When an initiator receives a directed connectable advertising event that contains a resolvable private address for the target's address (TargetA field) and address resolution is enabled, the Link Layer shall resolve the private address using the resolving list's Local IRK values (see Section 1.3.2.3). An initiator that has

been instructed by the Host to use Resolvable Private Addresses (e.g., using the HCI_LE_Create_Connection command; see [Vol 4] Part E, Section 7.8.12) shall not respond to directed connectable advertising events that contain Public or Static addresses for the target's address (TargetA field).

When an initiator receives a connectable advertising event that contains a device Identity Address for the advertiser's device address (AdvA field), and if that device is in the Resolving List with a non-zero peer IRK for which the Host has specified device privacy mode, then the initiator filter policy (see Section 4.3.4) shall determine if the initiator establishes a connection.

The Host may request that the Link Layer shall use resolvable private addresses for the initiator's device address (InitA field) when initiating connection establishment with an associated device that exists in the Resolving List. If so, then the initiator's device address (InitA field) in the initiating PDU is generated using the Resolving List Local IRK and the Resolvable Private Address Generation procedure (see Section 1.3.2.2). The Link Layer should not set the InitA field to the same value as the TargetA field in the received advertising PDU.

In all other circumstances, the Link Layer shall use the public address or the Host-provided address for the initiator's device address (InitA field) when initiating connection establishment.

## 6.5 Privacy of the device

A device that is required to maintain its privacy shall not use its Identity Address in any advertising PDU. The Host may command the Controller to advertise, scan, or initiate a connection using a Resolvable Private Address when the Controller is using the resolving list. If the local IRK in the resolving list associated with the peer Identity Address is all zeros, the Controller will use the Identity Address. If the peer IRK in the resolving list associated with the peer Identity Address is all zeros, the Controller will accept the Identity Address. If the Host has instructed the Controller to use device privacy mode with a peer Identity Address, the Controller will accept the peer's Identity Address. This implies that the device's network privacy is violated. To maintain a device's network privacy, the Controller's resolving list can only contain entries with non-zero IRKs and device privacy mode cannot be used.

## 6.6 Privacy in the Synchronization State

### 6.6.1 Periodic advertising trains

The requirements in this section apply when the Link Layer is directed by the Host to receive a periodic advertising train using the SyncInfo field of an AUX_ADV_IND PDU. The requirements in this section apply in addition to those in Section 4.4.5.1 and Section 4.7.

When a scanner receives an advertising packet that contains a resolvable private address for the advertiser's device address (AdvA field) and address resolution is enabled, the Link Layer shall resolve the private address (see Section 1.3.2.3). The scanner's periodic sync establishment filter policy (see Section 4.3.5) shall determine if the scanner processes the advertising packet.

# 7 ISO TEST MODE

This section contains information on testing a CIS or BIS in the Controller without the protocols and profiles in the Host that use the CIS Central, CIS Peripheral, Isochronous Broadcaster, and Synchronized Receiver role features of this specification. The command that enters ISO Test mode takes the connection handle of a CIS or BIS as a parameter.

## 7.1 ISO Transmit test mode

When instructed by the Host, the Link Layer shall generate and transmit the payloads in SDUs as follows:

1. When Payload_Type = 0b00 (zero size SDU), the length of every SDU shall be set to zero.
2. When Payload_Type = 0b01 (variable size SDU), every SDU shall contain the value of the 4-octet SDU counter of the SDU padded with vendor-specific data as shown in Figure 7.1. The size of the SDU can be different in every SDU_Interval by an algorithm chosen by the transmitting device. The range shall be between 4 and Max_SDU.

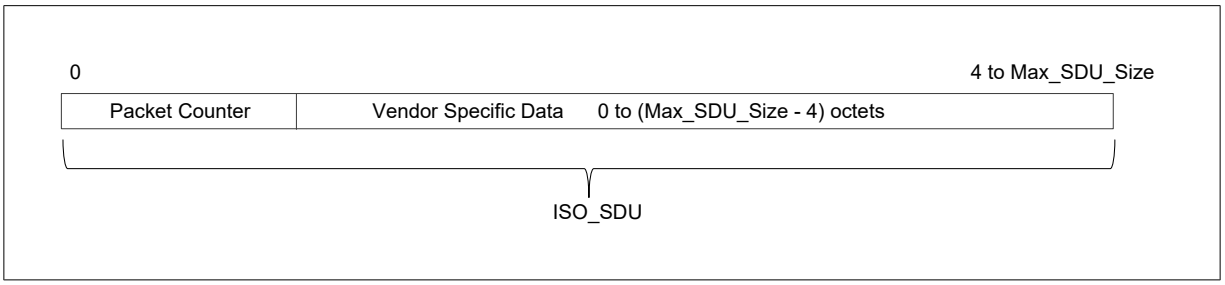



*Figure 7.1: Payload_Type = 1 (variable size payloads)*

3. When Payload_Type = 0b10 (maximum size SDU), every SDU shall contain the value of the 4-octet SDU counter of the SDU padded with vendor-specific data, as shown in Figure 7.2. The length of every SDU shall be equal to the value of the Max_SDU.

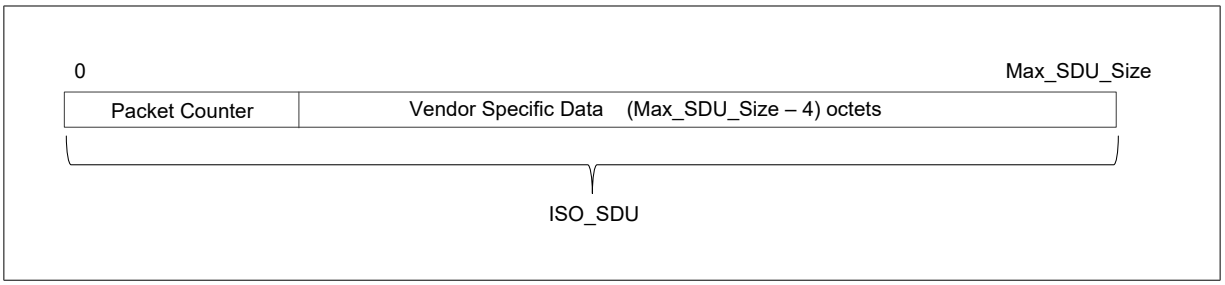



*Figure 7.2: Payload_Type = 2 (maximum size payloads)*

The ISO Transmit Test mode may be used in combination with the ISO Receive Test mode for a CIS configured for data flow from the Central to Peripheral and/or Peripheral to Central.

Note: For Payload_Type = 0b01 and 0b10, the minimum SDU size must be configured to be at least 4 octets, as described above in numbered paragraphs 2 and 3.

The SDU counter depends on the configuration (Max_SDU, SDU_Interval, or Max_PDU), of the logical link in the direction ISO Transmit Test mode is applied.

- When using unframed PDUs, the SDU counter shall be equal to the payload counter (i.e. the value of *bisPayloadCounter* in a BIS or of *cisPayloadCounter* in a CIS) of the PDU containing the first fragment from the SDU divided by UPPS (see [Vol 6] Part G, Section 2.1).

  Note: For UPPS=1 this results in the SDU counter being identical to the payload counter. The SDU counter is independent of the actual length of the SDUs when using variable length SDUs.
- When using framed PDUs the SDU counter cannot be derived from the payload counter of the PDU. It shall be initialized to 0 for the first test SDU transmitted and incremented by 1 for each additional SDU.

When instructed by the Host, the Link Layer shall stop generating test SDUs, exit the ISO Transmit Test mode, and notify the Host.

## 7.2 ISO Receive test mode

When instructed by the Host, the Link Layer shall initialize three 32-bit counters (Missed_SDU_Count, Received_SDU_Count, and Failed_SDU_Count) to zero and then increment these counters based on the received or missed test SDUs. Any vendor-specific data included in the SDU shall be ignored for the purpose of updating these counters.

When using framed PDUs the expected value of the SDU counter shall be initialized with the value of the SDU counter of the first valid received SDU. A valid SDU is one where all the PDUs composing the SDU are valid.

The counters shall be incremented as follows:

1. The Received_SDU_Count shall be incremented by one for every valid received test ISO Data SDU containing a size in the expected range and an SDU counter that matches the expected value as defined in Section 7.1. This includes empty SDUs when configured to zero size SDU.
2. The Missed_SDU_Count shall be incremented by one for each SDU that is made up of one or more invalid or missing PDUs.

3. The Failed_SDU_Count shall be incremented by one for each valid received test SDU where the size or SDU counter does not match the expected size or value as defined in Section 7.1.

Because the transmitter and receiver do not enter test mode simultaneously, it is not possible to determine whether the first test SDU received was the first one sent. As a consequence, at the moment the first valid test SDU is received (indicated by either Received_SDU_Count or Failed_SDU_Count being incremented), the value of Missed_SDU_Count is unpredictable. Once a valid test SDU has been received, any further changes in Missed_SDU_Count will be correct.

When instructed by the Host, the Link Layer shall exit the ISO Receive Test mode and notify the Host.

# 8 REFERENCES

[1] Core Specification Supplement, Part A, Data Types Specification

# SAMPLE DATA

*This Part contains sample data for the LE Controller; see [Vol 1] Part E, Section 4.*

# CONTENTS

**1 Encryption sample data ..... 3297**
1.1 Encrypt Command ..... 3300
1.2 Derivation of the MIC and encrypted data ..... 3300

**2 LE Coded PHY sample data ..... 3304**
2.1 Reference information packet ..... 3304
2.2 Forward Error Correction encoder ..... 3304
2.3 Transmitted symbols (S=2) ..... 3305
2.4 Transmitted symbols (S=8) ..... 3306

**3 LE Channel Selection algorithm #2 sample data ..... 3308**
3.1 Sample data 1 (37 used channels) ..... 3308
3.2 Sample data 2 (9 used channels) ..... 3308
3.3 Sample data 3 (3 used channels) ..... 3309

**4 Complete packets ..... 3311**
4.1 Whitening sequences ..... 3311
4.2 Advertising Physical channel PDUs ..... 3312
4.2.1 Legacy advertising PDUs ..... 3312
4.2.2 Extended advertising PDUs ..... 3313
4.3 Data channel PDUs ..... 3314
4.3.1 LL data PDUs ..... 3314
4.3.2 LL control PDUs ..... 3315

**5 Access Address generation for BISes ..... 3316**

**6 Group Session Key derivation for BIG ..... 3318**

**7 Deterministic Random Bit Generator sample data ..... 3319**

**8 Channel Sounding procedure sample data ..... 3325**
8.1 Channel Selection Algorithm #3b ..... 3325
8.1.1 Set 1 ..... 3325
8.1.2 Set 2 ..... 3347
8.1.3 Set 3 ..... 3394
8.1.4 Set 4 ..... 3437
8.2 Channel Selection Algorithm #3c ..... 3508
8.2.1 Set 1 ..... 3508
8.2.2 Set 2 ..... 3530
8.2.3 Set 3 ..... 3552

# 1 ENCRYPTION SAMPLE DATA

This section contains sample data for the Low Energy encryption process.

The following scenario describes the start of encryption, followed by the transfer of an encrypted data physical channel data packet in each direction. It describes:

- how the derived values are calculated (fixed values are given in red)
- which HCI command and events are exchanged if HCI is used (given in italic)
- which LL messages are exchanged over the air (given in green).

CRCs are not shown because they depend on a random CRC init value. Scrambling is disabled.

The following parameters are set to the fixed values below:

```
LTK = 0x4C68384139F574D836BCF34E9DFB01BF (MSO to LSO)
EDIV = 0x2474 (MSO to LSO)

RAND = 0xABCDEF1234567890 (MSO to LSO)
SKD_C = 0xACBDCEDFE0F10213 (MSO to LSO)
SKD_P = 0x0213243546576879 (MSO to LSO)
IV_C = 0xBADCAB24 (MSO to LSO)
IV_P = 0xDEAFBABE (MSO to LSO)
```

*HCI_LE_Enable_Encryption (length 0x1C) – Central HCI command*
*HCI parameters as octet sequence: 00 08 90 78 56 34 12 ef cd ab 74 24 bf 01 fb 9d 4e f3 bc 36 d8 74 f5 39 41 38 68 4c*
*Handle (2-octet value MSO to LSO) 0x0800*
*Random (8-octet value MSO to LSO) 0xabcdef1234567890*
*Encrypted Diversifier (2-octet value MSO to LSO) 0x2474*
*Long Term Key (16-octet value MSO to LSO) 0x4c68384139f574d836bcf34e9dfb01bf*

**SKD_C (LSO to MSO) :0x13:0x02:0xF1:0xE0:0xDF:0xCE:0xBD:0xAC:**

**IV_C (LSO to MSO) :0x24:0xAB:0xDC:0xBA**

*LL_ENC_REQ 03 17 03 90 78 56 34 12 ef cd ab 74 24 13 02 f1 e0 df ce bd ac 24 ab dc ba*
*Length 0x17*
*Control Type 0x03*
*Rand 90 78 56 34 12 ef cd ab*

*Sample Data*

*EDIV 74 24*
*SKD_C 13 02 f1 e0 df ce bd ac*
*IV_C 24 ab dc ba*

**SKD_P (LSO to MSO) :0x79:0x68:0x57:0x46:0x35:0x24:0x13:0x02:**

**IV_P (LSO to MSO) :0xBE:0xBA:0xAF:0xDE**

*LL_ENC_RSP 0b 0d 04 79 68 57 46 35 24 13 02 be ba af de*
*Length 0x0D*
*Control Type 0x04*
*SKD_P 79 68 57 46 35 24 13 02*
*IV_P be ba af de*

IV = IV_P || IV_C
IV (LSO to MSO) :0x24:0xAB:0xDC:0xBA:0xBE:0xBA:0xAF:0xDE

*HCI_Long_Term_Key_Request(length 0x0D) – Peripheral event*
*HCI parameters as octet sequence: 05 01 08 90 78 56 34 12 ef cd ab 74 24*
*LE_Event_Code 0x05*
*Handle (2-octet value MSO to LSO) 0x0801*
*Random (8-octet value MSO to LSO) 0xabcdef1234567890*
*Encrypted Diversifier (2-octet value MSO to LSO) 0x2474*

*HCI_LE_Long_Term_Key_Request_Reply (length 0x12) – Peripheral command*
*HCI parameters as octet sequence: 01 08 bf 01 fb 9d 4e f3 bc 36 d8 74 f5 39 41 38 68 4c*
*Handle (2-octet value MSO to LSO) 0x0801*
*Key (16-octet value MSO to LSO) 0x4C68384139F574D836BCF34E9DFB01BF*

SKD = SKD_P || SKD_C
SKD (LSO to MSO)
:0x13:0x02:0xF1:0xE0:0xDF:0xCE:0xBD:0xAC:0x79:0x68:0x57:0x46:0x35:0x24:0x13:0x02:

SK = Encrypt(LTK, SKD)
SK (LSO to MSO)
:0x66:0xC6:0xC2:0x27:0x8E:0x3B:0x8E:0x05:0x3E:0x7E:0xA3:0x26:0x52:0x1B:0xAD:0x99:

*LL_START_ENC_REQ 07 01 05*
*Length 0x01*
*Control Type 0x05*

*Sample Data*

```
LL_START_ENC_RSP1 0f 05 9f cd a7 f4 48
  Length 0x05
  Control Type Encrypted:0x9F Clear:0x06
  MIC (32-bit value MSO to LSO) 0xCDA7F448 (MICs are sent MSO first on the air)

LL_START_ENC_RSP2 07 05 a3 4c 13 a4 15
  Length 0x05S
  Control Type Encrypted:0xA3 Clear:0x06
  MIC (32-bit value MSO to LSO) 0x4C13A415

HCI_ACL_Data packet Central's Host to Controller
  00 08 1b 00 17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33 34 35 36
37 38 39 30
    Handle (12-bit value MSO to LSO) 0x0800
    Data Total Length (16-bit value MSO to LSO) 0x001B (27 dec)
    Data (LSO to MSO) 17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33
34 35 36 37 38 39 30

LL_DATA1 0e 1f 7a 70 d6 64 15 22 6d f2 6b 17 83 9a 06 04 05 59 6b d6 56 4f 79 6b 5b
9c e6 ff 32 f7 5a 6d 33
  Length 0x1F (i.e. 27 + 4 = 31 dec)
  Data (LSO to MSO)
    Clear     17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33 34 35 36
37 38 39 30
    Encrypted 7a 70 d6 64 15 22 6d f2 6b 17 83 9a 06 04 05 59 6b d6 56 4f 79 6b 5b
9c e6 ff 32
  MIC (32-bit value MSO to LSO)   0xF75A6D33

HCI_ACL_Data_Packet Peripheral's Host to Controller
  01 08 1b 00 17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d
4e 4f 50 51
    Handle (12-bit value MSO to LSO) 0x0801
    Data Total Length (16-bit value MSO to LSO) 0x001B (27 dec)
    Data (LSO to MSO) 17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a
4b 4c 4d 4e 4f 50 51


LL_DATA2 06 1f f3 88 81 e7 bd 94 c9 c3 69 b9 a6 68 46 dd 47 86 aa 8c 39 ce 54 0d 0d
ae 3a dc df 89 b9 60 88
  Length 0x1F (i.e. 27 + 4 = 31 dec)
  Data (LSO to MSO)
    Clear 17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d
```

*Sample Data*

```
4e 4f 50 51
    Encrypted f3 88 81 e7 bd 94 c9 c3 69 b9 a6 68 46 dd 47 86 aa 8c 39 ce 54 0d 0d
ae 3a dc df
  MIC (32-bit value MSO to LSO)   0x89B96088
```

## 1.1 Encrypt Command

```
HCI_LE_Encrypt (length 0x20) - command
  HCI parameters as octet sequence: bf 01 fb 9d 4e f3 bc 36 d8 74 f5 39 41 38 68 4c 13
  02 f1 e0 df ce bd ac 79 68 57 46 35 24 13 02
    Key (16-octet value MSO to LSO):            0x4C68384139F574D836BCF34E9DFB01BF
    Plaintext_Data (16-octet value MSO to LSO): 0x02132435465768798acbdcedfe0f10213

HCI_Command_Complete (length 0x14) - event
  HCI parameters as octet sequence: 02 17 20 00 66 c6 c2 27 8e 3b 8e 05 3e 7e a3 26
  52 1b ad 99
    Num_HCI_Command_Packets: 0x02
    Command_Opcode (2-octet value MSO to LSO): 0x2017
    Status: 0x00
    Encrypted_Data (16-octet value MSO to LSO): 0x99ad1b5226a37e3e058e3b8e27c2c666
```

## 1.2 Derivation of the MIC and encrypted data

All B/X/A/S values below follow the notation: LSbyte to MSbyte & msbit to lsbit.

```
IV = DEAFBABEBADCAB24
SK = 99AD1B5226A37E3E058E3B8E27C2C666

1.START_ENC_RSP1 (packet 0, Central → Peripheral)
----------------

B0 = 49000000008024ABDCBABEBAAFDE0001
B1 = 00010300000000000000000000000000
B2 = 06000000000000000000000000000000
X1 = 712eaaaae60603521d245e50786eefe4
X2 = debc43782a022675fca0aa6f0854f1ab
X3 = 6399913fede5fa111bdb993bbfb9be06
=> MIC = 6399913f

A0 = 01000000008024ABDCBABEBAAFDE0000
A1 = 01000000008024ABDCBABEBAAFDE0001

S0 = ae3e6577f64a8f25408c9c10d53acf8e
S1 = 99190d88f4aa1b60b97ecfe6f5fee777
```

*Sample Data*

```
So, encrypted packet payload = 9F
    encrypted MIC = CDA7F448

Which results in the following packet:

LL_START_ENC_RSP1 - 0f 05 9f cd a7 f4 48
  Length: 05
  Control Type:
    Clear:     06
    Encrypted: 9f
  MIC: CD A7 F4 48

2.START_ENC_RSP2 (packet 0, Peripheral → Central)
----------------

B0 = 49000000000024ABDCBABEBAAFDE0001
B1 = 00010300000000000000000000000000
B2 = 06000000000000000000000000000000

X1 = ddc86e3094f0c29cf341ef4c2c1e0088
X2 = fe960f5c93fba45a53959842ea8a0c0a
X3 = db403db3a32f39156faf6a6b472e1010
=> MIC = db403db3

A0 = 01000000000024ABDCBABEBAAFDE0000
A1 = 01000000000024ABDCBABEBAAFDE0001

S0 = 975399a66acdc39124886930d7bca95f
S1 = a5add4127b2f43788ddc9cd86b0b89d2

So, encrypted packet payload = A3
    encrypted MIC = 4c13a415

Which results in the following packet:

LL_START_ENC_RSP2 07 05 a3 4c 13 a4 15
  Length: 05
  Control Type:
    Clear:     06
    Encrypted: A3
  MIC: 4c 13 a4 15
```

*Sample Data*

```
3. Data packet1 (packet 1, Central → Peripheral)
---------------

B0 = 49010000008024ABDCBABEBAAFDE001B
B1 = 00010200000000000000000000000000
B2 = 17006364656667686 96A6B6C6D6E6F70
B3 = 71313233343536373839300000000000

X1 = 7c688612996de101f3eacb68b443969c
X2 = e3f1ef5c30161c0a9ec07274a0757fc8
X3 = e7e346f5b7c8a6072890a60dcf4ec20a
X4 = 3db113320b182f9fed635db14cac2df0
=> MIC = 3db11332

A0 = 01010000008024ABDCBABEBAAFDE0000
A1 = 01010000008024ABDCBABEBAAFDE0001
A2 = 01010000008024ABDCBABEBAAFDE0002

S0 = caeb7e017296dd2fa9a2ce789179501a
S1 = 6d70b50070440a9a027de8f66b6a6a29
S2 = 1ae7647c4d5e6dabdec602404c302341

So, encrypted packet payload =
7A70D66415226DF26B17839A060405596BD6564F796B5B9CE6FF32
    encrypted MIC = F75A6D33

which results in the following packet:

LL_DATA1 0E 1F 7A 70 D6 64 15 22 6D F2 6B 17 83 9A 06 04 05 59 6B D6
56 4F 79 6B 5B 9C E6 FF 32 F7 5A 6D 33
  Length: 1F
  Data:
    Clear:     17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31
32 33 34 35 36 37 38 39 30
    Encrypted: 7A 70 D6 64 15 22 6D F2 6B 17 83 9A 06 04 05 59 6B D6
56 4F 79 6B 5B 9C E6 FF 32
  MIC: F7 5A 6D 33

4. Data packet2 (packet 1, Peripheral → Central)
---------------

B0 = 49010000000024ABDCBABEBAAFDE001B
B1 = 00010200000000000000000000000000
```

*Sample Data*

```
B2 = 17003736353433323130414243444546
B3 = 4748494A4B4C4D4E4F50510000000000

X1 = 714234d50d6f1da5663be3e78460ad87
X2 = 96df1d97959e6176ac215c7baf90c674
X3 = 6cc52c3dcecdc2fa81eb347887960673
X4 = a776a26be617366496c391e36f6374a1 => MIC = a776a26b

A0 = 01010000000024ABDCBABEBAAFDE0000
A1 = 01010000000024ABDCBABEBAAFDE0001
A2 = 01010000000024ABDCBABEBAAFDE0002

S0 = 2ecfc2e31e01875653c0f306fc7bfb96
S1 = e488b6d188a0faf15889e72a059902c0
S2 = edc470841f4140e0758c8e8f708399bd

So, encrypted packet payload =
F38881E7BD94C9C369B9A66846DD4786AA8C39CE540D0DAE3ADCDF
    encrypted MIC = 89B96088

Which results in the following packet:

LL_DATA2 06 1F F3 88 81 E7 BD 94 C9 C3 69 B9 A6 68 46 DD 47 86 AA 8C
39 CE 54 0D 0D AE 3A DC DF 89 B9 60 88
  Length: 1F
   Data:
    Clear:     17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48
49 4a 4b 4c 4d 4e 4f 50 51
    Encrypted: F3 88 81 E7 BD 94 C9 C3 69 B9 A6 68 46 DD 47 86 AA 8C
39 CE 54 0D 0D AE 3A DC DF

MIC: 89 B9 60 88
```

# 2 LE CODED PHY SAMPLE DATA

Whenever bits are specified, they are in transmission order irrespective of spacing.

## 2.1 Reference information packet

The reference packet is described as bytes in transmission order (the leftmost byte in a line is transmitted first). Inside a byte, bits are transmitted LSB first.

```
Access address:     D6 BE 89 8E
PDU:                00 03 42 4C 45
CRC:                29 0A CE
```

## 2.2 Forward Error Correction encoder

This data shows the bits input to and output by the FEC encoder and its internal state.

The encoder state is expressed in octal notation where the LSB represents the rightmost bit store in Figure 3.6 of [Vol 6] Part B, Section 3.3.1. The state specified is that after the bits are output and the shift operations have taken place.

```
Access address
Input:   0   1   1   0   1   0   1   1   0   1   1   1   1   1   0   1
         1   0   0   1   0   0   0   1   0   1   1   1   0   0   0   1
State:   0   4   6   3   5   2   5   6   3   5   6   7   7   7   3   5
         6   3   1   4   2   1   0   4   2   5   6   7   3   1   0   4
Output:  0 0 1 1 0 1 0 1 1 1 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 0 1 1 0 1 1
         1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1

CI
         If S=2              If S=8
Input:   1   0               0   0
State:   6   3               2   1
Output:  0 1 0 1             1 0 1 1

TERM1
         If S=2              If S=8
Input:   0   0   0           0   0   0
State:   1   0   0           0   0   0
Output:  0 0 1 1 0 0         1 1 0 0 0 0

PDU
```

*Sample Data*

```
Input:   0   0   0   0   0   0   0   0   1   1   0   0   0   0   0   0
         0   1   0   0   0   0   1   0   0   0   1   1   0   0   1   0
         1   0   1   0   0   0   1   0
State:   0   0   0   0   0   0   0   0   4   6   3   1   0   0   0   0
         0   4   2   1   0   0   4   2   1   0   4   6   3   1   4   2
         5   2   5   2   1   0   4   2
Output:  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0
         0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 0
         0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 0
CRC
Input:   1   0   0   1   0   1   0   0   0   1   0   1   0   0   0   0
         0   1   1   1   0   0   1   1
State:   5   2   1   4   2   5   2   1   0   4   2   5   2   1   0   0
         0   4   6   7   3   1   4   6
Output:  0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0
         0 0 1 1 0 1 1 0 1 0 0 0 0 0 0 1
TERM2
Input:   0   0   0
State:   3   1   0
Output:  0 1 0 0 1 1
```

## 2.3 Transmitted symbols (S=2)

```
Preamble
0011 1100 0011 1100 0011 1100 0011 1100 0011 1100 0011 1100
0011 1100 0011 1100 0011 1100 0011 1100

Access Address
0011 0011 1100 1100 0011 1100 0011 1100 1100 1100 0011 1100
0011 0011 1100 0011 0011 1100 1100 1100 1100 0011 1100 0011
0011 1100 0011 1100 1100 0011 1100 1100 1100 0011 0011 1100
0011 0011 0011 0011 1100 0011 1100 1100 1100 1100 1100 1100
1100 0011 0011 0011 1100 0011 1100 0011 1100 0011 0011 0011
1100 1100 1100 1100

CI
0011 1100 0011 1100

TERM1
0011 0011 1100 1100 0011 0011

PDU
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 0 1
```

*Sample Data*

```
1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1
1 0

CRC
0 0 0 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 0 1 1
0 1 0 0 0 0 0 0 1

TERM2
0 1 0 0 1 1
```

Total Packet Duration 510 µs

## 2.4 Transmitted symbols (S=8)

```
Preamble
0011 1100 0011 1100 0011 1100 0011 1100 0011 1100 0011 1100
0011 1100 0011 1100 0011 1100 0011 1100

Access Address
0011 0011 1100 1100 0011 1100 0011 1100 1100 1100 0011 1100
0011 0011 1100 0011 0011 1100 1100 1100 1100 0011 1100 0011
0011 1100 0011 1100 1100 0011 1100 1100 1100 0011 0011 1100
0011 0011 0011 0011 1100 0011 1100 1100 1100 1100 1100 1100
1100 0011 0011 0011 1100 0011 1100 0011 1100 0011 0011 0011
1100 1100 1100 1100

CI
1100 0011 1100 1100

TERM1
1100 1100 0011 0011 0011 0011

PDU
0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011 0011
0011 0011 0011 0011 1100 1100 0011 1100 0011 1100 0011 0011
1100 1100 0011 0011 0011 0011 0011 0011 0011 0011 1100 1100
1100 0011 1100 1100 1100 1100 0011 0011 1100 1100 1100 0011
1100 1100 1100 1100 1100 1100 0011 1100 0011 1100 0011 0011
0011 0011 1100 0011 0011 0011 0011 1100 0011 0011 0011 1100
1100 1100 1100 1100 1100 1100 1100 0011

CRC
0011 0011 0011 1100 1100 1100 0011 0011 1100 0011 0011 0011
```

*Sample Data*

```
0011 1100 1100 1100 1100 1100 1100 1100 1100 0011 0011 0011
0011 1100 1100 1100 1100 1100 0011 0011 0011 0011 1100 1100
0011 1100 1100 0011 1100 0011 0011 0011 0011 0011 0011 1100

TERM2
0011 1100 0011 0011 1100 1100
```

Total Packet Duration 912 µs

# 3 LE CHANNEL SELECTION ALGORITHM #2 SAMPLE DATA

This section contains two sets of sample data with different channel maps for the LE Channel Selection Algorithm #2.

The test access address is 0x8E89BED6, meaning the *channelIdentifier* is 0x305F.

## 3.1 Sample data 1 (37 used channels)

Channel map [36:0] = 0b11111_11111111_11111111_11111111_11111111.

The minimum channel distance d = 11.

| | Event Counter | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|
| ACL or Periodic advertising event / CIS or BIS subevent 1 | prn_e | 56857 | 1685 | 38301 | 27475 |
| | unmappedChannel | 25 | 20 | 6 | 21 |
| | mappedChannel[1] | 25 | 20 | 6 | 21 |
| Subevent 2 | remappingIndexOfLastUsed-Channel | 25 | 20 | 6 | 21 |
| | prnSubEvent_se | 11710 | 20925 | 6541 | 40400 |
| | subEventIndex | 1 | 36 | 18 | 4 |
| | mappedSubEventChannel[1] | 1 | 36 | 18 | 4 |
| Subevent 3 | prnSubEvent_se | 16649 | 11081 | 14597 | 30015 |
| | subEventIndex | 16 | 12 | 32 | 22 |
| | mappedSubEventChannel[1] | 16 | 12 | 32 | 22 |
| Subevent 4 | prnSubEvent_se | 38198 | 48920 | 62982 | 49818 |
| | subEventIndex | 36 | 34 | 21 | 8 |
| | mappedSubEventChannel[1] | 36 | 34 | 21 | 8 |

*Table 3.1: Sample data 1 (37 used channels)*

[1]These values are the general purpose channel indices actually used for transmitting and receiving the relevant event or subevent.

## 3.2 Sample data 2 (9 used channels)

Channel map [36:0] =0b11110_00000000_11100000_00000110_00000000.

The remapping table is [9, 10, 21, 22, 23, 33, 34, 35, 36].

*Sample Data*

The minimum channel distance d = 3.

| | Event Counter | 6 | 7 | 8 |
|---|---|---|---|---|
| ACL or Periodic advertising event / CIS or BIS subevent 1 | prn_e | 10975 | 5490 | 46970 |
| | unmappedChannel | 23 | 14 | 17 |
| | mappedChannel[1] | 23 | 9 | 34 |
| Subevent 2 | remappingIndexOfLastUsed-Channel | 4 | 0 | 6 |
| | prnSubEvent_se | 14383 | 4108 | 7196 |
| | subEventIndex | 7 | 3 | 0 |
| | mappedSubEventChannel[1] | 35 | 22 | 9 |
| Subevent 3 | prnSubEvent_se | 28946 | 45462 | 33054 |
| | subEventIndex | 2 | 8 | 5 |
| | mappedSubEventChannel[1] | 21 | 36 | 33 |
| Subevent 4 | prnSubEvent_se | 61038 | 64381 | 42590 |
| | subEventIndex | 8 | 5 | 1 |
| | mappedSubEventChannel[1] | 36 | 33 | 10 |

*Table 3.2: Sample data (9 Used channels)*

[1]These values are the general purpose channel indices actually used for transmitting and receiving the relevant event or subevent.

## 3.3 Sample data 3 (3 used channels)

Channel map [36:0] = 0b00000_00000000_00000000_00000001_00000110.

The remapping table is [1, 2, 8].

The minimum channel distance d = 1.

| | Event Counter | 11 | 12 | 13 |
|---|---|---|---|---|
| ACL or Periodic advertising event / CIS or BIS subevent 1 | prn_e | 8628 | 34748 | 22072 |
| | unmappedChannel | 7 | 5 | 20 |
| | mappedChannel[1] | 1 | 2 | 2 |
| Subevent 2 | remappingIndexOfLastUsed-Channel | 0 | 1 | 1 |
| | prnSubEvent_se | 30457 | 51913 | 13818 |
| | subEventIndex | 1 | 0 | 2 |
| | mappedSubventChannel[1] | 2 | 1 | 8 |

*Sample Data*

| | Event Counter | 11 | 12 | 13 |
|---|---|---|---|---|
| Subevent 3 | prnSubEvent_se | 35147 | 46599 | 60171 |
| | subEventIndex | 0 | 2 | 1 |
| | mappedSubventChannel[1] | 1 | 8 | 2 |
| Subevent 4 | prnSubEvent_se | 36952 | 37702 | 36470 |
| | subEventIndex | 2 | 1 | 0 |
| | mappedSubventChannel[1] | 8 | 2 | 1 |

*Table 3.3: Sample data (3 Used channels)*

[1]These values are the general purpose channel indices actually used for transmitting and receiving the relevant event or subevent.

# 4 COMPLETE PACKETS

Examples in this section assume use of the LE 1M PHY. For the LE 2M PHY, the packet will be identical except for the longer preamble and longer (in number of bits) Constant Tone Extension. For the LE Coded PHY, examples of converting from the form used on the uncoded PHYs to those used on the air are shown in Section 2.

Bit sequences are always in transmission order irrespective of spacing. Decimal and hexadecimal numbers are in their normal form.

## 4.1 Whitening sequences

The whitening sequence (see [Vol 6] Part B, Section 3.2) depends only on the physical channel index of the channel being used. The sequence repeats after 127 bits.

| Channel | First 64 bits of the whitening sequence |
|---|---|
| 0 | 00000010 01001101 00111101 11000011 11111000 11101100 01010010 11111010 |
| 1 | 10010001 00000010 01001101 00111101 11000011 11111000 11101100 01010010 |
| 2 | 01001011 11101010 10000101 10111100 11100101 01100110 00001101 10101110 |
| 3 | 11011000 10100101 11110101 01000010 11011110 01110010 10110011 00000110 |
| 4 | 00100110 10011110 11100001 11111100 01110110 00101001 01111101 01010000 |
| 5 | 10110101 11010001 10010001 00000010 01001101 00111101 11000011 11111000 |
| 6 | 01101111 00111001 01011001 10000011 01101011 10100011 00100010 00000100 |
| 7 | 11111100 01110110 00101001 01111101 01010000 10110111 10011100 10101100 |
| 8 | 00010000 00100100 11010011 11011100 00111111 10001110 11000101 00101111 |
| 9 | 10000011 01101011 10100011 00100010 00000100 10011010 01111011 10000111 |
| 10 | 01011001 10000011 01101011 10100011 00100010 00000100 10011010 01111011 |
| 11 | 11001010 11001100 00011011 01011101 00011001 00010000 00100100 11010011 |
| 12 | 00110100 11110111 00001111 11100011 10110001 01001011 11101010 10000101 |
| 13 | 10100111 10111000 01111111 00011101 10001010 01011111 01010100 00101101 |
| 14 | 01111101 01010000 10110111 10011100 10101100 11000001 10110101 11010001 |
| 15 | 11101110 00011111 11000111 01100010 10010111 11010101 00001011 01111001 |
| 16 | 00001011 01111001 11001010 11001100 00011011 01011101 00011001 00010000 |
| 17 | 10011000 00110110 10111010 00110010 00100000 01001001 10100111 10111000 |
| 18 | 01000010 11011110 01110010 10110011 00000110 11010111 01000110 01000100 |
| 19 | 11010001 10010001 00000010 01001101 00111101 11000011 11111000 11101100 |
| 20 | 00101111 10101010 00010110 11110011 10010101 10011000 00110110 10111010 |
| 21 | 10111100 11100101 01100110 00001101 10101110 10001100 10001000 00010010 |
| 22 | 01100110 00001101 10101110 10001100 10001000 00010010 01101001 11101110 |
| 23 | 11110101 01000010 11011110 01110010 10110011 00000110 11010111 01000110 |
| 24 | 00011001 00010000 00100100 11010011 11011100 00111111 10001110 11000101 |
| 25 | 10001010 01011111 01010100 00101101 11100111 00101011 00110000 01101101 |

*Sample Data*

```
26  01010000 10110111 10011100 10101100 11000001 10110101 11010001 10010001
27  11000011 11111000 11101100 01010010 11111010 10100001 01101111 00111001
28  00111101 11000011 11111000 11101100 01010010 11111010 10100001 01101111
29  10101110 10001100 10001000 00010010 01101001 11101110 00011111 11000111
30  01110100 01100100 01000000 10010011 01001111 01110000 11111110 00111011
31  11100111 00101011 00110000 01101101 01110100 01100100 01000000 10010011
32  00000110 11010111 01000110 01000100 00001001 00110100 11110111 00001111
33  10010101 10011000 00110110 10111010 00110010 00100000 01001001 10100111
34  01001111 01110000 11111110 00111011 00010100 10111110 10101000 01011011
35  11011100 00111111 10001110 11000101 00101111 10101010 00010110 11110011
36  00100010 00000100 10011010 01111011 10000111 11110001 11011000 10100101
37  10110001 01001011 11101010 10000101 10111100 11100101 01100110 00001101
38  01101011 10100011 00100010 00000100 10011010 01111011 10000111 11110001
39  11111000 11101100 01010010 11111010 10100001 01101111 00111001 01011001
```

## 4.2 Advertising Physical channel PDUs

### 4.2.1 Legacy advertising PDUs

Example: ADV_NONCONN_IND PDU

```
PDU Type: 2
ChSel: RFU
TxAdd: 1 (random)
RxAdd: RFU
AdvA: 0xC1A2A3A4A5A6 (a static device address)
AdvData: (3 octets) 0x01 0x02 0x03
Channel index: 38

PDU header: 0100 0 0 1 0 10010000
PDU body: 01100101 10100101 00100101 11000101 01000101 10000011
10000000 01000000 11000000
CRC: 10110101 00101101 11010111

PDU and CRC before whitening:
01000010 10010000 01100101 10100101 00100101 11000101 01000101
10000011 10000000 01000000 11000000 10110101 00101101 11010111

PDU and CRC after whitening:
00101001 00110011 01000111 10100001 10111111 10111110 11000010
01110010 01011000 11100101 00110101 11110111 11110011 10100101

Complete packet:
```

*Sample Data*

```
01010101
01101011 01111101 10010001 01110001
00101001 00110011 01000111 10100001 10111111 10111110 11000010
01110010 01011000 11100101 00110101 11110111 11110011 10100101
```

### 4.2.2 Extended advertising PDUs

Example: connectable undirected AUX_ADV_IND PDU with AdvA, ADI, TxPower, and AdvData fields.

```
PDU Type: 7
ChSel: RFU
TxAdd: 0 (public)
RxAdd: RFU
AdvA: 0xA9AAABACADAE (a public device address)
Advertising DID: 0xABC
Advertising SID: 0xE
TxPower: 0xD6 (-42 dBm)
AdvData: (5 octets) 0x05 0x07 0x09 0x0B 0x0D
Channel index: 7
PDU header: 1110 0 0 0 0 00001000

Extended header length: 010100
AdvMode: 10

Extended header:
    Flags: 1 0 0 1 0 0 1 0
    AdvA: 01110101 10110101 00110101 11010101 01010101 10010101
    ADI: 001111010101 0111
    TxPower: 01101011

AdvData: 10100000 11100000 10010000 11010000 10110000

PDU: 11100000 00001000 01010010 10010010 01110101 10110101 00110101
11010101 01010101 10010101 00111101 01010111 01101011 10100000
11100000 10010000 11010000 10110000

CRC: 00011011 11000100 01110101

PDU and CRC after whitening:
00011100 01111110 01111011 11101111 00100101 00000010 10101001
01111001 10010100 00100000 11101100 11000110 01101001 11101101
```

*Sample Data*

```
11011101 01010011 00101000 01011100 01001001 00111110 11010100

Complete packet:
01010101
01101011 01111101 10010001 01110001
00011100 01111110 01111011 11101111 00100101 00000010 10101001
01111001 10010100 00100000 11101100 11000110 01101001 11101101
11011101 01010011 00101000 01011100 01001001 00111110 11010100
```

## 4.3 Data channel PDUs

The examples in this section are unencrypted. Examples of the encryption process can be seen in Section 1.

### 4.3.1 LL data PDUs

```
Access address: 0xAA08192B
CRCInit: 0xC4C181
LLID: 2
NESN: 1
SN: 0
MD: 1
Payload: 0x01 0x02 0x03 0x04 0x05
No Constant Tone Extension
Channel index 16

PDU header: 01 1 0 1 0 00 10100000

PDU: 01101000 10100000 10000000 01000000 11000000 00100000 10100000

CRC: 10100010 00001011 01001011

PDU and CRC after whitening:
01100011 11011001 01001010 10001100 11011011 01111101 10111001
10110010 00101111 10011000

Complete packet:
10101010
11010100 10011000 00010000 01010101
01100011 11011001 01001010 10001100 11011011 01111101 10111001
10110010 00101111 10011000
```

*Sample Data*

### 4.3.2 LL control PDUs

```
Access address: 0xAA173C42
CRCInit: 0xCD3F6C
Type: LL_CHANNEL_MAP_IND
NESN: 0
SN: 1
MD: 1
Map: channels 2, 5, and 33 to 36 unused, all other channels used
Instant: 0x4321
AoD Constant Tone Extension, length 40 µs, 2 µs switching and sampling
Channel index 29

PDU header: 11 0 1 1 1 00 00010000 10100 0 01

PDU: 11011100 00010000 10100001 10000000 11011011 11111111 11111111
11111111 10000000 10000100 11000010

CRC: 10011100 01000111 00101001

PDU and CRC after whitening:
01110010 10011100 00101001 10010010 10110010 00010001 11100000
00111000 11100010 00010011 00010111 10010111 00111110 11100011

Complete packet:
01010101
01000010 00111100 11101000 01010101
01110010 10011100 00101001 10010010 10110010 00010001 11100000
00111000 11100010 00010011 00010111 10010111 00111110 11100011
11111111 11111111 11111111 11111111 11111111
```

*Sample Data*

# 5 ACCESS ADDRESS GENERATION FOR BISES

The Access Addresses for the various BISes in a BIG with the SeedAccessAddress 0x78E52493 are:

| Num_BIS | Access Address |
|---|---|
| 1 | 0x85E32493 |
| 2 | 0x79D52493 |
| 3 | 0x86752493 |
| 4 | 0x7A572493 |
| 5 | 0x85F12493 |
| 6 | 0x79D32493 |
| 7 | 0x86732493 |
| 8 | 0x7B652493 |
| 9 | 0x85C72493 |
| 10 | 0x78E12493 |
| 11 | 0x86412493 |
| 12 | 0x7B632493 |
| 13 | 0x85D52493 |
| 14 | 0x78F72493 |
| 15 | 0x86572493 |
| 16 | 0x7B712493 |
| 17 | 0x85D32493 |
| 18 | 0x78C52493 |
| 19 | 0x87652493 |
| 20 | 0x7B472493 |
| 21 | 0x84E12493 |
| 22 | 0x78C32493 |
| 23 | 0x87632493 |
| 24 | 0x7B552493 |
| 25 | 0x84F72493 |
| 26 | 0x78D12493 |
| 27 | 0x87712493 |
| 28 | 0x7B532493 |

*Sample Data*

| Num_BIS | Access Address |
|---|---|
| 29 | 0x84C52493 |
| 30 | 0x79E72493 |
| 31 | 0x87472493 |

*Table 5.1: Access Address generation for BISes*

The Access Address of the BIG Control logical link is 0x7A412493.

# 6 GROUP SESSION KEY DERIVATION FOR BIG

In each data set in this section, the bytes are ordered from most significant on the left to least significant on the right.

### Group Session Key Derivation Function h8

```
K               ec0234a3 57c8ad05 341010a6 0a397d9b
S               1536d18d e3d20df9 9b7044c1 2f9ed5ba
keyID           cc030148
IK              fe77ab4e fa982991 c1486a3b 281fd4bc
h8              e5e5beba ae7228e7 22a38904 ed350f6d
```

### Derivation of Group Session Key from Broadcast Code

```
Bluetooth Broadcast Code: "Børne House"

Broadcast Code 00000000 6573756f 4820656e 72b8c342
GSKD            55188b3d 32f6bb9a 900afcfb eed4e72a
"BIG1"          00000000 00000000 00000000 42494731
"BIG2"          42494732
"BIG3"          42494733
IGLTK           4c0dd74c 2b19aa95 d8982385 5f1001b8
GLTK            c4cd4b83 49b5a18a 02de6620 9017aed3
GSK             be2a16fc 7ac464e7 52301bcc c818812c
```

# 7 DETERMINISTIC RANDOM BIT GENERATOR SAMPLE DATA

This section contains sample data for the Deterministic Random Bit Generator.

The following scenarios provide sample entropy input for the h9() instantiation function, which then invokes several subordinate security toolbox functions described in [Vol 6] Part E, Section 3. After this, CS procedure counters, step counters, transaction IDs, and transaction counters are incremented for further iterations of sample data. The CS Backtracking procedure is also invoked to generate additional sample data.

Unless otherwise noted, all fields are displayed in leftmost (MSO) to rightmost (LSO) orientation as described in [Vol 6] Part E, Section 3.1.1.

```
******************************** INSTANTIATION FUNCTION ********************************

h9() instantiation

************************************************************************************

Entropy input Peripheral (CS_IV_P):          E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):             3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                       E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral(CS_IN_P):                   9F F4 77 C1

Nonce Central(CS_IN_C):                      86 73 84 0D

Nonce (CS_IN):                               9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):    64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):              C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

    ********************** f8 function start **********************

        *************************** f7 function start ***************************
```

*Sample Data*

```
         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
00 00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96
78 68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

         f7 K2 output:   8B 2B 06 DC 52 2D 3E 0A F0 A5 0C AF 48 10 E0 35

        *************************** f7 function end ****************************

        *************************** f7 function start **************************

         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
00 00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96
78 68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

         f7 X output:    A3 4F BE 57 F8 F9 7E 34 9D 15 A3 76 79 60 74 64

        *************************** f7 function end ****************************

     f8 SM output:      B6 02 B1 B2 8C 6F 0A 3D DA E6 37 B4 84 25 08 7D DC 18 8C 89
A1 B0 CD FD A1 E8 FC 66 C9 99 97 50

    ************************ f8 function end ***********************

********************************** FIRST PROCEDURE ***********************************

K:               EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:               DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

*******************************************************************************************

 Proc. cnt. = 0; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:           DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28
```

*Sample Data*

```
 DRBG octets:    79 74 1F D1 8F 57 7B 45 D0 9A 66 5A 7F 1F 28 58


 Proc. cnt. = 0; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  1

 Vcnt:           DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 29

 DRBG octets:    D1 C1 D0 5A 40 B4 C4 81 EF BB 39 B2 61 D2 9C 4E


 Proc. cnt. = 0; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  2

 Vcnt:           DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 2A

 DRBG octets:    71 FD E8 68 E8 CA CA D1 18 E5 9B 18 5C EE FC 17


 Proc. cnt. = 0; Step cnt. = 10; Transaction ID = 4; Transaction cnt. =  0

 Vcnt:           DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 35 6D 28

 DRBG octets:    FF F8 74 38 4E 1C B9 D3 E6 9F 1F 2C 2A 22 CF 63

********************************** NEW PROCEDURE ***********************************

Procedure counter increment; f9() update:

V in:                                                DF 90 56 47 C1 06 6E 6F 52 C0
3E DF B8 2B 69 28

V'=V[127:32] || V[31:16] || V[15:8]+T_ID9 || V[7:0]  in:  DF 90 56 47 C1 06 6E 6F 52 C0
3E DF B8 2B 72 28

********************************************************************************

K:              F3 F4 CF 24 5F CE 86 B3 C6 9F AE 60 28 7E B5 AE

V:              08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 ED 54 B1

********************************************************************************

 Proc. cnt. = 1; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  0
```

*Sample Data*

```
 Vcnt:           08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 ED 54 B1

 DRBG octets:    5F 1B 11 5A 00 40 B4 2C 2F 8F B1 0C 9B 6F D7 AC


 Proc. cnt. = 1; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  1

 Vcnt:           08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 ED 54 B2

 DRBG octets:    55 73 A6 A5 20 91 B1 58 99 C8 64 AB 2F B0 E2 02


 Proc. cnt. = 1; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  2

 Vcnt:           08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 ED 54 B3

 DRBG octets:    46 20 D2 01 B7 41 9D 82 48 68 D5 A7 C8 20 BA 2C


 Proc. cnt. = 1; Step cnt. = 14; Transaction ID = 4; Transaction cnt. =  0

 Vcnt:           08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 FB 58 B1

 DRBG octets:    B4 20 00 40 63 B1 13 A2 3E D0 43 D0 3B 6E E7 34

********************************** NEW PROCEDURE ***********************************

Procedure counter increment; f9() update:

V in:                                                    08 56 98 94 89 1F 2F 30
1E EC DB 38 B6 ED 54 B1

V'=V[127:32] ||   V[31:16]  ||  V[15:8]+T_ID9   ||  V[7:0]  in:  08 56 98 94 89 1F 2F 30
1E EC DB 38 B6 ED 5D B1

********************************************************************************************

K:              A9 AA 68 C2 1D F4 09 00 1A F9 27 AC 14 C8 18 96

V:              16 07 DE 6C FC 80 D2 18 64 F7 08 A9 9C 49 39 96

********************************************************************************************
```

*Sample Data*

```
 Proc. cnt. = 2; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:          16 07 DE 6C FC 80 D2 18 64 F7 08 A9 9C 49 39 96

 DRBGoctets:    6A EA B7 5D 82 52 58 31 42 2C E2 F4 2E 85 07 88


 Proc. cnt. = 2; Step cnt. = 1; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:          16 07 DE 6C FC 80 D2 18 64 F7 08 A9 9C 4A 39 96

 DRBG octets:   14 40 EF 44 74 AF D8 49 32 FE 37 B0 A4 10 F2 3A


 Proc. cnt. = 2; Step cnt. = 2; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:          16 07 DE 6C FC 80 D2 18 64 F7 08 A9 9C 4B 39 96

 DRBG octets:   2A 83 EF 11 CC E5 99 31 4E FB 29 03 F4 DF D4 B8


 Proc. cnt. = 2; Step cnt. = 3; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:          16 07 DE 6C FC 80 D2 18 64 F7 08 A9 9C 4C 39 96

 DRBG octets:   19 6B E2 B6 43 98 53 FE D4 96 67 54 E1 40 15 1E

********************************** NEW PROCEDURE ***********************************

Procedure counter increment; f9() update:

V in:                                                    16 07 DE 6C FC 80 D2 18
64 F7 08 A9 9C 49 39 96

V'=V[127:32] ||   V[31:16]  ||  V[15:8]+T_ID9   ||  V[7:0]  in:  16 07 DE 6C FC 80 D2 18
64 F7 08 A9 9C 49 42 96

************************************************************************************

K:               77 71 3B B1 81 C3 B6 44 29 E8 98 B8 65 9B 85 A9

V:               D5 88 73 99 44 68 36 96 A9 3D 36 19 3C DA BA 11
```

*Sample Data*

```
********************************************************************************

 Proc. cnt. = 3; Step cnt. = 0; Transaction ID = 0; Transaction cnt. =  0

 Vcnt:          D5 88 73 99 44 68 36 96 A9 3D 36 19 3C DA BA 11

 DRBG octets:   58 04 C1 73 69 C5 64 DC 73 D6 E7 4E A0 C6 E1 66
```

# 8 CHANNEL SOUNDING PROCEDURE SAMPLE DATA

This section contains sample data for illustrative Channel Sounding procedures that make use of the Deterministic Random Bit Generator.

The data is generated as if the configuration and timing parameters listed in each subsection were negotiated or derived during the CS Configuration procedure ([Vol 6] Part B, Section 5.1.25 and [Vol 6] Part B, Section 2.4.2.45) and the CS Start procedure ([Vol 6] Part B, Section 5.1.26 and [Vol 6] Part B, Section 2.4.2.47).

The following scenarios provide sample entropy input for the h9() instantiation function, which then invokes several subordinate security toolbox functions described in [Vol 6] Part E, Section 3. After this, CS procedure counters, step counters, transaction IDs, and transaction counters are incremented for further iterations of sample data.

Unless otherwise noted, all fields are displayed in leftmost (MSO) to rightmost (LSO) orientation as described in [Vol 6] Part E, Section 3.1.1.

## 8.1 Channel Selection Algorithm #3b

### 8.1.1 Set 1

From the CS Configuration procedure:

- ChM = 0x00 00 00 00 00 00 00 1F FF FC
- ChM_Repetition = 2
- Main_Mode = 0x02
- Sub_Mode = 0x03
- Main_Mode_Min_Steps = 2
- Main_Mode_Max_Steps = 6
- Main_Mode_Repetition = 0
- Mode_0_Steps = 3
- RTT_Type = 0x01
- Role = 0b01
- ChSel = 0

*Sample Data*

From the CS Start procedure:

- Subevent_Len – setup to include at least 20 steps
- Procedure_Count = 1
- ACI = 7

```
******************************** INSTANTIATION FUNCTION ********************************

h9() instantiation

************************************************************************************

Entropy input Peripheral (CS_IV_P):            E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):               3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                         E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA
0B 3B

Nonce Peripheral (CS_IN_P):                    9F F4 77 C1

Nonce Central(CS_IN_C):                        86 73 84 0D

Nonce (CS_IN):                                 9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P):   C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):      64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):                C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68
F1 43

    ********************** f8 function start **********************

        *************************** f7 function start ***************************

         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00
00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78
68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

*Sample Data*

```
         f7 K2 output:   8B 2B 06 DC 52 2D 3E 0A F0 A5 0C AF 48 10 E0 35

        ************************** f7 function end ****************************

        ************************** f7 function start **************************

         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00
00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78
68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

         f7 X output:    A3 4F BE 57 F8 F9 7E 34 9D 15 A3 76 79 60 74 64

        ************************** f7 function end ****************************

     f8 SM output:      B6 02 B1 B2 8C 6F 0A 3D DA E6 37 B4 84 25 08 7D DC 18 8C 89 A1 B0
CD FD A1 E8 FC 66 C9 99 97 50


    ********************** f8 function end **********************

********************************** INITIAL K and V ***********************************

K:               EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:               DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

******************************************************************************************

************************************ CS SEQUENCE *************************************

************************************ Channel map *************************************

Bit-map:           00 00 00 00 00 00 00 1F FF FC

Filtered channels: 2 3  4  5 6  7  8 9  10  11 12  13  14 15  16  17 18  19  20

******************************************************************************************
```

*Sample Data*

```
Step=0 | Mode=0

T_ID 1 - mode0 channel (0 bits):    //   -  Range = // (decimal); hr1 = 0 (decimal)

**********************************************************************

New DRBG octets:   FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

**********************************************************************

T_ID 1 - mode0 channel (8 bits):    FF   -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):    BC   -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):    C1   -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):    CA   -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):    39   -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):    A6   -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):    9D   -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):    C4   -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):    07   -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8 bits):    38   -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):    EF   -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):    33   -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):    D9   -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):    D1   -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 1 - mode0 channel (8 bits):    35   -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):    32   -  Range = 17 (decimal); hr1 = 3 (decimal)

**********************************************************************
```

*Sample Data*

```
New DRBG octets:    51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

**********************************************************************

T_ID 1 - mode0 channel (8 bits):    51   -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):    EE   -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 - Shuffled channels:         11 7 14 18 9 19 10 8 5 2 4 15 16 13 12 6 17 20 3

**********************************************************************

New DRBG octets:    0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D F0 79 BC 3A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D
F0 79 BC 3A

T_ID 5 -     Initiator first  32-bit score: 12

T_ID 5 -     Initiator second 32-bit score: 8

T_ID 5 -     Initiator chosen PN pattern:   6C 37 6A B8

Access Address 0x6C376AB8 : 00011101 01010110 11101100 00110110  (Tx bit order, left
to right)
Followed by Trailer: 1010 (Tx bit order, left to right)

T_ID 5 -     Reflector first  32-bit score: 16

T_ID 5 -     Reflector second 32-bit score: 16

T_ID 5 -     Reflector chosen PN pattern:   F0 79 BC 3A

Access Address 0xF079BC3A : 01011100 00111101 10011110 00001111 (Tx bit order, left
to right)
Followed by Trailer: 0101 (Tx bit order, left to right)

Step=1 | Mode=0

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   01 1C AE 4E 99 47 D1 B5 D0 6A CD DA C7 F8 F9 37

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             01 1C AE 4E 99 47 D1 B5 D0 6A CD DA
C7 F8 F9 37

T_ID 5 -     Initiator first  32-bit score: 6

T_ID 5 -     Initiator second 32-bit score: 8

T_ID 5 -     Initiator chosen PN pattern:   01 1C AE 4E

T_ID 5 -     Reflector first  32-bit score: 10

T_ID 5 -     Reflector second 32-bit score: 14

T_ID 5 -     Reflector chosen PN pattern:   D0 6A CD DA

Step=2 | Mode=0

**********************************************************************

New DRBG octets:   64 06 12 14 86 05 AF EA 71 55 0B DD 28 94 2F 38

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             64 06 12 14 86 05 AF EA 71 55 0B DD
28 94 2F 38

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 12

T_ID 5 -     Initiator chosen PN pattern:   64 06 12 14

T_ID 5 -     Reflector first  32-bit score: 28

T_ID 5 -     Reflector second 32-bit score: 8

T_ID 5 -     Reflector chosen PN pattern:   28 94 2F 38
```

*Sample Data*

```
Step=3 | Mode=2

T_ID 0 - non-mode0 channel (0 bits):    //  -  Range = // (decimal); hr1 = 0 (decimal)

**********************************************************************

New DRBG octets:   32 73 A2 2D 88 47 DA 6A 1C 2C BD 8C E8 96 79 62

**********************************************************************

T_ID 0 - non-mode0 channel (8 bits):    32  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    73  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    A2  -  Range = 4 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    2D  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    88  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    47  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    DA  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    6A  -  Range = 9 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    1C  -  Range = 10 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    2C  -  Range = 11 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    BD  -  Range = 12 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    8C  -  Range = 13 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    E8  -  Range = 14 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    96  -  Range = 15 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    79  -  Range = 16 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    62  -  Range = 17 (decimal); hr1 = 6 (decimal)

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   AB D1 0C 59 DD BD D9 F3 05 20 F7 D0 EF 37 8E BE

**********************************************************************

T_ID 0 - non-mode0 channel (8 bits):    AB  -  Range = 18 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    D1  -  Range = 19 (decimal); hr1 = 15 (decimal)

T_ID 0 -   Shuffled channels: 6 12 5 10 3 2 18 17 16 8 11 7 19 4 13 20 9 15 14

**********************************************************************

New DRBG octets:  F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

**********************************************************************

T_ID 2 - Submode insertion (8 bits):    F8  -  Range = 5 (decimal); hr1 = 4 (decimal)

**********************************************************************

New DRBG octets:   3B 8E 7E 2A A8 D7 CF F1 CC 69 BE 13 75 1F B3 95

**********************************************************************

T_ID 3 - TPM extension (2 bits):        00

**********************************************************************

New DRBG octets:   B4 8D B4 8C 41 33 DE 1D 71 B7 46 A1 61 8C 4D D9

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=4 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=5 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=6 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   8C  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=7 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   41  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=8 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   33  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=9 | Mode=3

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   DE  -  Range = 24 (decimal); hr1 = 20 (decimal)

**********************************************************************

New DRBG octets:   A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A 6B 2E B7 48

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A
6B 2E B7 48

T_ID 5 -     Initiator first  32-bit score: 8

T_ID 5 -     Initiator second 32-bit score: 6

T_ID 5 -     Initiator chosen PN pattern:   16 9E ED DC
```

*Sample Data*

```
T_ID 5 -     Reflector first  32-bit score: 20

T_ID 5 -     Reflector second 32-bit score: 14

T_ID 5 -     Reflector chosen PN pattern:   6B 2E B7 48

**********************************************************************

New DRBG octets:   72 26 4D 65 1E 54 6E 3B A1 7E 46 61 A5 00 90 67

**********************************************************************

T_ID 6 - SS marker position initiator (8 bits):  72  -  Range = 29 (decimal); hr1 = 12
(decimal)

T_ID 6 - SS marker position reflector (8 bits):  26  -  Range = 29 (decimal); hr1 = 4
(decimal)

**********************************************************************

New DRBG octets:    7D CB DA 7B 6E 5E A2 0C 32 0D C2 4F 8E 2C 72 5D

**********************************************************************

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

Initiator Sounding Sequence: 0101 0101 0101 1100 0101 0101 0101 0101 (Tx bit order,
left to right)

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Reflector Sounding Sequence: 0101 0011 0101 0101 0101 0101 0101 0101 (Tx bit order,
left to right)

Step=10 | Mode=2

T_ID 2 - Submode insertion (8 bits):    82  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   1D  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=11 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   71  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=12 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   B7  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=13 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   46  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=14 | Mode=3

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   A1  -  Range = 24 (decimal); hr1 = 15 (decimal)

**********************************************************************

New DRBG octets:   F7 21 97 86 9D 82 D2 EC 80 69 7C 5B 57 17 64 70

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             F7 21 97 86 9D 82 D2 EC 80 69 7C 5B
57 17 64 70

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 12

T_ID 5 -     Initiator chosen PN pattern:   F7 21 97 86

T_ID 5 -     Reflector first  32-bit score: 14

T_ID 5 -     Reflector second 32-bit score: 10

T_ID 5 -     Reflector chosen PN pattern:   57 17 64 70
```

*Sample Data*

```
T_ID 6 - SS marker position initiator (8 bits):    4D  -  Range = 29 (decimal); hr1 = 8
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    65  -  Range = 29 (decimal); hr1 = 11
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=15 | Mode=2

T_ID 2 - Submode insertion (8 bits):    2A  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   61  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=16 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   8C  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=17 | Mode=3

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   4D   -  Range = 24 (decimal); hr1 = 7 (decimal)

**********************************************************************

New DRBG octets:   17 46 62 CD 92 22 DB 8B 0C 22 D1 19 C8 C5 97 E5

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             17 46 62 CD 92 22 DB 8B 0C 22 D1 19
C8 C5 97 E5

T_ID 5 -     Initiator first  32-bit score: 14

T_ID 5 -     Initiator second 32-bit score: 16
```

*Sample Data*

```
T_ID 5 -     Initiator chosen PN pattern:   17 46 62 CD

T_ID 5 -     Reflector first  32-bit score: 6

T_ID 5 -     Reflector second 32-bit score: 6

T_ID 5 -     Reflector chosen PN pattern:   C8 C5 97 E5

T_ID 6 - SS marker position initiator (8 bits):    1E  -  Range = 29 (decimal); hr1 = 3
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    54  -  Range = 29 (decimal); hr1 = 9
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=18 | Mode=2

T_ID 2 - Submode insertion (8 bits):    54  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   D9  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=19 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

**********************************************************************

New DRBG octets:    CE 5A DE D7 79 BC 1F DB E5 AE A3 FC 92 0B 05 31

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   CE  -  Range = 24 (decimal); hr1 = 19 (decimal)

Step=20 | Mode=0

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   33 3E 3C 9D 31 E0 44 B6 7B 3F 20 38 AF D6 73 AC

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             33 3E 3C 9D 31 E0 44 B6 7B 3F 20 38
AF D6 73 AC

T_ID 5 -     Initiator first  32-bit score: 22

T_ID 5 -     Initiator second 32-bit score: 10

T_ID 5 -     Initiator chosen PN pattern:   31 E0 44 B6

T_ID 5 -     Reflector first  32-bit score: 12

T_ID 5 -     Reflector second 32-bit score: 8

T_ID 5 -     Reflector chosen PN pattern:   AF D6 73 AC

Step=21 | Mode=0

**********************************************************************

New DRBG octets:   AF 9F 78 48 26 29 96 A3 85 E6 60 4F 82 3B 87 C2

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             AF 9F 78 48 26 29 96 A3 85 E6 60 4F
82 3B 87 C2

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 14

T_ID 5 -     Initiator chosen PN pattern:   AF 9F 78 48

T_ID 5 -     Reflector first  32-bit score: 12

T_ID 5 -     Reflector second 32-bit score: 18

T_ID 5 -     Reflector chosen PN pattern:   85 E6 60 4F

Step=22 | Mode=0
```

*Sample Data*

```
**********************************************************************

New DRBG octets:   3C 60 3B D2 E1 A3 41 67 B3 78 97 ED B4 B2 72 BF

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             3C 60 3B D2 E1 A3 41 67 B3 78 97 ED
B4 B2 72 BF

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 8

T_ID 5 -     Initiator chosen PN pattern:   E1 A3 41 67

T_ID 5 -     Reflector first  32-bit score: 12

T_ID 5 -     Reflector second 32-bit score: 12

T_ID 5 -     Reflector chosen PN pattern:   B4 B2 72 BF

Step=23 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   5A  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=24 | Mode=3

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   DE  -  Range = 24 (decimal); hr1 = 20 (decimal)

**********************************************************************

New DRBG octets:   A9 83 4E D5 F0 71 EC A6 35 17 86 10 A6 EE AA 04

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             A9 83 4E D5 F0 71 EC A6 35 17 86 10
A6 EE AA 04
```

*Sample Data*

```
T_ID 5 -     Initiator first  32-bit score: 14

T_ID 5 -     Initiator second 32-bit score: 14

T_ID 5 -     Initiator chosen PN pattern:   F0 71 EC A6

T_ID 5 -     Reflector first  32-bit score: 10

T_ID 5 -     Reflector second 32-bit score: 16

T_ID 5 -     Reflector chosen PN pattern:   35 17 86 10

T_ID 6 - SS marker position initiator (8 bits):    6E  -  Range = 29 (decimal); hr1 = 12
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    3B  -  Range = 29 (decimal); hr1 = 6
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=25 | Mode=2

T_ID 0 - non-mode0 channel (0 bits):    //  -  Range = // (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    0C  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    59  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    DD  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    BD  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    D9  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    F3  -  Range = 7 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    05  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    20  -  Range = 9 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    F7  -  Range = 10 (decimal); hr1 = 9 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):    D0  -  Range = 11 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    37  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    8E  -  Range = 14 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    BE  -  Range = 15 (decimal); hr1 = 11 (decimal)

**********************************************************************

New DRBG octets:    73 F3 11 6C E1 09 5A 29 BF 4D E7 AD D7 CC 2C D4

**********************************************************************

T_ID 0 - non-mode0 channel (8 bits):    73  -  Range = 16 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    F3  -  Range = 17 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):    11  -  Range = 18 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (16 bits):   6C E1 -  Range = 19 (decimal); hr1 = 16 (decimal)

T_ID 0 -    Shuffled channels: 9 19 14 6 5 7 8 17 12 11 4 16 2 3 13 15 20 10 18

T_ID 2 - Submode insertion (8 bits):    E0  -   Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   D7  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=26 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   79  -  Range = 24 (decimal); hr1 = 11 (decimal)

Step=27 | Mode=2

T_ID 3 - TPM extension (2 bits):        40
```

*Sample Data*

```
T_ID 4 - Antenna path perm. (8 bits):   BC  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=28 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   1F  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=29 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   DB  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=30 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   E5  -  Range = 24 (decimal); hr1 = 21 (decimal)

Step=31 | Mode=3

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   AE  -  Range = 24 (decimal); hr1 = 16 (decimal)

**********************************************************************

New DRBG octets:   C0 F4 37 CB 2E D8 01 47 35 AF 5E 5D 02 0B 67 A1

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             C0 F4 37 CB 2E D8 01 47 35 AF 5E 5D
02 0B 67 A1

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 16

T_ID 5 -     Initiator chosen PN pattern:   C0 F4 37 CB

T_ID 5 -     Reflector first  32-bit score: 12
```

*Sample Data*

```
T_ID 5 -     Reflector second 32-bit score: 16

T_ID 5 -     Reflector chosen PN pattern:   35 AF 5E 5D

T_ID 6 - SS marker position initiator (8 bits):    A1  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    7E  -  Range = 29 (decimal); hr1 = 14
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=32 | Mode=2

T_ID 2 - Submode insertion (8 bits):    7C  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   A3  -  Range = 24 (decimal); hr1 = 15 (decimal)

Step=33 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   FC  -  Range = 24 (decimal); hr1 = 23 (decimal)

Step=34 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   92  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=35 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (16 bits):  0B 05  -  Range = 24 (decimal); hr1 = 0 (decimal)

Step=36 | Mode=3

T_ID 3 - TPM extension (2 bits):        00
```

*Sample Data*

```
T_ID 4 - Antenna path perm. (8 bits):   31  -  Range = 24 (decimal); hr1 = 4 (decimal)

*********************************************************************

New DRBG octets:   2A 43 AB EB C6 D6 F2 E2 93 41 84 17 32 E6 82 31

*********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             2A 43 AB EB C6 D6 F2 E2 93 41 84 17
32 E6 82 31

T_ID 5 -     Initiator first  32-bit score: 22

T_ID 5 -     Initiator second 32-bit score: 8

T_ID 5 -     Initiator chosen PN pattern:   C6 D6 F2 E2

T_ID 5 -     Reflector first  32-bit score: 8

T_ID 5 -     Reflector second 32-bit score: 12

T_ID 5 -     Reflector chosen PN pattern:   93 41 84 17

T_ID 6 - SS marker position initiator (8 bits):    46  -  Range = 29 (decimal); hr1 = 7
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    61  -  Range = 29 (decimal); hr1 = 10
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

T_ID 7 - SS marker sig. sel. reflector (1 bits):   00

Step=37 | Mode=2

T_ID 2 - Submode insertion (8 bits):    50  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        40

*********************************************************************

New DRBG octets:   02 4C 7D 7E 4F 6A 09 9E C1 E0 6D F8 06 91 16 9B
```

*Sample Data*

```
**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   02  -  Range = 24 (decimal); hr1 = 0 (decimal)

Step=38 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   4C  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=39 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   7D  -  Range = 24 (decimal); hr1 = 11 (decimal)

Step=40 | Mode=3

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   7E  -  Range = 24 (decimal); hr1 = 11 (decimal)

**********************************************************************

New DRBG octets:   78 4C A9 80 0A 58 C3 56 12 75 95 0B 6F 84 0E C6

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             78 4C A9 80 0A 58 C3 56 12 75 95 0B
6F 84 0E C6

T_ID 5 -     Initiator first  32-bit score: 4

T_ID 5 -     Initiator second 32-bit score: 8

T_ID 5 -     Initiator chosen PN pattern:   78 4C A9 80

T_ID 5 -     Reflector first  32-bit score: 10

T_ID 5 -     Reflector second 32-bit score: 10

T_ID 5 -     Reflector chosen PN pattern:   6F 84 0E C6
```

*Sample Data*

```
T_ID 6 - SS marker position initiator (8 bits):    A5  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   00 90  -  Range = 29 (decimal); hr1 =
16 (decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   00

Step=41 | Mode=2

T_ID 2 - Submode insertion (8 bits):   15  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 3 - TPM extension (2 bits):       00

T_ID 4 - Antenna path perm. (8 bits):  4F  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=42 | Mode=2

T_ID 3 - TPM extension (2 bits):       40

T_ID 4 - Antenna path perm. (8 bits):  6A  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=43 | Mode=3

T_ID 3 - TPM extension (2 bits):       80

T_ID 4 - Antenna path perm. (8 bits):  09  -  Range = 24 (decimal); hr1 = 0 (decimal)

**********************************************************************

New DRBG octets:   51 17 E6 D9 19 D2 9F 36 65 42 19 F8 BC 7E 01 7A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             51 17 E6 D9 19 D2 9F 36 65 42 19 F8
BC 7E 01 7A

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 10
```

*Sample Data*

```
T_ID 5 -     Initiator chosen PN pattern:   19 D2 9F 36

T_ID 5 -     Reflector first  32-bit score: 6

T_ID 5 -     Reflector second 32-bit score: 20

T_ID 5 -     Reflector chosen PN pattern:   65 42 19 F8

T_ID 6 - SS marker position initiator (8 bits):    67  -  Range = 29 (decimal); hr1 = 11
(decimal)

**********************************************************************

New DRBG octets:   D1 93 AD A2 CE F3 2C 5F F0 29 1E 13 DB F5 9D 1D

**********************************************************************

T_ID 6 - SS marker position reflector (8 bits):    D1  -  Range = 29 (decimal); hr1 = 23
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80
```

### 8.1.2 Set 2

From the CS Configuration procedure:

- ChM = 0x1F FF FF FF FF FF FC 7F FF FC
- ChM_Repetition = 1
- Main_Mode = 0x02
- Sub_Mode = 0x01
- Main_Mode_Min_Steps = 2
- Main_Mode_Max_Steps = 6
- Main_Mode_Repetition = 1
- Mode_0_Steps = 3
- RTT_Type = 0x01
- Role = 0b01

*Sample Data*

- ChSel = 0

From the CS Start procedure:

- Subevent_Len – setup to include at least 20 steps
- Procedure_Count = 1
- ACI = 7

```
******************************** INSTANTIATION FUNCTION ********************************

h9() instantiation

************************************************************************************

Entropy input Peripheral (CS_IV_P):          E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):             3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                       E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral (CS_IN_P):                  9F F4 77 C1

Nonce Central (CS_IN_C):                     86 73 84 0D

Nonce (CS_IN):                               9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):    64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):              C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

    ********************** f8 function start **********************

        ************************** f7 function start **************************

        f7 K input:    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

        f7 V||S input:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00 00 00
20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                        9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1
43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

*Sample Data*

```
        f7 K2 output:   8B 2B 06 DC 52 2D 3E 0A F0 A5 0C AF 48 10 E0 35

       ************************** f7 function end ***************************

        ************************** f7 function start **************************

        f7 K input:    00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

        f7 V||S input: 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00 00
00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                        9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68
F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00

        f7 X output:   A3 4F BE 57 F8 F9 7E 34 9D 15 A3 76 79 60 74 64

        ************************** f7 function end ***************************

    f8 SM output:     B6 02 B1 B2 8C 6F 0A 3D DA E6 37 B4 84 25 08 7D DC 18 8C 89 A1 B0
CD FD A1 E8 FC 66 C9 99 97 50

    ********************** f8 function end **********************

********************************** INITIAL K and V ***********************************

K:              EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:              DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

******************************************************************************************

****************************** CHANNEL SOUNDING SEQUENCE *****************************

************************************ Channel Map *************************************

Bit-map:          1F FF FF FF FF FF FC 7F FF FC

Filtered channels: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
63 64 65 66 67 68 69 70 71 72 73 74 75 76

************************************ CS Procedure: 0 *********************************
```

*Sample Data*

```
***  INITIAL K and V  ***

K:   EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:   DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

Step=0 | Mode=0

**********************************************************************

New DRBG octets:   FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

**********************************************************************

T_ID 1 - mode0 channel (8bits):         FF  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):         BC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):         C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         CA  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):         A6  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):         9D  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):         C4  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8bits):         07  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8bits):         38  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):         EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         33  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):         D9  -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         D1  -  Range = 15 (decimal); hr1 = 12 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8bits):         35  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         32  -  Range = 17 (decimal); hr1 = 3 (decimal)

************************************************************************

New DRBG octets:   51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

************************************************************************

T_ID 1 - mode0 channel (8bits):         51  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         EE  -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8bits):         4B  -  Range = 20 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         B0  -  Range = 21 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8bits):         3E  -  Range = 22 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         E6  -  Range = 23 (decimal); hr1 = 20 (decimal)

T_ID 1 - mode0 channel (8bits):         D0  -  Range = 24 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8bits):         A1  -  Range = 25 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8bits):         71  -  Range = 26 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         87  -  Range = 27 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8bits):         1C  -  Range = 28 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         2E  -  Range = 29 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         60  -  Range = 30 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         46  -  Range = 31 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8bits):         8F  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8bits):         6C  -  Range = 33 (decimal); hr1 = 13 (decimal)

************************************************************************
```

*Sample Data*

```
New DRBG octets:   7D 75 2F 17 23 57 34 22 EF C7 CB C2 1F 18 90 8F

**********************************************************************

T_ID 1 - mode0 channel (8bits):         7D  -  Range = 34 (decimal); hr1 = 16 (decimal)

T_ID 1 - mode0 channel (8bits):         75  -  Range = 35 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8bits):         2F  -  Range = 36 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8bits):         17  -  Range = 37 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         23  -  Range = 38 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         57  -  Range = 39 (decimal); hr1 = 13 (decimal)

T_ID 1 - mode0 channel (8bits):         34  -  Range = 40 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8bits):         22  -  Range = 41 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         EF  -  Range = 42 (decimal); hr1 = 39 (decimal)

T_ID 1 - mode0 channel (8bits):         C7  -  Range = 43 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8bits):         CB  -  Range = 44 (decimal); hr1 = 34 (decimal)

T_ID 1 - mode0 channel (16bits):        C2 1F -  Range = 45 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):         18  -  Range = 46 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):         90  -  Range = 47 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8bits):         8F  -  Range = 48 (decimal); hr1 = 26 (decimal)

**********************************************************************

New DRBG octets:   64 74 2B E9 80 0D 87 22 36 1E 6F 55 61 A8 7C 17

**********************************************************************

T_ID 1 - mode0 channel (8bits):         64  -  Range = 49 (decimal); hr1 = 19 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8bits):         74  -  Range = 50 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (8bits):         2B  -  Range = 51 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8bits):         E9  -  Range = 52 (decimal); hr1 = 47 (decimal)

T_ID 1 - mode0 channel (8bits):         80  -  Range = 53 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8bits):         0D  -  Range = 54 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (16bits):        87 22 -  Range = 55 (decimal); hr1 = 7 (decimal)

T_ID 1 - mode0 channel (8bits):         36  -  Range = 56 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         1E  -  Range = 57 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8bits):         6F  -  Range = 58 (decimal); hr1 = 25 (decimal)

T_ID 1 - mode0 channel (8bits):         55  -  Range = 59 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8bits):         61  -  Range = 60 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (16bits):        A8 7C -  Range = 61 (decimal); hr1 = 29 (decimal)

T_ID 1 - mode0 channel (8bits):         17  -  Range = 62 (decimal); hr1 = 5 (decimal)

*****************************************************************

New DRBG octets:   0F 2F 55 C1 9B BD 7C 71 EC 79 0A 97 FD 0D 93 69

*****************************************************************

T_ID 1 - mode0 channel (8bits):         0F  -  Range = 63 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):         2F  -  Range = 64 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):         55  -  Range = 65 (decimal); hr1 = 21 (decimal)

T_ID 1 - mode0 channel (8bits):         C1  -  Range = 66 (decimal); hr1 = 49 (decimal)

T_ID 1 - mode0 channel (8bits):         9B  -  Range = 67 (decimal); hr1 = 40 (decimal)

T_ID 1 - mode0 channel (8bits):         BD  -  Range = 68 (decimal); hr1 = 50 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8bits):         7C  -  Range = 69 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8bits):         71  -  Range = 70 (decimal); hr1 = 30 (decimal)

T_ID 1 - mode0 channel (8bits):         EC  -  Range = 71 (decimal); hr1 = 65 (decimal)

T_ID 1 - mode0 channel (16bits):        79 0A -  Range = 72 (decimal); hr1 = 2 (decimal)

T_ID 1 -    Shuffled channels: 11 7 76 67 50 66 61 59 55 2 4 68 16 43 31 39 38 36 3 63 27
69 64 19 6 62 57 18 26 65 74 20 13 73 48 10 32 33 37 46 71 35 17 29 45 9 22 56 28 70 72
51 52 14 8 34 40 15 53 54 30 49 41 60 21 75 42 44 47 5 12 58

**********************************************************************

New DRBG octets:   0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D F0 79 BC 3A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D
F0 79 BC 3A

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   6C 37 6A B8

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   F0 79 BC 3A

Step=1 | Mode=0

**********************************************************************

New DRBG octets:   01 1C AE 4E 99 47 D1 B5 D0 6A CD DA C7 F8 F9 37

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             01 1C AE 4E 99 47 D1 B5 D0 6A CD DA
```

*Sample Data*

```
C7 F8 F9 37

T_ID 5 -    Initiator first  32-bit score: 6

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   01 1C AE 4E

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   D0 6A CD DA

Step=2 | Mode=0

********************************************************************

New DRBG octets:   64 06 12 14 86 05 AF EA 71 55 0B DD 28 94 2F 38

********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             64 06 12 14 86 05 AF EA 71 55 0B DD
28 94 2F 38

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   64 06 12 14

T_ID 5 -    Reflector first  32-bit score: 28

T_ID 5 -    Reflector second 32-bit score: 8

T_ID 5 -    Reflector chosen PN pattern:   28 94 2F 38

Step=3 | Mode=2

********************************************************************

New DRBG octets:   32 73 A2 2D 88 47 DA 6A 1C 2C BD 8C E8 96 79 62
```

*Sample Data*

```
**********************************************************************

T_ID 0 - non-mode0 channel (8bits):     32  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):     73  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):     A2  -  Range = 4 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):     2D  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):     88  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):     47  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):     DA  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):     6A  -  Range = 9 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):     1C  -  Range = 10 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):     2C  -  Range = 11 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):     BD  -  Range = 12 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):     8C  -  Range = 13 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):     E8  -  Range = 14 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8bits):     96  -  Range = 15 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):     79  -  Range = 16 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):     62  -  Range = 17 (decimal); hr1 = 6 (decimal)

**********************************************************************

New DRBG octets:   AB D1 0C 59 DD BD D9 F3 05 20 F7 D0 EF 37 8E BE

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):     AB  -  Range = 18 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8bits):     D1  -  Range = 19 (decimal); hr1 = 15 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):     0C  -  Range = 20 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):     59  -  Range = 21 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):     DD  -  Range = 22 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8bits):     BD  -  Range = 23 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8bits):     D9  -  Range = 24 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8bits):     F3  -  Range = 25 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8bits):     05  -  Range = 26 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):     20  -  Range = 27 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):     F7  -  Range = 28 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8bits):     D0  -  Range = 29 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (16bits):    EF 37 -  Range = 30 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):     8E  -  Range = 31 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8bits):     BE  -  Range = 32 (decimal); hr1 = 23 (decimal)

**********************************************************************

New DRBG octets:   3D FD C0 DF 3B F7 A9 A1 91 71 B5 33 C8 83 37 C5

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):     3D  -  Range = 33 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):     FD  -  Range = 34 (decimal); hr1 = 33 (decimal)

T_ID 0 - non-mode0 channel (8bits):     C0  -  Range = 35 (decimal); hr1 = 26 (decimal)

T_ID 0 - non-mode0 channel (8bits):     DF  -  Range = 36 (decimal); hr1 = 31 (decimal)

T_ID 0 - non-mode0 channel (8bits):     3B  -  Range = 37 (decimal); hr1 = 8 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):     F7  -  Range = 38 (decimal); hr1 = 36 (decimal)

T_ID 0 - non-mode0 channel (8bits):     A9  -  Range = 39 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8bits):     A1  -  Range = 40 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8bits):     91  -  Range = 41 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8bits):     71  -  Range = 42 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8bits):     B5  -  Range = 43 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8bits):     33  -  Range = 44 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):     C8  -  Range = 45 (decimal); hr1 = 35 (decimal)

T_ID 0 - non-mode0 channel (8bits):     83  -  Range = 46 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8bits):     37  -  Range = 47 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8bits):     C5  -  Range = 48 (decimal); hr1 = 36 (decimal)

*********************************************************************

New DRBG octets:   B3 97 AA E8 E5 02 C6 D3 BE 18 97 48 15 08 BB D5

*********************************************************************

T_ID 0 - non-mode0 channel (8bits):     B3  -  Range = 49 (decimal); hr1 = 34 (decimal)

T_ID 0 - non-mode0 channel (8bits):     97  -  Range = 50 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (8bits):     AA  -  Range = 51 (decimal); hr1 = 33 (decimal)

T_ID 0 - non-mode0 channel (16bits):    E8 E5 -  Range = 52 (decimal); hr1 = 46 (decimal)

T_ID 0 - non-mode0 channel (8bits):     02  -  Range = 53 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):     C6  -  Range = 54 (decimal); hr1 = 41 (decimal)

T_ID 0 - non-mode0 channel (8bits):     D3  -  Range = 55 (decimal); hr1 = 45 (decimal)

T_ID 0 - non-mode0 channel (8bits):     BE  -  Range = 56 (decimal); hr1 = 41 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):     18  -  Range = 57 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):     97  -  Range = 58 (decimal); hr1 = 34 (decimal)

T_ID 0 - non-mode0 channel (8bits):     48  -  Range = 59 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8bits):     15  -  Range = 60 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8bits):     08  -  Range = 61 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):     BB  -  Range = 62 (decimal); hr1 = 45 (decimal)

T_ID 0 - non-mode0 channel (8bits):     D5  -  Range = 63 (decimal); hr1 = 52 (decimal)

**********************************************************************

New DRBG octets:   74 77 BB 6D 51 76 FF 51 5F 13 93 86 FB FA 7F 41

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):     74  -  Range = 64 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (16bits):    77 BB -  Range = 65 (decimal); hr1 = 47 (decimal)

T_ID 0 - non-mode0 channel (16bits):    6D 51 -  Range = 66 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8bits):     76  -  Range = 67 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8bits):     FF  -  Range = 68 (decimal); hr1 = 67 (decimal)

T_ID 0 - non-mode0 channel (8bits):     51  -  Range = 69 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8bits):     5F  -  Range = 70 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8bits):     13  -  Range = 71 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):     93  -  Range = 72 (decimal); hr1 = 41 (decimal)

T_ID 0 -    Shuffled channels: 57 65 5 31 64 75 34 37 48 8 51 7 19 4 13 20 63 35 46 6 70 73
9 50 17 74 39 32 29 68 71 40 22 55 62 49 52 16 21 43 36 76 15 41 33 66 56 69 10 18 38 11 67
26 45 58 2 53 27 3 12 59 30 54 42 28 47 72 14 44 61 60
```

*Sample Data*

```
**********************************************************************

New DRBG octets:   F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

**********************************************************************

T_ID 2 - Submode insertion (8 bits):    F8  -  Range = 5 (decimal);  hr1 = 4 (decimal)

**********************************************************************

New DRBG octets:   3B 8E 7E 2A A8 D7 CF F1 CC 69 BE 13 75 1F B3 95

**********************************************************************

T_ID 3 - TPM extension (2 bits):        00

**********************************************************************

New DRBG octets:   B4 8D B4 8C 41 33 DE 1D 71 B7 46 A1 61 8C 4D D9

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=4 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=5 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   B4   -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=6 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   8C  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=7 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   41   -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=8 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   33   -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=9 | Mode=1

**********************************************************************

New DRBG octets:   A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A 6B 2E B7 48

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A
6B 2E B7 48

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 6

T_ID 5 -    Initiator chosen PN pattern:   16 9E ED DC

T_ID 5 -    Reflector first  32-bit score: 20

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   6B 2E B7 48

**********************************************************************

New DRBG octets:   72 26 4D 65 1E 54 6E 3B A1 7E 46 61 A5 00 90 67

**********************************************************************

T_ID 6 - SS marker position initiator (8 bits):   72  -  Range = 29 (decimal); hr1 = 12
(decimal)
```

*Sample Data*

```
T_ID 6 - SS marker position reflector (8 bits):   26  -  Range = 29 (decimal); hr1 = 4
(decimal)

**********************************************************************

New DRBG octets:   7D CB DA 7B 6E 5E A2 0C 32 0D C2 4F 8E 2C 72 5D

**********************************************************************

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=10 | Mode=2

T_ID 2 - Submode insertion (8 bits):    82  -  Range = 5 (decimal);  hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   DE  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=11 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   1D  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=12 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   71  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=13 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   B7  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=14 | Mode=1

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   F7 21 97 86 9D 82 D2 EC 80 69 7C 5B 57 17 64 70

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             F7 21 97 86 9D 82 D2 EC 80 69 7C 5B
57 17 64 70

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   F7 21 97 86

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   57 17 64 70

T_ID 6 - SS marker position initiator (8 bits):    4D  -  Range = 29 (decimal); hr1 = 8
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    65  -  Range = 29 (decimal); hr1 = 11
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=15 | Mode=2

T_ID 2 - Submode insertion (8 bits):    2A  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   46  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=16 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   A1  -  Range = 24 (decimal); hr1 = 15 (decimal)
```

*Sample Data*

```
Step=17 | Mode=1

**********************************************************************

New DRBG octets:   17 46 62 CD 92 22 DB 8B 0C 22 D1 19 C8 C5 97 E5

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             17 46 62 CD 92 22 DB 8B 0C 22 D1 19
C8 C5 97 E5

T_ID 5 -    Initiator first  32-bit score: 14

T_ID 5 -    Initiator second 32-bit score: 16

T_ID 5 -    Initiator chosen PN pattern:   17 46 62 CD

T_ID 5 -    Reflector first  32-bit score: 6

T_ID 5 -    Reflector second 32-bit score: 6

T_ID 5 -    Reflector chosen PN pattern:   C8 C5 97 E5

T_ID 6 - SS marker position initiator (8 bits):   1E  -  Range = 29 (decimal); hr1 = 3
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   54  -  Range = 29 (decimal); hr1 = 9
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=18 | Mode=2

T_ID 2 - Submode insertion (8 bits):    54  -  Range = 5 (decimal);  hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   61  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=19 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   8C   -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=20 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   4D   -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=21 | Mode=0

**********************************************************************

New DRBG octets:   AF 9F 78 48 26 29 96 A3 85 E6 60 4F 82 3B 87 C2

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             AF 9F 78 48 26 29 96 A3 85 E6 60 4F
82 3B 87 C2

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   AF 9F 78 48

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 18

T_ID 5 -    Reflector chosen PN pattern:   85 E6 60 4F

Step=22 | Mode=0

**********************************************************************

New DRBG octets:   3C 60 3B D2 E1 A3 41 67 B3 78 97 ED B4 B2 72 BF

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             3C 60 3B D2 E1 A3 41 67 B3 78 97 ED
B4 B2 72 BF
```

*Sample Data*

```
T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   E1 A3 41 67

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   B4
B2 72 BF

Step=23 | Mode=0

**********************************************************************

New DRBG octets:   59 5F E1 AA 5F 57 BA 90 82 5F 07 64 35 91 56 A4

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             59 5F E1 AA 5F 57 BA 90 82 5F 07 64
35 91 56 A4

T_ID 5 -    Initiator first  32-bit score: 18

T_ID 5 -    Initiator second 32-bit score: 18

T_ID 5 -    Initiator chosen PN pattern:   5F 57 BA 90

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   82 5F 07 64

Step=24 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   D9  -  Range = 24 (decimal); hr1 = 20 (decimal)
```

*Sample Data*

```
Step=25 | Mode=1

**********************************************************************

New DRBG octets:   72 4B 10 38 16 33 91 44 07 98 41 98 61 3B C2 E8

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             72 4B 10 38 16 33 91 44 07 98 41 98
61 3B C2 E8

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   72 4B 10 38

T_ID 5 -    Reflector first  32-bit score: 20

T_ID 5 -   Reflector second 32-bit score: 6

T_ID 5 -    Reflector chosen PN pattern:   61 3B C2 E8

T_ID 6 - SS marker position initiator (8 bits):    6E  -  Range = 29 (decimal); hr1 = 12
(decimal)

T_ID 6 - SS marker position reflector (8 bits):    3B  -  Range = 29 (decimal); hr1 = 6
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=26 | Mode=2

T_ID 2 - Submode insertion (8 bits):    E0  -  Range = 5 (decimal);  hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):        80

**********************************************************************

New DRBG octets:   1B 36 9E 49 12 D6 92 B3 BB F8 47 DD B4 8D 59 9B
```

*Sample Data*

```
**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   1B  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=27 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   36  -  Range = 24 (decimal); hr1 = 5 (decimal)

Step=28 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   9E  -  Range = 24 (decimal); hr1 = 14 (decimal)

Step=29 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   49  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=30 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   12  -  Range = 24 (decimal); hr1 = 1 (decimal)

Step=31 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   D6  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=32 | Mode=1

**********************************************************************

New DRBG octets:   2C E0 B1 90 2B FA F8 37 7A B0 91 BF A7 37 8B DC

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):            2C E0 B1 90 2B FA F8 37 7A B0 91 BF
```

*Sample Data*

```
A7 37 8B DC

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 16

T_ID 5 -    Initiator chosen PN pattern:   2C E0 B1 90

T_ID 5 -    Reflector first  32-bit score: 8

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   7A B0 91 BF

T_ID 6 - SS marker position initiator (8 bits):   A1  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   7E  -  Range = 29 (decimal); hr1 = 14
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=33 | Mode=2

T_ID 2 - Submode insertion (8 bits):    7C  -  Range = 5 (decimal);  hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   92  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=34 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   B3  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=35 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   BB  -  Range = 24 (decimal); hr1 = 17 (decimal)
```

*Sample Data*

```
Step=36 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   F8  -  Range = 24 (decimal); hr1 = 23 (decimal)

Step=37 | Mode=1

**********************************************************************

New DRBG octets:   A3 5E 66 E1 FC 19 2D CF 4C 1C 10 C0 41 10 AB 72

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             A3 5E 66 E1 FC 19 2D CF 4C 1C 10 C0
41 10 AB 72

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   A3 5E 66 E1

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   41 10 AB 72

T_ID 6 - SS marker position initiator (8 bits):   46  -  Range = 29 (decimal); hr1 = 7
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   61  -  Range = 29 (decimal); hr1 = 10
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=38 | Mode=2

T_ID 2 - Submode insertion (8 bits):    50  -  Range = 5 (decimal);  hr1 = 1 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   47  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=39 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   DD  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=40 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=41 | Mode=0

**********************************************************************

New DRBG octets:   F9 75 C7 DA 8C C1 CB C7 5E A9 A3 F9 C2 EF B6 90

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             F9 75 C7 DA 8C C1 CB C7 5E A9 A3 F9
C2 EF B6 90

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 26

T_ID 5 -    Initiator chosen PN pattern:   F9 75 C7 DA

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   C2 EF B6 90

Step=42 | Mode=0

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   94 27 94 24 2B 7D 51 F3 3E 76 6F CB 96 52 E3 61

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             94 27 94 24 2B 7D 51 F3 3E 76 6F CB
96 52 E3 61

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   94 27 94 24

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   96 52 E3 61

Step=43 | Mode=0

**********************************************************************

New DRBG octets:   51 17 E6 D9 19 D2 9F 36 65 42 19 F8 BC 7E 01 7A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             51 17 E6 D9 19 D2 9F 36 65 42 19 F8
BC 7E 01 7A

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 10

T_ID 5 -    Initiator chosen PN pattern:   19 D2 9F 36

T_ID 5 -    Reflector first  32-bit score: 6

T_ID 5 -    Reflector second 32-bit score: 20

T_ID 5 -    Reflector chosen PN pattern:   65 42 19 F8
```

*Sample Data*

```
Step=44 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=45 | Mode=1

**********************************************************************

New DRBG octets:   3A FC C9 66 2A 84 09 59 B4 39 B7 28 A5 56 83 81

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             3A FC C9 66 2A 84 09 59 B4 39 B7 28
A5 56 83 81

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 16

T_ID 5 -    Initiator chosen PN pattern:   3A FC C9 66

T_ID 5 -    Reflector first  32-bit score: 8

T_ID 5 -    Reflector second 32-bit score: 20

T_ID 5 -    Reflector chosen PN pattern:   B4 39 B7 28

T_ID 6 - SS marker position initiator (8 bits):   A5  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   00 90  -  Range = 29 (decimal); hr1 = 16
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   00

Step=46 | Mode=2

T_ID 2 - Submode insertion (8 bits):    15  -  Range = 5 (decimal);  hr1 = 0 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   59  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=47 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   9B  -  Range = 24 (decimal); hr1 = 14 (decimal)

Step=48 | Mode=1

************************************************************************

New DRBG octets:   87 D7 D1 62 B8 30 DB BE 77 34 24 24 C7 AE 18 FE

************************************************************************

T_ID 5 - RTT AA candidates (128 bits):             87 D7 D1 62 B8 30 DB BE 77 34 24 24
C7 AE 18 FE

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   B8 30 DB BE

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 20

T_ID 5 -    Reflector chosen PN pattern:   77 34 24 24

T_ID 6 - SS marker position initiator (8 bits):   67  -  Range = 29 (decimal); hr1 = 11
(decimal)

************************************************************************

New DRBG octets:   32 27 9A 3D 88 04 52 18 AF DA 47 5C 50 DB 6F 3C

************************************************************************

T_ID 6 - SS marker position reflector (8 bits):   32  -  Range = 29 (decimal); hr1 = 5
```

*Sample Data*

```
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=49 | Mode=2

T_ID 2 - Submode insertion (8 bits):    57  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

**********************************************************************

New DRBG octets:   4A B2 C3 2F 35 67 74 81 52 56 29 D0 32 8A 1E BF

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   4A  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=50 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   B2  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=51 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   C3  -  Range = 24 (decimal); hr1 = 18 (decimal)

Step=52 | Mode=1

**********************************************************************

New DRBG octets:   29 3A 71 72 62 7E C2 74 4A 63 9D 3C 78 9D 64 25

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             29 3A 71 72 62 7E C2 74 4A 63 9D 3C
78 9D 64 25
```

*Sample Data*

```
T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 10

T_ID 5 -    Initiator chosen PN pattern:   62 7E C2 74

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 6

T_ID 5 -    Reflector chosen PN pattern:   78 9D 64 25

T_ID 6 - SS marker position initiator (8 bits):   27  -  Range = 29 (decimal); hr1 = 4
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   9A  -  Range = 29 (decimal); hr1 = 17
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=53 | Mode=2

T_ID 2 - Submode insertion (8 bits):    CD  -  Range = 5 (decimal);  hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   2F  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=54 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   35  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=55 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   67  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=56 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   74  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=57 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   81  -  Range = 24 (decimal); hr1 = 12 (decimal)

Step=58 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   52  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=59 | Mode=1

**********************************************************************

New DRBG octets:   AE 8A 62 36 0B 80 CD 62 1F BB FF 19 A6 E7 E5 E2

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             AE 8A 62 36 0B 80 CD 62 1F BB FF 19
A6 E7 E5 E2

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 4

T_ID 5 -    Initiator chosen PN pattern:   0B 80 CD 62

T_ID 5 -    Reflector first  32-bit score: 20

T_ID 5 -    Reflector second 32-bit score: 4

T_ID 5 -    Reflector chosen PN pattern:   A6 E7 E5 E2

T_ID 6 - SS marker position initiator (8 bits):   3D  -  Range = 29 (decimal); hr1 = 6
(decimal)
```

*Sample Data*

```
T_ID 6 - SS marker position reflector (8 bits):   88  -  Range = 29 (decimal); hr1 = 15
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=60 | Mode=0

**********************************************************************

New DRBG octets:   59 D1 03 4C 30 03 73 84 75 24 FA 34 78 DD AC 66

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             59 D1 03 4C 30 03 73 84 75 24 FA 34
78 DD AC 66

T_ID 5 -    Initiator first  32-bit score: 6

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   59 D1 03 4C

T_ID 5 -    Reflector first  32-bit score: 8

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   75 24 FA 34

Step=61 | Mode=0

**********************************************************************

New DRBG octets:   3B 07 5A 7F 92 91 D5 EB 76 C5 32 55 7E 2B 6E DB

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             3B 07 5A 7F 92 91 D5 EB 76 C5 32 55
7E 2B 6E DB

T_ID 5 -    Initiator first  32-bit score: 8
```

*Sample Data*

```
T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   3B 07 5A 7F

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   7E 2B 6E DB

Step=62 | Mode=0

**********************************************************************

New DRBG octets:   B1 74 51 AC 18 FE 87 44 AC 7E 27 CF 18 9D 44 87

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             B1 74 51 AC 18 FE 87 44 AC 7E 27 CF
18 9D 44 87

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   B1 74 51 AC

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 8

T_ID 5 -    Reflector chosen PN pattern:   18 9D 44 87

Step=63 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   56  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=64 | Mode=2

T_ID 2 - Submode insertion (8 bits):    98  -  Range = 5 (decimal);  hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   29  -  Range = 24 (decimal); hr1 = 3 (decimal)

Step=65 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   D0  -  Range = 24 (decimal); hr1 = 19 (decimal)

Step=66 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   32  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=67 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   8A  -  Range = 24 (decimal); hr1 = 12 (decimal)

Step=68 | Mode=1

**********************************************************************

New DRBG octets:   76 44 70 57 97 3E 93 E5 DC 7A 6D 6A 8C D6 38 ED

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             76 44 70 57 97 3E 93 E5 DC 7A 6D 6A
8C D6 38 ED

T_ID 5 -    Initiator first  32-bit score: 6

T_ID 5 -    Initiator second 32-bit score: 6

T_ID 5 -    Initiator chosen PN pattern:   97 3E 93 E5

T_ID 5 -    Reflector first  32-bit score: 8

T_ID 5 -    Reflector second 32-bit score: 20
```

*Sample Data*

```
T_ID 5 -    Reflector chosen PN pattern:   DC 7A 6D 6A

T_ID 6 - SS marker position initiator (8 bits):   04  -  Range = 29 (decimal); hr1 = 0
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   52  -  Range = 29 (decimal); hr1 = 9
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=69 | Mode=2

T_ID 2 - Submode insertion (8 bits):    51  -  Range = 5 (decimal);  hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   1E  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=70 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   BF  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=71 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

**********************************************************************

New DRBG octets:   36 77 E6 E4 59 5B 6C EF 8A AC 2E CF 59 8D 5E C0

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):   36  -  Range = 24 (decimal); hr1 = 5 (decimal)

Step=72 | Mode=1

**********************************************************************

New DRBG octets:   72 54 52 61 69 F2 5A 0D CC CA 4F CE BD 8A 88 1E
```

*Sample Data*

```
**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             72 54 52 61 69 F2 5A 0D CC CA 4F CE
BD 8A 88 1E

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   72 54 52 61

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   BD 8A 88 1E

T_ID 6 - SS marker position initiator (8 bits):   18  -  Range = 29 (decimal); hr1 = 2
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   AF  -  Range = 29 (decimal); hr1 = 19
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=73 | Mode=2

T_ID 2 - Submode insertion (8 bits):    69  -  Range = 5 (decimal);  hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   77  -  Range = 24 (decimal); hr1 = 11 (decimal)

Step=74 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   E6  -  Range = 24 (decimal); hr1 = 21 (decimal)

Step=75 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   E4  -  Range = 24 (decimal); hr1 = 21 (decimal)

Step=76 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   59  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=77 | Mode=1

**********************************************************************

New DRBG octets:   9F B8 94 20 27 F3 AA 36 81 84 73 90 7C 34 35 07

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             9F B8 94 20 27 F3 AA 36 81 84 73 90
7C 34 35 07

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 2

T_ID 5 -    Initiator chosen PN pattern:   27 F3 AA 36

T_ID 5 -    Reflector first  32-bit score: 18

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   7C 34 35 07

T_ID 6 - SS marker position initiator (8 bits):   DA  -  Range = 29 (decimal); hr1 = 24
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   47 5C  -  Range = 29 (decimal); hr1 = 10
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80
```

*Sample Data*

```
Step=78 | Mode=2

T_ID 2 - Submode insertion (8 bits):    D4  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   5B  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=79 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   6C  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=80 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   EF  -  Range = 24 (decimal); hr1 = 22 (decimal)

Step=81 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   8A  -  Range = 24 (decimal); hr1 = 12 (decimal)

Step=82 | Mode=0

**********************************************************************

New DRBG octets:   83 70 FC 0A A7 A6 53 65 E1 13 9F 67 7B B9 53 86

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             83 70 FC 0A A7 A6 53 65 E1 13 9F 67
7B B9 53 86

T_ID 5 -    Initiator first  32-bit score: 18

T_ID 5 -    Initiator second 32-bit score: 20

T_ID 5 -    Initiator chosen PN pattern:   83 70 FC 0A
```

*Sample Data*

```
T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   7B B9 53 86

Step=83 | Mode=0

********************************************************************

New DRBG octets:   AE D2 9E 4D 5E A1 01 54 C3 9B 02 8E 21 3D C0 DD

********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             AE D2 9E 4D 5E A1 01 54 C3 9B 02 8E
21 3D C0 DD

T_ID 5 -    Initiator first  32-bit score: 16

T_ID 5 -    Initiator second 32-bit score: 24

T_ID 5 -    Initiator chosen PN pattern:   AE D2 9E 4D

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 8

T_ID 5 -    Reflector chosen PN pattern:   21 3D C0 DD

Step=84 | Mode=0

********************************************************************

New DRBG octets:   7A 61 88 D2 B3 D8 C1 09 1C 1B 06 35 A7 9B 42 2F

********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             7A 61 88 D2 B3 D8 C1 09 1C 1B 06 35
A7 9B 42 2F

T_ID 5 -    Initiator first  32-bit score: 8
```

*Sample Data*

```
T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   7A 61 88 D2

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 4

T_ID 5 -    Reflector chosen PN pattern:   A7 9B 42 2F

Step=85 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   AC  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=86 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   2E  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=87 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   CF  -  Range = 24 (decimal); hr1 = 19 (decimal)

Step=88 | Mode=1

**********************************************************************

New DRBG octets:   05 81 5B E5 5E 76 9B EA C5 67 5D CF EC 89 79 12

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             05 81 5B E5 5E 76 9B EA C5 67 5D CF
EC 89 79 12

T_ID 5 -    Initiator first  32-bit score: 14

T_ID 5 -    Initiator second 32-bit score: 8
```

*Sample Data*

```
T_ID 5 -    Initiator chosen PN pattern:   5E 76 9B EA

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   C5 67 5D CF

T_ID 6 - SS marker position initiator (16 bits):   50 DB  -  Range = 29 (decimal); hr1 = 24
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   6F  -  Range = 29 (decimal); hr1 = 12
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=89 | Mode=2

T_ID 2 - Submode insertion (8 bits):    AC  -  Range = 5 (decimal);  hr1 = 3 (decimal)

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   59  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=90 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=91 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   5E  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=92 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

*****************************************************************
```

*Sample Data*

```
New DRBG octets:   85 C3 62 27 BA 61 FE 15 53 3D DF A0 BD 1A DC 23

**********************************************************************

T_ID 4 - Antenna path perm. (16 bits):   C0 85  -  Range = 24 (decimal); hr1 = 12 (decimal)


Step=93 | Mode=2

**********************************************************************

New DRBG octets:   74 30 D2 5E 70 AE 7C F7 B5 44 F9 33 E6 5A F3 25

**********************************************************************

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   C3  -  Range = 24 (decimal); hr1 = 18 (decimal)

Step=94 | Mode=1

**********************************************************************

New DRBG octets:   CF A8 D9 FB 13 04 07 7D 66 26 C7 0E 22 42 67 02

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             CF A8 D9 FB 13 04 07 7D 66 26 C7 0E
22 42 67 02

T_ID 5 -    Initiator first  32-bit score: 6

T_ID 5 -    Initiator second 32-bit score: 18

T_ID 5 -    Initiator chosen PN pattern:   CF A8 D9 FB

T_ID 5 -    Reflector first  32-bit score: 24

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   22 42 67 02
```

*Sample Data*

```
T_ID 6 - SS marker position initiator (8 bits):   3C  -  Range = 29 (decimal); hr1 = 6
(decimal)

**********************************************************************

New DRBG octets:   28 84 FE 25 8A 58 C6 5C 5B 6F 45 D8 83 94 8A B4

**********************************************************************

T_ID 6 - SS marker position reflector (8 bits):    28  -  Range = 29 (decimal); hr1 = 4
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):   80

T_ID 7 - SS marker sig. sel. reflector (1 bits):   00

Step=95 | Mode=2

T_ID 2 - Submode insertion (8 bits):    AB  -  Range = 5 (decimal);  hr1 = 3 (decimal)

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   62  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=96 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   27  -  Range = 24 (decimal); hr1 = 3 (decimal)

Step=97 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   BA  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=98 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   61  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=99 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   FE  -  Range = 24 (decimal); hr1 = 23 (decimal)

Step=100 | Mode=1

**********************************************************************

New DRBG octets:   B6 E1 30 8B 38 37 25 B0 CE 5D 24 0A 10 9E FE 07

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             B6 E1 30 8B 38 37 25 B0 CE 5D 24 0A
10 9E FE 07

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -   Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   B6 E1 30 8B

T_ID 5 -    Reflector first  32-bit score: 6

T_ID 5 -    Reflector second 32-bit score: 22

T_ID 5 -    Reflector chosen PN pattern:   CE 5D 24 0A

T_ID 6 - SS marker position initiator (8 bits):   84  -  Range = 29 (decimal); hr1 = 14
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   FE  -  Range = 29 (decimal); hr1 = 28
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=101 | Mode=2

**********************************************************************

New DRBG octets:   54 AD 8A 11 F2 83 AD BE 5E 1D 5A E3 3C A4 B2 B4
```

*Sample Data*

```
**********************************************************************

T_ID 2 - Submode insertion (8 bits):    54  -  Range = 5 (decimal);  hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   15  -  Range = 24 (decimal); hr1 = 1 (decimal)

Step=102 | Mode=2

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (8 bits):   53  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=103 | Mode=2

T_ID 3 - TPM extension (2 bits):        C0

T_ID 4 - Antenna path perm. (8 bits):   3D  -  Range = 24 (decimal); hr1 = 5 (decimal)

Step=104 | Mode=0

**********************************************************************

New DRBG octets:   14 74 46 A3 21 4B 6A C5 AC 95 65 F2 A4 1E 82 3D

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             14 74 46 A3 21 4B 6A C5 AC 95 65 F2
A4 1E 82 3D

T_ID 5 -    Initiator first  32-bit score: 16

T_ID 5 -    Initiator second 32-bit score: 18

T_ID 5 -    Initiator chosen PN pattern:   14 74 46 A3

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   A4 1E 82 3D
```

*Sample Data*

```
Step=105 | Mode=0

**********************************************************************

New DRBG octets:   99 72 D6 75 CA 00 A1 61 94 E3 8F 9B 06 A2 9C 82

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             99 72 D6 75 CA 00 A1 61 94 E3 8F 9B
06 A2 9C 82

T_ID 5 -    Initiator first  32-bit score: 16

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   CA 00 A1 61

T_ID 5 -    Reflector first  32-bit score: 20

T_ID 5 -    Reflector second 32-bit score: 6

T_ID 5 -    Reflector chosen PN pattern:   06 A2 9C 82

Step=106 | Mode=0

**********************************************************************

New DRBG octets:   A9 18 D2 F4 BF 93 95 5D 2E 5D C9 9D 6A 7D 0D 47

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             A9 18 D2 F4 BF 93 95 5D 2E 5D C9 9D
6A 7D 0D 47

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   A9 18 D2 F4

T_ID 5 -    Reflector first  32-bit score: 12
```

*Sample Data*

```
T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   6A 7D 0D 47

Step=107 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   DF  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=108 | Mode=1

**********************************************************************

New DRBG octets:   9E 51 FD E2 B2 36 1E BE 3B 9E 68 DF CE D3 70 80

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):             9E 51 FD E2 B2 36 1E BE 3B 9E 68 DF
CE D3 70 80

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 4

T_ID 5 -    Initiator chosen PN pattern:   B2 36 1E BE

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   CE D3 70 80

T_ID 6 - SS marker position initiator (8 bits):   25  -  Range = 29 (decimal); hr1 = 4
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   8A  -  Range = 29 (decimal); hr1 = 15
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80
```

*Sample Data*

```
Step=109 | Mode=2

T_ID 2 - Submode insertion (8 bits):    AD  -  Range = 5 (decimal);  hr1 = 3 (decimal)

T_ID 3 - TPM extension (2 bits):        00

T_ID 4 - Antenna path perm. (16 bits):   A0 BD  -  Range = 24 (decimal); hr1 = 17 (decimal)


Step=110 | Mode=2

T_ID 3 - TPM extension (2 bits):        80

T_ID 4 - Antenna path perm. (8 bits):   1A  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=111 | Mode=2

T_ID 3 - TPM extension (2 bits):        40

T_ID 4 - Antenna path perm. (8 bits):   DC  -  Range = 24 (decimal); hr1 = 20 (decimal)
```

### 8.1.3 Set 3

From the CS Configuration procedure:

- ChM = 0x00 00 00 00 00 00 00 1F FF FC
- ChM_Repetition = 2
- Main_Mode = 0x02
- Sub_Mode = 0x03
- Main_Mode_Min_Steps = 2
- Main_Mode_Max_Steps = 6
- Main_Mode_Repetition = 0
- Mode_0_Steps = 3
- RTT_Type = 0x01
- Role = 0b01
- ChSel = 0

*Sample Data*

From the CS Start procedure:

- Subevent_Len – setup to include at least 20 steps
- Procedure_Count = 2
- ACI = 7

```
****************************** INSTANTIATION FUNCTION *********************************

h9() instantiation

*******************************************************************************************

Entropy input Peripheral (CS_IV_P):          E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):             3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                       E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral (CS_IN_P):                  9F F4 77 C1

Nonce Central (CS_IN_C):                     86 73 84 0D

Nonce (CS_IN):                               9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):    64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):              C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

    ********************** f8 function start **********************

        *************************** f7 function start ***************************

         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00
00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78
68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

         f7 K2 output:   8B 2B 06 DC 52 2D 3E 0A F0 A5 0C AF 48 10 E0 35
```

*Sample Data*

```
        ************************** f7 function end ***************************

        ************************** f7 function start **************************

         f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

         f7 V||S input:  00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 28 00
00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                         9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96 78
68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

         f7 X output:    A3 4F BE 57 F8 F9 7E 34 9D 15 A3 76 79 60 74 64

        ************************** f7 function end ***************************

     f8 SM output:       B6 02 B1 B2 8C 6F 0A 3D DA E6 37 B4 84 25 08 7D DC 18 8C 89 A1
B0 CD FD A1 E8 FC 66 C9 99 97 50

    ********************** f8 function end **********************

********************************** INITIAL K and V ***********************************

K:               EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:               DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

******************************************************************************************

****************************** CHANNEL SOUNDING SEQUENCE *****************************

************************************ Channel map *************************************

Bit-map:           00 00 00 00 00 00 00 1F FF FC

Filtered channels: 2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20

********************************** CS Procedure: 0 ***********************************

*** INITIAL K and V ***

K:    EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27
```

*Sample Data*

```
V:    DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

Step=0 | Mode=0

*********************************************************************

New DRBG octets:   FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

*********************************************************************

T_ID 1 - mode0 channel (8bits):          FF  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):          BC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):          CA  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):          39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):          A6  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):          9D  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):          C4  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8bits):          07  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8bits):          38  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):          33  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          D9  -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):          D1  -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 1 - mode0 channel (8bits):          35  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):          32  -  Range = 17 (decimal); hr1 = 3 (decimal)
```

*Sample Data*

```
**********************************************************************

New DRBG octets:    51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

**********************************************************************

T_ID 1 - mode0 channel (8bits):          51  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8bits):          EE  -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 -     Shuffled channels: 11 7 14 18 9 19 10 8 5 2 4 15 16 13 12 6 17 20 3

**********************************************************************

New DRBG octets:    0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D F0 79 BC 3A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D
F0 79 BC 3A

T_ID 5 -    Initiator first  32-bit score: 12

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   6C 37 6A B8

T_ID 5 -    Reflector first  32-bit score: 16

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   F0 79 BC 3A

Step=1 | Mode=0

**********************************************************************

New DRBG octets:   01 1C AE 4E 99 47 D1 B5 D0 6A CD DA C7 F8 F9 37

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              01 1C AE 4E 99 47 D1 B5 D0 6A CD DA
C7 F8 F9 37
```

*Sample Data*

```
T_ID 5 -    Initiator first  32-bit score: 6

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   01 1C AE 4E

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   D0 6A CD DA

Step=2 | Mode=0

**********************************************************************

New DRBG octets:   64 06 12 14 86 05 AF EA 71 55 0B DD 28 94 2F 38

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              64 06 12 14 86 05 AF EA 71 55 0B DD
28 94 2F 38

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   64 06 12 14

T_ID 5 -    Reflector first  32-bit score: 28

T_ID 5 -    Reflector second 32-bit score: 8

T_ID 5 -    Reflector chosen PN pattern:   28 94 2F 38

Step=3 | Mode=2

**********************************************************************

New DRBG octets:   32 73 A2 2D 88 47 DA 6A 1C 2C BD 8C E8 96 79 62

**********************************************************************
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):      32  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      73  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      A2  -  Range = 4 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      2D  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      88  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      47  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      DA  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):      6A  -  Range = 9 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      1C  -  Range = 10 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      2C  -  Range = 11 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      BD  -  Range = 12 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):      8C  -  Range = 13 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      E8  -  Range = 14 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8bits):      96  -  Range = 15 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):      79  -  Range = 16 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      62  -  Range = 17 (decimal); hr1 = 6 (decimal)

**********************************************************************

New DRBG octets:   AB D1 0C 59 DD BD D9 F3 05 20 F7 D0 EF 37 8E BE

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):      AB  -  Range = 18 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8bits):      D1  -  Range = 19 (decimal); hr1 = 15 (decimal)
```

*Sample Data*

```
T_ID 0 -    Shuffled channels: 6 12 5 10 3 2 18 17 16 8 11 7 19 4 13 20 9 15 14

**********************************************************************

New DRBG octets:   F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

**********************************************************************

T_ID 2 - Submode insertion (8 bits):     F8  -  Range = 5 (decimal); hr1 = 4 (decimal)

**********************************************************************

New DRBG octets:   3B 8E 7E 2A A8 D7 CF F1 CC 69 BE 13 75 1F B3 95

**********************************************************************

T_ID 3 - TPM extension (2 bits):         00

**********************************************************************

New DRBG octets:   B4 8D B4 8C 41 33 DE 1D 71 B7 46 A1 61 8C 4D D9

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=4 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=5 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=6 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    8C  -  Range = 24 (decimal); hr1 = 13 (decimal)
```

*Sample Data*

```
Step=7 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    41  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=8 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    33  -  Range = 24 (decimal); hr1 = 4 (decimal)

Step=9 | Mode=3

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    DE  -  Range = 24 (decimal); hr1 = 20 (decimal)

**********************************************************************

New DRBG octets:   A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A 6B 2E B7 48

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A
6B 2E B7 48

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 6

T_ID 5 -    Initiator chosen PN pattern:   16 9E ED DC

T_ID 5 -    Reflector first  32-bit score: 20

T_ID 5 -    Reflector second 32-bit score: 14

T_ID 5 -    Reflector chosen PN pattern:   6B 2E B7 48

**********************************************************************

New DRBG octets:   72 26 4D 65 1E 54 6E 3B A1 7E 46 61 A5 00 90 67
```

*Sample Data*

```
**********************************************************************

T_ID 6 - SS marker position initiator (8 bits):   72  -  Range = 29 (decimal); hr1 = 12
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   26  -  Range = 29 (decimal); hr1 = 4
(decimal)

**********************************************************************

New DRBG octets:    7D CB DA 7B 6E 5E A2 0C 32 0D C2 4F 8E 2C 72 5D

**********************************************************************

T_ID 7 - SS marker sig. sel. initiator (1 bits):   00

T_ID 7 - SS marker sig. sel. reflector (1 bits):   80

Step=10 | Mode=2

T_ID 2 - Submode insertion (8 bits):     82  -  Range = 5 (decimal);  hr1 = 2 (decimal)

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    1D  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=11 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    71  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=12 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    B7  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=13 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0
```

*Sample Data*

```
T_ID 4 - Antenna path perm. (8 bits):    46  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=14 | Mode=3

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    A1  -  Range = 24 (decimal); hr1 = 15 (decimal)

**********************************************************************

New DRBG octets:   F7 21 97 86 9D 82 D2 EC 80 69 7C 5B 57 17 64 70

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              F7 21 97 86 9D 82 D2 EC 80 69 7C 5B
57 17 64 70

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 12

T_ID 5 -    Initiator chosen PN pattern:   F7 21 97 86

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   57 17 64 70

T_ID 6 - SS marker position initiator (8 bits):   4D  -  Range = 29 (decimal); hr1 = 8
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   65  -  Range = 29 (decimal); hr1 = 11
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=15 | Mode=2

T_ID 2 - Submode insertion (8 bits):     2A  -  Range = 5 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    61  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=16 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    8C  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=17 | Mode=3

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    4D  -  Range = 24 (decimal); hr1 = 7 (decimal)

**********************************************************************

New DRBG octets:   17 46 62 CD 92 22 DB 8B 0C 22 D1 19 C8 C5 97 E5

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              17 46 62 CD 92 22 DB 8B 0C 22 D1 19
C8 C5 97 E5

T_ID 5 -    Initiator first  32-bit score: 14

T_ID 5 -    Initiator second 32-bit score: 16

T_ID 5 -    Initiator chosen PN pattern:   17 46 62 CD

T_ID 5 -    Reflector first  32-bit score: 6

T_ID 5 -    Reflector second 32-bit score: 6

T_ID 5 -    Reflector chosen PN pattern:   C8 C5 97 E5

T_ID 6 - SS marker position initiator (8 bits):   1E  -  Range = 29 (decimal); hr1 = 3
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   54  -  Range = 29 (decimal); hr1 = 9
(decimal)
```

*Sample Data*

```
T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=18 | Mode=2

T_ID 2 - Submode insertion (8 bits):     54  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    D9  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=19 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

**********************************************************************

New DRBG octets:    CE 5A DE D7 79 BC 1F DB E5 AE A3 FC 92 0B 05 31

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    CE  -  Range = 24 (decimal); hr1 = 19 (decimal)

Step=20 | Mode=0

**********************************************************************

New DRBG octets:   33 3E 3C 9D 31 E0 44 B6 7B 3F 20 38 AF D6 73 AC

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              33 3E 3C 9D 31 E0 44 B6 7B 3F 20 38
AF D6 73 AC

T_ID 5 -    Initiator first  32-bit score: 22

T_ID 5 -    Initiator second 32-bit score: 10

T_ID 5 -    Initiator chosen PN pattern:   31 E0 44 B6

T_ID 5 -    Reflector first  32-bit score: 12
```

*Sample Data*

```
T_ID 5 -    Reflector second 32-bit score: 8

T_ID 5 -    Reflector chosen PN pattern:   AF D6 73 AC

Step=21 | Mode=0

**********************************************************************

New DRBG octets:   AF 9F 78 48 26 29 96 A3 85 E6 60 4F 82 3B 87 C2

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              AF 9F 78 48 26 29 96 A3 85 E6 60 4F
82 3B 87 C2

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   AF 9F 78 48

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 18

T_ID 5 -    Reflector chosen PN pattern:   85 E6 60 4F

Step=22 | Mode=0

**********************************************************************

New DRBG octets:   3C 60 3B D2 E1 A3 41 67 B3 78 97 ED B4 B2 72 BF

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              3C 60 3B D2 E1 A3 41 67 B3 78 97 ED
B4 B2 72 BF

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   E1 A3 41 67
```

*Sample Data*

```
T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   B4 B2 72 BF

Step=23 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    5A  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=24 | Mode=3

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    DE  -  Range = 24 (decimal); hr1 = 20 (decimal)

*********************************************************************

New DRBG octets:   A9 83 4E D5 F0 71 EC A6 35 17 86 10 A6 EE AA 04

*********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              A9 83 4E D5 F0 71 EC A6 35 17 86 10
A6 EE AA 04

T_ID 5 -    Initiator first  32-bit score: 14

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   F0 71 EC A6

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   35 17 86 10

T_ID 6 - SS marker position initiator (8 bits):   6E  -  Range = 29 (decimal); hr1 = 12
(decimal)
```

*Sample Data*

```
T_ID 6 - SS marker position reflector (8 bits):   3B  -  Range = 29 (decimal); hr1 = 6
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=25 | Mode=2

T_ID 0 - non-mode0 channel (8bits):      0C  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      59  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      DD  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      BD  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      D9  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):      F3  -  Range = 7 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):      05  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      20  -  Range = 9 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      F7  -  Range = 10 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8bits):      D0  -  Range = 11 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):      EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8bits):      37  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      8E  -  Range = 14 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      BE  -  Range = 15 (decimal); hr1 = 11 (decimal)

**********************************************************************

New DRBG octets:    73 F3 11 6C E1 09 5A 29 BF 4D E7 AD D7 CC 2C D4

**********************************************************************
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):      73  -  Range = 16 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      F3  -  Range = 17 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8bits):      11  -  Range = 18 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (16bits):     6C E1 - Range = 19 (decimal); hr1 = 16 (decimal)

T_ID 0 -     Shuffled channels: 9 19 14 6 5 7 8 17 12 11 4 16 2 3 13 15 20 10 18

T_ID 2 - Submode insertion (8 bits):     E0  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    D7  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=26 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    79  -  Range = 24 (decimal); hr1 = 11 (decimal)

Step=27 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    BC  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=28 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    1F  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=29 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    DB  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=30 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0
```

*Sample Data*

```
T_ID 4 - Antenna path perm. (8 bits):    E5  -  Range = 24 (decimal); hr1 = 21 (decimal)

Step=31 | Mode=3

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    AE  -  Range = 24 (decimal); hr1 = 16 (decimal)

**********************************************************************

New DRBG octets:   C0 F4 37 CB 2E D8 01 47 35 AF 5E 5D 02 0B 67 A1

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              C0 F4 37 CB 2E D8 01 47 35 AF 5E 5D
02 0B 67 A1

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 16

T_ID 5 -    Initiator chosen PN pattern:   C0 F4 37 CB

T_ID 5 -    Reflector first  32-bit score: 12

T_ID 5 -    Reflector second 32-bit score: 16

T_ID 5 -    Reflector chosen PN pattern:   35 AF 5E 5D

T_ID 6 - SS marker position initiator (8 bits):   A1  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   7E  -  Range = 29 (decimal); hr1 = 14
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=32 | Mode=2

T_ID 2 - Submode insertion (8 bits):     7C  -  Range = 5 (decimal);  hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    A3  -  Range = 24 (decimal); hr1 = 15 (decimal)

Step=33 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    FC  -  Range = 24 (decimal); hr1 = 23 (decimal)

Step=34 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    92  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=35 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (16 bits):    0B 05  -  Range = 24 (decimal);
hr1 = 0 (decimal)

Step=36 | Mode=3

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    31  -  Range = 24 (decimal); hr1 = 4 (decimal)

**********************************************************************

New DRBG octets:   2A 43 AB EB C6 D6 F2 E2 93 41 84 17 32 E6 82 31

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              2A 43 AB EB C6 D6 F2 E2 93 41 84 17
32 E6 82 31

T_ID 5 -    Initiator first  32-bit score: 22

T_ID 5 -    Initiator second 32-bit score: 8
```

*Sample Data*

```
T_ID 5 -    Initiator chosen PN pattern:   C6 D6 F2 E2

T_ID 5 -    Reflector first  32-bit score: 8

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   93 41 84 17

T_ID 6 - SS marker position initiator (8 bits):   46  -  Range = 29 (decimal); hr1 = 7
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   61  -  Range = 29 (decimal); hr1 = 10
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=37 | Mode=2

T_ID 2 - Submode insertion (8 bits):     50  -  Range = 5 (decimal);  hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):         40

**********************************************************************

New DRBG octets:   02 4C 7D 7E 4F 6A 09 9E C1 E0 6D F8 06 91 16 9B

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    02  -  Range = 24 (decimal); hr1 = 0 (decimal)

Step=38 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    4C  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=39 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    7D  -  Range = 24 (decimal); hr1 = 11 (decimal)
```

*Sample Data*

```
Step=40 | Mode=0

**********************************************************************

New DRBG octets:   78 4C A9 80 0A 58 C3 56 12 75 95 0B 6F 84 0E C6

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              78 4C A9 80 0A 58 C3 56 12 75 95 0B
6F 84 0E C6

T_ID 5 -    Initiator first  32-bit score: 4

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   78 4C A9 80

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 10

T_ID 5 -    Reflector chosen PN pattern:   6F 84 0E C6

Step=41 | Mode=0

**********************************************************************

New DRBG octets:   F9 75 C7 DA 8C C1 CB C7 5E A9 A3 F9 C2 EF B6 90

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              F9 75 C7 DA 8C C1 CB C7 5E A9 A3 F9
C2 EF B6 90

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 26

T_ID 5 -    Initiator chosen PN pattern:   F9 75 C7 DA

T_ID 5 -    Reflector first  32-bit score: 12
```

*Sample Data*

```
T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   C2 EF B6 90

Step=42 | Mode=0

**********************************************************************

New DRBG octets:   94 27 94 24 2B 7D 51 F3 3E 76 6F CB 96 52 E3 61

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              94 27 94 24 2B 7D 51 F3 3E 76 6F CB
96 52 E3 61

T_ID 5 -    Initiator first  32-bit score: 8

T_ID 5 -    Initiator second 32-bit score: 14

T_ID 5 -    Initiator chosen PN pattern:   94 27 94 24

T_ID 5 -    Reflector first  32-bit score: 14

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   96 52 E3 61

Step=43 | Mode=3

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    7E   -  Range = 24 (decimal); hr1 = 11 (decimal)

**********************************************************************

New DRBG octets:   51 17 E6 D9 19 D2 9F 36 65 42 19 F8 BC 7E 01 7A

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              51 17 E6 D9 19 D2 9F 36 65 42 19 F8
BC 7E 01 7A

T_ID 5 -    Initiator first  32-bit score: 10
```

*Sample Data*

```
T_ID 5 -    Initiator second 32-bit score: 10

T_ID 5 -    Initiator chosen PN pattern:   19 D2 9F 36

T_ID 5 -    Reflector first  32-bit score: 6

T_ID 5 -    Reflector second 32-bit score: 20

T_ID 5 -    Reflector chosen PN pattern:   65 42 19 F8

T_ID 6 - SS marker position initiator (8 bits):   A5  -  Range = 29 (decimal); hr1 = 18
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   00 90  -  Range = 29 (decimal);
hr1 = 16 (decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=44 | Mode=2

T_ID 2 - Submode insertion (8 bits):     15  -  Range = 5 (decimal);  hr1 = 0 (decimal)

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    4F  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=45 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    6A  -  Range = 24 (decimal); hr1 = 9 (decimal)

Step=46 | Mode=3

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    09  -  Range = 24 (decimal); hr1 = 0 (decimal)

**********************************************************************
```

*Sample Data*

```
New DRBG octets:   0B 4D 44 7F BF CD 8A 16 0D 8C 37 94 7D 89 D2 44

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              0B 4D 44 7F BF CD 8A 16 0D 8C 37 94
7D 89 D2 44

T_ID 5 -    Initiator first  32-bit score: 10

T_ID 5 -    Initiator second 32-bit score: 8

T_ID 5 -    Initiator chosen PN pattern:   BF CD 8A 16

T_ID 5 -    Reflector first  32-bit score: 10

T_ID 5 -    Reflector second 32-bit score: 12

T_ID 5 -    Reflector chosen PN pattern:   0D 8C 37 94

T_ID 6 - SS marker position initiator (8 bits):   67  -  Range = 29 (decimal); hr1 = 11
(decimal)

**********************************************************************

New DRBG octets:   85 85 4E 9A 21 C0 EB 14 70 9B F0 75 D6 B2 D2 1A

**********************************************************************

T_ID 6 - SS marker position reflector (16 bits):   85 85  -  Range = 29 (decimal);
hr1 = 15 (decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

********************************** CS Procedure: 1 **********************************

*** UPDATED K and V ***

K:    F3 F4 CF 24 5F CE 86 B3 C6 9F AE 60 28 7E B5 AE

V:    08 56 98 94 89 1F 2F 30 1E EC DB 38 B6 ED 54 B1
```

*Sample Data*

```
Step=0 | Mode=0

********************************************************************

New DRBG octets:    70 C4 6C 6B 39 EB 34 06 0B 4B F5 3E 37 2D B8 3C

********************************************************************

T_ID 1 - mode0 channel (8bits):          70  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8bits):          C4  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          6C  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):          6B  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):          EB  -  Range = 7 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8bits):          34  -  Range = 8 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8bits):          06  -  Range = 9 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8bits):          0B  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8bits):          4B  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8bits):          F5  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8bits):          3E  -  Range = 13 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (16bits):         37 2D -  Range = 14 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8bits):          B8  -  Range = 15 (decimal); hr1 = 10 (decimal)

T_ID 1 - mode0 channel (8bits):          3C  -  Range = 16 (decimal); hr1 = 3 (decimal)

********************************************************************

New DRBG octets:    E9 42 5B B2 07 D9 3F 82 08 44 AB 9D 59 29 7D 94

********************************************************************
```

*Sample Data*

```
T_ID 1 - mode0 channel (8bits):          E9  -  Range = 17 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8bits):          42  -  Range = 18 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8bits):          5B  -  Range = 19 (decimal); hr1 = 6 (decimal)

T_ID 1 -     Shuffled channels: 11 9 15 17 19 5 20 7 3 10 16 13 12 6 2 18 14 4 8

**********************************************************************

New DRBG octets:    A8 83 E9 3C 5E 8F 03 B0 97 92 05 0C F0 8D A8 8C

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              A8 83 E9 3C 5E 8F 03 B0 97 92 05 0C
F0 8D A8 8C

T_ID 5 -     Initiator first  32-bit score: 6

T_ID 5 -     Initiator second 32-bit score: 16

T_ID 5 -     Initiator chosen PN pattern:   A8 83 E9 3C

T_ID 5 -     Reflector first  32-bit score: 6

T_ID 5 -     Reflector second 32-bit score: 6

T_ID 5 -     Reflector chosen PN pattern:   F0 8D A8 8C

Step=1 | Mode=0

**********************************************************************

New DRBG octets:    B6 3B F5 5D 77 25 FE AB 35 69 E3 05 F8 C6 13 4C

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              B6 3B F5 5D 77 25 FE AB 35 69 E3 05
F8 C6 13 4C

T_ID 5 -     Initiator first  32-bit score: 14
```

*Sample Data*

```
T_ID 5 -     Initiator second 32-bit score: 12

T_ID 5 -     Initiator chosen PN pattern:   77 25 FE AB

T_ID 5 -     Reflector first  32-bit score: 8

T_ID 5 -     Reflector second 32-bit score: 18

T_ID 5 -     Reflector chosen PN pattern:   35 69 E3 05

Step=2 | Mode=0

**********************************************************************

New DRBG octets:    EA 85 C1 73 5D 9C 54 E9 35 7D 25 A0 0C 50 EE A1

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              EA 85 C1 73 5D 9C 54 E9 35 7D 25 A0
0C 50 EE A1

T_ID 5 -     Initiator first  32-bit score: 16

T_ID 5 -     Initiator second 32-bit score: 16

T_ID 5 -     Initiator chosen PN pattern:   5D 9C 54 E9

T_ID 5 -     Reflector first  32-bit score: 18

T_ID 5 -     Reflector second 32-bit score: 14

T_ID 5 -     Reflector chosen PN pattern:   0C 50 EE A1

Step=3 | Mode=2

**********************************************************************

New DRBG octets:    09 66 3B 5C 15 02 92 2F 12 B1 9E 80 97 07 59 2E

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):      09  -  Range = 2 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8bits):      66  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      3B  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      5C  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      15  -  Range = 6 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      02  -  Range = 7 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      92  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8bits):      2F  -  Range = 9 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      12  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      B1  -  Range = 11 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      9E  -  Range = 12 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      80  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):      97  -  Range = 14 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8bits):      07  -  Range = 15 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      59  -  Range = 16 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):      2E  -  Range = 17 (decimal); hr1 = 3 (decimal)

**********************************************************************

New DRBG octets:    71 32 82 FB F9 DA 15 66 58 00 50 86 3A 4F 64 66

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):      71  -  Range = 18 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8bits):      32  -  Range = 19 (decimal); hr1 = 3 (decimal)

T_ID 0 -     Shuffled channels: 16 10 2 20 9 17 14 19 15 8 4 12 7 6 11 5 3 13 18

**********************************************************************
```

*Sample Data*

```
New DRBG octets:    D0 D2 CA 3E A1 E4 B2 3E 5E 1A AA 76 CE CD 27 80

**********************************************************************

T_ID 2 - Submode insertion (8 bits):     D0  -  Range = 5 (decimal);  hr1 = 4 (decimal)

**********************************************************************

New DRBG octets:    53 14 F5 C9 8D 61 9E 80 A0 43 47 65 34 4A 61 F3

**********************************************************************

T_ID 3 - TPM extension (2 bits):         40

**********************************************************************

New DRBG octets:    E8 1E B4 0E 75 3B 86 C4 6F C3 55 5B CB 3F B7 B8

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    E8  -  Range = 24 (decimal); hr1 = 21 (decimal)

Step=4 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    1E  -  Range = 24 (decimal); hr1 = 2 (decimal)

Step=5 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    B4  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=6 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    0E  -  Range = 24 (decimal); hr1 = 1 (decimal)

Step=7 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    75  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=8 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    3B  -  Range = 24 (decimal); hr1 = 5 (decimal)

Step=9 | Mode=3

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    86  -  Range = 24 (decimal); hr1 = 12 (decimal)

**********************************************************************

New DRBG octets:    36 1A F9 D3 08 D4 BE 76 32 4E CC EE 59 EB D0 1B

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              36 1A F9 D3 08 D4 BE 76 32 4E CC EE
59 EB D0 1B

T_ID 5 -     Initiator first  32-bit score: 8

T_ID 5 -     Initiator second 32-bit score: 2

T_ID 5 -     Initiator chosen PN pattern:   08 D4 BE 76

T_ID 5 -     Reflector first  32-bit score: 20

T_ID 5 -     Reflector second 32-bit score: 8

T_ID 5 -     Reflector chosen PN pattern:   59 EB D0 1B

**********************************************************************

New DRBG octets:    F8 B1 E5 EA 82 88 D4 57 3B D7 74 49 8F B2 3C 5D

**********************************************************************
```

*Sample Data*

```
T_ID 6 - SS marker position initiator (8 bits):   F8  -  Range = 29 (decimal); hr1 = 28
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   B1 E5  -  Range = 29 (decimal);
hr1 = 26 (decimal)

**********************************************************************

New DRBG octets:    EC C2 57 DF E9 97 C5 CF 63 E4 4C 71 E3 23 34 38

**********************************************************************

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=10 | Mode=2

T_ID 2 - Submode insertion (8 bits):     D2  -  Range = 5 (decimal);  hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    C4  -  Range = 24 (decimal); hr1 = 18 (decimal)

Step=11 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    6F  -  Range = 24 (decimal); hr1 = 10 (decimal)

Step=12 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    C3  -  Range = 24 (decimal); hr1 = 18 (decimal)

Step=13 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    55  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=14 | Mode=2
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    5B  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=15 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (16 bits):    CB 3F  -  Range = 24 (decimal); hr1 = 5 (decimal)


Step=16 | Mode=3

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    B7  -  Range = 24 (decimal); hr1 = 17 (decimal)

**********************************************************************

New DRBG octets:    08 3C 13 AD 40 5C C5 EA F1 90 58 08 0B DF C4 07

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              08 3C 13 AD 40 5C C5 EA F1 90 58 08
0B DF C4 07

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 10

T_ID 5 -     Initiator chosen PN pattern:   40 5C C5 EA

T_ID 5 -     Reflector first  32-bit score: 14

T_ID 5 -     Reflector second 32-bit score: 26

T_ID 5 -     Reflector chosen PN pattern:   F1 90 58 08

T_ID 6 - SS marker position initiator (8 bits):   EA  -  Range = 29 (decimal); hr1 = 26
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   82  -  Range = 29 (decimal); hr1 = 14
```

*Sample Data*

```
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=17 | Mode=2

T_ID 2 - Submode insertion (8 bits):     CA  -  Range = 5 (decimal);  hr1 = 3 (decimal)

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    B8  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=18 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

**********************************************************************

New DRBG octets:    B5 08 D4 4C 3C 86 DC C7 AF 58 4E 5B 8D 79 56 A9

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    B5  -  Range = 24 (decimal); hr1 = 16 (decimal)

Step=19 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    08  -  Range = 24 (decimal); hr1 = 0 (decimal)

Step=20 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    D4    -  Range = 24 (decimal); hr1 = 19 (decimal)

Step=21 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    4C  -  Range = 24 (decimal); hr1 = 7 (decimal)
```

*Sample Data*

```
Step=22 | Mode=0

**********************************************************************

New DRBG octets:    DD A6 1F 35 5C 73 11 35 F4 26 10 C5 E1 4D C8 03

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              DD A6 1F 35 5C 73 11 35 F4 26 10 C5
E1 4D C8 03

T_ID 5 -     Initiator first  32-bit score: 6

T_ID 5 -     Initiator second 32-bit score: 18

T_ID 5 -     Initiator chosen PN pattern:   DD A6 1F 35

T_ID 5 -     Reflector first  32-bit score: 4

T_ID 5 -     Reflector second 32-bit score: 12

T_ID 5 -     Reflector chosen PN pattern:   F4 26 10 C5

Step=23 | Mode=0

**********************************************************************

New DRBG octets:    3B A5 CC 83 BE 04 2A 8A A4 80 0D F2 31 ED 01 6F

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              3B A5 CC 83 BE 04 2A 8A A4 80 0D F2
31 ED 01 6F

T_ID 5 -     Initiator first  32-bit score: 8

T_ID 5 -     Initiator second 32-bit score: 22

T_ID 5 -     Initiator chosen PN pattern:   3B A5 CC 83

T_ID 5 -     Reflector first  32-bit score: 26
```

*Sample Data*

```
T_ID 5 -     Reflector second 32-bit score: 12

T_ID 5 -     Reflector chosen PN pattern:   31 ED 01 6F

Step=24 | Mode=0

**********************************************************************

New DRBG octets:    62 EA 77 67 34 E0 8A D3 A8 AC E8 0E 31 C7 3C 5D

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              62 EA 77 67 34 E0 8A D3 A8 AC E8 0E
31 C7 3C 5D

T_ID 5 -     Initiator first  32-bit score: 12

T_ID 5 -     Initiator second 32-bit score: 6

T_ID 5 -     Initiator chosen PN pattern:   34 E0 8A D3

T_ID 5 -     Reflector first  32-bit score: 16

T_ID 5 -     Reflector second 32-bit score: 30

T_ID 5 -     Reflector chosen PN pattern:   A8 AC E8 0E

Step=25 | Mode=3

T_ID 0 - non-mode0 channel (8bits):      82  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8bits):      FB  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      F9  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      DA  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8bits):      15  -  Range = 6 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8bits):      66  -  Range = 7 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      58  -  Range = 8 (decimal); hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (16bits):     00 50 -  Range = 9 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      86  -  Range = 10 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):      3A  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8bits):      4F  -  Range = 12 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      64  -  Range = 13 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):      66  -  Range = 14 (decimal); hr1 = 5 (decimal)

**********************************************************************

New DRBG octets:    6D 34 4D A5 C3 64 54 34 96 51 E5 B4 58 86 07 0E

**********************************************************************

T_ID 0 - non-mode0 channel (8bits):      6D  -  Range = 15 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8bits):      34  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8bits):      4D  -  Range = 17 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8bits):      A5  -  Range = 18 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8bits):      C3  -  Range = 19 (decimal); hr1 = 14 (decimal)

T_ID 0 -     Shuffled channels: 7 3 12 17 6 18 16 8 9 2 10 19 11 14 20 13 15 5 4

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    3C  -  Range = 24 (decimal); hr1 = 5 (decimal)

**********************************************************************

New DRBG octets:    84 97 9C FC 7B FA 84 EC CE 00 EB 40 2C 5D E0 0C

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              84 97 9C FC 7B FA 84 EC CE 00 EB 40
2C 5D E0 0C
```

*Sample Data*

```
T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 10

T_ID 5 -     Initiator chosen PN pattern:   7B FA 84 EC

T_ID 5 -     Reflector first  32-bit score: 16

T_ID 5 -     Reflector second 32-bit score: 10

T_ID 5 -     Reflector chosen PN pattern:   2C 5D E0 0C

T_ID 6 - SS marker position initiator (8 bits):   88  -  Range = 29 (decimal); hr1 = 15
(decimal)

T_ID 6 - SS marker position reflector (16 bits):   D4 57  -  Range = 29 (decimal);
hr1 = 9 (decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=26 | Mode=2

T_ID 2 - Submode insertion (8 bits):     3E  -  Range = 5 (decimal);  hr1 = 1 (decimal)

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    86  -  Range = 24 (decimal); hr1 = 12 (decimal)

Step=27 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    DC  -  Range = 24 (decimal); hr1 = 20 (decimal)

Step=28 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    C7  -  Range = 24 (decimal); hr1 = 18 (decimal)

Step=29 | Mode=3
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    AF  -  Range = 24 (decimal); hr1 = 16 (decimal)

**********************************************************************

New DRBG octets:    E2 6C F2 67 2D 8A 75 9C 0D 0C 53 5D DB 18 15 83

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              E2 6C F2 67 2D 8A 75 9C 0D 0C 53 5D
DB 18 15 83

T_ID 5 -     Initiator first  32-bit score: 18

T_ID 5 -     Initiator second 32-bit score: 14

T_ID 5 -     Initiator chosen PN pattern:   2D 8A 75 9C

T_ID 5 -     Reflector first  32-bit score: 6

T_ID 5 -     Reflector second 32-bit score: 6

T_ID 5 -     Reflector chosen PN pattern:   DB 18 15 83

T_ID 6 - SS marker position initiator (8 bits):   3B  -  Range = 29 (decimal); hr1 = 6
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   D7  -  Range = 29 (decimal); hr1 = 24
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=30 | Mode=2

T_ID 2 - Submode insertion (8 bits):     A1  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    58  -  Range = 24 (decimal); hr1 = 8 (decimal)
```

*Sample Data*

```
Step=31 | Mode=2

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    4E  -  Range = 24 (decimal); hr1 = 7 (decimal)

Step=32 | Mode=2

T_ID 3 - TPM extension (2 bits):         C0

T_ID 4 - Antenna path perm. (8 bits):    5B  -  Range = 24 (decimal); hr1 = 8 (decimal)

Step=33 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    8D  -  Range = 24 (decimal); hr1 = 13 (decimal)

Step=34 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    79  -  Range = 24 (decimal); hr1 = 11 (decimal)

Step=35 | Mode=3

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    56  -  Range = 24 (decimal); hr1 = 8 (decimal)

**********************************************************************

New DRBG octets:    54 D5 4A 4F C8 3A 49 AF 6A A1 10 42 56 0B A5 AD

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              54 D5 4A 4F C8 3A 49 AF 6A A1 10 42
56 0B A5 AD

T_ID 5 -     Initiator first  32-bit score: 26

T_ID 5 -     Initiator second 32-bit score: 10
```

*Sample Data*

```
T_ID 5 -     Initiator chosen PN pattern:   C8 3A 49 AF

T_ID 5 -     Reflector first  32-bit score: 16

T_ID 5 -     Reflector second 32-bit score: 20

T_ID 5 -     Reflector chosen PN pattern:   6A A1 10 42

T_ID 6 - SS marker position initiator (8 bits):   74  -  Range = 29 (decimal); hr1 = 13
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   49  -  Range = 29 (decimal); hr1 = 8
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  80

T_ID 7 - SS marker sig. sel. reflector (1 bits):  80

Step=36 | Mode=2

T_ID 2 - Submode insertion (8 bits):     E4  -  Range = 5 (decimal);  hr1 = 4 (decimal)

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    A9  -  Range = 24 (decimal); hr1 = 15 (decimal)

Step=37 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

**********************************************************************

New DRBG octets:    EF B6 A1 07 43 B2 B9 66 0E DE 52 97 74 F0 33 84

**********************************************************************

T_ID 4 - Antenna path perm. (8 bits):    EF  -  Range = 24 (decimal); hr1 = 22 (decimal)

Step=38 | Mode=2

T_ID 3 - TPM extension (2 bits):         80
```

*Sample Data*

```
T_ID 4 - Antenna path perm. (8 bits):    B6  -  Range = 24 (decimal); hr1 = 17 (decimal)

Step=39 | Mode=2

T_ID 3 - TPM extension (2 bits):         80

T_ID 4 - Antenna path perm. (8 bits):    A1  -  Range = 24 (decimal); hr1 = 15 (decimal)

Step=40 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    07  -  Range = 24 (decimal); hr1 = 0 (decimal)

Step=41 | Mode=2

T_ID 3 - TPM extension (2 bits):         00

T_ID 4 - Antenna path perm. (8 bits):    43  -  Range = 24 (decimal); hr1 = 6 (decimal)

Step=42 | Mode=0

**********************************************************************

New DRBG octets:    78 68 87 E5 EE E6 7F 3E CC 7C C4 7E 5C 63 92 29

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              78 68 87 E5 EE E6 7F 3E CC 7C C4 7E
5C 63 92 29

T_ID 5 -     Initiator first  32-bit score: 10

T_ID 5 -     Initiator second 32-bit score: 14

T_ID 5 -     Initiator chosen PN pattern:   78 68 87 E5

T_ID 5 -     Reflector first  32-bit score: 24

T_ID 5 -     Reflector second 32-bit score: 16

T_ID 5 -     Reflector chosen PN pattern:   5C 63 92 29
```

*Sample Data*

```
Step=43 | Mode=0

**********************************************************************

New DRBG octets:    9A D2 89 1B 16 E0 63 3F 65 19 BD BA 84 60 71 98

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              9A D2 89 1B 16 E0 63 3F 65 19 BD BA
84 60 71 98

T_ID 5 -     Initiator first  32-bit score: 20

T_ID 5 -     Initiator second 32-bit score: 12

T_ID 5 -     Initiator chosen PN pattern:   16 E0 63 3F

T_ID 5 -     Reflector first  32-bit score: 14

T_ID 5 -     Reflector second 32-bit score: 26

T_ID 5 -     Reflector chosen PN pattern:   65 19 BD BA

Step=44 | Mode=0

**********************************************************************

New DRBG octets:    D2 2D 15 56 BF 28 7E 3A 2D 6F C3 29 0D 51 E9 5B

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              D2 2D 15 56 BF 28 7E 3A 2D 6F C3 29
0D 51 E9 5B

T_ID 5 -     Initiator first  32-bit score: 26

T_ID 5 -     Initiator second 32-bit score: 16

T_ID 5 -     Initiator chosen PN pattern:   BF 28 7E 3A

T_ID 5 -     Reflector first  32-bit score: 8

T_ID 5 -     Reflector second 32-bit score: 16
```

*Sample Data*

```
T_ID 5 -     Reflector chosen PN pattern:   2D 6F C3 29

Step=45 | Mode=3

T_ID 3 - TPM extension (2 bits):         40

T_ID 4 - Antenna path perm. (8 bits):    B2  -  Range = 24 (decimal); hr1 = 16 (decimal)

**********************************************************************

New DRBG octets:    46 6E 07 57 46 BB 98 F2 69 A2 66 BA 3C 6F EE 5E

**********************************************************************

T_ID 5 - RTT AA candidates (128 bits):              46 6E 07 57 46 BB 98 F2 69 A2 66 BA
3C 6F EE 5E

T_ID 5 -     Initiator first  32-bit score: 8

T_ID 5 -     Initiator second 32-bit score: 14

T_ID 5 -     Initiator chosen PN pattern:   46 6E 07 57

T_ID 5 -     Reflector first  32-bit score: 18

T_ID 5 -     Reflector second 32-bit score: 8

T_ID 5 -     Reflector chosen PN pattern:   3C 6F EE 5E

T_ID 6 - SS marker position initiator (8 bits):   8F  -  Range = 29 (decimal); hr1 = 16
(decimal)

T_ID 6 - SS marker position reflector (8 bits):   B2  -  Range = 29 (decimal); hr1 = 20
(decimal)

T_ID 7 - SS marker sig. sel. initiator (1 bits):  00

T_ID 7 - SS marker sig. sel. reflector (1 bits):  00

Step=46 | Mode=2

T_ID 2 - Submode insertion (8 bits):     B2  -  Range = 5 (decimal); hr1 = 3 (decimal)
```

*Sample Data*

```
T_ID 3 - TPM extension (2 bits):          00

T_ID 4 - Antenna path perm. (8 bits):     B9  -  Range = 24 (decimal); hr1 = 17 (decimal)
```

### 8.1.4 Set 4

The sample data in this section includes CS scheduling considerations.

From the CS Configuration procedure:

- T_SW, Central and Peripheral = 10 µs, 10 µs

From the CS Configuration procedure:

- ChM = 0x00 00 00 00 00 00 00 1F 1F FC (Used: 2-12, 16-20, Unused: 0-1, 13-15, 21-79)
- ChM_Repetition = 2
- Main_Mode = 0x03 (Mode-3)
- Sub Mode = 0x02 (Mode-2)
- Main_Mode_Min_Steps = 1
- Main_Mode_Max_Steps = 6
- Main_Mode_Repetition = 2
- Mode_0_Steps = 1
- CS_SYNC_PHY = 0x02 (2M PHY)
- RTT_Type = 0x02 (96-bit sounding sequence)
- Role = 0b01 (Reflector)
- ChSel = 0 (Channel Selection Algorithm 3b)
- Ch3c Shape = 0
- Ch3c Jump = 2
- T IP1 = 0x07 (145 µs)
- T IP2 = 0x07 (145 µs)
- T FCS = 0x09 (150 µs)
- T PM = 0x00 (10 µs)

From the CS Start procedure:

- T_EVENT_OFFSET (from Offset_Min and Offset_Max) = 0 ns

*Sample Data*

- T_EVENT_INTERVAL (from Event_Interval) = 1
- N_SUBEVENTS_PER_EVENT (from Subvents_Per_Event) = 3
- T_SUBEVENT_INTERVAL (from Subevent_Interval) = 5.000 ms
- T_SUBEVENT_LEN (from Subevent_Len) = 4.000 ms
- T_MAX_PROCEDURE_LEN (from Max_Procedure_Len) = 40.4000 s
- ACI = 7 (ACI 7)

Other scheduling parameters:

- ACL connection interval = 40.000 ms

```
****************************** INSTANTIATION FUNCTION ******************************

h9() instantiation

********************************************************************************

Entropy input Peripheral (CS_IV_P): E1:0B:C2:8A:0B:FD:DF:E9

Entropy input Central (CS_IV_C): 3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV): E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral (CS_IN_P): 9F F4 77 C1

Nonce Central(CS_IN_C): 86 73 84 0D

Nonce (CS_IN): 9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C): 64 A6 74 96 78 68 F1 43

Personalization string (CS_PV): C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

    ********************** f8 function start **********************

        *************************** f7 function start ***************************

        f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

        f7 V||S input:  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
```

*Sample Data*

```
00 00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                        9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96
78 68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

        f7 K2 output:   8B 2B 06 DC 52 2D 3E 0A F0 A5 0C AF 48 10 E0 35

        ************************** f7 function end ****************************

        ************************** f7 function start **************************

        f7 K input:     00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

        f7 V||S input:  00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 28
00 00 00 20 E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

                        9F F4 77 C1 86 73 84 0D C9 80 DE DF 98 82 ED 44 64 A6 74 96
78 68 F1 43 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

        f7 X output:    A3 4F BE 57 F8 F9 7E 34 9D 15 A3 76 79 60 74 64

        ************************** f7 function end ****************************

        f8 SM output: B6 02 B1 B2 8C 6F 0A 3D DA E6 37 B4 84 25 08 7D DC 18 8C 89 A1
B0 CD FD A1 E8 FC 66 C9 99 97 50

    ********************** f8 function end **********************

********************************* INITIAL K and V ***********************************

K:              EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:              DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

*******************************************************************************

***************************** CHANNEL SOUNDING SEQUENCE *****************************
    cr1 start. nChannels 16, CSStepCount 0, Mode-0, channels 2 3 4 5 6 7 8 9 10 11 12
16 17 18 19 20
*************************************************************
CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
New DRBG octets: FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32; 8-bit value FF
*************************************************************
```

*Sample Data*

```
    hr1. output 0x01, Trand 0x01FE; range 2, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte BC
    hr1. output 0x02, Trand 0x0234; range 3, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte C1
    hr1. output 0x03, Trand 0x0304; range 4, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte CA
    hr1. output 0x03, Trand 0x03F2; range 5, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte 39
    hr1. output 0x01, Trand 0x0156; range 6, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte A6
    hr1. output 0x04, Trand 0x048A; range 7, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte 9D
    hr1. output 0x04, Trand 0x04E8; range 8, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte C4
    hr1. output 0x06, Trand 0x06E4; range 9, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte 07
    hr1. output 0x00, Trand 0x0046; range 10, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte 38
    hr1. output 0x02, Trand 0x0268; range 11, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte EF
    hr1. output 0x0B, Trand 0x0B34; range 12, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
```

*Sample Data*

```
Byte 33
    hr1. output 0x02, Trand 0x0297; range 13, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte D9
    hr1. output 0x0B, Trand 0x0BDE; range 14, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte D1
    hr1. output 0x0C, Trand 0x0C3F; range 15, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 0, Transaction ID = 1 (Mode0 channels).
Byte 35
    hr1. output 0x03, Trand 0x0350; range 16, Step Counter = 0, Transaction ID = 1
(Mode0 channels)
  cr1 end. channels 11 7 17 20 9 3 10 8 5 2 4 18 19 16 12 6
  cr1 start. nChannels 16, CSStepCount 1, Non-Mode-0, channels 2 3 4 5 6 7 8 9 10 11
12 16 17 18 19 20
************************************************************
CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0 channels).
New DRBG octets: 20 7C E5 39 E1 DA 42 2A 36 C5 11 44 DC 1E 08 03; 8-bit value 20
************************************************************
    hr1. output 0x00, Trand 0x0040; range 2, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 7C
    hr1. output 0x01, Trand 0x0174; range 3, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte E5
    hr1. output 0x03, Trand 0x0394; range 4, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 39
    hr1. output 0x01, Trand 0x011D; range 5, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte E1
    hr1. output 0x05, Trand 0x0546; range 6, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte DA
    hr1. output 0x05, Trand 0x05F6; range 7, Step Counter = 1, Transaction ID = 0
```

*Sample Data*

```
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 42
    hr1. output 0x02, Trand 0x0210; range 8, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 2A
    hr1. output 0x01, Trand 0x017A; range 9, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 36
    hr1. output 0x02, Trand 0x021C; range 10, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte C5
    hr1. output 0x08, Trand 0x0877; range 11, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 11
    hr1. output 0x00, Trand 0x00CC; range 12, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 44
    hr1. output 0x03, Trand 0x0374; range 13, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte DC
    hr1. output 0x0C, Trand 0x0C08; range 14, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 1E
    hr1. output 0x01, Trand 0x01C2; range 15, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 0 (Non-Mode0
channels). Byte 08
    hr1. output 0x00, Trand 0x0080; range 16, Step Counter = 1, Transaction ID = 0
(Non-Mode0 channels)
  cr1 end. channels 20 19 11 17 4 8 7 2 12 9 6 3 18 5 10 16
************************************************************
CS DRBG for 128 bits, Step Counter = 0, Transaction ID = 5 (AA generation).
New DRGB octets: values 0C 00 E4 AA 6C 37 6A B8 00 B8 C5 5D F0 79 BC 3A
************************************************************
  AA generation, CSStepCount 0. AA_I 0x6C376AB8, AA_R 0xF079BC3A; s0 0x0C00E4AA
```

*Sample Data*

```
(score 12), s1 0x6C376AB8 (score 8), s2 0x00B8C55D (score 16), s3 0xF079BC3A (score
16)
************************************************************
CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 2 (Subevent submode).
New DRBG octets: 1F 97 C7 7F 86 64 B4 9D DE 97 33 FB 39 23 02 64; 8-bit value 1F
************************************************************
    hr1. output 0x00, Trand 0x00BA; range 6, Step Counter = 1, Transaction ID = 2
(Subevent submode)
  Submode insertion: 1 Main-Mode steps. CSStepCount 1, Main_Mode_Min 1, Main_Mode_Max
6
************************************************************
CS DRBG for 128 bits, Step Counter = 1, Transaction ID = 5 (AA generation).
New DRGB octets: values 01 1C AE 4E 99 47 D1 B5 D0 6A CD DA C7 F8 F9 37
************************************************************
  AA generation, CSStepCount 1. AA_I 0x011CAE4E, AA_R 0xD06ACDDA; s0 0x011CAE4E
(score 6), s1 0x9947D1B5 (score 8), s2 0xD06ACDDA (score 10), s3 0xC7F8F937 (score 14)
************************************************************
CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 2D 5E 35 B8 D9 17 3A 3F 48 3F 8E 53 0C 42 1D 95; 8-bit value 2D
************************************************************
    hr1. output 0x0B, Trand 0x0B40; range 64, Step Counter = 1, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 6 (Sounding sequence
marker position). Byte 5E
    hr1. output 0x1B, Trand 0x1B8A; range 75, Step Counter = 1, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 11. CSStepCount 1, sounding length 96
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 6 (Sounding sequence
marker position). Byte 35
    hr1. output 0x0D, Trand 0x0D40; range 64, Step Counter = 1, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 6 (Sounding sequence
marker position). Byte B8
    hr1. output 0x35, Trand 0x35E8; range 75, Step Counter = 1, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 13. CSStepCount 1, sounding length 96
************************************************************
CS DRBG for 1 bits, Step Counter = 1, Transaction ID = 7 (Sounding sequence marker
signal).
New DRBG octets: AC C3 B3 47 DF 5C 15 FC 2F 65 25 A4 D0 0C 40 F9; 1-bit value 1 bit:
0b1
************************************************************
```

*Sample Data*

```
  Marker bits in SS values. CSStepCount 1, sounding length 96, value 1
    CS DRBG for 1 bits, Step Counter = 1, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 1, sounding length 96, value 0
************************************************************
CS DRBG for 8 bits, Step Counter = 1, Transaction ID = 4 (Antenna permutation).
New DRBG octets: EF 4F 17 9B 63 8A DC 61 C7 E3 16 5D B1 EC 10 A8; 8-bit value EF
************************************************************
    hr1. output 0x16, Trand 0x1668; range 24, Step Counter = 1, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 22. N_AP = 4, CSStepCount 1
************************************************************
CS DRBG for 2 bits, Step Counter = 1, Transaction ID = 3 (Tone extension presence).
New DRBG octets: 1E 94 C0 D2 A9 92 B9 C9 0D 82 16 BD 4A 12 76 FD; 2-bit value 2 bits:
0b00
************************************************************
  Tone extensions: I 0, R 0. CSStepCount 1
        CS DRBG for 2 bits, Step Counter = 2, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 2
        CS DRBG for 8 bits, Step Counter = 2, Transaction ID = 4 (Antenna
permutation). Byte 4F
    hr1. output 0x07, Trand 0x0768; range 24, Step Counter = 2, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 2
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 2 (Subevent submode).
Byte 97
    hr1. output 0x03, Trand 0x038A; range 6, Step Counter = 3, Transaction ID = 2
(Subevent submode)
  Submode insertion: 4 Main-Mode steps. CSStepCount 3, Main_Mode_Min 1, Main_Mode_Max
6
************************************************************
CS DRBG for 128 bits, Step Counter = 3, Transaction ID = 5 (AA generation).
New DRGB octets: values 3F 1C 25 6E F2 F1 63 0E 36 66 EE E1 4E 84 2F E9
************************************************************
  AA generation, CSStepCount 3. AA_I 0x3F1C256E, AA_R 0x4E842FE9; s0 0x3F1C256E
(score 8), s1 0xF2F1630E (score 18), s2 0x3666EEE1 (score 14), s3 0x4E842FE9 (score
12)
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 6 (Sounding sequence
marker position). Byte D9
    hr1. output 0x36, Trand 0x3640; range 64, Step Counter = 3, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 6 (Sounding sequence
```

*Sample Data*

```
marker position). Byte 17
    hr1. output 0x06, Trand 0x06BD; range 75, Step Counter = 3, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 54, 73. CSStepCount 3, sounding length 96
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 6 (Sounding sequence
marker position). Byte 3A
    hr1. output 0x0E, Trand 0x0E80; range 64, Step Counter = 3, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 6 (Sounding sequence
marker position). Byte 3F
    hr1. output 0x12, Trand 0x1275; range 75, Step Counter = 3, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 14, 85. CSStepCount 3, sounding length 96
        CS DRBG for 1 bits, Step Counter = 3, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 3, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 3, sounding length 96, values 1, 0
        CS DRBG for 1 bits, Step Counter = 3, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 3, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 3, sounding length 96, values 1, 1
        CS DRBG for 8 bits, Step Counter = 3, Transaction ID = 4 (Antenna
permutation). Byte 17
    hr1. output 0x02, Trand 0x0228; range 24, Step Counter = 3, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 2. N_AP = 4, CSStepCount 3
        CS DRBG for 2 bits, Step Counter = 3, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 3
************************************************************
CS DRBG for 128 bits, Step Counter = 4, Transaction ID = 5 (AA generation).
New DRGB octets: values 90 BE 37 AB 55 D2 BB D3 AF 2E F4 6B D0 82 B8 54
************************************************************
  AA generation, CSStepCount 4. AA_I 0x90BE37AB, AA_R 0xAF2EF46B; s0 0x90BE37AB
(score 6), s1 0x55D2BBD3 (score 24), s2 0xAF2EF46B (score 12), s3 0xD082B854 (score
18)
        CS DRBG for 8 bits, Step Counter = 4, Transaction ID = 6 (Sounding sequence
marker position). Byte 48
    hr1. output 0x12, Trand 0x1200; range 64, Step Counter = 4, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 4, Transaction ID = 6 (Sounding sequence
```

*Sample Data*

```
marker position). Byte 3F
    hr1. output 0x12, Trand 0x1275; range 75, Step Counter = 4, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 18, 85. CSStepCount 4, sounding length 96
        CS DRBG for 8 bits, Step Counter = 4, Transaction ID = 6 (Sounding sequence
marker position). Byte 8E
    hr1. output 0x23, Trand 0x2380; range 64, Step Counter = 4, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 4, Transaction ID = 6 (Sounding sequence
marker position). Byte 53
    hr1. output 0x18, Trand 0x1851; range 75, Step Counter = 4, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 35, 91. CSStepCount 4, sounding length 96
        CS DRBG for 1 bits, Step Counter = 4, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 4, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 4, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 4, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 4, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 4, sounding length 96, values 1, 1
        CS DRBG for 8 bits, Step Counter = 4, Transaction ID = 4 (Antenna
permutation). Byte 9B
    hr1. output 0x0E, Trand 0x0E88; range 24, Step Counter = 4, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 14. N_AP = 4, CSStepCount 4
        CS DRBG for 2 bits, Step Counter = 4, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 4
************************************************************
CS DRBG for 128 bits, Step Counter = 5, Transaction ID = 5 (AA generation).
New DRGB octets: values 36 73 03 D7 26 2F 29 07 37 34 75 8A 94 E4 6E 08
************************************************************
  AA generation, CSStepCount 5. AA_I 0x262F2907, AA_R 0x94E46E08; s0 0x367303D7
(score 14), s1 0x262F2907 (score 4), s2 0x3734758A (score 16), s3 0x94E46E08 (score 6)
        CS DRBG for 8 bits, Step Counter = 5, Transaction ID = 6 (Sounding sequence
marker position). Byte 0C
    hr1. output 0x03, Trand 0x0300; range 64, Step Counter = 5, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 5, Transaction ID = 6 (Sounding sequence
marker position). Byte 42
```

*Sample Data*

```
    hr1. output 0x13, Trand 0x1356; range 75, Step Counter = 5, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 3, 86. CSStepCount 5, sounding length 96
        CS DRBG for 8 bits, Step Counter = 5, Transaction ID = 6 (Sounding sequence
marker position). Byte 1D
    hr1. output 0x07, Trand 0x0740; range 64, Step Counter = 5, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 5, Transaction ID = 6 (Sounding sequence
marker position). Byte 95
    hr1. output 0x2B, Trand 0x2BA7; range 75, Step Counter = 5, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 7. CSStepCount 5, sounding length 96
        CS DRBG for 1 bits, Step Counter = 5, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 5, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 5, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 5, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 5, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 5, Transaction ID = 4 (Antenna
permutation). Byte 63
    hr1. output 0x09, Trand 0x0948; range 24, Step Counter = 5, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 9. N_AP = 4, CSStepCount 5
        CS DRBG for 2 bits, Step Counter = 5, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 5
************************************************************
CS DRBG for 128 bits, Step Counter = 6, Transaction ID = 5 (AA generation).
New DRGB octets: values F3 C9 2E 6B FA D7 FB 2A F1 D2 AA 34 51 9D 04 48
************************************************************
  AA generation, CSStepCount 6. AA_I 0xF3C92E6B, AA_R 0x519D0448; s0 0xF3C92E6B
(score 10), s1 0xFAD7FB2A (score 20), s2 0xF1D2AA34 (score 24), s3 0x519D0448 (score
2)
************************************************************
CS DRBG for 128 bits, Step Counter = 7, Transaction ID = 5 (AA generation).
New DRGB octets: values C9 04 A9 35 4F 0E 6B 3B A0 88 EB D9 43 27 CC 87
************************************************************
  AA generation, CSStepCount 7. AA_I 0x4F0E6B3B, AA_R 0xA088EBD9; s0 0xC904A935
(score 18), s1 0x4F0E6B3B (score 14), s2 0xA088EBD9 (score 4), s3 0x4327CC87 (score
14)
************************************************************
```

*Sample Data*

```
CS DRBG for 8 bits, Step Counter = 7, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: D5 E1 B2 C1 7F 9C 60 80 6B E6 66 B4 0E 96 F5 1C; 8-bit value D5
************************************************************
    hr1. output 0x35, Trand 0x3540; range 64, Step Counter = 7, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 7, Transaction ID = 6 (Sounding sequence
marker position). Byte E1
    hr1. output 0x41, Trand 0x41EB; range 75, Step Counter = 7, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 53. CSStepCount 7, sounding length 96
        CS DRBG for 8 bits, Step Counter = 7, Transaction ID = 6 (Sounding sequence
marker position). Byte B2
    hr1. output 0x2C, Trand 0x2C80; range 64, Step Counter = 7, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 7, Transaction ID = 6 (Sounding sequence
marker position). Byte C1
    hr1. output 0x38, Trand 0x388B; range 75, Step Counter = 7, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 44. CSStepCount 7, sounding length 96
        CS DRBG for 1 bits, Step Counter = 7, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 7, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 7, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 7, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 7, Transaction ID = 4 (Antenna
permutation). Byte 8A
    hr1. output 0x0C, Trand 0x0CF0; range 24, Step Counter = 7, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 12. N_AP = 4, CSStepCount 7
        CS DRBG for 2 bits, Step Counter = 7, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 7
************************************************************
CS DRBG for 128 bits, Step Counter = 8, Transaction ID = 5 (AA generation).
New DRGB octets: values 8B 55 C7 D8 BD 91 D2 12 BB 8D AE FA 2A B9 07 75
************************************************************
  AA generation, CSStepCount 8. AA_I 0xBD91D212, AA_R 0xBB8DAEFA; s0 0x8B55C7D8
(score 16), s1 0xBD91D212 (score 12), s2 0xBB8DAEFA (score 6), s3 0x2AB90775 (score
24)
        CS DRBG for 8 bits, Step Counter = 8, Transaction ID = 6 (Sounding sequence
marker position). Byte 7F
```

*Sample Data*

```
    hr1. output 0x1F, Trand 0x1FC0; range 64, Step Counter = 8, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 8, Transaction ID = 6 (Sounding sequence
marker position). Byte 9C
    hr1. output 0x2D, Trand 0x2DB4; range 75, Step Counter = 8, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 31. CSStepCount 8, sounding length 96
        CS DRBG for 8 bits, Step Counter = 8, Transaction ID = 6 (Sounding sequence
marker position). Byte 60
    hr1. output 0x18, Trand 0x1800; range 64, Step Counter = 8, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 8, Transaction ID = 6 (Sounding sequence
marker position). Byte 80
    hr1. output 0x25, Trand 0x2580; range 75, Step Counter = 8, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 24. CSStepCount 8, sounding length 96
        CS DRBG for 1 bits, Step Counter = 8, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 8, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 8, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 8, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 8, Transaction ID = 4 (Antenna
permutation). Byte DC
    hr1. output 0x14, Trand 0x14A0; range 24, Step Counter = 8, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 20. N_AP = 4, CSStepCount 8
        CS DRBG for 2 bits, Step Counter = 8, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 8
************************************************************
CS DRBG for 128 bits, Step Counter = 9, Transaction ID = 5 (AA generation).
New DRGB octets: values A6 93 A0 3E 16 9E ED DC 3D 54 2C 1A 6B 2E B7 48
************************************************************
  AA generation, CSStepCount 9. AA_I 0x169EEDDC, AA_R 0x6B2EB748; s0 0xA693A03E
(score 8), s1 0x169EEDDC (score 6), s2 0x3D542C1A (score 20), s3 0x6B2EB748 (score 14)
        CS DRBG for 8 bits, Step Counter = 9, Transaction ID = 6 (Sounding sequence
marker position). Byte 6B
    hr1. output 0x1A, Trand 0x1AC0; range 64, Step Counter = 9, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 9, Transaction ID = 6 (Sounding sequence
marker position). Byte E6
    hr1. output 0x43, Trand 0x4362; range 75, Step Counter = 9, Transaction ID = 6
```

*Sample Data*

```
(Sounding sequence marker position)
  Marker bits in SS position 26. CSStepCount 9, sounding length 96
        CS DRBG for 8 bits, Step Counter = 9, Transaction ID = 6 (Sounding sequence
marker position). Byte 66
    hr1. output 0x19, Trand 0x1980; range 64, Step Counter = 9, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 9, Transaction ID = 6 (Sounding sequence
marker position). Byte B4
    hr1. output 0x34, Trand 0x34BC; range 75, Step Counter = 9, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 25. CSStepCount 9, sounding length 96
        CS DRBG for 1 bits, Step Counter = 9, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 9, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 9, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 9, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 9, Transaction ID = 4 (Antenna
permutation). Byte 61
    hr1. output 0x09, Trand 0x0918; range 24, Step Counter = 9, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 9. N_AP = 4, CSStepCount 9
        CS DRBG for 2 bits, Step Counter = 9, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 9
        CS DRBG for 2 bits, Step Counter = 10, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 10
        CS DRBG for 8 bits, Step Counter = 10, Transaction ID = 4 (Antenna
permutation). Byte C7
    hr1. output 0x12, Trand 0x12A8; range 24, Step Counter = 10, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 18. N_AP = 4, CSStepCount 10
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 2 (Subevent submode).
Byte C7
    hr1. output 0x04, Trand 0x04AA; range 6, Step Counter = 11, Transaction ID = 2
(Subevent submode)
  Submode insertion: 5 Main-Mode steps. CSStepCount 11, Main_Mode_Min 1,
Main_Mode_Max 6
************************************************************
CS DRBG for 128 bits, Step Counter = 11, Transaction ID = 5 (AA generation).
New DRGB octets: values 73 11 B1 24 CA 69 75 5F 70 C8 26 79 2D DC CC D3
************************************************************
```

*Sample Data*

```
  AA generation, CSStepCount 11. AA_I 0xCA69755F, AA_R 0x70C82679; s0 0x7311B124
(score 16), s1 0xCA69755F (score 16), s2 0x70C82679 (score 16), s3 0x2DDCCCD3 (score
20)
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 6 (Sounding sequence
marker position). Byte 0E
    hr1. output 0x03, Trand 0x0380; range 64, Step Counter = 11, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 6 (Sounding sequence
marker position). Byte 96
    hr1. output 0x2B, Trand 0x2BF2; range 75, Step Counter = 11, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 3. CSStepCount 11, sounding length 96
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 6 (Sounding sequence
marker position). Byte F5
    hr1. output 0x3D, Trand 0x3D40; range 64, Step Counter = 11, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 6 (Sounding sequence
marker position). Byte 1C
    hr1. output 0x08, Trand 0x0834; range 75, Step Counter = 11, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 61, 75. CSStepCount 11, sounding length 96
        CS DRBG for 1 bits, Step Counter = 11, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 11, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 11, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 11, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 11, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 11, Transaction ID = 4 (Antenna
permutation). Byte E3
    hr1. output 0x15, Trand 0x1548; range 24, Step Counter = 11, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 21. N_AP = 4, CSStepCount 11
        CS DRBG for 2 bits, Step Counter = 11, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 11
************************************************************
CS DRBG for 128 bits, Step Counter = 12, Transaction ID = 5 (AA generation).
New DRGB octets: values DB 20 3C 13 38 B5 04 81 CE 7D 20 3F 8A CD D0 46
************************************************************
  AA generation, CSStepCount 12. AA_I 0x38B50481, AA_R 0x8ACDD046; s0 0xDB203C13
(score 18), s1 0x38B50481 (score 6), s2 0xCE7D203F (score 16), s3 0x8ACDD046 (score 6)
```

*Sample Data*

```
************************************************************
CS DRBG for 128 bits, Step Counter = 13, Transaction ID = 5 (AA generation).
New DRGB octets: values D9 AE 74 FF 98 EA 9C 6F D0 52 CC 49 B7 A7 6D 63
************************************************************
  AA generation, CSStepCount 13. AA_I 0xD9AE74FF, AA_R 0xB7A76D63; s0 0xD9AE74FF
(score 8), s1 0x98EA9C6F (score 16), s2 0xD052CC49 (score 16), s3 0xB7A76D63 (score
14)
************************************************************
CS DRBG for 8 bits, Step Counter = 13, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 54 CF 67 DD 3B 4A 95 4B 01 55 E5 B8 C0 23 33 6E; 8-bit value 54
************************************************************
    hr1. output 0x15, Trand 0x1500; range 64, Step Counter = 13, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 13, Transaction ID = 6 (Sounding sequence
marker position). Byte CF
    hr1. output 0x3C, Trand 0x3CA5; range 75, Step Counter = 13, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 21. CSStepCount 13, sounding length 96
        CS DRBG for 8 bits, Step Counter = 13, Transaction ID = 6 (Sounding sequence
marker position). Byte 67
    hr1. output 0x19, Trand 0x19C0; range 64, Step Counter = 13, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 13, Transaction ID = 6 (Sounding sequence
marker position). Byte DD
    hr1. output 0x40, Trand 0x40BF; range 75, Step Counter = 13, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 25. CSStepCount 13, sounding length 96
        CS DRBG for 1 bits, Step Counter = 13, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 13, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 13, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 13, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 13, Transaction ID = 4 (Antenna
permutation). Byte 16
    hr1. output 0x02, Trand 0x0210; range 24, Step Counter = 13, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 2. N_AP = 4, CSStepCount 13
        CS DRBG for 2 bits, Step Counter = 13, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 13
************************************************************
```

*Sample Data*

```
CS DRBG for 128 bits, Step Counter = 14, Transaction ID = 5 (AA generation).
New DRGB octets: values F7 21 97 86 9D 82 D2 EC 80 69 7C 5B 57 17 64 70
************************************************************
  AA generation, CSStepCount 14. AA_I 0xF7219786, AA_R 0x57176470; s0 0xF7219786
(score 10), s1 0x9D82D2EC (score 12), s2 0x80697C5B (score 14), s3 0x57176470 (score
10)
        CS DRBG for 8 bits, Step Counter = 14, Transaction ID = 6 (Sounding sequence
marker position). Byte 3B
    hr1. output 0x0E, Trand 0x0EC0; range 64, Step Counter = 14, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 14, Transaction ID = 6 (Sounding sequence
marker position). Byte 4A
    hr1. output 0x15, Trand 0x15AE; range 75, Step Counter = 14, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 14, 88. CSStepCount 14, sounding length 96
        CS DRBG for 8 bits, Step Counter = 14, Transaction ID = 6 (Sounding sequence
marker position). Byte 95
    hr1. output 0x25, Trand 0x2540; range 64, Step Counter = 14, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 14, Transaction ID = 6 (Sounding sequence
marker position). Byte 4B
    hr1. output 0x15, Trand 0x15F9; range 75, Step Counter = 14, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 37, 88. CSStepCount 14, sounding length 96
        CS DRBG for 1 bits, Step Counter = 14, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 14, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 14, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 14, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 14, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 14, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 14, Transaction ID = 4 (Antenna
permutation). Byte 5D
    hr1. output 0x08, Trand 0x08B8; range 24, Step Counter = 14, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 8. N_AP = 4, CSStepCount 14
        CS DRBG for 2 bits, Step Counter = 14, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 14
************************************************************
```

*Sample Data*

```
CS DRBG for 128 bits, Step Counter = 15, Transaction ID = 5 (AA generation).
New DRGB octets: values D5 8F 34 7D AA 8B 79 C1 D3 72 7F 5D 78 7E 62 31
************************************************************
  AA generation, CSStepCount 15. AA_I 0xD58F347D, AA_R 0xD3727F5D; s0 0xD58F347D
(score 8), s1 0xAA8B79C1 (score 20), s2 0xD3727F5D (score 6), s3 0x787E6231 (score 24)
        CS DRBG for 8 bits, Step Counter = 15, Transaction ID = 6 (Sounding sequence
marker position). Byte 01
    hr1. output 0x00, Trand 0x0040; range 64, Step Counter = 15, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 15, Transaction ID = 6 (Sounding sequence
marker position). Byte 55
    hr1. output 0x18, Trand 0x18E7; range 75, Step Counter = 15, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 0, 91. CSStepCount 15, sounding length 96
        CS DRBG for 8 bits, Step Counter = 15, Transaction ID = 6 (Sounding sequence
marker position). Byte E5
    hr1. output 0x39, Trand 0x3940; range 64, Step Counter = 15, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 15, Transaction ID = 6 (Sounding sequence
marker position). Byte B8
    hr1. output 0x35, Trand 0x35E8; range 75, Step Counter = 15, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 57. CSStepCount 15, sounding length 96
        CS DRBG for 1 bits, Step Counter = 15, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 15, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 15, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 15, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 15, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 15, Transaction ID = 4 (Antenna
permutation). Byte B1
    hr1. output 0x10, Trand 0x1098; range 24, Step Counter = 15, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 16. N_AP = 4, CSStepCount 15
        CS DRBG for 2 bits, Step Counter = 15, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 15
************************************************************
CS DRBG for 128 bits, Step Counter = 16, Transaction ID = 5 (AA generation).
New DRGB octets: values 47 A8 15 62 8A 2C F2 7C 0E 49 E3 47 B3 64 D0 CC
************************************************************
```

*Sample Data*

```
  AA generation, CSStepCount 16. AA_I 0x8A2CF27C, AA_R 0x0E49E347; s0 0x47A81562
(score 22), s1 0x8A2CF27C (score 8), s2 0x0E49E347 (score 16), s3 0xB364D0CC (score
26)
        CS DRBG for 8 bits, Step Counter = 16, Transaction ID = 6 (Sounding sequence
marker position). Byte C0
    hr1. output 0x30, Trand 0x3000; range 64, Step Counter = 16, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 16, Transaction ID = 6 (Sounding sequence
marker position). Byte 23
    hr1. output 0x0A, Trand 0x0A41; range 75, Step Counter = 16, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 48, 77. CSStepCount 16, sounding length 96
        CS DRBG for 8 bits, Step Counter = 16, Transaction ID = 6 (Sounding sequence
marker position). Byte 33
    hr1. output 0x0C, Trand 0x0CC0; range 64, Step Counter = 16, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 16, Transaction ID = 6 (Sounding sequence
marker position). Byte 6E
    hr1. output 0x20, Trand 0x203A; range 75, Step Counter = 16, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 12. CSStepCount 16, sounding length 96
        CS DRBG for 1 bits, Step Counter = 16, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 16, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 16, sounding length 96, values 1, 1
        CS DRBG for 1 bits, Step Counter = 16, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 16, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 16, Transaction ID = 4 (Antenna
permutation). Byte EC
    hr1. output 0x16, Trand 0x1620; range 24, Step Counter = 16, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 22. N_AP = 4, CSStepCount 16
        CS DRBG for 2 bits, Step Counter = 16, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 16
************************************************************
CS DRBG for 128 bits, Step Counter = 17, Transaction ID = 5 (AA generation).
New DRGB octets: values 17 46 62 CD 92 22 DB 8B 0C 22 D1 19 C8 C5 97 E5
************************************************************
  AA generation, CSStepCount 17. AA_I 0x174662CD, AA_R 0xC8C597E5; s0 0x174662CD
(score 14), s1 0x9222DB8B (score 16), s2 0x0C22D119 (score 6), s3 0xC8C597E5 (score 6)
```

*Sample Data*

```
************************************************************
CS DRBG for 8 bits, Step Counter = 17, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 59 83 F2 4E 7D 9B FE 8C 8B C1 21 96 18 FB 86 18; 8-bit value 59
************************************************************
    hr1. output 0x16, Trand 0x1640; range 64, Step Counter = 17, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 17, Transaction ID = 6 (Sounding sequence
marker position). Byte 83
    hr1. output 0x26, Trand 0x2661; range 75, Step Counter = 17, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 22. CSStepCount 17, sounding length 96
        CS DRBG for 8 bits, Step Counter = 17, Transaction ID = 6 (Sounding sequence
marker position). Byte F2
    hr1. output 0x3C, Trand 0x3C80; range 64, Step Counter = 17, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 17, Transaction ID = 6 (Sounding sequence
marker position). Byte 4E
    hr1. output 0x16, Trand 0x16DA; range 75, Step Counter = 17, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 60, 89. CSStepCount 17, sounding length 96
        CS DRBG for 1 bits, Step Counter = 17, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 17, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 17, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 17, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 17, sounding length 96, values 1, 1
        CS DRBG for 8 bits, Step Counter = 17, Transaction ID = 4 (Antenna
permutation). Byte 10
    hr1. output 0x01, Trand 0x0180; range 24, Step Counter = 17, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 1. N_AP = 4, CSStepCount 17
        CS DRBG for 2 bits, Step Counter = 17, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 17
************************************************************
CS DRBG for 128 bits, Step Counter = 18, Transaction ID = 5 (AA generation).
New DRGB octets: values A7 3F 8A F4 F9 87 B5 C0 05 CB B8 53 2F BD 25 04
************************************************************
  AA generation, CSStepCount 18. AA_I 0xA73F8AF4, AA_R 0x05CBB853; s0 0xA73F8AF4
(score 10), s1 0xF987B5C0 (score 12), s2 0x05CBB853 (score 6), s3 0x2FBD2504 (score
```

*Sample Data*

```
16)
************************************************************
CS DRBG for 128 bits, Step Counter = 19, Transaction ID = 5 (AA generation).
New DRGB octets: values 5F 1B 30 8F F1 89 96 06 93 6D 1D 34 E9 77 0F 47
************************************************************
  AA generation, CSStepCount 19. AA_I 0x5F1B308F, AA_R 0xE9770F47; s0 0x5F1B308F
(score 12), s1 0xF1899606 (score 14), s2 0x936D1D34 (score 24), s3 0xE9770F47 (score
12)
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 6 (Sounding sequence
marker position). Byte 7D
    hr1. output 0x1F, Trand 0x1F40; range 64, Step Counter = 19, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 6 (Sounding sequence
marker position). Byte 9B
    hr1. output 0x2D, Trand 0x2D69; range 75, Step Counter = 19, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 31. CSStepCount 19, sounding length 96
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 6 (Sounding sequence
marker position). Byte FE
    hr1. output 0x3F, Trand 0x3F80; range 64, Step Counter = 19, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 6 (Sounding sequence
marker position). Byte 8C
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 6 (Sounding sequence
marker position). Byte 8B
        hr1 using two bytes. output 28, Trand1 2904; range 75, counters: Step 19,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 63. CSStepCount 19, sounding length 96
        CS DRBG for 1 bits, Step Counter = 19, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 19, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 19, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 19, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 19, Transaction ID = 4 (Antenna
permutation). Byte A8
    hr1. output 0x0F, Trand 0x0FC0; range 24, Step Counter = 19, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 15. N_AP = 4, CSStepCount 19
        CS DRBG for 2 bits, Step Counter = 19, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 19
************************************************************
```

*Sample Data*

```
CS DRBG for 128 bits, Step Counter = 20, Transaction ID = 5 (AA generation).
New DRGB octets: values 33 3E 3C 9D 31 E0 44 B6 7B 3F 20 38 AF D6 73 AC
************************************************************
  AA generation, CSStepCount 20. AA_I 0x31E044B6, AA_R 0xAFD673AC; s0 0x333E3C9D
(score 22), s1 0x31E044B6 (score 10), s2 0x7B3F2038 (score 12), s3 0xAFD673AC (score
8)
        CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 6 (Sounding sequence
marker position). Byte C1
    hr1. output 0x30, Trand 0x3040; range 64, Step Counter = 20, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 6 (Sounding sequence
marker position). Byte 21
    hr1. output 0x09, Trand 0x09AB; range 75, Step Counter = 20, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 48, 76. CSStepCount 20, sounding length 96
        CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 6 (Sounding sequence
marker position). Byte 96
    hr1. output 0x25, Trand 0x2580; range 64, Step Counter = 20, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 6 (Sounding sequence
marker position). Byte 18
        CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 6 (Sounding sequence
marker position). Byte FB
        hr1 using two bytes. output 49, Trand1 0708; range 75, counters: Step 20,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 37. CSStepCount 20, sounding length 96
        CS DRBG for 1 bits, Step Counter = 20, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 20, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 20, sounding length 96, values 1, 0
        CS DRBG for 1 bits, Step Counter = 20, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 20, sounding length 96, value 1
************************************************************
CS DRBG for 8 bits, Step Counter = 20, Transaction ID = 4 (Antenna permutation).
New DRBG octets: F6 F9 7A 5D 55 E7 58 1E 9E F3 66 E7 D6 26 5D 55; 8-bit value F6
************************************************************
    hr1. output 0x17, Trand 0x1710; range 24, Step Counter = 20, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 23. N_AP = 4, CSStepCount 20
        CS DRBG for 2 bits, Step Counter = 20, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
```

*Sample Data*

```
  Tone extensions: I 1, R 0. CSStepCount 20
************************************************************
CS DRBG for 128 bits, Step Counter = 21, Transaction ID = 5 (AA generation).
New DRGB octets: values AF 9F 78 48 26 29 96 A3 85 E6 60 4F 82 3B 87 C2
************************************************************
  AA generation, CSStepCount 21. AA_I 0xAF9F7848, AA_R 0x85E6604F; s0 0xAF9F7848
(score 10), s1 0x262996A3 (score 14), s2 0x85E6604F (score 12), s3 0x823B87C2 (score
18)
        CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 6 (Sounding sequence
marker position). Byte 86
    hr1. output 0x21, Trand 0x2180; range 64, Step Counter = 21, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 6 (Sounding sequence
marker position). Byte 18
************************************************************
CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: C8 82 7E E7 A8 74 52 9C 10 55 12 10 1B BD 4B 7E; 8-bit value C8
************************************************************
    hr1 using two bytes. output 3A, Trand1 0708; range 75, counters: Step 21,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 33. CSStepCount 21, sounding length 96
        CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 6 (Sounding sequence
marker position). Byte 82
    hr1. output 0x20, Trand 0x2080; range 64, Step Counter = 21, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 6 (Sounding sequence
marker position). Byte 7E
    hr1. output 0x24, Trand 0x24EA; range 75, Step Counter = 21, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 32. CSStepCount 21, sounding length 96
        CS DRBG for 1 bits, Step Counter = 21, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 21, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 21, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 21, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 21, Transaction ID = 4 (Antenna
permutation). Byte F9
    hr1. output 0x17, Trand 0x1758; range 24, Step Counter = 21, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 23. N_AP = 4, CSStepCount 21
        CS DRBG for 2 bits, Step Counter = 21, Transaction ID = 3 (Tone extension
```

*Sample Data*

```
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 21
        CS DRBG for 2 bits, Step Counter = 22, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 22
        CS DRBG for 8 bits, Step Counter = 22, Transaction ID = 4 (Antenna
permutation). Byte 7A
    hr1. output 0x0B, Trand 0x0B70; range 24, Step Counter = 22, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 11. N_AP = 4, CSStepCount 22
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 2 (Subevent submode).
Byte 7F
    hr1. output 0x02, Trand 0x02FA; range 6, Step Counter = 23, Transaction ID = 2
(Subevent submode)
  Submode insertion: 3 Main-Mode steps. CSStepCount 23, Main_Mode_Min 1,
Main_Mode_Max 6
************************************************************
CS DRBG for 128 bits, Step Counter = 23, Transaction ID = 5 (AA generation).
New DRGB octets: values 59 5F E1 AA 5F 57 BA 90 82 5F 07 64 35 91 56 A4
************************************************************
  AA generation, CSStepCount 23. AA_I 0x5F57BA90, AA_R 0x825F0764; s0 0x595FE1AA
(score 18), s1 0x5F57BA90 (score 18), s2 0x825F0764 (score 10), s3 0x359156A4 (score
16)
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 6 (Sounding sequence
marker position). Byte E7
    hr1. output 0x39, Trand 0x39C0; range 64, Step Counter = 23, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 6 (Sounding sequence
marker position). Byte A8
    hr1. output 0x31, Trand 0x3138; range 75, Step Counter = 23, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 57. CSStepCount 23, sounding length 96
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 6 (Sounding sequence
marker position). Byte 74
    hr1. output 0x1D, Trand 0x1D00; range 64, Step Counter = 23, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 6 (Sounding sequence
marker position). Byte 52
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 6 (Sounding sequence
marker position). Byte 9C
        hr1 using two bytes. output 2D, Trand1 1806; range 75, counters: Step 23,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 29. CSStepCount 23, sounding length 96
```

*Sample Data*

```
        CS DRBG for 1 bits, Step Counter = 23, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 23, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 23, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 23, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 23, Transaction ID = 4 (Antenna
permutation). Byte 5D
    hr1. output 0x08, Trand 0x08B8; range 24, Step Counter = 23, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 8. N_AP = 4, CSStepCount 23
        CS DRBG for 2 bits, Step Counter = 23, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 23
************************************************************
CS DRBG for 128 bits, Step Counter = 24, Transaction ID = 5 (AA generation).
New DRGB octets: values A9 83 4E D5 F0 71 EC A6 35 17 86 10 A6 EE AA 04
************************************************************
  AA generation, CSStepCount 24. AA_I 0xF071ECA6, AA_R 0x35178610; s0 0xA9834ED5
(score 14), s1 0xF071ECA6 (score 14), s2 0x35178610 (score 10), s3 0xA6EEAA04 (score
16)
************************************************************
CS DRBG for 128 bits, Step Counter = 25, Transaction ID = 5 (AA generation).
New DRGB octets: values 72 4B 10 38 16 33 91 44 07 98 41 98 61 3B C2 E8
************************************************************
  AA generation, CSStepCount 25. AA_I 0x724B1038, AA_R 0x613BC2E8; s0 0x724B1038
(score 8), s1 0x16339144 (score 12), s2 0x07984198 (score 20), s3 0x613BC2E8 (score 6)
        CS DRBG for 8 bits, Step Counter = 25, Transaction ID = 6 (Sounding sequence
marker position). Byte 10
    hr1. output 0x04, Trand 0x0400; range 64, Step Counter = 25, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 25, Transaction ID = 6 (Sounding sequence
marker position). Byte 55
    hr1. output 0x18, Trand 0x18E7; range 75, Step Counter = 25, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 4, 91. CSStepCount 25, sounding length 96
        CS DRBG for 8 bits, Step Counter = 25, Transaction ID = 6 (Sounding sequence
marker position). Byte 12
    hr1. output 0x04, Trand 0x0480; range 64, Step Counter = 25, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 25, Transaction ID = 6 (Sounding sequence
marker position). Byte 10
    hr1. output 0x04, Trand 0x04B0; range 75, Step Counter = 25, Transaction ID = 6
```

*Sample Data*

```
(Sounding sequence marker position)
  Marker bits in SS positions 4, 71. CSStepCount 25, sounding length 96
        CS DRBG for 1 bits, Step Counter = 25, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 25, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 25, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 25, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 25, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 25, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 25, Transaction ID = 4 (Antenna
permutation). Byte 55
    hr1. output 0x07, Trand 0x07F8; range 24, Step Counter = 25, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 25
        CS DRBG for 2 bits, Step Counter = 25, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 25
************************************************************
CS DRBG for 128 bits, Step Counter = 26, Transaction ID = 5 (AA generation).
New DRGB octets: values D4 FD E6 0E 54 46 3E 92 58 22 3D 9F 00 51 D1 81
************************************************************
  AA generation, CSStepCount 26. AA_I 0xD4FDE60E, AA_R 0x58223D9F; s0 0xD4FDE60E
(score 8), s1 0x54463E92 (score 10), s2 0x58223D9F (score 6), s3 0x0051D181 (score
18)
        CS DRBG for 8 bits, Step Counter = 26, Transaction ID = 6 (Sounding sequence
marker position). Byte 1B
    hr1. output 0x06, Trand 0x06C0; range 64, Step Counter = 26, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 26, Transaction ID = 6 (Sounding sequence
marker position). Byte BD
    hr1. output 0x37, Trand 0x375F; range 75, Step Counter = 26, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 6. CSStepCount 26, sounding length 96
        CS DRBG for 8 bits, Step Counter = 26, Transaction ID = 6 (Sounding sequence
marker position). Byte 4B
    hr1. output 0x12, Trand 0x12C0; range 64, Step Counter = 26, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 26, Transaction ID = 6 (Sounding sequence
marker position). Byte 7E
    hr1. output 0x24, Trand 0x24EA; range 75, Step Counter = 26, Transaction ID = 6
```

*Sample Data*

```
(Sounding sequence marker position)
  Marker bits in SS position 18. CSStepCount 26, sounding length 96
        CS DRBG for 1 bits, Step Counter = 26, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 26, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 26, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 26, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 26, Transaction ID = 4 (Antenna
permutation). Byte E7
    hr1. output 0x15, Trand 0x15A8; range 24, Step Counter = 26, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 21. N_AP = 4, CSStepCount 26
        CS DRBG for 2 bits, Step Counter = 26, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 26
**************************************************************
CS DRBG for 128 bits, Step Counter = 27, Transaction ID = 5 (AA generation).
New DRGB octets: values 3D 89 58 94 3A 6E F4 A1 EB 4B 7E E0 F9 40 73 9F
**************************************************************
  AA generation, CSStepCount 27. AA_I 0x3A6EF4A1, AA_R 0xEB4B7EE0; s0 0x3D895894
(score 8), s1 0x3A6EF4A1 (score 4), s2 0xEB4B7EE0 (score 14), s3 0xF940739F (score
16)
**************************************************************
CS DRBG for 8 bits, Step Counter = 27, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 88 12 42 F5 19 44 C4 F3 50 A4 45 6A 1E 9E 28 33; 8-bit value 88
**************************************************************
    hr1. output 0x22, Trand 0x2200; range 64, Step Counter = 27, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 27, Transaction ID = 6 (Sounding sequence
marker position). Byte 12
    hr1. output 0x05, Trand 0x0546; range 75, Step Counter = 27, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 34, 72. CSStepCount 27, sounding length 96
        CS DRBG for 8 bits, Step Counter = 27, Transaction ID = 6 (Sounding sequence
marker position). Byte 42
    hr1. output 0x10, Trand 0x1080; range 64, Step Counter = 27, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 27, Transaction ID = 6 (Sounding sequence
marker position). Byte F5
    hr1. output 0x47, Trand 0x47C7; range 75, Step Counter = 27, Transaction ID = 6
(Sounding sequence marker position)
```

*Sample Data*

```
  Marker bits in SS position 16. CSStepCount 27, sounding length 96
        CS DRBG for 1 bits, Step Counter = 27, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 27, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 27, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 27, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 27, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 27, Transaction ID = 4 (Antenna
permutation). Byte 58
    hr1. output 0x08, Trand 0x0840; range 24, Step Counter = 27, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 8. N_AP = 4, CSStepCount 27
        CS DRBG for 2 bits, Step Counter = 27, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 27
************************************************************
CS DRBG for 128 bits, Step Counter = 28, Transaction ID = 5 (AA generation).
New DRGB octets: values 7B D8 09 20 2F E7 7B 50 C9 6E 79 FF EE 47 0D B8
************************************************************
  AA generation, CSStepCount 28. AA_I 0x2FE77B50, AA_R 0xEE470DB8; s0 0x7BD80920
(score 24), s1 0x2FE77B50 (score 12), s2 0xC96E79FF (score 16), s3 0xEE470DB8 (score
16)
        CS DRBG for 8 bits, Step Counter = 28, Transaction ID = 6 (Sounding sequence
marker position). Byte 19
    hr1. output 0x06, Trand 0x0640; range 64, Step Counter = 28, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 28, Transaction ID = 6 (Sounding sequence
marker position). Byte 44
    hr1. output 0x13, Trand 0x13EC; range 75, Step Counter = 28, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 6, 86. CSStepCount 28, sounding length 96
        CS DRBG for 8 bits, Step Counter = 28, Transaction ID = 6 (Sounding sequence
marker position). Byte C4
    hr1. output 0x31, Trand 0x3100; range 64, Step Counter = 28, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 28, Transaction ID = 6 (Sounding sequence
marker position). Byte F3
    hr1. output 0x47, Trand 0x4731; range 75, Step Counter = 28, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 49. CSStepCount 28, sounding length 96
        CS DRBG for 1 bits, Step Counter = 28, Transaction ID = 7 (Sounding sequence
```

*Sample Data*

```
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 28, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 28, sounding length 96, values 1, 1
        CS DRBG for 1 bits, Step Counter = 28, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 28, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 28, Transaction ID = 4 (Antenna
permutation). Byte 1E
    hr1. output 0x02, Trand 0x02D0; range 24, Step Counter = 28, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 2. N_AP = 4, CSStepCount 28
        CS DRBG for 2 bits, Step Counter = 28, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 28
  cr1 start. nChannels 16, CSStepCount 29, Non-Mode-0, channels 2 3 4 5 6 7 8 9 10 11
12 16 17 18 19 20
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 03
    hr1. output 0x00, Trand 0x0006; range 2, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
************************************************************
CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0 channels).
New DRBG octets: 19 58 6B 7C A5 9B 9F 54 28 81 15 A6 1C E3 F7 CC; 8-bit value 19
************************************************************
    hr1. output 0x00, Trand 0x004B; range 3, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 58
    hr1. output 0x01, Trand 0x0160; range 4, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 6B
    hr1. output 0x02, Trand 0x0217; range 5, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 7C
    hr1. output 0x02, Trand 0x02E8; range 6, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte A5
    hr1. output 0x04, Trand 0x0483; range 7, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
```

*Sample Data*

```
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 9B
    hr1. output 0x04, Trand 0x04D8; range 8, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 9F
    hr1. output 0x05, Trand 0x0597; range 9, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 54
    hr1. output 0x03, Trand 0x0348; range 10, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 28
    hr1. output 0x01, Trand 0x01B8; range 11, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 81
    hr1. output 0x06, Trand 0x060C; range 12, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 15
    hr1. output 0x01, Trand 0x0111; range 13, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte A6
    hr1. output 0x09, Trand 0x0914; range 14, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte 1C
    hr1. output 0x01, Trand 0x01A4; range 15, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 0 (Non-Mode0
channels). Byte E3
    hr1. output 0x0E, Trand 0x0E30; range 16, Step Counter = 29, Transaction ID = 0
(Non-Mode0 channels)
  cr1 end. channels 4 19 7 11 9 10 16 8 6 18 5 3 12 2 20 17
        CS DRBG for 2 bits, Step Counter = 29, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 29
        CS DRBG for 8 bits, Step Counter = 29, Transaction ID = 4 (Antenna
permutation). Byte 9E
    hr1. output 0x0E, Trand 0x0ED0; range 24, Step Counter = 29, Transaction ID = 4
```

*Sample Data*

```
(Antenna permutation)
  Antenna Path Permutation Index: 14. N_AP = 4, CSStepCount 29
**************************************************************
CS DRBG for 128 bits, Step Counter = 30, Transaction ID = 5 (AA generation).
New DRGB octets: values BC A0 66 6E 2C F2 DF 02 95 FF DB D9 B9 E8 18 06
**************************************************************
  AA generation, CSStepCount 30. AA_I 0x2CF2DF02, AA_R 0xB9E81806; s0 0xBCA0666E
(score 8), s1 0x2CF2DF02 (score 8), s2 0x95FFDBD9 (score 22), s3 0xB9E81806 (score 12)
**************************************************************
CS DRBG for 128 bits, Step Counter = 31, Transaction ID = 5 (AA generation).
New DRGB octets: values C0 F4 37 CB 2E D8 01 47 35 AF 5E 5D 02 0B 67 A1
**************************************************************
  AA generation, CSStepCount 31. AA_I 0xC0F437CB, AA_R 0x35AF5E5D; s0 0xC0F437CB
(score 10), s1 0x2ED80147 (score 16), s2 0x35AF5E5D (score 12), s3 0x020B67A1 (score
16)
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 6 (Sounding sequence
marker position). Byte 50
    hr1. output 0x14, Trand 0x1400; range 64, Step Counter = 31, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 6 (Sounding sequence
marker position). Byte A4
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 6 (Sounding sequence
marker position). Byte 45
        hr1 using two bytes. output 14, Trand1 300C; range 75, counters: Step 31,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS positions 20, 87. CSStepCount 31, sounding length 96
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 6 (Sounding sequence
marker position). Byte 6A
    hr1. output 0x1A, Trand 0x1A80; range 64, Step Counter = 31, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 6 (Sounding sequence
marker position). Byte 1E
    hr1. output 0x08, Trand 0x08CA; range 75, Step Counter = 31, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 26, 75. CSStepCount 31, sounding length 96
        CS DRBG for 1 bits, Step Counter = 31, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 31, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 31, sounding length 96, values 1, 1
        CS DRBG for 1 bits, Step Counter = 31, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 31, Transaction ID = 7 (Sounding sequence
```

*Sample Data*

```
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 31, sounding length 96, values 1, 1
        CS DRBG for 8 bits, Step Counter = 31, Transaction ID = 4 (Antenna
permutation). Byte F3
    hr1. output 0x16, Trand 0x16C8; range 24, Step Counter = 31, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 22. N_AP = 4, CSStepCount 31
        CS DRBG for 2 bits, Step Counter = 31, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 31
************************************************************
CS DRBG for 128 bits, Step Counter = 32, Transaction ID = 5 (AA generation).
New DRGB octets: values 2C E0 B1 90 2B FA F8 37 7A B0 91 BF A7 37 8B DC
************************************************************
  AA generation, CSStepCount 32. AA_I 0x2CE0B190, AA_R 0x7AB091BF; s0 0x2CE0B190
(score 10), s1 0x2BFAF837 (score 16), s2 0x7AB091BF (score 8), s3 0xA7378BDC (score
10)
        CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 6 (Sounding sequence
marker position). Byte 9E
    hr1. output 0x27, Trand 0x2780; range 64, Step Counter = 32, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 6 (Sounding sequence
marker position). Byte 28
    hr1. output 0x0B, Trand 0x0BB8; range 75, Step Counter = 32, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 39, 78. CSStepCount 32, sounding length 96
        CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 6 (Sounding sequence
marker position). Byte 33
    hr1. output 0x0C, Trand 0x0CC0; range 64, Step Counter = 32, Transaction ID = 6
(Sounding sequence marker position)
************************************************************
CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: AB 5D A9 65 12 A9 DE 3E 1C FB B0 3B A4 DD A2 76; 8-bit value AB
************************************************************
        CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 6 (Sounding sequence
marker position). Byte 5D
     hr1 using two bytes. output 1B, Trand1 3219; range 75, counters: Step 32,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 12. CSStepCount 32, sounding length 96
        CS DRBG for 1 bits, Step Counter = 32, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 32, Transaction ID = 7 (Sounding sequence
```

*Sample Data*

```
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 32, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 32, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 32, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 32, Transaction ID = 4 (Antenna
permutation). Byte 66
    hr1. output 0x09, Trand 0x0990; range 24, Step Counter = 32, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 9. N_AP = 4, CSStepCount 32
        CS DRBG for 2 bits, Step Counter = 32, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 32
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 2 (Subevent submode).
Byte 86
    hr1. output 0x03, Trand 0x0324; range 6, Step Counter = 33, Transaction ID = 2
(Subevent submode)
  Submode insertion: 4 Main-Mode steps. CSStepCount 33, Main_Mode_Min 1,
Main_Mode_Max 6
************************************************************
CS DRBG for 128 bits, Step Counter = 33, Transaction ID = 5 (AA generation).
New DRGB octets: values 02 4B 9B 77 E9 8D AA D0 FF F0 9D 22 BE 1F CF 12
************************************************************
  AA generation, CSStepCount 33. AA_I 0xE98DAAD0, AA_R 0xBE1FCF12; s0 0x024B9B77
(score 14), s1 0xE98DAAD0 (score 10), s2 0xFFF09D22 (score 26), s3 0xBE1FCF12 (score
12)
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 6 (Sounding sequence
marker position). Byte A9
    hr1. output 0x2A, Trand 0x2A40; range 64, Step Counter = 33, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 6 (Sounding sequence
marker position). Byte 65
    hr1. output 0x1D, Trand 0x1D97; range 75, Step Counter = 33, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 42. CSStepCount 33, sounding length 96
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 6 (Sounding sequence
marker position). Byte 12
    hr1. output 0x04, Trand 0x0480; range 64, Step Counter = 33, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 6 (Sounding sequence
marker position). Byte A9
    hr1. output 0x31, Trand 0x3183; range 75, Step Counter = 33, Transaction ID = 6
(Sounding sequence marker position)
```

*Sample Data*

```
  Marker bits in SS position 4. CSStepCount 33, sounding length 96
        CS DRBG for 1 bits, Step Counter = 33, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 33, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 33, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 33, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 33, Transaction ID = 4 (Antenna
permutation). Byte E7
    hr1. output 0x15, Trand 0x15A8; range 24, Step Counter = 33, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 21. N_AP = 4, CSStepCount 33
        CS DRBG for 2 bits, Step Counter = 33, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 33
************************************************************
CS DRBG for 128 bits, Step Counter = 34, Transaction ID = 5 (AA generation).
New DRGB octets: values C9 E4 D3 83 7B 25 88 FD 9A F2 FE E1 88 8B 1F 25
************************************************************
  AA generation, CSStepCount 34. AA_I 0x7B2588FD, AA_R 0x888B1F25; s0 0xC9E4D383
(score 14), s1 0x7B2588FD (score 8), s2 0x9AF2FEE1 (score 6), s3 0x888B1F25 (score 4)
        CS DRBG for 8 bits, Step Counter = 34, Transaction ID = 6 (Sounding sequence
marker position). Byte DE
    hr1. output 0x37, Trand 0x3780; range 64, Step Counter = 34, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 34, Transaction ID = 6 (Sounding sequence
marker position). Byte 3E
    hr1. output 0x12, Trand 0x122A; range 75, Step Counter = 34, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 55, 85. CSStepCount 34, sounding length 96
        CS DRBG for 8 bits, Step Counter = 34, Transaction ID = 6 (Sounding sequence
marker position). Byte 1C
    hr1. output 0x07, Trand 0x0700; range 64, Step Counter = 34, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 34, Transaction ID = 6 (Sounding sequence
marker position). Byte FB
    hr1. output 0x49, Trand 0x4989; range 75, Step Counter = 34, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 7. CSStepCount 34, sounding length 96
        CS DRBG for 1 bits, Step Counter = 34, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 34, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
```

*Sample Data*

```
  Marker bits in SS values. CSStepCount 34, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 34, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 34, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 34, Transaction ID = 4 (Antenna
permutation). Byte D6
    hr1. output 0x14, Trand 0x1410; range 24, Step Counter = 34, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 20. N_AP = 4, CSStepCount 34
        CS DRBG for 2 bits, Step Counter = 34, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 34
************************************************************
CS DRBG for 128 bits, Step Counter = 35, Transaction ID = 5 (AA generation).
New DRGB octets: values 34 47 B8 7D FC 69 A8 DD 0C 25 42 B1 E6 47 E2 AD
************************************************************
  AA generation, CSStepCount 35. AA_I 0xFC69A8DD, AA_R 0xE647E2AD; s0 0x3447B87D
(score 10), s1 0xFC69A8DD (score 2), s2 0x0C2542B1 (score 12), s3 0xE647E2AD (score 6)
        CS DRBG for 8 bits, Step Counter = 35, Transaction ID = 6 (Sounding sequence
marker position). Byte B0
    hr1. output 0x2C, Trand 0x2C00; range 64, Step Counter = 35, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 35, Transaction ID = 6 (Sounding sequence
marker position). Byte 3B
    hr1. output 0x11, Trand 0x1149; range 75, Step Counter = 35, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 44, 84. CSStepCount 35, sounding length 96
        CS DRBG for 8 bits, Step Counter = 35, Transaction ID = 6 (Sounding sequence
marker position). Byte A4
    hr1. output 0x29, Trand 0x2900; range 64, Step Counter = 35, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 35, Transaction ID = 6 (Sounding sequence
marker position). Byte DD
    hr1. output 0x40, Trand 0x40BF; range 75, Step Counter = 35, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 41. CSStepCount 35, sounding length 96
        CS DRBG for 1 bits, Step Counter = 35, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 35, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 35, sounding length 96, values 1, 1
        CS DRBG for 1 bits, Step Counter = 35, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
```

*Sample Data*

```
  Marker bits in SS values. CSStepCount 35, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 35, Transaction ID = 4 (Antenna
permutation). Byte 26
    hr1. output 0x03, Trand 0x0390; range 24, Step Counter = 35, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 3. N_AP = 4, CSStepCount 35
        CS DRBG for 2 bits, Step Counter = 35, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 35
************************************************************
CS DRBG for 128 bits, Step Counter = 36, Transaction ID = 5 (AA generation).
New DRGB octets: values 2A 43 AB EB C6 D6 F2 E2 93 41 84 17 32 E6 82 31
************************************************************
  AA generation, CSStepCount 36. AA_I 0xC6D6F2E2, AA_R 0x93418417; s0 0x2A43ABEB
(score 22), s1 0xC6D6F2E2 (score 8), s2 0x93418417 (score 8), s3 0x32E68231 (score
12)
************************************************************
CS DRBG for 128 bits, Step Counter = 37, Transaction ID = 5 (AA generation).
New DRGB octets: values A3 5E 66 E1 FC 19 2D CF 4C 1C 10 C0 41 10 AB 72
************************************************************
  AA generation, CSStepCount 37. AA_I 0xA35E66E1, AA_R 0x4110AB72; s0 0xA35E66E1
(score 12), s1 0xFC192DCF (score 14), s2 0x4C1C10C0 (score 12), s3 0x4110AB72 (score
12)
        CS DRBG for 8 bits, Step Counter = 37, Transaction ID = 6 (Sounding sequence
marker position). Byte A2
    hr1. output 0x28, Trand 0x2880; range 64, Step Counter = 37, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 37, Transaction ID = 6 (Sounding sequence
marker position). Byte 76
    hr1. output 0x22, Trand 0x2292; range 75, Step Counter = 37, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 40. CSStepCount 37, sounding length 96
************************************************************
CS DRBG for 8 bits, Step Counter = 37, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: D6 EB C0 15 35 50 E1 C2 9A E2 50 F6 23 EA 33 27; 8-bit value D6
************************************************************
    hr1. output 0x35, Trand 0x3580; range 64, Step Counter = 37, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 37, Transaction ID = 6 (Sounding sequence
marker position). Byte EB
    hr1. output 0x44, Trand 0x44D9; range 75, Step Counter = 37, Transaction ID = 6
(Sounding sequence marker position)
```

*Sample Data*

```
  Marker bits in SS position 53. CSStepCount 37, sounding length 96
        CS DRBG for 1 bits, Step Counter = 37, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 37, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 37, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 37, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 37, Transaction ID = 4 (Antenna
permutation). Byte 5D
    hr1. output 0x08, Trand 0x08B8; range 24, Step Counter = 37, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 8. N_AP = 4, CSStepCount 37
        CS DRBG for 2 bits, Step Counter = 37, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 37
************************************************************
CS DRBG for 128 bits, Step Counter = 38, Transaction ID = 5 (AA generation).
New DRGB octets: values F6 3F D7 0B A1 5C 65 33 0F A0 C6 9B 90 D2 92 49
************************************************************
  AA generation, CSStepCount 38. AA_I 0xF63FD70B, AA_R 0x0FA0C69B; s0 0xF63FD70B
(score 14), s1 0xA15C6533 (score 16), s2 0x0FA0C69B (score 8), s3 0x90D29249 (score
32)
        CS DRBG for 8 bits, Step Counter = 38, Transaction ID = 6 (Sounding sequence
marker position). Byte C0
    hr1. output 0x30, Trand 0x3000; range 64, Step Counter = 38, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 38, Transaction ID = 6 (Sounding sequence
marker position). Byte 15
    hr1. output 0x06, Trand 0x0627; range 75, Step Counter = 38, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 48, 73. CSStepCount 38, sounding length 96
        CS DRBG for 8 bits, Step Counter = 38, Transaction ID = 6 (Sounding sequence
marker position). Byte 35
    hr1. output 0x0D, Trand 0x0D40; range 64, Step Counter = 38, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 38, Transaction ID = 6 (Sounding sequence
marker position). Byte 50
    hr1. output 0x17, Trand 0x1770; range 75, Step Counter = 38, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 13, 90. CSStepCount 38, sounding length 96
        CS DRBG for 1 bits, Step Counter = 38, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 38, Transaction ID = 7 (Sounding sequence
```

*Sample Data*

```
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 38, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 38, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 38, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 38, sounding length 96, values 1, 0
        CS DRBG for 8 bits, Step Counter = 38, Transaction ID = 4 (Antenna
permutation). Byte 55
    hr1. output 0x07, Trand 0x07F8; range 24, Step Counter = 38, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 38
        CS DRBG for 2 bits, Step Counter = 38, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 38
************************************************************
CS DRBG for 128 bits, Step Counter = 39, Transaction ID = 5 (AA generation).
New DRGB octets: values A1 C6 20 DF C0 71 16 E6 3C 2A 25 4B 20 28 36 EB
************************************************************
  AA generation, CSStepCount 39. AA_I 0xC07116E6, AA_R 0x202836EB; s0 0xA1C620DF
(score 12), s1 0xC07116E6 (score 12), s2 0x3C2A254B (score 18), s3 0x202836EB (score
14)
        CS DRBG for 8 bits, Step Counter = 39, Transaction ID = 6 (Sounding sequence
marker position). Byte E1
    hr1. output 0x38, Trand 0x3840; range 64, Step Counter = 39, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 39, Transaction ID = 6 (Sounding sequence
marker position). Byte C2
    hr1. output 0x38, Trand 0x38D6; range 75, Step Counter = 39, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 56. CSStepCount 39, sounding length 96
        CS DRBG for 8 bits, Step Counter = 39, Transaction ID = 6 (Sounding sequence
marker position). Byte 9A
    hr1. output 0x26, Trand 0x2680; range 64, Step Counter = 39, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 39, Transaction ID = 6 (Sounding sequence
marker position). Byte E2
    hr1. output 0x42, Trand 0x4236; range 75, Step Counter = 39, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 38. CSStepCount 39, sounding length 96
        CS DRBG for 1 bits, Step Counter = 39, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 39, sounding length 96, value 1
```

*Sample Data*

```
        CS DRBG for 1 bits, Step Counter = 39, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 39, sounding length 96, value 0
************************************************************
CS DRBG for 8 bits, Step Counter = 39, Transaction ID = 4 (Antenna permutation).
New DRBG octets: 9F CF 53 85 F3 79 51 51 46 E1 45 5E 9F 02 B4 25; 8-bit value 9F
************************************************************
    hr1. output 0x0E, Trand 0x0EE8; range 24, Step Counter = 39, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 14. N_AP = 4, CSStepCount 39
        CS DRBG for 2 bits, Step Counter = 39, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 39
        CS DRBG for 2 bits, Step Counter = 40, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 40
        CS DRBG for 8 bits, Step Counter = 40, Transaction ID = 4 (Antenna
permutation). Byte CF
    hr1. output 0x13, Trand 0x1368; range 24, Step Counter = 40, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 19. N_AP = 4, CSStepCount 40
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 2 (Subevent submode).
Byte 64
    hr1. output 0x02, Trand 0x0258; range 6, Step Counter = 41, Transaction ID = 2
(Subevent submode)
  Submode insertion: 3 Main-Mode steps. CSStepCount 41, Main_Mode_Min 1,
Main_Mode_Max 6
************************************************************
CS DRBG for 128 bits, Step Counter = 41, Transaction ID = 5 (AA generation).
New DRGB octets: values F9 75 C7 DA 8C C1 CB C7 5E A9 A3 F9 C2 EF B6 90
************************************************************
  AA generation, CSStepCount 41. AA_I 0xF975C7DA, AA_R 0xC2EFB690; s0 0xF975C7DA
(score 8), s1 0x8CC1CBC7 (score 26), s2 0x5EA9A3F9 (score 12), s3 0xC2EFB690 (score
12)
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 6 (Sounding sequence
marker position). Byte 50
    hr1. output 0x14, Trand 0x1400; range 64, Step Counter = 41, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 6 (Sounding sequence
marker position). Byte F6
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 6 (Sounding sequence
marker position). Byte 23
    hr1 using two bytes. output 0A, Trand1 4812; range 75, counters: Step 41,
```

*Sample Data*

```
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS positions 20, 77. CSStepCount 41, sounding length 96
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 6 (Sounding sequence
marker position). Byte EA
    hr1. output 0x3A, Trand 0x3A80; range 64, Step Counter = 41, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 6 (Sounding sequence
marker position). Byte 33
    hr1. output 0x0E, Trand 0x0EF1; range 75, Step Counter = 41, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 58, 81. CSStepCount 41, sounding length 96
        CS DRBG for 1 bits, Step Counter = 41, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 41, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 41, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 41, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 41, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 41, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 41, Transaction ID = 4 (Antenna
permutation). Byte 53
    hr1. output 0x07, Trand 0x07C8; range 24, Step Counter = 41, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 41
        CS DRBG for 2 bits, Step Counter = 41, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
  Tone extensions: I 1, R 1. CSStepCount 41
**************************************************************
CS DRBG for 128 bits, Step Counter = 42, Transaction ID = 5 (AA generation).
New DRGB octets: values 94 27 94 24 2B 7D 51 F3 3E 76 6F CB 96 52 E3 61
**************************************************************
  AA generation, CSStepCount 42. AA_I 0x94279424, AA_R 0x9652E361; s0 0x94279424
(score 8), s1 0x2B7D51F3 (score 14), s2 0x3E766FCB (score 14), s3 0x9652E361 (score
12)
**************************************************************
CS DRBG for 128 bits, Step Counter = 43, Transaction ID = 5 (AA generation).
New DRGB octets: values 51 17 E6 D9 19 D2 9F 36 65 42 19 F8 BC 7E 01 7A
**************************************************************
  AA generation, CSStepCount 43. AA_I 0x19D29F36, AA_R 0x654219F8; s0 0x5117E6D9
(score 10), s1 0x19D29F36 (score 10), s2 0x654219F8 (score 6), s3 0xBC7E017A (score
20)
```

*Sample Data*

```
        CS DRBG for 8 bits, Step Counter = 43, Transaction ID = 6 (Sounding sequence
marker position). Byte 27
    hr1. output 0x09, Trand 0x09C0; range 64, Step Counter = 43, Transaction ID = 6
(Sounding sequence marker position)
************************************************************
CS DRBG for 8 bits, Step Counter = 43, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: D1 93 AD A2 CE F3 2C 5F F0 29 1E 13 DB F5 9D 1D; 8-bit value D1
************************************************************
    hr1. output 0x3D, Trand 0x3D3B; range 75, Step Counter = 43, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 9. CSStepCount 43, sounding length 96
        CS DRBG for 8 bits, Step Counter = 43, Transaction ID = 6 (Sounding sequence
marker position). Byte 93
    hr1. output 0x24, Trand 0x24C0; range 64, Step Counter = 43, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 43, Transaction ID = 6 (Sounding sequence
marker position). Byte AD
    hr1. output 0x32, Trand 0x32AF; range 75, Step Counter = 43, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 36. CSStepCount 43, sounding length 96
        CS DRBG for 1 bits, Step Counter = 43, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 43, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 43, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 43, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 43, Transaction ID = 4 (Antenna
permutation). Byte 85
    hr1. output 0x0C, Trand 0x0C78; range 24, Step Counter = 43, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 12. N_AP = 4, CSStepCount 43
        CS DRBG for 2 bits, Step Counter = 43, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 43
************************************************************
CS DRBG for 128 bits, Step Counter = 44, Transaction ID = 5 (AA generation).
New DRGB octets: values FA 3B 44 CE AC 66 77 B5 98 20 19 66 25 B5 E5 88
************************************************************
  AA generation, CSStepCount 44. AA_I 0xFA3B44CE, AA_R 0x25B5E588; s0 0xFA3B44CE
(score 6), s1 0xAC6677B5 (score 14), s2 0x98201966 (score 18), s3 0x25B5E588 (score
14)
        CS DRBG for 8 bits, Step Counter = 44, Transaction ID = 6 (Sounding sequence
```

*Sample Data*

```
marker position). Byte A2
    hr1. output 0x28, Trand 0x2880; range 64, Step Counter = 44, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 44, Transaction ID = 6 (Sounding sequence
marker position). Byte CE
    hr1. output 0x3C, Trand 0x3C5A; range 75, Step Counter = 44, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 40. CSStepCount 44, sounding length 96
        CS DRBG for 8 bits, Step Counter = 44, Transaction ID = 6 (Sounding sequence
marker position). Byte F3
    hr1. output 0x3C, Trand 0x3CC0; range 64, Step Counter = 44, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 44, Transaction ID = 6 (Sounding sequence
marker position). Byte 2C
    hr1. output 0x0C, Trand 0x0CE4; range 75, Step Counter = 44, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 60, 79. CSStepCount 44, sounding length 96
        CS DRBG for 1 bits, Step Counter = 44, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 44, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 44, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
        CS DRBG for 1 bits, Step Counter = 44, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 44, sounding length 96, values 1, 0
        CS DRBG for 8 bits, Step Counter = 44, Transaction ID = 4 (Antenna
permutation). Byte F3
    hr1. output 0x16, Trand 0x16C8; range 24, Step Counter = 44, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 22. N_AP = 4, CSStepCount 44
        CS DRBG for 2 bits, Step Counter = 44, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 44
************************************************************
CS DRBG for 128 bits, Step Counter = 45, Transaction ID = 5 (AA generation).
New DRGB octets: values 3A FC C9 66 2A 84 09 59 B4 39 B7 28 A5 56 83 81
************************************************************
  AA generation, CSStepCount 45. AA_I 0x3AFCC966, AA_R 0xB439B728; s0 0x3AFCC966
(score 12), s1 0x2A840959 (score 16), s2 0xB439B728 (score 8), s3 0xA5568381 (score
20)
        CS DRBG for 8 bits, Step Counter = 45, Transaction ID = 6 (Sounding sequence
marker position). Byte 5F
    hr1. output 0x17, Trand 0x17C0; range 64, Step Counter = 45, Transaction ID = 6
```

*Sample Data*

```
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 45, Transaction ID = 6 (Sounding sequence
marker position). Byte F0
    hr1. output 0x46, Trand 0x4650; range 75, Step Counter = 45, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 23. CSStepCount 45, sounding length 96
        CS DRBG for 8 bits, Step Counter = 45, Transaction ID = 6 (Sounding sequence
marker position). Byte 29
    hr1. output 0x0A, Trand 0x0A40; range 64, Step Counter = 45, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 45, Transaction ID = 6 (Sounding sequence
marker position). Byte 1E
    hr1. output 0x08, Trand 0x08CA; range 75, Step Counter = 45, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 10, 75. CSStepCount 45, sounding length 96
        CS DRBG for 1 bits, Step Counter = 45, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 45, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 45, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 45, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 45, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 45, Transaction ID = 4 (Antenna
permutation). Byte 79
    hr1. output 0x0B, Trand 0x0B58; range 24, Step Counter = 45, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 11. N_AP = 4, CSStepCount 45
        CS DRBG for 2 bits, Step Counter = 45, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 45
************************************************************
CS DRBG for 128 bits, Step Counter = 46, Transaction ID = 5 (AA generation).
New DRGB octets: values 0B 4D 44 7F BF CD 8A 16 0D 8C 37 94 7D 89 D2 44
************************************************************
  AA generation, CSStepCount 46. AA_I 0xBFCD8A16, AA_R 0x0D8C3794; s0 0x0B4D447F
(score 10), s1 0xBFCD8A16 (score 8), s2 0x0D8C3794 (score 10), s3 0x7D89D244 (score
12)
        CS DRBG for 8 bits, Step Counter = 46, Transaction ID = 6 (Sounding sequence
marker position). Byte 13
    hr1. output 0x04, Trand 0x04C0; range 64, Step Counter = 46, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 46, Transaction ID = 6 (Sounding sequence
```

*Sample Data*

```
marker position). Byte DB
    hr1. output 0x40, Trand 0x4029; range 75, Step Counter = 46, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 4. CSStepCount 46, sounding length 96
        CS DRBG for 8 bits, Step Counter = 46, Transaction ID = 6 (Sounding sequence
marker position). Byte F5
    hr1. output 0x3D, Trand 0x3D40; range 64, Step Counter = 46, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 46, Transaction ID = 6 (Sounding sequence
marker position). Byte 9D
    hr1. output 0x2D, Trand 0x2DFF; range 75, Step Counter = 46, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 61. CSStepCount 46, sounding length 96
        CS DRBG for 1 bits, Step Counter = 46, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 46, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 46, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 46, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 46, Transaction ID = 4 (Antenna
permutation). Byte 51
    hr1. output 0x07, Trand 0x0798; range 24, Step Counter = 46, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 46
        CS DRBG for 2 bits, Step Counter = 46, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 46
        CS DRBG for 2 bits, Step Counter = 47, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 47
        CS DRBG for 8 bits, Step Counter = 47, Transaction ID = 4 (Antenna
permutation). Byte 51
    hr1. output 0x07, Trand 0x0798; range 24, Step Counter = 47, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 7. N_AP = 4, CSStepCount 47
************************************************************
CS DRBG for 128 bits, Step Counter = 48, Transaction ID = 5 (AA generation).
New DRGB octets: values 87 D7 D1 62 B8 30 DB BE 77 34 24 24 C7 AE 18 FE
************************************************************
  AA generation, CSStepCount 48. AA_I 0xB830DBBE, AA_R 0x77342424; s0 0x87D7D162
(score 10), s1 0xB830DBBE (score 8), s2 0x77342424 (score 12), s3 0xC7AE18FE (score
20)
************************************************************
```

*Sample Data*

```
CS DRBG for 128 bits, Step Counter = 49, Transaction ID = 5 (AA generation).
New DRGB octets: values 07 4A 4D A6 4A A7 4A D3 F5 E6 FF 5B 49 A7 13 B8
************************************************************
  AA generation, CSStepCount 49. AA_I 0x074A4DA6, AA_R 0x49A713B8; s0 0x074A4DA6
(score 18), s1 0x4AA74AD3 (score 20), s2 0xF5E6FF5B (score 16), s3 0x49A713B8 (score
10)
        CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 6 (Sounding sequence
marker position). Byte 1D
    hr1. output 0x07, Trand 0x0740; range 64, Step Counter = 49, Transaction ID = 6
(Sounding sequence marker position)
************************************************************
CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 8C E6 93 A8 AC 5B 0D AF F3 FD 50 83 7A 7A 8D BA; 8-bit value 8C
************************************************************
        CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 6 (Sounding sequence
marker position). Byte E6
    hr1 using two bytes. output 43, Trand1 2904; range 75, counters: Step 49,
Transaction ID = 6 (Sounding sequence marker position)
  Marker bits in SS position 7. CSStepCount 49, sounding length 96
        CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 6 (Sounding sequence
marker position). Byte 93
    hr1. output 0x24, Trand 0x24C0; range 64, Step Counter = 49, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 6 (Sounding sequence
marker position). Byte A8
    hr1. output 0x31, Trand 0x3138; range 75, Step Counter = 49, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 36. CSStepCount 49, sounding length 96
        CS DRBG for 1 bits, Step Counter = 49, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 49, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 49, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 49, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 49, Transaction ID = 4 (Antenna
permutation). Byte 46
    hr1. output 0x06, Trand 0x0690; range 24, Step Counter = 49, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 6. N_AP = 4, CSStepCount 49
        CS DRBG for 2 bits, Step Counter = 49, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
  Tone extensions: I 0, R 0. CSStepCount 49
```

*Sample Data*

```
************************************************************
CS DRBG for 128 bits, Step Counter = 50, Transaction ID = 5 (AA generation).
New DRGB octets: values BD 48 17 2D B3 19 C9 46 CC C0 16 1A EC 49 67 F8
************************************************************
  AA generation, CSStepCount 50. AA_I 0xBD48172D, AA_R 0xCCC0161A; s0 0xBD48172D
(score 16), s1 0xB319C946 (score 20), s2 0xCCC0161A (score 12), s3 0xEC4967F8 (score
14)
        CS DRBG for 8 bits, Step Counter = 50, Transaction ID = 6 (Sounding sequence
marker position). Byte AC
    hr1. output 0x2B, Trand 0x2B00; range 64, Step Counter = 50, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 50, Transaction ID = 6 (Sounding sequence
marker position). Byte 5B
    hr1. output 0x1A, Trand 0x1AA9; range 75, Step Counter = 50, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 43. CSStepCount 50, sounding length 96
        CS DRBG for 8 bits, Step Counter = 50, Transaction ID = 6 (Sounding sequence
marker position). Byte 0D
    hr1. output 0x03, Trand 0x0340; range 64, Step Counter = 50, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 50, Transaction ID = 6 (Sounding sequence
marker position). Byte AF
    hr1. output 0x33, Trand 0x3345; range 75, Step Counter = 50, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 3. CSStepCount 50, sounding length 96
        CS DRBG for 1 bits, Step Counter = 50, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 50, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 50, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 50, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 50, Transaction ID = 4 (Antenna
permutation). Byte E1
    hr1. output 0x15, Trand 0x1518; range 24, Step Counter = 50, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 21. N_AP = 4, CSStepCount 50
        CS DRBG for 2 bits, Step Counter = 50, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 50
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 2 (Subevent submode).
Byte B4
    hr1. output 0x04, Trand 0x0438; range 6, Step Counter = 51, Transaction ID = 2
(Subevent submode)
```

*Sample Data*

```
  Submode insertion: 5 Main-Mode steps. CSStepCount 51, Main_Mode_Min 1,
Main_Mode_Max 6
************************************************************
CS DRBG for 128 bits, Step Counter = 51, Transaction ID = 5 (AA generation).
New DRGB octets: values F1 80 A9 42 86 90 B9 3A 90 F4 28 7D B9 ED FB 99
************************************************************
  AA generation, CSStepCount 51. AA_I 0x8690B93A, AA_R 0x90F4287D; s0 0xF180A942
(score 12), s1 0x8690B93A (score 6), s2 0x90F4287D (score 10), s3 0xB9EDFB99 (score
12)
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 6 (Sounding sequence
marker position). Byte F3
    hr1. output 0x3C, Trand 0x3CC0; range 64, Step Counter = 51, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 6 (Sounding sequence
marker position). Byte FD
    hr1. output 0x4A, Trand 0x4A1F; range 75, Step Counter = 51, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 60. CSStepCount 51, sounding length 96
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 6 (Sounding sequence
marker position). Byte 50
    hr1. output 0x14, Trand 0x1400; range 64, Step Counter = 51, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 6 (Sounding sequence
marker position). Byte 83
    hr1. output 0x26, Trand 0x2661; range 75, Step Counter = 51, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 20. CSStepCount 51, sounding length 96
        CS DRBG for 1 bits, Step Counter = 51, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 51, sounding length 96, value 1
        CS DRBG for 1 bits, Step Counter = 51, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 51, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 51, Transaction ID = 4 (Antenna
permutation). Byte 45
    hr1. output 0x06, Trand 0x0678; range 24, Step Counter = 51, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 6. N_AP = 4, CSStepCount 51
        CS DRBG for 2 bits, Step Counter = 51, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 51
************************************************************
CS DRBG for 128 bits, Step Counter = 52, Transaction ID = 5 (AA generation).
```

*Sample Data*

```
New DRGB octets: values 29 3A 71 72 62 7E C2 74 4A 63 9D 3C 78 9D 64 25
************************************************************
  AA generation, CSStepCount 52. AA_I 0x627EC274, AA_R 0x789D6425; s0 0x293A7172
(score 12), s1 0x627EC274 (score 10), s2 0x4A639D3C (score 16), s3 0x789D6425 (score
6)
        CS DRBG for 8 bits, Step Counter = 52, Transaction ID = 6 (Sounding sequence
marker position). Byte 7A
    hr1. output 0x1E, Trand 0x1E80; range 64, Step Counter = 52, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 52, Transaction ID = 6 (Sounding sequence
marker position). Byte 7A
    hr1. output 0x23, Trand 0x23BE; range 75, Step Counter = 52, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 30. CSStepCount 52, sounding length 96
        CS DRBG for 8 bits, Step Counter = 52, Transaction ID = 6 (Sounding sequence
marker position). Byte 8D
    hr1. output 0x23, Trand 0x2340; range 64, Step Counter = 52, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 52, Transaction ID = 6 (Sounding sequence
marker position). Byte BA
    hr1. output 0x36, Trand 0x367E; range 75, Step Counter = 52, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 35. CSStepCount 52, sounding length 96
        CS DRBG for 1 bits, Step Counter = 52, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 52, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 52, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 52, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 52, Transaction ID = 4 (Antenna
permutation). Byte 5E
    hr1. output 0x08, Trand 0x08D0; range 24, Step Counter = 52, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 8. N_AP = 4, CSStepCount 52
        CS DRBG for 2 bits, Step Counter = 52, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 52
************************************************************
CS DRBG for 128 bits, Step Counter = 53, Transaction ID = 5 (AA generation).
New DRGB octets: values 42 BB F0 C2 3C 3B 56 00 5B C6 47 A8 4B 07 22 15
************************************************************
  AA generation, CSStepCount 53. AA_I 0x42BBF0C2, AA_R 0x4B072215; s0 0x42BBF0C2
(score 12), s1 0x3C3B5600 (score 14), s2 0x5BC647A8 (score 6), s3 0x4B072215 (score 6)
```

*Sample Data*

```
************************************************************
CS DRBG for 8 bits, Step Counter = 53, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 48 58 B3 19 B3 28 A9 81 5D 09 D8 49 56 E1 1B A5; 8-bit value 48
************************************************************
    hr1. output 0x12, Trand 0x1200; range 64, Step Counter = 53, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 53, Transaction ID = 6 (Sounding sequence
marker position). Byte 58
    hr1. output 0x19, Trand 0x19C8; range 75, Step Counter = 53, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 18, 92. CSStepCount 53, sounding length 96
        CS DRBG for 8 bits, Step Counter = 53, Transaction ID = 6 (Sounding sequence
marker position). Byte B3
    hr1. output 0x2C, Trand 0x2CC0; range 64, Step Counter = 53, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 53, Transaction ID = 6 (Sounding sequence
marker position). Byte 19
    hr1. output 0x07, Trand 0x0753; range 75, Step Counter = 53, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 44, 74. CSStepCount 53, sounding length 96
        CS DRBG for 1 bits, Step Counter = 53, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 53, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 53, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 53, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 53, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 53, sounding length 96, values 0, 0
        CS DRBG for 8 bits, Step Counter = 53, Transaction ID = 4 (Antenna
permutation). Byte 9F
    hr1. output 0x0E, Trand 0x0EE8; range 24, Step Counter = 53, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 14. N_AP = 4, CSStepCount 53
        CS DRBG for 2 bits, Step Counter = 53, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b10
  Tone extensions: I 1, R 0. CSStepCount 53
************************************************************
CS DRBG for 128 bits, Step Counter = 54, Transaction ID = 5 (AA generation).
New DRGB octets: values 03 64 D0 56 79 3D 51 8F 3B DA 61 F4 22 61 3D BE
************************************************************
```

*Sample Data*

```
  AA generation, CSStepCount 54. AA_I 0x793D518F, AA_R 0x3BDA61F4; s0 0x0364D056
(score 10), s1 0x793D518F (score 10), s2 0x3BDA61F4 (score 6), s3 0x22613DBE (score
14)
**************************************************************
CS DRBG for 128 bits, Step Counter = 55, Transaction ID = 5 (AA generation).
New DRGB octets: values 05 72 35 B1 00 2C 96 79 32 F5 3C E3 92 36 F9 46
**************************************************************
  AA generation, CSStepCount 55. AA_I 0x057235B1, AA_R 0x9236F946; s0 0x057235B1
(score 4), s1 0x002C9679 (score 16), s2 0x32F53CE3 (score 16), s3 0x9236F946 (score
16)
        CS DRBG for 8 bits, Step Counter = 55, Transaction ID = 6 (Sounding sequence
marker position). Byte B3
    hr1. output 0x2C, Trand 0x2CC0; range 64, Step Counter = 55, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 55, Transaction ID = 6 (Sounding sequence
marker position). Byte 28
    hr1. output 0x0B, Trand 0x0BB8; range 75, Step Counter = 55, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 44, 78. CSStepCount 55, sounding length 96
        CS DRBG for 8 bits, Step Counter = 55, Transaction ID = 6 (Sounding sequence
marker position). Byte A9
    hr1. output 0x2A, Trand 0x2A40; range 64, Step Counter = 55, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 55, Transaction ID = 6 (Sounding sequence
marker position). Byte 81
    hr1. output 0x25, Trand 0x25CB; range 75, Step Counter = 55, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 42. CSStepCount 55, sounding length 96
        CS DRBG for 1 bits, Step Counter = 55, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 55, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 55, sounding length 96, values 0, 1
        CS DRBG for 1 bits, Step Counter = 55, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 55, sounding length 96, value 1
        CS DRBG for 8 bits, Step Counter = 55, Transaction ID = 4 (Antenna
permutation). Byte 02
    hr1. output 0x00, Trand 0x0030; range 24, Step Counter = 55, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 0. N_AP = 4, CSStepCount 55
        CS DRBG for 2 bits, Step Counter = 55, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
```

*Sample Data*

```
  Tone extensions: I 1, R 1. CSStepCount 55
************************************************************
CS DRBG for 128 bits, Step Counter = 56, Transaction ID = 5 (AA generation).
New DRGB octets: values 54 4F A4 E0 63 61 16 58 CF 83 E6 6E FE 08 D4 7C
************************************************************
  AA generation, CSStepCount 56. AA_I 0x544FA4E0, AA_R 0xFE08D47C; s0 0x544FA4E0
(score 6), s1 0x63611658 (score 10), s2 0xCF83E66E (score 20), s3 0xFE08D47C (score
18)
        CS DRBG for 8 bits, Step Counter = 56, Transaction ID = 6 (Sounding sequence
marker position). Byte 5D
    hr1. output 0x17, Trand 0x1740; range 64, Step Counter = 56, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 56, Transaction ID = 6 (Sounding sequence
marker position). Byte 09
    hr1. output 0x02, Trand 0x02A3; range 75, Step Counter = 56, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 23, 69. CSStepCount 56, sounding length 96
        CS DRBG for 8 bits, Step Counter = 56, Transaction ID = 6 (Sounding sequence
marker position). Byte D8
    hr1. output 0x36, Trand 0x3600; range 64, Step Counter = 56, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 56, Transaction ID = 6 (Sounding sequence
marker position). Byte 49
    hr1. output 0x15, Trand 0x1563; range 75, Step Counter = 56, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 54, 88. CSStepCount 56, sounding length 96
        CS DRBG for 1 bits, Step Counter = 56, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 56, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 56, sounding length 96, values 0, 0
        CS DRBG for 1 bits, Step Counter = 56, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 56, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b1
  Marker bits in SS values. CSStepCount 56, sounding length 96, values 0, 1
        CS DRBG for 8 bits, Step Counter = 56, Transaction ID = 4 (Antenna
permutation). Byte B4
    hr1. output 0x10, Trand 0x10E0; range 24, Step Counter = 56, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 16. N_AP = 4, CSStepCount 56
        CS DRBG for 2 bits, Step Counter = 56, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b11
```

*Sample Data*

```
  Tone extensions: I 1, R 1. CSStepCount 56
************************************************************
CS DRBG for 128 bits, Step Counter = 57, Transaction ID = 5 (AA generation).
New DRGB octets: values D9 15 06 34 31 B8 17 94 91 F1 D9 E2 D3 6E C1 D3
************************************************************
  AA generation, CSStepCount 57. AA_I 0xD9150634, AA_R 0xD36EC1D3; s0 0xD9150634
(score 6), s1 0x31B81794 (score 8), s2 0x91F1D9E2 (score 18), s3 0xD36EC1D3 (score 12)
        CS DRBG for 8 bits, Step Counter = 57, Transaction ID = 6 (Sounding sequence
marker position). Byte 56
    hr1. output 0x15, Trand 0x1580; range 64, Step Counter = 57, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 57, Transaction ID = 6 (Sounding sequence
marker position). Byte E1
    hr1. output 0x41, Trand 0x41EB; range 75, Step Counter = 57, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 21. CSStepCount 57, sounding length 96
        CS DRBG for 8 bits, Step Counter = 57, Transaction ID = 6 (Sounding sequence
marker position). Byte 1B
    hr1. output 0x06, Trand 0x06C0; range 64, Step Counter = 57, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 57, Transaction ID = 6 (Sounding sequence
marker position). Byte A5
    hr1. output 0x30, Trand 0x3057; range 75, Step Counter = 57, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 6. CSStepCount 57, sounding length 96
        CS DRBG for 1 bits, Step Counter = 57, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 57, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 57, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 57, sounding length 96, value 0
        CS DRBG for 8 bits, Step Counter = 57, Transaction ID = 4 (Antenna
permutation). Byte 25
    hr1. output 0x03, Trand 0x0378; range 24, Step Counter = 57, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 3. N_AP = 4, CSStepCount 57
        CS DRBG for 2 bits, Step Counter = 57, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 57
************************************************************
CS DRBG for 128 bits, Step Counter = 58, Transaction ID = 5 (AA generation).
New DRGB octets: values 7B 03 0B B4 14 30 62 18 59 1E 03 15 A1 5F 89 A6
************************************************************
```

*Sample Data*

```
  AA generation, CSStepCount 58. AA_I 0x7B030BB4, AA_R 0xA15F89A6; s0 0x7B030BB4
(score 6), s1 0x14306218 (score 16), s2 0x591E0315 (score 8), s3 0xA15F89A6 (score 6)
************************************************************
CS DRBG for 8 bits, Step Counter = 58, Transaction ID = 6 (Sounding sequence marker
position).
New DRBG octets: 09 F5 79 13 BD 3A 01 E8 76 3F 42 6E 2D 48 0C 0C; 8-bit value 09
************************************************************
    hr1. output 0x02, Trand 0x0240; range 64, Step Counter = 58, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 58, Transaction ID = 6 (Sounding sequence
marker position). Byte F5
    hr1. output 0x47, Trand 0x47C7; range 75, Step Counter = 58, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS position 2. CSStepCount 58, sounding length 96
        CS DRBG for 8 bits, Step Counter = 58, Transaction ID = 6 (Sounding sequence
marker position). Byte 79
    hr1. output 0x1E, Trand 0x1E40; range 64, Step Counter = 58, Transaction ID = 6
(Sounding sequence marker position)
        CS DRBG for 8 bits, Step Counter = 58, Transaction ID = 6 (Sounding sequence
marker position). Byte 13
    hr1. output 0x05, Trand 0x0591; range 75, Step Counter = 58, Transaction ID = 6
(Sounding sequence marker position)
  Marker bits in SS positions 30, 72. CSStepCount 58, sounding length 96
        CS DRBG for 1 bits, Step Counter = 58, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 58, sounding length 96, value 0
        CS DRBG for 1 bits, Step Counter = 58, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
        CS DRBG for 1 bits, Step Counter = 58, Transaction ID = 7 (Sounding sequence
marker signal). Value 1 bit: 0b0
  Marker bits in SS values. CSStepCount 58, sounding length 96, values 0, 0
************************************************************
CS DRBG for 8 bits, Step Counter = 58, Transaction ID = 4 (Antenna permutation).
New DRBG octets: FC EC D6 F2 B5 25 45 6F 6F B1 67 0D 45 3A B6 66; 8-bit value FC
************************************************************
    hr1. output 0x17, Trand 0x17A0; range 24, Step Counter = 58, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 23. N_AP = 4, CSStepCount 58
        CS DRBG for 2 bits, Step Counter = 58, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b01
  Tone extensions: I 0, R 1. CSStepCount 58
        CS DRBG for 2 bits, Step Counter = 59, Transaction ID = 3 (Tone extension
presence). Value 2 bits: 0b00
```

*Sample Data*

```
  Tone extensions: I 0, R 0. CSStepCount 59
        CS DRBG for 8 bits, Step Counter = 59, Transaction ID = 4 (Antenna
permutation). Byte EC
    hr1. output 0x16, Trand 0x1620; range 24, Step Counter = 59, Transaction ID = 4
(Antenna permutation)
  Antenna Path Permutation Index: 22. N_AP = 4, CSStepCount 59

************************************************************
************************************************************
  Resulting Procedure
    Expected CS Procedure: Mode-3 & Mode-2; Ch Sel #3b; 96-bit sounding sequence
      Summary
        Event 0, 0 ns
                  SubEvent 0, 0 ns-3.121 ms, steps 0-5 = 1 Mode-2, 4 Mode-3: 0, 3, 2,
3, 3, 3
                SubEvent 1, 5.000 ms-8.121 ms, steps 6-11 = 1 Mode-2, 4 Mode-3: 0, 3,
3, 3, 2, 3
              SubEvent 2, 10.000 ms-13.289 ms, steps 12-17 = 5 Mode-3: 0, 3, 3, 3, 3,
3

        Event 1, 40.000 ms
             SubEvent 3, 40.000 ms-43.121 ms, steps 18-23 = 1 Mode-2, 4 Mode-3: 0, 3,
3, 3, 2, 3
              SubEvent 4, 45.000 ms-48.289 ms, steps 24-29 = 1 Mode-2, 4 Mode-3: 0, 3,
3, 3, 3, 2
              SubEvent 5, 50.000 ms-53.289 ms, steps 30-35 = 5 Mode-3: 0, 3, 3, 3, 3,
3

        Event 2, 80.000 ms
               SubEvent 6, 80.000 ms-83.121 ms, steps 36-41 = 1 Mode-2, 4 Mode-3: 0,
3, 3, 3, 2, 3
               SubEvent 7, 85.000 ms-88.289 ms, steps 42-47 = 1 Mode-2, 4 Mode-3: 0,
3, 3, 3, 3, 2
               SubEvent 8, 90.000 ms-93.289 ms, steps 48-53 = 5 Mode-3: 0, 3, 3, 3, 3,
3

        Event 3, 120.000 ms
               SubEvent 9, 120.000 ms-123.289 ms, steps 54-59 = 1 Mode-2, 4 Mode-3: 0,
3, 3, 3, 3, 2

************************************************************
  Detail
    Event 0, 0 ns
```

*Sample Data*

```
      SubEvent 0, 0 ns-3.121 ms, steps 0-5
        Step # 0: Mode-0 step on ch 11, +150.00 µs after subevent anchor
                        Initiator Sync AA only, AA 0x6C376AB8 = No payload, Access
Address only
              Reflector Sync AA only, AA 0xF079BC3A, 176 ns after step start = No
payload, Access Address only
                    Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step # 1: Mode-3 step on ch 20, +597.00 µs after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x011CAE4E = Marker 0011
(DRBG bit 1) at bits 11-14 (1:3). Bits   0: {01010101}    1: {01000111}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 22
                        Reflector 80 µs tone, 334 ns after step start = APPI 22
              Reflector 96-bit sounding sequence, AA 0xD06ACDDA, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 13-16 (1:5). Bits   0: {01010101}    1:
{01010110}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step # 2: Mode-2 step on ch 19, +1.270 ms after subevent anchor
                                        Initiator 80 µs tone =  APPI 7
                          Reflector 100 µs tone, 250 ns after step start = APPI 7,
tone extension present

        Step # 3: Mode-3 step on ch 11, +1.775 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x3F1C256E = Marker 0011
(DRBG bit 1) at bits 54-57 (6:6); Marker 1100 (DRBG bit 0) at bits 73-76 (9:1).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010100}    7: {11010101}    8: {01010101}    9:
{01100101}   10: {01010101}   11: {01010101}
                        Initiator 100 µs tone, 84 ns after step start = APPI 2, tone
extension present
                        Reflector 100 µs tone, 334 ns after step start = APPI 2, tone
extension present
              Reflector 96-bit sounding sequence, AA 0x4E842FE9, 444 ns after step
start = Marker 0011 (DRBG bit 1) at bits 14-17 (1:6); Marker 0011 (DRBG bit 1) at
bits 85-88 (10:5). Bits   0: {01010101}    1: {01010100}    2: {11010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {01010101}   10: {01010001}   11: {11010101}

        Step # 4: Mode-3 step on ch 17, +2.448 ms after subevent anchor
```

*Sample Data*

```
                    Initiator 96-bit sounding sequence, AA 0x90BE37AB = Marker 1100
(DRBG bit 0) at bits 18-21 (2:2); Marker 1100 (DRBG bit 0) at bits 85-88 (10:5).
Bits   0: {01010101}    1: {01010101}    2: {01110001}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010101}   10: {01010110}   11: {01010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 14, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 14
            Reflector 96-bit sounding sequence, AA 0xAF2EF46B, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 35-38 (4:3); Marker 0011 (DRBG bit 1) at
bits 91-94 (11:3). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01000111}    5: {01010101}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {01010101}   10: {01010101}   11: {01000111}

        Step # 5: Mode-3 step on ch 4, +3.121 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x262F2907 = Marker 1100
(DRBG bit 0) at bits 3-6 (0:3); Marker 1100 (DRBG bit 0) at bits 86-89 (10:6). Bits
0: {01011001}    1: {01010101}    2: {01010101}    3: {01010101}    4: {01010101}
5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9: {01010101}
10: {01010111}   11: {00010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 9, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 9
            Reflector 96-bit sounding sequence, AA 0x94E46E08, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 7-10 (0:7). Bits   0: {01010101}    1:
{10010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

************************************************************
  SubEvent 1, 5.000 ms-8.121 ms, steps 6-11
       Step # 6: Mode-0 step on ch 7, +150.00 µs after subevent anchor
                           Initiator Sync AA only, AA 0xF3C92E6B = No payload,
Access Address only
                  Reflector Sync AA only, AA 0x519D0448, 176 ns after step start = No
payload, Access Address only
                          Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step # 7: Main-Mode Repetition, Mode-3 step on ch 17, +597.00 µs after
subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x4F0E6B3B = Marker
1100 (DRBG bit 0) at bits 53-56 (6:5). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010110}    7:
```

*Sample Data*

{01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}
Initiator 80 µs tone, 84 ns after step start = APPI 12
Reflector 100 µs tone, 334 ns after step start = APPI 12, tone extension present
Reflector 96-bit sounding sequence, AA 0xA088EBD9, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 44-47 (5:4). Bits 0: {01010101}  1: {01010101}  2: {01010101}  3: {01010101}  4: {01010101}  5: {01010011}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}

Step # 8: Main-Mode Repetition, Mode-3 step on ch 4, +1.270 ms after subevent anchor
Initiator 96-bit sounding sequence, AA 0xBD91D212 = Marker 0011 (DRBG bit 1) at bits 31-34 (3:7). Bits 0: {01010101}  1: {01010101}  2: {01010101}  3: {01010100}  4: {01110101}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}
Initiator 80 µs tone, 84 ns after step start = APPI 20
Reflector 100 µs tone, 334 ns after step start = APPI 20, tone extension present
Reflector 96-bit sounding sequence, AA 0xBB8DAEFA, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 24-27 (3:0). Bits 0: {01010101}  1: {01010101}  2: {01010101}  3: {00110101}  4: {01010101}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}

Step # 9: Mode-3 step on ch 8, +1.943 ms after subevent anchor
Initiator 96-bit sounding sequence, AA 0x169EEDDC = Marker 1100 (DRBG bit 0) at bits 26-29 (3:2). Bits 0: {01010101}  1: {01010101}  2: {01010101}  3: {01110001}  4: {01010101}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}
Initiator 80 µs tone, 84 ns after step start = APPI 9
Reflector 80 µs tone, 334 ns after step start = APPI 9
Reflector 96-bit sounding sequence, AA 0x6B2EB748, 424 ns after step start = Marker 0011 (DRBG bit 1) at bits 25-28 (3:1). Bits 0: {01010101}  1: {01010101}  2: {01010101}  3: {00011101}  4: {01010101}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}

Step #10: Mode-2 step on ch 7, +2.616 ms after subevent anchor
Initiator 100 µs tone = APPI 18, tone extension present
Reflector 100 µs tone, 250 ns after step start = APPI 18, tone extension present

*Sample Data*

```
        Step #11: Mode-3 step on ch 2, +3.121 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xCA69755F = Marker 0011
(DRBG bit 1) at bits 3-6 (0:3). Bits   0: {01000111}    1: {01010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 21
                        Reflector 80 µs tone, 334 ns after step start = APPI 21
                Reflector 96-bit sounding sequence, AA 0x70C82679, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 61-64 (7:5); Marker 1100 (DRBG bit 0) at
bits 75-78 (9:3). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01010110}    8:
{01010101}    9: {01011001}   10: {01010101}   11: {01010101}

************************************************************
  SubEvent 2, 10.000 ms-13.289 ms, steps 12-17
        Step #12: Mode-0 step on ch 17, +150.00 µs after subevent anchor
                                Initiator Sync AA only, AA 0x38B50481 = No payload,
Access Address only
                    Reflector Sync AA only, AA 0x8ACDD046, 176 ns after step start = No
payload, Access Address only
                               Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #13: Main-Mode Repetition, Mode-3 step on ch 8, +597.00 µs after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xD9AE74FF = Marker 0011
(DRBG bit 1) at bits 21-24 (2:5). Bits   0: {01010101}    1: {01010101}    2:
{01010001}    3: {11010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 2
                        Reflector 80 µs tone, 334 ns after step start = APPI 2
                   Reflector 96-bit sounding sequence, AA 0xB7A76D63, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 25-28 (3:1). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {00011101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #14: Main-Mode Repetition, Mode-3 step on ch 2, +1.270 ms after subevent
anchor
                        Initiator 96-bit sounding sequence, AA 0xF7219786 = Marker 1100
(DRBG bit 0) at bits 14-17 (1:6); Marker 0011 (DRBG bit 1) at bits 88-91 (11:0).
Bits   0: {01010101}    1: {01010111}    2: {00010101}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
```

*Sample Data*

{01010101}  10: {01010101}  11: {00110101}

Initiator 80 µs tone, 84 ns after step start =  APPI 8

Reflector 80 µs tone, 334 ns after step start =  APPI 8

Reflector 96-bit sounding sequence, AA 0x57176470, 424 ns after step start = Marker 1100 (DRBG bit 0) at bits 37-40 (4:5); Marker 1100 (DRBG bit 0) at bits 88-91 (11:0). Bits  0: {01010101}  1: {01010101}  2: {01010101}  3: {01010101}  4: {01010110}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {11000101}

Step #15: Mode-3 step on ch 12, +1.943 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0xD58F347D = Marker 1100 (DRBG bit 0) at bits 0-3 (0:0); Marker 0011 (DRBG bit 1) at bits 91-94 (11:3). Bits 0: {11000101}  1: {01010101}  2: {01010101}  3: {01010101}  4: {01010101} 5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101} 10: {01010101}  11: {01000111}

Initiator 100 µs tone, 84 ns after step start = APPI 16, tone extension present

Reflector 100 µs tone, 334 ns after step start = APPI 16, tone extension present

Reflector 96-bit sounding sequence, AA 0xD3727F5D, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 57-60 (7:1). Bits  0: {01010101}  1: {01010101}  2: {01010101}  3: {01010101}  4: {01010101}  5: {01010101}  6: {01010101}  7: {00011101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}

Step #16: Mode-3 step on ch 9, +2.616 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0x8A2CF27C = Marker 0011 (DRBG bit 1) at bits 48-51 (6:0); Marker 0011 (DRBG bit 1) at bits 77-80 (9:5). Bits  0: {01010101}  1: {01010101}  2: {01010101}  3: {01010101}  4: {01010101}  5: {01010101}  6: {00110101}  7: {01010101}  8: {01010101}  9: {01010001}  10: {11010101}  11: {01010101}

Initiator 80 µs tone, 84 ns after step start = APPI 22

Reflector 100 µs tone, 334 ns after step start = APPI 22, tone extension present

Reflector 96-bit sounding sequence, AA 0x0E49E347, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 12-15 (1:4). Bits  0: {01010101}  1: {01010011}  2: {01010101}  3: {01010101}  4: {01010101}  5: {01010101}  6: {01010101}  7: {01010101}  8: {01010101}  9: {01010101}  10: {01010101}  11: {01010101}

Step #17: Mode-3 step on ch 6, +3.289 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0x174662CD = Marker 1100 (DRBG bit 0) at bits 22-25 (2:6). Bits  0: {01010101}  1: {01010101}  2:

*Sample Data*

```
{01010111}    3: {00010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 1
                    Reflector 80 µs tone, 334 ns after step start = APPI 1
               Reflector 96-bit sounding sequence, AA 0xC8C597E5, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 60-63 (7:4); Marker 0011 (DRBG bit 1) at
bits 89-92 (11:1). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01010011}    8:
{01010101}    9: {01010101}   10: {01010101}   11: {00011101}


**************************************************************
  Event 1, 40.000 ms
    SubEvent 3, 40.000 ms-43.121 ms, steps 18-23
        Step #18: Mode-0 step on ch 20, +150.00 µs after subevent anchor
                            Initiator Sync AA only, AA 0xA73F8AF4 = No payload,
Access Address only
                  Reflector Sync AA only, AA 0x05CBB853, 176 ns after step start = No
payload, Access Address only
                           Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #19: Main-Mode Repetition, Mode-3 step on ch 9, +597.00 µs after
subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x5F1B308F = Marker 0011
(DRBG bit 1) at bits 31-34 (3:7). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010100}    4: {01110101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 15, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 15
              Reflector 96-bit sounding sequence, AA 0xE9770F47, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 63-66 (7:7). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010100}    8: {01110101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #20: Main-Mode Repetition, Mode-3 step on ch 6, +1.270 ms after subevent
anchor
                    Initiator 96-bit sounding sequence, AA 0x31E044B6 = Marker 0011
(DRBG bit 1) at bits 48-51 (6:0); Marker 1100 (DRBG bit 0) at bits 76-79 (9:4).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {00110101}    7: {01010101}    8: {01010101}    9:
{01011100}   10: {01010101}   11: {01010101}
```

*Sample Data*

```
                    Initiator 100 µs tone, 84 ns after step start = APPI 23, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 23
             Reflector 96-bit sounding sequence, AA 0xAFD673AC, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 37-40 (4:5). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010001}    5: {11010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #21: Mode-3 step on ch 3, +1.943 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0xAF9F7848 = Marker 1100
(DRBG bit 0) at bits 33-36 (4:1). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010101}    4: {01100101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 23, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 23
             Reflector 96-bit sounding sequence, AA 0x85E6604F, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 32-35 (4:0). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {00110101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #22: Mode-2 step on ch 18, +2.616 ms after subevent anchor
                                    Initiator 100 µs tone = APPI 11, tone
extension present
                       Reflector 80 µs tone, 250 ns after step start = APPI 11

        Step #23: Mode-3 step on ch 5, +3.121 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x5F57BA90 = Marker 0011
(DRBG bit 1) at bits 57-60 (7:1). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{00011101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 8
                    Reflector 100 µs tone, 334 ns after step start = APPI 8, tone
extension present
             Reflector 96-bit sounding sequence, AA 0x825F0764, 444 ns after step
start = Marker 0011 (DRBG bit 1) at bits 29-32 (3:5). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010001}    4: {11010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

************************************************************
```

*Sample Data*

```
  SubEvent 4, 45.000 ms–48.289 ms, steps 24–29
        Step #24: Mode-0 step on ch 9, +150.00 µs after subevent anchor
                                Initiator Sync AA only, AA 0xF071ECA6 = No payload,
Access Address only
                    Reflector Sync AA only, AA 0x35178610, 176 ns after step start = No
payload, Access Address only
                               Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #25: Main-Mode Repetition, Mode-3 step on ch 3, +597.00 µs after
subevent anchor
                         Initiator 96-bit sounding sequence, AA 0x724B1038 = Marker
1100 (DRBG bit 0) at bits 4-7 (0:4); Marker 1100 (DRBG bit 0) at bits 91-94 (11:3).
Bits   0: {01011100}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010101}   10: {01010101}   11: {01011001}
                         Initiator 100 µs tone, 84 ns after step start = APPI 7, tone
extension present
                         Reflector 80 µs tone, 334 ns after step start = APPI 7
              Reflector 96-bit sounding sequence, AA 0x613BC2E8, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 4-7 (0:4); Marker 1100 (DRBG bit 0) at bits
71-74 (8:7). Bits   0: {01011100}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {10010101}   10: {01010101}   11: {01010101}

        Step #26: Main-Mode Repetition, Mode-3 step on ch 5, +1.270 ms after subevent
anchor
                         Initiator 96-bit sounding sequence, AA 0xD4FDE60E = Marker 1100
(DRBG bit 0) at bits 6-9 (0:6). Bits   0: {01010111}    1: {00010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                         Initiator 80 µs tone, 84 ns after step start = APPI 21
                         Reflector 100 µs tone, 334 ns after step start = APPI 21, tone
extension present
              Reflector 96-bit sounding sequence, AA 0x58223D9F, 444 ns after step
start = Marker 0011 (DRBG bit 1) at bits 18-21 (2:2). Bits   0: {01010101}    1:
{01010101}    2: {01001101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #27: Mode-3 step on ch 10, +1.943 ms after subevent anchor
                         Initiator 96-bit sounding sequence, AA 0x3A6EF4A1 = Marker 1100
(DRBG bit 0) at bits 34-37 (4:2); Marker 0011 (DRBG bit 1) at bits 72-75 (9:0).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
```

*Sample Data*

```
{01110001}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{00110101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 8
                    Reflector 80 µs tone, 334 ns after step start = APPI 8
            Reflector 96-bit sounding sequence, AA 0xEB4B7EE0, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 16-19 (2:0). Bits   0: {01010101}    1:
{01010101}    2: {11000101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #28: Mode-3 step on ch 16, +2.616 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x2FE77B50 = Marker 0011
(DRBG bit 1) at bits 6-9 (0:6); Marker 0011 (DRBG bit 1) at bits 86-89 (10:6). Bits
0: {01010100}    1: {11010101}    2: {01010101}    3: {01010101}    4: {01010101}
5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9: {01010101}
10: {01010100}   11: {11010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 2, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 2
            Reflector 96-bit sounding sequence, AA 0xEE470DB8, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 49-52 (6:1). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{00011101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #29: Mode-2 step on ch 4, +3.289 ms after subevent anchor
                                    Initiator 100 µs tone = APPI 14, tone
extension present
                        Reflector 80 µs tone, 250 ns after step start = APPI 14

************************************************************
  SubEvent 5, 50.000 ms-53.289 ms, steps 30-35
        Step #30: Mode-0 step on ch 3, +150.00 µs after subevent anchor
                            Initiator Sync AA only, AA 0x2CF2DF02 = No payload,
Access Address only
                  Reflector Sync AA only, AA 0xB9E81806, 176 ns after step start = No
payload, Access Address only
                            Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #31: Main-Mode Repetition, Mode-3 step on ch 10, +597.00 µs after
subevent anchor
                    Initiator 96-bit sounding sequence, AA 0xC0F437CB = Marker 0011
(DRBG bit 1) at bits 20-23 (2:4); Marker 0011 (DRBG bit 1) at bits 87-90 (10:7).
```

*Sample Data*

Bits 0: {01010101} 1: {01010101} 2: {01010011} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010100} 11: {01110101}

Initiator 100 µs tone, 84 ns after step start = APPI 22, tone extension present

Reflector 100 µs tone, 334 ns after step start = APPI 22, tone extension present

Reflector 96-bit sounding sequence, AA 0x35AF5E5D, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 26-29 (3:2); Marker 0011 (DRBG bit 1) at bits 75-78 (9:3). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01001101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01000111} 10: {01010101} 11: {01010101}

Step #32: Main-Mode Repetition, Mode-3 step on ch 16, +1.270 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0x2CE0B190 = Marker 1100 (DRBG bit 0) at bits 39-42 (4:7); Marker 1100 (DRBG bit 0) at bits 78-81 (9:6). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {10010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010111} 10: {00010101} 11: {01010101}

Initiator 100 µs tone, 84 ns after step start = APPI 9, tone extension present

Reflector 80 µs tone, 334 ns after step start = APPI 9

Reflector 96-bit sounding sequence, AA 0x7AB091BF, 424 ns after step start = Marker 1100 (DRBG bit 0) at bits 12-15 (1:4). Bits 0: {01010101} 1: {01011100} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Step #33: Mode-3 step on ch 19, +1.943 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0xE98DAAD0 = Marker 1100 (DRBG bit 0) at bits 42-45 (5:2). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01110001} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Initiator 80 µs tone, 84 ns after step start = APPI 21

Reflector 100 µs tone, 334 ns after step start = APPI 21, tone extension present

Reflector 96-bit sounding sequence, AA 0xBE1FCF12, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 4-7 (0:4). Bits 0: {01010011} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

*Sample Data*

```
        Step #34: Mode-3 step on ch 7, +2.616 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x7B2588FD = Marker 1100
(DRBG bit 0) at bits 55-58 (6:7); Marker 0011 (DRBG bit 1) at bits 85-88 (10:5).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {10010101}    8: {01010101}    9:
{01010101}   10: {01010001}   11: {11010101}
                        Initiator 100 µs tone, 84 ns after step start = APPI 20, tone
extension present
                        Reflector 100 µs tone, 334 ns after step start = APPI 20, tone
extension present
                    Reflector 96-bit sounding sequence, AA 0x888B1F25, 444 ns after step
start = Marker 0011 (DRBG bit 1) at bits 7-10 (0:7). Bits   0: {01010100}    1:
{01110101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #35: Mode-3 step on ch 11, +3.289 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xFC69A8DD = Marker 0011
(DRBG bit 1) at bits 44-47 (5:4); Marker 0011 (DRBG bit 1) at bits 84-87 (10:4).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01010011}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010101}   10: {01010011}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 3
                        Reflector 80 µs tone, 334 ns after step start = APPI 3
                 Reflector 96-bit sounding sequence, AA 0xE647E2AD, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 41-44 (5:1). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01100101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

************************************************************
  Event 2, 80.000 ms
    SubEvent 6, 80.000 ms-83.121 ms, steps 36-41
        Step #36: Mode-0 step on ch 10, +150.00 µs after subevent anchor
                              Initiator Sync AA only, AA 0xC6D6F2E2 = No payload,
Access Address only
                    Reflector Sync AA only, AA 0x93418417, 176 ns after step start = No
payload, Access Address only
                             Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #37: Main-Mode Repetition, Mode-3 step on ch 7, +597.00 µs after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xA35E66E1 = Marker 0011
```

*Sample Data*

(DRBG bit 1) at bits 40-43 (5:0). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {00110101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Initiator 100 µs tone, 84 ns after step start = APPI 8, tone extension present

Reflector 80 µs tone, 334 ns after step start = APPI 8

Reflector 96-bit sounding sequence, AA 0x4110AB72, 424 ns after step start = Marker 0011 (DRBG bit 1) at bits 53-56 (6:5). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010001} 7: {11010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Step #38: Main-Mode Repetition, Mode-3 step on ch 11, +1.270 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0xF63FD70B = Marker 1100 (DRBG bit 0) at bits 48-51 (6:0); Marker 1100 (DRBG bit 0) at bits 73-76 (9:1). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {11000101} 7: {01010101} 8: {01010101} 9: {01100101} 10: {01010101} 11: {01010101}

Initiator 80 µs tone, 84 ns after step start = APPI 7

Reflector 100 µs tone, 334 ns after step start = APPI 7, tone extension present

Reflector 96-bit sounding sequence, AA 0x0FA0C69B, 444 ns after step start = Marker 0011 (DRBG bit 1) at bits 13-16 (1:5); Marker 1100 (DRBG bit 0) at bits 90-93 (11:2). Bits 0: {01010101} 1: {01010001} 2: {11010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01110001}

Step #39: Mode-3 step on ch 9, +1.943 ms after subevent anchor

Initiator 96-bit sounding sequence, AA 0xC07116E6 = Marker 0011 (DRBG bit 1) at bits 56-59 (7:0). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {00110101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Initiator 80 µs tone, 84 ns after step start = APPI 14

Reflector 80 µs tone, 334 ns after step start = APPI 14

Reflector 96-bit sounding sequence, AA 0x202836EB, 424 ns after step start = Marker 1100 (DRBG bit 0) at bits 38-41 (4:6). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010111} 5: {00010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Step #40: Mode-2 step on ch 10, +2.616 ms after subevent anchor

Initiator 80 µs tone = APPI 19

*Sample Data*

```
                        Reflector 80 µs tone, 250 ns after step start = APPI 19

        Step #41: Mode-3 step on ch 16, +3.121 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xF975C7DA = Marker 1100
(DRBG bit 0) at bits 20-23 (2:4); Marker 0011 (DRBG bit 1) at bits 77-80 (9:5).
Bits   0: {01010101}    1: {01010101}    2: {01011100}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010001}   10: {11010101}   11: {01010101}
                        Initiator 100 µs tone, 84 ns after step start = APPI 7, tone
extension present
                        Reflector 100 µs tone, 334 ns after step start = APPI 7, tone
extension present
              Reflector 96-bit sounding sequence, AA 0xC2EFB690, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 58-61 (7:2); Marker 1100 (DRBG bit 0) at
bits 81-84 (10:1). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01110001}    8:
{01010101}    9: {01010101}   10: {01100101}   11: {01010101}

************************************************************
  SubEvent 7, 85.000 ms-88.289 ms, steps 42-47
        Step #42: Mode-0 step on ch 8, +150.00 µs after subevent anchor
                              Initiator Sync AA only, AA 0x94279424 = No payload,
Access Address only
                   Reflector Sync AA only, AA 0x9652E361, 176 ns after step start = No
payload, Access Address only
                             Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #43: Main-Mode Repetition, Mode-3 step on ch 9, +597.00 µs after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x19D29F36 = Marker 0011
(DRBG bit 1) at bits 9-12 (1:1). Bits   0: {01010101}    1: {00011101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 12
                        Reflector 100 µs tone, 334 ns after step start = APPI 12, tone
extension present
              Reflector 96-bit sounding sequence, AA 0x654219F8, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 36-39 (4:4). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01011100}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #44: Main-Mode Repetition, Mode-3 step on ch 16, +1.270 ms after
```

*Sample Data*

subevent anchor
Initiator 96-bit sounding sequence, AA 0xFA3B44CE = Marker 0011 (DRBG bit 1) at bits 40-43 (5:0). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {00110101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}
Initiator 100 µs tone, 84 ns after step start = APPI 22, tone extension present
Reflector 80 µs tone, 334 ns after step start = APPI 22
Reflector 96-bit sounding sequence, AA 0x25B5E588, 424 ns after step start = Marker 0011 (DRBG bit 1) at bits 60-63 (7:4); Marker 1100 (DRBG bit 0) at bits 79-82 (9:7). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010011} 8: {01010101} 9: {01010101} 10: {10010101} 11: {01010101}

Step #45: Mode-3 step on ch 8, +1.943 ms after subevent anchor
Initiator 96-bit sounding sequence, AA 0x3AFCC966 = Marker 0011 (DRBG bit 1) at bits 23-26 (2:7). Bits 0: {01010101} 1: {01010101} 2: {01010100} 3: {01110101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}
Initiator 80 µs tone, 84 ns after step start = APPI 11
Reflector 80 µs tone, 334 ns after step start = APPI 11
Reflector 96-bit sounding sequence, AA 0xB439B728, 424 ns after step start = Marker 1100 (DRBG bit 0) at bits 10-13 (1:2); Marker 1100 (DRBG bit 0) at bits 75-78 (9:3). Bits 0: {01010101} 1: {01110001} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01011001} 10: {01010101} 11: {01010101}

Step #46: Mode-3 step on ch 6, +2.616 ms after subevent anchor
Initiator 96-bit sounding sequence, AA 0xBFCD8A16 = Marker 0011 (DRBG bit 1) at bits 4-7 (0:4). Bits 0: {01010011} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010101} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}
Initiator 80 µs tone, 84 ns after step start = APPI 7
Reflector 80 µs tone, 334 ns after step start = APPI 7
Reflector 96-bit sounding sequence, AA 0x0D8C3794, 424 ns after step start = Marker 1100 (DRBG bit 0) at bits 61-64 (7:5). Bits 0: {01010101} 1: {01010101} 2: {01010101} 3: {01010101} 4: {01010101} 5: {01010101} 6: {01010101} 7: {01010110} 8: {01010101} 9: {01010101} 10: {01010101} 11: {01010101}

Step #47: Mode-2 step on ch 18, +3.289 ms after subevent anchor
Initiator 100 µs tone = APPI 7, tone extension present

*Sample Data*

```
                        Reflector 80 µs tone, 250 ns after step start = APPI 7

************************************************************
  SubEvent 8, 90.000 ms-93.289 ms, steps 48-53
        Step #48: Mode-0 step on ch 5, +150.00 µs after subevent anchor
                              Initiator Sync AA only, AA 0xB830DBBE = No payload,
Access Address only
                   Reflector Sync AA only, AA 0x77342424, 176 ns after step start = No
payload, Access Address only
                             Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #49: Main-Mode Repetition, Mode-3 step on ch 8, +597.00 µs after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x074A4DA6 = Marker 1100
(DRBG bit 0) at bits 7-10 (0:7). Bits   0: {01010101}    1: {10010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 6
                        Reflector 80 µs tone, 334 ns after step start = APPI 6
              Reflector 96-bit sounding sequence, AA 0x49A713B8, 424 ns after step
start = Marker 0011 (DRBG bit 1) at bits 36-39 (4:4). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010011}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #50: Main-Mode Repetition, Mode-3 step on ch 6, +1.270 ms after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xBD48172D = Marker 0011
(DRBG bit 1) at bits 43-46 (5:3). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010101}    4: {01010101}    5: {01000111}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                        Initiator 80 µs tone, 84 ns after step start = APPI 21
                        Reflector 100 µs tone, 334 ns after step start = APPI 21, tone
extension present
              Reflector 96-bit sounding sequence, AA 0xCCC0161A, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 3-6 (0:3). Bits   0: {01011001}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #51: Mode-3 step on ch 5, +1.943 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x8690B93A = Marker 0011
(DRBG bit 1) at bits 60-63 (7:4). Bits   0: {01010101}    1: {01010101}    2:
```

*Sample Data*

```
{01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010011}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 6
                    Reflector 100 µs tone, 334 ns after step start = APPI 6, tone
extension present
            Reflector 96-bit sounding sequence, AA 0x90F4287D, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 20-23 (2:4). Bits   0: {01010101}    1:
{01010101}    2: {01011100}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #52: Mode-3 step on ch 3, +2.616 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x627EC274 = Marker 1100
(DRBG bit 0) at bits 30-33 (3:6). Bits   0: {01010101}    1: {01010101}    2:
{01010101}    3: {01010111}    4: {00010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 100 µs tone, 84 ns after step start = APPI 8, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 8
            Reflector 96-bit sounding sequence, AA 0x789D6425, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 35-38 (4:3). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01011001}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #53: Mode-3 step on ch 12, +3.289 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x42BBF0C2 = Marker 1100
(DRBG bit 0) at bits 18-21 (2:2); Marker 1100 (DRBG bit 0) at bits 92-95 (11:4).
Bits   0: {01010101}    1: {01010101}    2: {01110001}    3: {01010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010101}   10: {01010101}   11: {01011100}
                    Initiator 100 µs tone, 84 ns after step start = APPI 14, tone
extension present
                    Reflector 80 µs tone, 334 ns after step start = APPI 14
            Reflector 96-bit sounding sequence, AA 0x4B072215, 424 ns after step
start = Marker 1100 (DRBG bit 0) at bits 44-47 (5:4); Marker 1100 (DRBG bit 0) at
bits 74-77 (9:2). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01011100}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {01110001}   10: {01010101}   11: {01010101}

************************************************************
  Event 3, 120.000 ms
    SubEvent 9, 120.000 ms-123.289 ms, steps 54-59
```

*Sample Data*

```
        Step #54: Mode-0 step on ch 2, +150.00 µs after subevent anchor
                            Initiator Sync AA only, AA 0x793D518F = No payload,
Access Address only
                  Reflector Sync AA only, AA 0x3BDA61F4, 176 ns after step start = No
payload, Access Address only
                            Reflector 80 µs tone, 212 ns after step start = APPI 0

        Step #55: Main-Mode Repetition, Mode-3 step on ch 3, +597.00 µs after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x057235B1 = Marker 1100
(DRBG bit 0) at bits 44-47 (5:4); Marker 0011 (DRBG bit 1) at bits 78-81 (9:6).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {01010101}    4:
{01010101}    5: {01011100}    6: {01010101}    7: {01010101}    8: {01010101}    9:
{01010100}   10: {11010101}   11: {01010101}
                        Initiator 100 µs tone, 84 ns after step start = APPI 0, tone
extension present
                        Reflector 100 µs tone, 334 ns after step start = APPI 0, tone
extension present
             Reflector 96-bit sounding sequence, AA 0x9236F946, 444 ns after step
start = Marker 0011 (DRBG bit 1) at bits 42-45 (5:2). Bits   0: {01010101}    1:
{01010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01001101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #56: Main-Mode Repetition, Mode-3 step on ch 12, +1.270 ms after
subevent anchor
                        Initiator 96-bit sounding sequence, AA 0x544FA4E0 = Marker 1100
(DRBG bit 0) at bits 23-26 (2:7); Marker 1100 (DRBG bit 0) at bits 69-72 (8:5).
Bits   0: {01010101}    1: {01010101}    2: {01010101}    3: {10010101}    4:
{01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8: {01010110}    9:
{01010101}   10: {01010101}   11: {01010101}
                        Initiator 100 µs tone, 84 ns after step start = APPI 16, tone
extension present
                        Reflector 100 µs tone, 334 ns after step start = APPI 16, tone
extension present
             Reflector 96-bit sounding sequence, AA 0xFE08D47C, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 54-57 (6:6); Marker 0011 (DRBG bit 1) at
bits 88-91 (11:0). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010101}    4: {01010101}    5: {01010101}    6: {01010111}    7: {00010101}    8:
{01010101}    9: {01010101}   10: {01010101}   11: {00110101}

        Step #57: Mode-3 step on ch 2, +1.943 ms after subevent anchor
                        Initiator 96-bit sounding sequence, AA 0xD9150634 = Marker 1100
```

*Sample Data*

```
(DRBG bit 0) at bits 21-24 (2:5). Bits   0: {01010101}    1: {01010101}    2:
{01010110}    3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7:
{01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 3
                    Reflector 100 µs tone, 334 ns after step start = APPI 3, tone
extension present
            Reflector 96-bit sounding sequence, AA 0xD36EC1D3, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 6-9 (0:6). Bits   0: {01010111}    1:
{00010101}    2: {01010101}    3: {01010101}    4: {01010101}    5: {01010101}    6:
{01010101}    7: {01010101}    8: {01010101}    9: {01010101}   10: {01010101}   11:
{01010101}

        Step #58: Mode-3 step on ch 20, +2.616 ms after subevent anchor
                    Initiator 96-bit sounding sequence, AA 0x7B030BB4 = Marker 1100
(DRBG bit 0) at bits 2-5 (0:2). Bits   0: {01110001}    1: {01010101}    2: {01010101}
3: {01010101}    4: {01010101}    5: {01010101}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {01010101}   10: {01010101}   11: {01010101}
                    Initiator 80 µs tone, 84 ns after step start = APPI 23
                    Reflector 100 µs tone, 334 ns after step start = APPI 23, tone
extension present
            Reflector 96-bit sounding sequence, AA 0xA15F89A6, 444 ns after step
start = Marker 1100 (DRBG bit 0) at bits 30-33 (3:6); Marker 1100 (DRBG bit 0) at
bits 72-75 (9:0). Bits   0: {01010101}    1: {01010101}    2: {01010101}    3:
{01010111}    4: {00010101}    5: {01010101}    6: {01010101}    7: {01010101}    8:
{01010101}    9: {11000101}   10: {01010101}   11: {01010101}

        Step #59: Mode-2 step on ch 17, +3.289 ms after subevent anchor
                                Initiator 80 µs tone = APPI 22
                    Reflector 80 µs tone, 250 ns after step start = APPI 22
```

## 8.2 Channel Selection Algorithm #3c

The timing parameters provided with each set are used to actively calculate the number of steps included in each CS subevent.

### 8.2.1 Set 1

Time parameters:

| | |
|---|---|
| Sync packet: | T_SY = 26 |
| Frequency measurement: | T_FM = 80 µs |
| Guard Time: | T_GD = 10 µs |
| Time to hop: | T_FCS = 15 µs |

*Sample Data*

| | |
|---|---|
| Ramp down time: | T_RD = 5 µs |
| Interlude period 1: | T_IP1 = 10 µs |
| Interlude period 2: | T_IP2 = 10 µs |
| Phase measurement time: | T_PM = 10 µs |
| Number of antenna path: | N_AP = 1 |
| Antenna switch time: | T_SW = 0 µs |
| --- | --- |
| CSShapeSelection: | X |
| CSChannelJump: | 2 |
| CSNumRepetitions: | 1 |
| --- | --- |
| Mode-0 duration: | 177 µs |
| Main-mode (Mode-2) duration: | 75 µs |
| Sub-Mode (Mode-3) duration: | 147 µs |
| Minimum main mode steps: | 2 |
| Maximum main mode steps: | 4 |
| PHY: | 2 Mbps |
| RTT Type: | AA-only |
| --- | --- |
| Event duration: | 5000 µs |
| Mode_0_Steps: | 3 |
| Main_Mode_Repetition: | 3 |

```
************************************ Channel Map ************************************

Bit-map:           1F FF FF FF FF FF FC 7F FF FC

Filtered channels: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76

****************************** INSTANTIATION FUNCTION *********************************

h9() instantiation

*******************************************************************************************

Entropy input Peripheral (CS_IV_P):          E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):             3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                       E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B
```

*Sample Data*

```
Nonce Peripheral (CS_IN_P):                  9F F4 77 C1

Nonce Central (CS_IN_C):                     86 73 84 0D

Nonce (CS_IN):                               9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):    64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):              C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

********************************** INITIAL K and V **********************************

K: EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V: DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

*******************************************************************************************

Step=0 | Mode=0

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 0;

New DRBG octets: FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

************************************************************

T_ID 1 - mode0 channel (8 bits):  FF  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  BC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  CA  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  A6  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  9D  -  Range = 8 (decimal); hr1 = 4 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  C4  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  07  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8 bits):  38  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  33  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  D9  -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  D1  -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 1 - mode0 channel (8 bits):  35  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  32  -  Range = 17 (decimal); hr1 = 3 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 1;

New DRBG octets: 51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

************************************************************

T_ID 1 - mode0 channel (8 bits):  51  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EE  -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  4B  -  Range = 20 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  B0  -  Range = 21 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  3E  -  Range = 22 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  E6  -  Range = 23 (decimal); hr1 = 20 (decimal)

T_ID 1 - mode0 channel (8 bits):  D0  -  Range = 24 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  A1  -  Range = 25 (decimal); hr1 = 15 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  71  -  Range = 26 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  87  -  Range = 27 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  1C  -  Range = 28 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2E  -  Range = 29 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  60  -  Range = 30 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  46  -  Range = 31 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  6C  -  Range = 33 (decimal); hr1 = 13 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 2;

New DRBG octets: 7D 75 2F 17 23 57 34 22 EF C7 CB C2 1F 18 90 8F

************************************************************

T_ID 1 - mode0 channel (8 bits):  7D  -  Range = 34 (decimal); hr1 = 16 (decimal)

T_ID 1 - mode0 channel (8 bits):  75  -  Range = 35 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 36 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 37 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  23  -  Range = 38 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  57  -  Range = 39 (decimal); hr1 = 13 (decimal)

T_ID 1 - mode0 channel (8 bits):  34  -  Range = 40 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  22  -  Range = 41 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 42 (decimal); hr1 = 39 (decimal)

T_ID 1 - mode0 channel (8 bits):  C7  -  Range = 43 (decimal); hr1 = 33 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  CB  -  Range = 44 (decimal); hr1 = 34 (decimal)

T_ID 1 - mode0 channel (16 bits): C2 1F -  Range = 45 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  18  -  Range = 46 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  90  -  Range = 47 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 48 (decimal); hr1 = 26 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 3;

New DRBG octets: 64 74 2B E9 80 0D 87 22 36 1E 6F 55 61 A8 7C 17

**************************************************************

T_ID 1 - mode0 channel (8 bits):  64  -  Range = 49 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  74  -  Range = 50 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (8 bits):  2B  -  Range = 51 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  E9  -  Range = 52 (decimal); hr1 = 47 (decimal)

T_ID 1 - mode0 channel (8 bits):  80  -  Range = 53 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8 bits):  0D  -  Range = 54 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (16 bits): 87 22 -  Range = 55 (decimal); hr1 = 7 (decimal)

T_ID 1 - mode0 channel (8 bits):  36  -  Range = 56 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  1E  -  Range = 57 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  6F  -  Range = 58 (decimal); hr1 = 25 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 59 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  61  -  Range = 60 (decimal); hr1 = 22 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (16 bits): A8 7C -  Range = 61 (decimal); hr1 = 29 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 62 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 4;

New DRBG octets: 0F 2F 55 C1 9B BD 7C 71 EC 79 0A 97 FD 0D 93 69

************************************************************

T_ID 1 - mode0 channel (8 bits):  0F  -  Range = 63 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 64 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 65 (decimal); hr1 = 21 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 66 (decimal); hr1 = 49 (decimal)

T_ID 1 - mode0 channel (8 bits):  9B  -  Range = 67 (decimal); hr1 = 40 (decimal)

T_ID 1 - mode0 channel (8 bits):  BD  -  Range = 68 (decimal); hr1 = 50 (decimal)

T_ID 1 - mode0 channel (8 bits):  7C  -  Range = 69 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8 bits):  71  -  Range = 70 (decimal); hr1 = 30 (decimal)

T_ID 1 - mode0 channel (8 bits):  EC  -  Range = 71 (decimal); hr1 = 65 (decimal)

T_ID 1 - mode0 channel (16 bits): 79 0A -  Range = 72 (decimal); hr1 = 2 (decimal)

Mode0ShuffledchannelArray:

11 7 76 67 50 66 61 59 55 2 4 68 16 43 31 39 38 36 3 63 27 69 64 19 6 62 57 18 26
65 74 20 13 73 48 10 32 33 37 46 71 35 17 29 45 9 22 56 28 70 72 51 52 14 8 34 40
15 53 54 30 49 41 60 21 75 42 44 47 5 12 58

Jitter selection:

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 0;
```

*Sample Data*

```
New DRBG octets: 79 74 1F D1 8F 57 7B 45 D0 9A 66 5A 7F 1F 28 58

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  79  -  Range = 2 (decimal); hr1 = 0 (decimal)

ShapeChSeq:

1 76 3 74 5 72 7 70 9 68 11 66 13 64 15 62 17 60 19 58 21 56 23 54 25 52 27 50 29
48 31 46 33 44 35 42 37 40 39 38 41 36 43 34 45 32 47 30 49 28 51 26 53 24 55 22 57
20 59 18 61 16 63 14 65 12 67 10 69 8 71 6 73 4 75 2 77 0

firstAndEndAllChSeq:

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  57  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 7 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  45  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 9A 66 - Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 12 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 13 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 14 (decimal); hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 15 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 1;

New DRBG octets: D1 C1 D0 5A 40 B4 C4 81 EF BB 39 B2 61 D2 9C 4E

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 16 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C1  -  Range = 17 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 18 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 19 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 40 B4 - Range = 20 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 21 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 22 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EF  -  Range = 23 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BB  -  Range = 24 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 25 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): B2 61 - Range = 26 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D2  -  Range = 27 (decimal); hr1 = 22 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9C  -  Range = 28 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4E  -  Range = 29 (decimal); hr1 = 8 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 2;
```

*Sample Data*

```
New DRBG octets: 71 FD E8 68 E8 CA CA D1 18 E5 9B 18 5C EE FC 17

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  71  -  Range = 30 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FD  -  Range = 31 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 32 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 33 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 34 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 35 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 36 (decimal); hr1 = 28 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 37 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 38 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E5  -  Range = 39 (decimal); hr1 = 34 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9B  -  Range = 40 (decimal); hr1 = 24 (decimal)

firstAndEndAllChSeq Shuffled:

22 32 33 57 29 44 38 28 48 45 26 41 36 52 39 27 40 47 23 37 20 42 46 31 59 21 24 54
55 51 56 35 49 50 58 25 53 30 43 34

middleAllChSeq:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EE  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FC  -  Range = 5 (decimal); hr1 = 4 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 6 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 3;

New DRBG octets: B5 0E CB BF 12 99 7C 4B 02 63 35 D0 81 C5 6B D0

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  B5  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0E  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BF  -  Range = 10 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  12  -  Range = 11 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  99  -  Range = 12 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7C  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 14 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 15 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  63  -  Range = 16 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  35  -  Range = 17 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 18 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 19 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 20 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6B  -  Range = 21 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 22 (decimal); hr1 = 17 (decimal)
```

*Sample Data*

```
************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 4;

New DRBG octets: C3 3C B1 23 C6 A2 E0 8A BE 0E 8A BD 22 CB 24 1F

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 23 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3C  -  Range = 24 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B1  -  Range = 25 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 26 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C6  -  Range = 27 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A2  -  Range = 28 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E0  -  Range = 29 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 30 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (16 bits): BE 0E  -  Range = 31 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BD  -  Range = 33 (decimal); hr1 = 24 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  22  -  Range = 34 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 35 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  24  -  Range = 36 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 37 (decimal); hr1 = 4 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 5;

New DRBG octets: 68 87 A9 32 7B E1 61 B4 CD 1E 92 4D AC 00 74 05
```

*Sample Data*

```
************************************************************

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 38 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  87  -  Range = 39 (decimal); hr1 = 20 (decimal)

middleAllChSeq Shuffled:

14 70 0 65 76 75 15 11 60 18 7 9 4 6 17 77 69 71 8 12 78 10 61 1 72 68 5 74 16
3 2 67 62 13 64 63 73 19 66

firstAndEndUnusedChSeq:

firstAndEndUnusedChSeq Shuffled:

middleUnusedChSeq:

78

middleUnusedChSeq Shuffled:

78

FirstAndEndSaltChSeq:

22 32 33 57 29 44 38 28 48 45 26 41 36 52 39 27 40 47 23 37 20 42 46 31 59 21 24
54 55 51 56 35 49 50 58 25 53 30 43 34

MiddleSaltChSeq:

78 14 70 0 65 76 75 15 11 60 18 7 9 4 6 17 77 69 71 8 12 78 10 61 1 72 68 5 74 16
3 2 67 62 13 64 63 73 19 66

Random number for initial and end salting

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 10 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  32  -  Range = 5 (decimal); hr1 = 0 (decimal)

SaltedChSeq:

22 78 32 14 33 70 57 1 76 29 3 74 44 5 72 38 7 70 28 9 68 48 11 66 45 13 64 26 15
```

*Sample Data*

```
62 41 17 60 36 19 58 0 21 56 65 23 54 76 25 52 75 27 50 15 29 48 11 31 46 60 33
44 18 35 42 7 37 40 9 39 38 4 41 36 6 43 34 17 45 32 77 47 30 69 49 28 71 51 26 8
53 24 12 55 22 78 57 20 10 59 18 52 61 16 39 63 14 27 65 12 40 67 10 47 69 8 23 71
6 37 73 4 20 75 2 42 77 0 46

tempBlockSeq:

22 32 14 33 70 57 76 29 3 74 44 5 72 38 7 70 28 9 68 48 11 66 45 13 64 26 15 62

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E1  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B4  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CD  -  Range = 6 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 7 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  92  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4D  -  Range = 9 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AC  -  Range = 10 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 00 74  - Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  05  -  Range = 12 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 6;

New DRBG octets: 2B BC 5A 8F FB 1E 39 AA 02 8F F4 4C 0A 11 7D F9

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  2B  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 14 (decimal); hr1 = 10 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 15 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 16 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FB  -  Range = 17 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 18 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 19 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AA  -  Range = 20 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 21 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 22 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F4  -  Range = 23 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4C  -  Range = 24 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 25 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  11  -  Range = 26 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 27 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 28 (decimal); hr1 = 27 (decimal)

tempBlockSeq Shuffled:

64 26 9 70 68 7 74 13 70 32 38 76 66 15 22 14 28 72 44 29 5 45 3 57 11 33 48 62

tempBlockSeq:

41 17 60 36 19 58 21 56 65 54 76 52 75 27 50 15 29 48 11 31 46 60 33 44 18 35 42 7

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 7;

New DRBG octets: 58 A1 10 C9 6F 8A 88 43 76 61 17 07 D7 61 DF A3

************************************************************
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A1  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C9  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6F  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 7 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  88  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  43  -  Range = 9 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  76  -  Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 12 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  07  -  Range = 13 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D7  -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 15 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DF  -  Range = 16 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 17 (decimal); hr1 = 10 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 8;

New DRBG octets: 53 23 37 62 4B C5 D9 66 1A C4 34 BC 41 36 C0 FE

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  53  -  Range = 18 (decimal); hr1 = 5 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 19 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  37  -  Range = 20 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  62  -  Range = 21 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 22 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 23 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 24 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 25 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1A  -  Range = 26 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 27 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  34  -  Range = 28 (decimal); hr1 = 5 (decimal)

tempBlockSeq Shuffled:

75 52 35 21 31 7 60 17 46 18 29 27 36 15 41 60 54 33 65 76 42 19 50 58 56 11 44 48

tempBlockSeq:

37 40 9 39 38 4 41 36 6 43 34 17 45 32 47 30 69 49 28 71 51 26 8 53 12 55 22 57

T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  41  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  36  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C0  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FE  -  Range = 6 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 9;

New DRBG octets: 28 BE 39 8A 7D 3F DA 86 15 0A 83 3E 4B 8B 3D D9
```

*Sample Data*

```
************************************************************

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BE  -  Range = 8 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 39 8A  -  Range = 9 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3F  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DA  -  Range = 12 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  15  -  Range = 14 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 15 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  83  -  Range = 16 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3E  -  Range = 17 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 19 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3D  -  Range = 20 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 21 (decimal); hr1 = 17 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 10;

New DRBG octets: 86 AD C3 21 02 CF 55 74 D9 58 09 10 16 F9 64 97

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 22 (decimal); hr1 = 11 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  AD  -  Range = 23 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 24 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 25 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 26 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CF  -  Range = 27 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  55  -  Range = 28 (decimal); hr1 = 9 (decimal)

tempBlockSeq Shuffled:

55 32 34 12 71 49 45 4 30 57 28 26 40 41 39 8 43 51 53 69 36 22 9 17 38 47 37 6

tempBlockSeq:

20 10 59 18 52 61 16 39 63 14 27 65 12 40 67 10 47 69 8 71 6 37 73 4 20 75 2 42 46

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  09  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 6 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  16  -  Range = 7 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 8 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  64  -  Range = 9 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  97  -  Range = 10 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 11;

New DRBG octets: A9 B9 82 8A 7F 2A EC D9 59 D1 74 8B C0 31 4A 9F
```

*Sample Data*

```
**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 11 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B9  -  Range = 12 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  82  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 14 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 15 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2A  -  Range = 16 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EC  -  Range = 17 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 18 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 19 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 20 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 21 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 22 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C0  -  Range = 23 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  31  -  Range = 24 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4A  -  Range = 25 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9F  -  Range = 26 (decimal); hr1 = 16 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 12;

New DRBG octets: 59 E0 5C 66 A0 CE 21 CB 23 81 24 96 31 01 37 50

**************************************************************
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 27 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E0  -  Range = 28 (decimal); hr1 = 24 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5C  -  Range = 29 (decimal); hr1 = 10 (decimal)

tempBlockSeq Shuffled:

16 18 10 63 4 14 8 20 65 2 46 37 61 27 40 69 75 73 12 59 52 20 47 10 42 71 6 67 39

NonMode0ShuffledChannelArray:

64 26 9 70 68 7 74 13 70 32 38 76 66 15 22 14 28 72 44 29 5 45 3 57 11 33 48 62 75
52 35 21 31 7 60 17 46 18 29 27 36 15 41 60 54 33 65 76 42 19 50 58 56 11 44 48 55
32 34 12 71 49 45 4 30 57 28 26 40 41 39 8 43 51 53 69 36 22 9 17 38 47 37 6 16 18
10 63 4 14 8 20 65 2 46 37 61 27 40 69 75 73 12 59 52 20 47 10 42 71 6 67 39

************************************************************

Step Counter = 3; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

************************************************************

T_ID 2 - Submode insertion (8 bits):  F8  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  82  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  2A  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  54  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  E0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  7C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  50  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  15  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  57  -  Range = 3 (decimal); hr1 = 1 (decimal)
```

*Sample Data*

```
T_ID 2 - Submode insertion (8 bits):  CD  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  98  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  51  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  69  -  Range = 3 (decimal); hr1 = 1 (decimal)

Event 1 is full. step_ctr = 51, seq_ctr = 48, time = 4995

T_ID 2 - Submode insertion (8 bits):  D4  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AB  -  Range = 3 (decimal); hr1 = 2 (decimal)

************************************************************

Step Counter = 68; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: 99 24 D0 DE B1 04 EF EE 43 9F 91 BD 9F 61 D0 3A

************************************************************

T_ID 2 - Submode insertion (8 bits):  99  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  24  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  D0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  DE  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  B1  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  04  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  EF  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  EE  -  Range = 3 (decimal); hr1 = 2 (decimal)

Event 2 is full. step_ctr = 52, seq_ctr = 94, time = 4926

T_ID 2 - Submode insertion (8 bits):  43  -  Range = 3 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
T_ID 2 - Submode insertion (8 bits):  9F  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  91  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  BD  -  Range = 3 (decimal); hr1 = 2 (decimal)

steps per event: 25, time = 2469

Event 1 channels:

11 7 76 64 26 9 70 68 7 74 13 70 32 38 76 66 15 22 14 28 72 44 29 5 45 3 57 11 33
48 62 75 52 35 21 31 7 60 17 46 18 29 27 36 15 41 60 54 33 65 76

Event 2 channels:

67 50 66 54 65 76 42 19 50 58 56 11 44 48 55 32 34 12 71 49 45 4 30 57 28 26 40 41
39 8 43 51 53 69 36 22 9 17 38 47 37 6 16 18 10 63 4 14 8 20 65 2

Event 3 channels:

61 59 55 8 20 2 46 37 61 27 40 69 75 73 12 59 52 20 47 10 42 71 6 67 39
```

### 8.2.2 Set 2

Time parameters:

| | |
|---|---|
| Sync packet: | T_SY = 26 |
| Frequency measurement: | T_FM = 80 µs |
| Guard Time: | T_GD = 10 µs |
| Time to hop: | T_FCS = 15 µs |
| Ramp down time: | T_RD = 5 µs |
| Interlude period 1: | T_IP1 = 10 µs |
| Interlude period 2: | T_IP2 = 10 µs |
| Phase measurement time: | T_PM = 10 µs |
| Number of antenna path: | N_AP = 1 |
| Antenna switch time: | T_SW = 0 µs |
| --- | --- |
| CSShapeSelection: | Hat |
| CSChannelJump: | 2 |
| CSNumRepetitions: | 1 |
| --- | --- |
| Mode-0 duration: | 177 µs |

*Sample Data*

Main-mode (Mode-2) duration: 75 µs
Sub-Mode (Mode-3) duration: 147 µs
Minimum main mode steps: 2
Maximum main mode steps: 4
PHY: 2 Mbps
RTT Type: AA-only
---
Event duration: 5000 µs
Mode_0_Steps: 3
Main_Mode_Repetition: 3

```
********************************** Channel Map ****************************************

Bit-map:           1F FF FF FF FF FF FC 7F FF FC

Filtered channels: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76

****************************** INSTANTIATION FUNCTION *********************************

h9() instantiation

*******************************************************************************************

Entropy input Peripheral (CS_IV_P):           E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):              3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                        E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral (CS_IN_P):                   9F F4 77 C1

Nonce Central (CS_IN_C):                      86 73 84 0D

Nonce (CS_IN):                                9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P):  C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):     64 A6 74 96 78 68 F1 43

Personalization string (CS_PV):               C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43
```

*Sample Data*

```
********************************** INITIAL K and V **********************************

K:    EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:    DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

*****************************************************************************************

Step=0 | Mode=0

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 0;

New DRBG octets: FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

************************************************************

T_ID 1 - mode0 channel (8 bits):  FF  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  BC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  CA  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  A6  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  9D  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  C4  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  07  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8 bits):  38  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  33  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  D9  -  Range = 14 (decimal); hr1 = 11 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  D1  -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 1 - mode0 channel (8 bits):  35  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  32  -  Range = 17 (decimal); hr1 = 3 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 1;

New DRBG octets: 51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

**************************************************************

T_ID 1 - mode0 channel (8 bits):  51  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EE  -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  4B  -  Range = 20 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  B0  -  Range = 21 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  3E  -  Range = 22 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  E6  -  Range = 23 (decimal); hr1 = 20 (decimal)

T_ID 1 - mode0 channel (8 bits):  D0  -  Range = 24 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  A1  -  Range = 25 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8 bits):  71  -  Range = 26 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  87  -  Range = 27 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  1C  -  Range = 28 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2E  -  Range = 29 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  60  -  Range = 30 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  46  -  Range = 31 (decimal); hr1 = 8 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  6C  -  Range = 33 (decimal); hr1 = 13 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 2;

New DRBG octets: 7D 75 2F 17 23 57 34 22 EF C7 CB C2 1F 18 90 8F

************************************************************

T_ID 1 - mode0 channel (8 bits):  7D  -  Range = 34 (decimal); hr1 = 16 (decimal)

T_ID 1 - mode0 channel (8 bits):  75  -  Range = 35 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 36 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 37 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  23  -  Range = 38 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  57  -  Range = 39 (decimal); hr1 = 13 (decimal)

T_ID 1 - mode0 channel (8 bits):  34  -  Range = 40 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  22  -  Range = 41 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 42 (decimal); hr1 = 39 (decimal)

T_ID 1 - mode0 channel (8 bits):  C7  -  Range = 43 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8 bits):  CB  -  Range = 44 (decimal); hr1 = 34 (decimal)

T_ID 1 - mode0 channel (16 bits): C2 1F  -  Range = 45 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  18  -  Range = 46 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  90  -  Range = 47 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 48 (decimal); hr1 = 26 (decimal)

************************************************************
```

*Sample Data*

```
Step Counter = 0; Transaction ID = 1; Transaction Counter = 3;

New DRBG octets: 64 74 2B E9 80 0D 87 22 36 1E 6F 55 61 A8 7C 17

**************************************************************

T_ID 1 - mode0 channel (8 bits):  64  -  Range = 49 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  74  -  Range = 50 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (8 bits):  2B  -  Range = 51 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  E9  -  Range = 52 (decimal); hr1 = 47 (decimal)

T_ID 1 - mode0 channel (8 bits):  80  -  Range = 53 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8 bits):  0D  -  Range = 54 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (16 bits): 87 22  -  Range = 55 (decimal); hr1 = 7 (decimal)

T_ID 1 - mode0 channel (8 bits):  36  -  Range = 56 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  1E  -  Range = 57 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  6F  -  Range = 58 (decimal); hr1 = 25 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 59 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bis):  61  -  Range = 60 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (16 bits): A8 7C  -  Range = 61 (decimal); hr1 = 29 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 62 (decimal); hr1 = 5 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 4;

New DRBG octets: 0F 2F 55 C1 9B BD 7C 71 EC 79 0A 97 FD 0D 93 69

**************************************************************
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  0F  -  Range = 63 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 64 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 65 (decimal); hr1 = 21 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 66 (decimal); hr1 = 49 (decimal)

T_ID 1 - mode0 channel (8 bits):  9B  -  Range = 67 (decimal); hr1 = 40 (decimal)

T_ID 1 - mode0 channel (8 bits):  BD  -  Range = 68 (decimal); hr1 = 50 (decimal)

T_ID 1 - mode0 channel (8 bits):  7C  -  Range = 69 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8 bits):  71  -  Range = 70 (decimal); hr1 = 30 (decimal)

T_ID 1 - mode0 channel (8 bits):  EC  -  Range = 71 (decimal); hr1 = 65 (decimal)

T_ID 1 - mode0 channel (16 bits): 79 0A  -  Range = 72 (decimal); hr1 = 2 (decimal)

Mode0ShuffledChannelArray:

11 7 76 67 50 66 61 59 55 2 4 68 16 43 31 39 38 36 3 63 27 69 64 19 6 62 57 18 26 65
74 20 13 73 48 10 32 33 37 46 71 35 17 29 45 9 22 56 28 70 72 51 52 14 8 34 40 15 53
54 30 49 41 60 21 75 42 44 47 5 12 58

Jitter selection:

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 0;

New DRBG octets: 79 74 1F D1 8F 57 7B 45 D0 9A 66 5A 7F 1F 28 58

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  79  -  Range = 2 (decimal); hr1 = 0 (decimal)

ShapeChSeq :

1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59
61 63 65 67 69 71 73 75 77 76 74 72 70 68 66 64 62 60 58 56 54 52 50 48 46 44 42 40
38 36 34 32 30 28 26 24 22 20 18 16 14 12 10 8 6 4 2 0
```

*Sample Data*

```
firstAndEndAllChSeq:

40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67
68 69 70 71 72 73 74 75 76 77 78

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  57  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 7 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  45  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 9A 66  - Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 12 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 13 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 14 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 15 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 1;

New DRBG octets: D1 C1 D0 5A 40 B4 C4 81 EF BB 39 B2 61 D2 9C 4E

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 16 (decimal); hr1 = 13 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  C1  -  Range = 17 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 18 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 19 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 40 B4  - Range = 20 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 21 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 22 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EF  -  Range = 23 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BB  -  Range = 24 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 25 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): B2 61  - Range = 26 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D2  -  Range = 27 (decimal); hr1 = 22 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9C  -  Range = 28 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4E  -  Range = 29 (decimal); hr1 = 8 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 2;

New DRBG octets: 71 FD E8 68 E8 CA CA D1 18 E5 9B 18 5C EE FC 17

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  71  -  Range = 30 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FD  -  Range = 31 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 32 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 33 (decimal); hr1 = 13 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 34 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 35 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 36 (decimal); hr1 = 28 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 37 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 38 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E5  -  Range = 39 (decimal); hr1 = 34 (decimal)

firstAndEndAllChSeq Shuffled:

42 52 53 77 49 64 58 48 68 65 46 61 56 72 59 47 60 67 43 57 40 62 66 51 54 41 44 74 75
71 76 55 69 70 78 45 73 50 63

middleAllChSeq:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39

T_ID 0 - non-mode0 channel (8 bits):  9B  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5C  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EE  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FC  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 7 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 3;

New DRBG octets: B5 0E CB BF 12 99 7C 4B 02 63 35 D0 81 C5 6B D0

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  B5  -  Range = 8 (decimal); hr1 = 5 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  0E  -  Range = 9 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 10 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BF  -  Range = 11 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  12  -  Range = 12 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  99  -  Range = 13 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7C  -  Range = 14 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 15 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 16 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  63  -  Range = 17 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  35  -  Range = 18 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 19 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 20 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 21 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6B  -  Range = 22 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 23 (decimal); hr1 = 18 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 4;

New DRBG octets: C3 3C B1 23 C6 A2 E0 8A BE 0E 8A BD 22 CB 24 1F

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 24 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3C  -  Range = 25 (decimal); hr1 = 5 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  B1  -  Range = 26 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 27 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C6  -  Range = 28 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A2  -  Range = 29 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E0  -  Range = 30 (decimal); hr1 = 26 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 31 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BE  -  Range = 32 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0E  -  Range = 33 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 34 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BD  -  Range = 35 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  22  -  Range = 36 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 37 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  24  -  Range = 38 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 39 (decimal); hr1 = 4 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 5;

New DRBG octets: 68 87 A9 32 7B E1 61 B4 CD 1E 92 4D AC 00 74 05

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 40 (decimal); hr1 = 16 (decimal)

middleAllChSeq Shuffled:

15 32 0 26 38 37 16 12 10 21 19 8 9 2 4 18 39 25 33 6 13 27 11 31 7 34 29 5 23 36 20
22 3 28 1 14 17 24 35 30
```

*Sample Data*

```
firstAndEndUnusedChSeq:

78

firstAndEndUnusedChSeq Shuffled:

78

middleUnusedChSeq:

middleUnusedChSeq Shuffled:

FirstAndEndSaltChSeq:

78 42 52 53 77 49 64 58 48 68 65 46 61 56 72 59 47 60 67 43 57 40 62 66 51 54 41 44
74 75 71 76 55 69 70 78 45 73 50 63

MiddleSaltChSeq:

15 32 0 26 38 37 16 12 10 21 19 8 9 2 4 18 39 25 33 6 13 27 11 31 7 34 29 5 23 36 20
22 3 28 1 14 17 24 35 30

Random number for initial and end salting

T_ID 0 - non-mode0 channel (8 bits):  87  -  Range = 10 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 5 (decimal); hr1 = 3 (decimal)

SaltedChSeq:

78 15 42 32 52 53 1 3 77 5 7 49 9 11 64 13 15 58 17 19 48 21 23 68 25 27 65 29 31 46
33 35 61 37 39 0 41 43 26 45 47 38 49 51 37 53 55 16 57 59 12 61 63 10 65 67 21 69 71
19 73 75 8 77 76 9 74 72 2 70 68 4 66 64 18 62 60 39 58 56 25 54 52 33 50 48 6 46 44
13 42 40 56 38 36 72 34 32 59 30 28 47 26 24 60 22 20 67 18 16 43 14 12 57 10 8 40 6
4 62 2 0 66 27 51 11 31 7

tempBlockSeq:

15 42 32 52 53 3 5 7 49 9 11 64 13 15 58 17 19 48 21 68 27 65 29 31 46 33 35 61 37

T_ID 0 - non-mode0 channel (8 bits):  32  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 3 (decimal); hr1 = 1 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  E1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B4  -  Range = 6 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CD  -  Range = 7 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  92  -  Range = 9 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 4D AC  - Range = 10 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 00 74  - Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  05  -  Range = 12 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 6;

New DRBG octets: 2B BC 5A 8F FB 1E 39 AA 02 8F F4 4C 0A 11 7D F9

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  2B  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 14 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 15 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 16 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FB  -  Range = 17 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 18 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 19 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AA  -  Range = 20 (decimal); hr1 = 13 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 21 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 22 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F4  -  Range = 23 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4C  -  Range = 24 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 25 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  11  -  Range = 26 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 27 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 28 (decimal); hr1 = 27 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 7;

New DRBG octets: 58 A1 10 C9 6F 8A 88 43 76 61 17 07 D7 61 DF A3

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 29 (decimal); hr1 = 9 (decimal)

tempBlockSeq Shuffled:

46 33 48 52 21 58 9 31 17 37 15 7 65 35 49 5 19 13 11 3 64 29 15 42 27 53 68 61 32

tempBlockSeq:

39 41 43 26 45 47 38 49 51 37 53 55 16 57 59 12 61 63 10 65 67 21 69 71 19 73 75 8 76

T_ID 0 - non-mode0 channel (8 bits):  A1  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C9  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 6 (decimal); hr1 = 3 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  88  -  Range = 7 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  43  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  76  -  Range = 9 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 10 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 11 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  07  -  Range = 12 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D7  -  Range = 13 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 14 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DF  -  Range = 15 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 16 (decimal); hr1 = 10 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 8;

New DRBG octets: 53 23 37 62 4B C5 D9 66 1A C4 34 BC 41 36 C0 FE

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  53  -  Range = 17 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 18 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  37  -  Range = 19 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  62  -  Range = 20 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 21 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 22 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 23 (decimal); hr1 = 19 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 24 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1A  -  Range = 25 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 26 (decimal); hr1 = 19 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  34  -  Range = 27 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 28 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  41  -  Range = 29 (decimal); hr1 = 7 (decimal)

tempBlockSeq Shuffled:

55 41 19 37 10 75 67 76 39 71 12 53 43 59 26 16 21 49 51 73 8 57 45 38 63 69 61 47 65

tempBlockSeq:

9 74 72 2 70 68 4 66 64 18 62 60 39 58 56 54 52 33 50 48 6 46 44 13 42 40 56 38 36

T_ID 0 - non-mode0 channel (8 bits):  36  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FE  -  Range = 4 (decimal); hr1 = 3 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 9;

New DRBG octets: 28 BE 39 8A 7D 3F DA 86 15 0A 83 3E 4B 8B 3D D9

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BE  -  Range = 6 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 9 (decimal); hr1 = 4 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  3F  -  Range = 10 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DA  -  Range = 11 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 12 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  15  -  Range = 13 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 14 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  83  -  Range = 15 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3E  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 17 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 18 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3D  -  Range = 19 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 20 (decimal); hr1 = 16 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 10;

New DRBG octets: 86 AD C3 21 02 CF 55 74 D9 58 09 10 16 F9 64 97

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 21 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AD  -  Range = 22 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 23 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 24 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 25 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (16 bits): CF 55  -  Range = 26 (decimal); hr1 = 8 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 27 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 28 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 29 (decimal); hr1 = 9 (decimal)

tempBlockSeq Shuffled:

42 39 18 13 50 74 60 56 40 36 6 9 56 70 46 2 48 44 52 64 72 68 62 38 58 66 4 54 33

tempBlockSeq:

72 34 32 59 30 28 47 26 60 22 20 67 18 16 43 14 12 57 10 8 40 6 4 62 2 66 27 51 11
31 7

T_ID 0 - non-mode0 channel (8 bits):  09  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  16  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  64  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  97  -  Range = 7 (decimal); hr1 = 4 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 11;

New DRBG octets: A9 B9 82 8A 7F 2A EC D9 59 D1 74 8B C0 31 4A 9F

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 8 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B9  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  82  -  Range = 10 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 11 (decimal); hr1 = 5 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 12 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2A  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EC  -  Range = 14 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 16 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 17 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 18 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 19 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (16 bits): C0 31  - Range = 20 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4A  -  Range = 21 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9F  -  Range = 22 (decimal); hr1 = 13 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 12;

New DRBG octets: 59 E0 5C 66 A0 CE 21 CB 23 81 24 96 31 01 37 50

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 23 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (16 bits): E0 5C  - Range = 24 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 25 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A0  -  Range = 26 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CE  -  Range = 27 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 28 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 29 (decimal); hr1 = 22 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 30 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 31 (decimal); hr1 = 15 (decimal)

tempBlockSeq Shuffled:

59 72 18 51 31 14 40 4 62 2 10 20 43 6 16 7 66 30 22 32 60 27 11 57 26 28 12 8 34 47
67

NonMode0ShuffledChannelArray:

46 33 48 52 21 58 9 31 17 37 15 7 65 35 49 5 19 13 11 3 64 29 15 42 27 53 68 61 32 55
41 19 37 10 75 67 76 39 71 12 53 43 59 26 16 21 49 51 73 8 57 45 38 63 69 61 47 65 42
39 18 13 50 74 60 56 40 36 6 9 56 70 46 2 48 44 52 64 72 68 62 38 58 66 4 54 33 59 72
18 51 31 14 40 4 62 2 10 20 43 6 16 7 66 30 22 32 60 27 11 57 26 28 12 8 34 47 67

************************************************************

Step Counter = 3; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

************************************************************

T_ID 2 - Submode insertion (8 bits):  F8  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  82  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  2A  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  54  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  E0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  7C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  50  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  15  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  57  -  Range = 3 (decimal); hr1 = 1 (decimal)
```

*Sample Data*

```
T_ID 2 - Submode insertion (8 bits):  CD  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  98  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  51  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  69  -  Range = 3 (decimal); hr1 = 1 (decimal)

Event 1 is full. step_ctr = 51, seq_ctr = 48, time = 4995

T_ID 2 - Submode insertion (8 bits):  D4  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AB  -  Range = 3 (decimal); hr1 = 2 (decimal)

************************************************************

Step Counter = 68; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: 99 24 D0 DE B1 04 EF EE 43 9F 91 BD 9F 61 D0 3A

************************************************************

T_ID 2 - Submode insertion (8 bits):  99  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  24  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  D0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  DE  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  B1  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  04  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  EF  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  EE  -  Range = 3 (decimal); hr1 = 2 (decimal)

Event 2 is full. step_ctr = 52, seq_ctr = 94, time = 4926

T_ID 2 - Submode insertion (8 bits):  43  -  Range = 3 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
T_ID 2 - Submode insertion (8 bits):  9F  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  91  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  BD  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  9F  -  Range = 3 (decimal); hr1 = 1 (decimal)

steps per event: 30, time = 2916

Event 1 channels:

11 7 76 46 33 48 52 21 58 9 31 17 37 15 7 65 35 49 5 19 13 11 3 64 29 15 42 27 53 68
61 32 55 41 19 37 10 75 67 76 39 71 12 53 43 59 26 16 21 49 51

Event 2 channels:

67 50 66 16 49 51 73 8 57 45 38 63 69 61 47 65 42 39 18 13 50 74 60 56 40 36 6 9 56
70 46 2 48 44 52 64 72 68 62 38 58 66 4 54 33 59 72 18 51 31 14 40

Event 3 channels:

61 59 55 51 31 40 4 62 2 10 20 43 6 16 7 66 30 22 32 60 27 11 57 26 28 12 8 34 47 67
```

### 8.2.3 Set 3

Time parameters:

| | |
|---|---|
| Sync packet: | T_SY = 26 |
| Frequency measurement: | T_FM = 80 µs |
| Guard Time: | T_GD = 10 µs |
| Time to hop: | T_FCS = 15 µs |
| Ramp down time: | T_RD = 5 µs |
| Interlude period 1: | T_IP1 = 10 µs |
| Interlude period 2: | T_IP2 = 10 µs |
| Phase measurement time: | T_PM = 10 µs |
| Number of antenna path: | N_AP = 1 |
| Antenna switch time: | T_SW = 0 µs |

---

| | |
|---|---|
| CSShapeSelection: | X |
| CSChannelJump: | 4 |
| CSNumRepetitions: | 2 |

*Sample Data*

```
----------------------------------------
Mode-0 duration:                177 µs
Main-mode (Mode-2) duration:    75 µs
Sub-Mode (Mode-3) duration:     147 µs
Minimum main mode steps:        2
Maximum main mode steps:        4
PHY:                            2 Mbps
RTT Type:                       AA-only
----------------------------------------
Event duration:                 5000 µs
Mode_0_Steps:                   3
Main_Mode_Repetition:           3

********************************** Channel Map ****************************************

Bit-map:           1F FF FF FF FF FF FC 7F FF FC

Filtered channels: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76


****************************** INSTANTIATION FUNCTION *********************************

h9() instantiation

*******************************************************************************************

Entropy input Peripheral (CS_IV_P):          E1 0B C2 8A 0B FD DF E9

Entropy input Central (CS_IV_C):             3E 7F 51 86 E0 CA 0B 3B

Entropy input (CS_IV):                       E1 0B C2 8A 0B FD DF E9 3E 7F 51 86 E0 CA 0B 3B

Nonce Peripheral (CS_IN_P):                  9F F4 77 C1

Nonce Central (CS_IN_C):                     86 73 84 0D

Nonce (CS_IN):                               9F F4 77 C1 86 73 84 0D

Personalization string Peripheral (CS_PV_P): C9 80 DE DF 98 82 ED 44

Personalization string Central (CS_PV_C):    64 A6 74 96 78 68 F1 43
```

*Sample Data*

```
Personalization string (CS_PV):              C9 80 DE DF 98 82 ED 44 64 A6 74 96 78 68 F1 43

********************************** INITIAL K and V ***********************************

K:    EE E0 4D 7C 76 11 3A 5C EC 99 2A E3 20 C2 4D 27

V:    DF 90 56 47 C1 06 6E 6F 52 C0 3E DF B8 2B 69 28

********************************************************************************************

Step=0 | Mode=0

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 0;

New DRBG octets: FF BC C1 CA 39 A6 9D C4 07 38 EF 33 D9 D1 35 32

************************************************************

T_ID 1 - mode0 channel (8 bits):  FF  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  BC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  CA  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  39  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 1 - mode0 channel (8 bits):  A6  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  9D  -  Range = 8 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  C4  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  07  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 1 - mode0 channel (8 bits):  38  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 12 (decimal); hr1 = 11 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  33  -  Range = 13 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (8 bits):  D9  -  Range = 14 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  D1  -  Range = 15 (decimal); hr1 = 12 (decimal)

T_ID 1 - mode0 channel (8 bits):  35  -  Range = 16 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  32  -  Range = 17 (decimal); hr1 = 3 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 1;

New DRBG octets: 51 EE 4B B0 3E E6 D0 A1 71 87 1C 2E 60 46 8F 6C

************************************************************

T_ID 1 - mode0 channel (8 bits):  51  -  Range = 18 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EE  -  Range = 19 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  4B  -  Range = 20 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  B0  -  Range = 21 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  3E  -  Range = 22 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  E6  -  Range = 23 (decimal); hr1 = 20 (decimal)

T_ID 1 - mode0 channel (8 bits):  D0  -  Range = 24 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  A1  -  Range = 25 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8 bits):  71  -  Range = 26 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  87  -  Range = 27 (decimal); hr1 = 14 (decimal)

T_ID 1 - mode0 channel (8 bits):  1C  -  Range = 28 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2E  -  Range = 29 (decimal); hr1 = 5 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  60  -  Range = 30 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  46  -  Range = 31 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 1 - mode0 channel (8 bits):  6C  -  Range = 33 (decimal); hr1 = 13 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 2;

New DRBG octets: 7D 75 2F 17 23 57 34 22 EF C7 CB C2 1F 18 90 8F

************************************************************

T_ID 1 - mode0 channel (8 bits):  7D  -  Range = 34 (decimal); hr1 = 16 (decimal)

T_ID 1 - mode0 channel (8 bits):  75  -  Range = 35 (decimal); hr1 = 15 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 36 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 37 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  23  -  Range = 38 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  57  -  Range = 39 (decimal); hr1 = 13 (decimal)

T_ID 1 - mode0 channel (8 bits):  34  -  Range = 40 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  22  -  Range = 41 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  EF  -  Range = 42 (decimal); hr1 = 39 (decimal)

T_ID 1 - mode0 channel (8 bits):  C7  -  Range = 43 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8 bits):  CB  -  Range = 44 (decimal); hr1 = 34 (decimal)

T_ID 1 - mode0 channel (16 bits): C2 1F  -  Range = 45 (decimal); hr1 = 5 (decimal)

T_ID 1 - mode0 channel (8 bits):  18  -  Range = 46 (decimal); hr1 = 4 (decimal)

T_ID 1 - mode0 channel (8 bits):  90  -  Range = 47 (decimal); hr1 = 26 (decimal)
```

*Sample Data*

```
T_ID 1 - mode0 channel (8 bits):  8F  -  Range = 48 (decimal); hr1 = 26 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 3;

New DRBG octets: 64 74 2B E9 80 0D 87 22 36 1E 6F 55 61 A8 7C 17

************************************************************

T_ID 1 - mode0 channel (8 bits):  64  -  Range = 49 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  74  -  Range = 50 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (8 bits):  2B  -  Range = 51 (decimal); hr1 = 8 (decimal)

T_ID 1 - mode0 channel (8 bits):  E9  -  Range = 52 (decimal); hr1 = 47 (decimal)

T_ID 1 - mode0 channel (8 bits):  80  -  Range = 53 (decimal); hr1 = 26 (decimal)

T_ID 1 - mode0 channel (8 bits):  0D  -  Range = 54 (decimal); hr1 = 2 (decimal)

T_ID 1 - mode0 channel (16 bits): 87 22  -  Range = 55 (decimal); hr1 = 7 (decimal)

T_ID 1 - mode0 channel (8 bits):  36  -  Range = 56 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  1E  -  Range = 57 (decimal); hr1 = 6 (decimal)

T_ID 1 - mode0 channel (8 bits):  6F  -  Range = 58 (decimal); hr1 = 25 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 59 (decimal); hr1 = 19 (decimal)

T_ID 1 - mode0 channel (8 bits):  61  -  Range = 60 (decimal); hr1 = 22 (decimal)

T_ID 1 - mode0 channel (16 bits): A8 7C  -  Range = 61 (decimal); hr1 = 29 (decimal)

T_ID 1 - mode0 channel (8 bits):  17  -  Range = 62 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 1; Transaction Counter = 4;
```

*Sample Data*

```
New DRBG octets: 0F 2F 55 C1 9B BD 7C 71 EC 79 0A 97 FD 0D 93 69

************************************************************

T_ID 1 - mode0 channel (8 bits):  0F  -  Range = 63 (decimal); hr1 = 3 (decimal)

T_ID 1 - mode0 channel (8 bits):  2F  -  Range = 64 (decimal); hr1 = 11 (decimal)

T_ID 1 - mode0 channel (8 bits):  55  -  Range = 65 (decimal); hr1 = 21 (decimal)

T_ID 1 - mode0 channel (8 bits):  C1  -  Range = 66 (decimal); hr1 = 49 (decimal)

T_ID 1 - mode0 channel (8 bits):  9B  -  Range = 67 (decimal); hr1 = 40 (decimal)

T_ID 1 - mode0 channel (8 bits):  BD  -  Range = 68 (decimal); hr1 = 50 (decimal)

T_ID 1 - mode0 channel (8 bits):  7C  -  Range = 69 (decimal); hr1 = 33 (decimal)

T_ID 1 - mode0 channel (8 bits):  71  -  Range = 70 (decimal); hr1 = 30 (decimal)

T_ID 1 - mode0 channel (8 bits):  EC  -  Range = 71 (decimal); hr1 = 65 (decimal)

T_ID 1 - mode0 channel (16 bits): 79 0A  -  Range = 72 (decimal); hr1 = 2 (decimal)

Mode0ShuffledChannelArray:

11 7 76 67 50 66 61 59 55 2 4 68 16 43 31 39 38 36 3 63 27 69 64 19 6 62 57 18 26 65
74 20 13 73 48 10 32 33 37 46 71 35 17 29 45 9 22 56 28 70 72 51 52 14 8 34 40 15 53
54 30 49 41 60 21 75 42 44 47 5 12 58

Jitter selection:

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 0;

New DRBG octets: 79 74 1F D1 8F 57 7B 45 D0 9A 66 5A 7F 1F 28 58

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  79  -  Range = 4 (decimal); hr1 = 1 (decimal)

ShapeChSeq :
```

*Sample Data*

```
1 75 5 71 9 67 13 63 17 59 21 55 25 51 29 47 33 43 37 39 41 35 45 31 49 27 53 23 57
19 61 15 65 11 69 7 73 3 77

firstAndEndAllChSeq:

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  57  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 7 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  45  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 9A 66  - Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 12 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 13 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 14 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 15 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 1;

New DRBG octets: D1 C1 D0 5A 40 B4 C4 81 EF BB 39 B2 61 D2 9C 4E
```

*Sample Data*

```
************************************************************

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 16 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C1  -  Range = 17 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 18 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 19 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 40 B4  - Range = 20 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 21 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 22 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EF  -  Range = 23 (decimal); hr1 = 21 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BB  -  Range = 24 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 25 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): B2 61  - Range = 26 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D2  -  Range = 27 (decimal); hr1 = 22 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9C  -  Range = 28 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4E  -  Range = 29 (decimal); hr1 = 8 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 2;

New DRBG octets: 71 FD E8 68 E8 CA CA D1 18 E5 9B 18 5C EE FC 17

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  71  -  Range = 30 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FD  -  Range = 31 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 32 (decimal); hr1 = 29 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 33 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 34 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 35 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 36 (decimal); hr1 = 28 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 37 (decimal); hr1 = 30 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 38 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E5  -  Range = 39 (decimal); hr1 = 34 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9B  -  Range = 40 (decimal); hr1 = 24 (decimal)

firstAndEndAllChSeq Shuffled:

22 32 33 57 29 44 38 28 48 45 26 41 36 52 39 27 40 47 23 37 20 42 46 31 59 21 24 54
55 51 56 35 49 50 58 25 53 30 43 34

middleAllChSeq:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78

T_ID 0 - non-mode0 channel (8 bits):  18  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EE  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FC  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 6 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 3;

New DRBG octets: B5 0E CB BF 12 99 7C 4B 02 63 35 D0 81 C5 6B D0
```

*Sample Data*

```
************************************************************

T_ID 0 - non-mode0 channel (8 bits):  B5  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0E  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BF  -  Range = 10 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  12  -  Range = 11 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  99  -  Range = 12 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7C  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 14 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 15 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  63  -  Range = 16 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  35  -  Range = 17 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 18 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 19 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 20 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6B  -  Range = 21 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 22 (decimal); hr1 = 17 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 4;

New DRBG octets: C3 3C B1 23 C6 A2 E0 8A BE 0E 8A BD 22 CB 24 1F

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 23 (decimal); hr1 = 17 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  3C  -  Range = 24 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B1  -  Range = 25 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 26 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C6  -  Range = 27 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A2  -  Range = 28 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E0  -  Range = 29 (decimal); hr1 = 25 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 30 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (16 bits): BE 0E  - Range = 31 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 32 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BD  -  Range = 33 (decimal); hr1 = 24 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  22  -  Range = 34 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 35 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  24  -  Range = 36 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 37 (decimal); hr1 = 4 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 5;

New DRBG octets: 68 87 A9 32 7B E1 61 B4 CD 1E 92 4D AC 00 74 05

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 38 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  87  -  Range = 39 (decimal); hr1 = 20 (decimal)

middleAllChSeq Shuffled:
```

*Sample Data*

```
14 70 0 65 76 75 15 11 60 18 7 9 4 6 17 77 69 71 8 12 78 10 61 1 72 68 5 74 16 3 2
67 62 13 64 63 73 19 66

firstAndEndUnusedChSeq:

20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  32  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7B  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E1  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B4  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CD  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 9 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  92  -  Range = 10 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4D  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AC  -  Range = 12 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 00 74  - Range = 13 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  05  -  Range = 14 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 6;

New DRBG octets: 2B BC 5A 8F FB 1E 39 AA 02 8F F4 4C 0A 11 7D F9

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  2B  -  Range = 15 (decimal); hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 16 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5A  -  Range = 17 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 18 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FB  -  Range = 19 (decimal); hr1 = 18 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 20 (decimal); hr1 = 2 (decimal)

firstAndEndUnusedChSeq Shuffled:

46 36 58 40 32 52 34 28 42 20 54 50 38 24 30 26 44 22 56 48

middleUnusedChSeq:

0 2 4 6 8 10 12 14 16 18 60 62 64 66 68 70 72 74 76 78

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AA  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F4  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4C  -  Range = 7 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  11  -  Range = 9 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 11 (decimal); hr1 = 10 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 7;

New DRBG octets: 58 A1 10 C9 6F 8A 88 43 76 61 17 07 D7 61 DF A3
```

*Sample Data*

```
************************************************************

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 12 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A1  -  Range = 13 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 14 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C9  -  Range = 15 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6F  -  Range = 16 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 17 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  88  -  Range = 18 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  43  -  Range = 19 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  76  -  Range = 20 (decimal); hr1 = 9 (decimal)

middleUnusedChSeq Shuffled:

66 4 12 2 76 10 70 6 64 78 60 68 14 16 18 8 0 72 62 74

FirstAndEndSaltChSeq:

46 36 58 40 32 52 34 28 42 20 54 50 38 24 30 26 44 22 56 48 22 32 33 57 29 44 38 28
48 45 26 41 36 52 39 27 40 47 23 37 20 42 46 31 59 21 24 54 55 51 56 35 49 50 58 25
53 30 43 34

MiddleSaltChSeq:

66 4 12 2 76 10 70 6 64 78 60 68 14 16 18 8 0 72 62 74 14 70 0 65 76 75 15 11 60 18
7 9 4 6 17 77 69 71 8 12 78 10 61 1 72 68 5 74 16 3 2 67 62 13 64 63 73 19 66

Random number for initial and end salting

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 10 (decimal); hr1 = 3 (decimal)

SaltedChSeq:

46 66 36 58 1 75 40 5 71 32 9 67 52 13 63 34 17 59 4 21 55 12 25 51 2 29 47 76 33 43
```

*Sample Data*

```
10 37 39 70 41 35 6 45 31 64 49 27 78 53 23 60 57 19 28 61 15 42 65 11 20 69 7 54 73
3 50 77 68

tempBlockSeq:

46 66 36 58 75 40 5 71 32 9 67 52 13 63

T_ID 0 - non-mode0 channel (8 bits):  17  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  07  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D7  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DF  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 7 (decimal); hr1 = 4 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 8;

New DRBG octets: 53 23 37 62 4B C5 D9 66 1A C4 34 BC 41 36 C0 FE

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  53  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 9 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  37  -  Range = 10 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  62  -  Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 12 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (16 bits): C5 D9  - Range = 13 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 14 (decimal); hr1 = 5 (decimal)

tempBlockSeq Shuffled:
```

*Sample Data*

```
36 32 9 52 67 63 46 66 75 71 5 13 58 40

tempBlockSeq:

34 17 59 4 21 55 12 51 2 29 47 76 33 43

T_ID 0 - non-mode0 channel (8 bits):  1A  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  34  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BC  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  41  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  36  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C0  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FE  -  Range = 9 (decimal); hr1 = 8 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 9;

New DRBG octets: 28 BE 39 8A 7D 3F DA 86 15 0A 83 3E 4B 8B 3D D9

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  28  -  Range = 10 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BE  -  Range = 11 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 12 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 8A 7D  - Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3F  -  Range = 14 (decimal); hr1 = 3 (decimal)

tempBlockSeq Shuffled:

4 29 76 43 17 34 33 55 47 12 2 59 51 21
```

*Sample Data*

```
tempBlockSeq:

10 37 39 70 41 35 6 45 31 64 49 27 53 60

T_ID 0 - non-mode0 channel (8 bits):  DA  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  15  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  83  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3E  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4B  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 9 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3D  -  Range = 10 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 11 (decimal); hr1 = 9 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 10;

New DRBG octets: 86 AD C3 21 02 CF 55 74 D9 58 09 10 16 F9 64 97

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  86  -  Range = 12 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AD  -  Range = 13 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C3  -  Range = 14 (decimal); hr1 = 10 (decimal)

tempBlockSeq Shuffled:

41 6 64 35 31 10 27 37 53 49 60 39 70 45
```

*Sample Data*

```
tempBlockSeq:

57 19 28 61 15 42 65 11 20 69 7 54 73 3 50 68

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  02  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CF  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  55  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 6 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 7 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  58  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  09  -  Range = 9 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  10  -  Range = 10 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  16  -  Range = 11 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  64  -  Range = 13 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  97  -  Range = 14 (decimal); hr1 = 8 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 11;

New DRBG octets: A9 B9 82 8A 7F 2A EC D9 59 D1 74 8B C0 31 4A 9F

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  A9  -  Range = 15 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B9  -  Range = 16 (decimal); hr1 = 11 (decimal)

tempBlockSeq Shuffled:
```

*Sample Data*

```
7 15 11 61 57 73 19 42 3 50 69 68 65 28 20 54

ShapeChSeq :

2 76 6 72 10 68 14 64 18 60 22 56 26 52 30 48 34 44 38 40 42 36 46 32 50 28 54 24
58 20 62 16 66 12 70 8 74 4 78 0

firstAndEndAllChSeq:

20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55 56 57 58 59

T_ID 0 - non-mode0 channel (8 bits):  82  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8A  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7F  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2A  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EC  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 7 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 8 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D1  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 10 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8B  -  Range = 11 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (16 bits): C0 31  - Range = 12 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  4A  -  Range = 13 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9F  -  Range = 14 (decimal); hr1 = 8 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 12;
```

*Sample Data*

```
New DRBG octets: 59 E0 5C 66 A0 CE 21 CB 23 81 24 96 31 01 37 50

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  59  -  Range = 15 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E0  -  Range = 16 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  5C  -  Range = 17 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 18 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A0  -  Range = 19 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CE  -  Range = 20 (decimal); hr1 = 16 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 21 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CB  -  Range = 22 (decimal); hr1 = 17 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 23 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 24 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  24  -  Range = 25 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  96  -  Range = 26 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  31  -  Range = 27 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  01  -  Range = 28 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  37  -  Range = 29 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  50  -  Range = 30 (decimal); hr1 = 9 (decimal)

**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 13;

New DRBG octets: 4E 1E D6 DF C2 11 39 08 66 A7 0E 2F 60 6F 16 B4

**************************************************************
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  4E  -  Range = 31 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1E  -  Range = 32 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D6  -  Range = 33 (decimal); hr1 = 27 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DF  -  Range = 34 (decimal); hr1 = 29 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C2  -  Range = 35 (decimal); hr1 = 26 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  11  -  Range = 36 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 37 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  08  -  Range = 38 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  66  -  Range = 39 (decimal); hr1 = 15 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A7  -  Range = 40 (decimal); hr1 = 26 (decimal)

firstAndEndAllChSeq Shuffled:

47 57 55 51 29 46 48 37 56 50 26 38 43 21 35 58 39 41 27 25 31 28 32 22 42 30 59 52
36 53 49 44 24 20 34 40 33 23 45 54

middleAllChSeq:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 60 61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78

T_ID 0 - non-mode0 channel (8 bits):  0E  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2F  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  60  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6F  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  16  -  Range = 6 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B4  -  Range = 7 (decimal); hr1 = 4 (decimal)
```

*Sample Data*

```
************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 14;

New DRBG octets: A7 74 D3 DB 61 79 2A C2 D0 3E 3A 03 AD 20 ED 05

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  A7  -  Range = 8 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 9 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D3  -  Range = 10 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DB  -  Range = 11 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 12 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  79  -  Range = 13 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2A  -  Range = 14 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C2  -  Range = 15 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D0  -  Range = 16 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3E  -  Range = 17 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3A  -  Range = 18 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  03  -  Range = 19 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AD  -  Range = 20 (decimal); hr1 = 13 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  20  -  Range = 21 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  ED  -  Range = 22 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  05  -  Range = 23 (decimal); hr1 = 0 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 15;
```

*Sample Data*

```
New DRBG octets: 74 CA 60 71 81 08 B1 2E D3 2B 96 EE 27 A3 0C 45

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  74  -  Range = 24 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  CA  -  Range = 25 (decimal); hr1 = 19 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  60  -  Range = 26 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  71  -  Range = 27 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 28 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  08  -  Range = 29 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B1  -  Range = 30 (decimal); hr1 = 20 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2E  -  Range = 31 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D3  -  Range = 32 (decimal); hr1 = 26 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2B  -  Range = 33 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  96  -  Range = 34 (decimal); hr1 = 19 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  EE  -  Range = 35 (decimal); hr1 = 32 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  27  -  Range = 36 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 37 (decimal); hr1 = 23 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  0C  -  Range = 38 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  45  -  Range = 39 (decimal); hr1 = 10 (decimal)

middleAllChSeq Shuffled:

68 77 60 0 17 75 12 2 9 65 78 66 1 19 67 4 11 16 5 73 69 13 18 76 15 10 71 8 62 61
7 14 74 64 70 72 6 3 63

firstAndEndUnusedChSeq:
```

*Sample Data*

```
21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 16;

New DRBG octets: 7D 54 12 20 C7 3C 36 B3 9D 7E 19 E5 E3 7E 8D 98

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  7D  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  54  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  12  -  Range = 4 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  20  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C7  -  Range = 6 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3C  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  36  -  Range = 8 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B3  -  Range = 9 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  9D  -  Range = 10 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7E  -  Range = 11 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  19  -  Range = 12 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E5  -  Range = 13 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E3  -  Range = 14 (decimal); hr1 = 12 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  7E  -  Range = 15 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8D  -  Range = 16 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  98  -  Range = 17 (decimal); hr1 = 10 (decimal)
```

*Sample Data*

```
************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 17;

New DRBG octets: 0A 22 C0 85 DC 55 47 8E DA 06 D5 D4 B2 61 3E 70

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 18 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  22  -  Range = 19 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (16 bits): C0 85  - Range = 20 (decimal); hr1 = 10 (decimal)

firstAndEndUnusedChSeq Shuffled:

55 43 57 25 31 41 39 49 51 37 59 45 47 35 33 21 27 29 23 53

middleUnusedChSeq:

1 3 5 7 9 11 13 15 17 19 61 63 65 67 69 71 73 75 77

T_ID 0 - non-mode0 channel (8 bits):  DC  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  55  -   Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  47  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  8E  -  Range = 5 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  DA  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  06  -  Range = 7 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D5  -  Range = 8 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D4  -  Range = 9 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B2  -  Range = 10 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3E  -  Range = 12 (decimal); hr1 = 2 (decimal)
```

*Sample Data*

```
T_ID 0 - non-mode0 channel (8 bits):  70  -  Range = 13 (decimal); hr1 = 5 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 18;

New DRBG octets: 0A 71 2E 1F A3 C1 D8 DE 81 C1 AB A0 29 F7 3C B2

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  0A  -  Range = 14 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  71  -  Range = 15 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  2E  -  Range = 16 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  1F  -  Range = 17 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 18 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C1  -  Range = 19 (decimal); hr1 = 14 (decimal)

middleUnusedChSeq Shuffled:

67 7 73 3 61 65 69 17 5 15 1 75 11 13 77 63 71 9 19

FirstAndEndSaltChSeq:

55 43 57 25 31 41 39 49 51 37 59 45 47 35 33 21 27 29 23 53 47 57 55 51 29 46 48 37
56 50 26 38 43 21 35 58 39 41 27 25 31 28 32 22 42 30 59 52 36 53 49 44 24 20 34 40
33 23 45 54

MiddleSaltChSeq:

67 7 73 3 61 65 69 17 5 15 1 75 11 13 77 63 71 9 19 68 77 60 0 17 75 12 2 9 65 78 66
1 19 67 4 11 16 5 73 69 13 18 76 15 10 71 8 62 61 7 14 74 64 70 72 6 3 63

Random number for initial and end salting

T_ID 0 - non-mode0 channel (8 bits):  D8  -  Range = 5 (decimal); hr1 = 4 (decimal)

SaltedChSeq:
```

*Sample Data*

```
55 2 76 43 6 72 57 10 68 25 14 64 31 18 60 67 22 56 7 26 52 73 30 48 3 34 44 61 38
40 65 42 36 69 46 32 17 50 28 5 54 24 15 58 20 41 62 16 39 66 12 49 70 8 51 74 4 37
78 0 59 1 45 75

tempBlockSeq:

55 2 76 43 6 72 57 10 68 14 64 31 18 60

T_ID 0 - non-mode0 channel (8 bits):  DE  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  81  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C1  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AB  -  Range = 5 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A0  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  29  -  Range = 7 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F7  -  Range = 8 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3C  -  Range = 9 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B2  -  Range = 10 (decimal); hr1 = 6 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 19;

New DRBG octets: 40 F9 D7 29 EB B9 61 15 45 AD 05 00 0B 40 68 C5

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  40  -  Range = 11 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 12 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D7  -  Range = 13 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  29  -  Range = 14 (decimal); hr1 = 2 (decimal)
```

*Sample Data*

```
tempBlockSeq Shuffled:

55 57 60 72 43 6 14 10 2 76 18 31 68 64

tempBlockSeq:

67 22 56 7 26 52 73 30 48 3 34 44 61 38

T_ID 0 - non-mode0 channel (8 bits):  EB  -  Range = 2 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  B9  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  61  -  Range = 4 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  15  -  Range = 5 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  45  -  Range = 6 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  AD  -  Range = 7 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  05  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (16 bits): 00 0B  - Range = 9 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  40  -  Range = 10 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  68  -  Range = 11 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C5  -  Range = 12 (decimal); hr1 = 9 (decimal)

************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 20;

New DRBG octets: 60 85 51 F9 BA 39 98 FD 14 83 F9 C4 E8 E7 3D 4D

************************************************************

T_ID 0 - non-mode0 channel (8 bits):  60  -  Range = 13 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  85  -  Range = 14 (decimal); hr1 = 7 (decimal)

tempBlockSeq Shuffled:
```

*Sample Data*

```
48 52 3 22 61 7 67 38 30 44 73 56 34 26

tempBlockSeq:

40 65 42 36 69 46 32 17 50 28 5 54 15 58

T_ID 0 - non-mode0 channel (8 bits):  51  -  Range = 2 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  BA  -  Range = 4 (decimal); hr1 = 2 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  39  -  Range = 5 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  98  -  Range = 6 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  FD  -  Range = 7 (decimal); hr1 = 6 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  14  -  Range = 8 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  83  -  Range = 9 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F9  -  Range = 10 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  C4  -  Range = 11 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E8  -  Range = 12 (decimal); hr1 = 10 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E7  -  Range = 13 (decimal); hr1 = 11 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3D  -  Range = 14 (decimal); hr1 = 3 (decimal)

tempBlockSeq Shuffled:

17 69 36 58 50 42 32 65 5 28 54 15 40 46

tempBlockSeq:

20 41 62 16 39 66 12 49 70 8 51 74 4 37 59 45 75

T_ID 0 - non-mode0 channel (8 bits):  4D  -  Range = 2 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
**************************************************************

Step Counter = 0; Transaction ID = 0; Transaction Counter = 21;

New DRBG octets: 90 D4 D7 D9 21 A3 F5 F1 47 99 25 6A 3F E6 23 A6

**************************************************************

T_ID 0 - non-mode0 channel (8 bits):  90  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D4  -  Range = 4 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D7  -  Range = 5 (decimal); hr1 = 4 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  D9  -  Range = 6 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  21  -  Range = 7 (decimal); hr1 = 0 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  A3  -  Range = 8 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F5  -  Range = 9 (decimal); hr1 = 8 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  F1  -  Range = 10 (decimal); hr1 = 9 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  47  -  Range = 11 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  99  -  Range = 12 (decimal); hr1 = 7 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  25  -  Range = 13 (decimal); hr1 = 1 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  6A  -  Range = 14 (decimal); hr1 = 5 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  3F  -  Range = 15 (decimal); hr1 = 3 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  E6  -  Range = 16 (decimal); hr1 = 14 (decimal)

T_ID 0 - non-mode0 channel (8 bits):  23  -  Range = 17 (decimal); hr1 = 2 (decimal)

tempBlockSeq Shuffled:

12 4 75 59 39 37 41 74 70 8 16 66 62 49 45 51 20

NonMode0ShuffledChannelArray:
```

*Sample Data*

```
36 32 9 52 67 63 46 66 75 71 5 13 58 40 4 29 76 43 17 34 33 55 47 12 2 59 51 21 41
6 64 35 31 10 27 37 53 49 60 39 70 45 7 15 11 61 57 73 19 42 3 50 69 68 65 28 20 54
55 57 60 72 43 6 14 10 2 76 18 31 68 64 48 52 3 22 61 7 67 38 30 44 73 56 34 26 17
69 36 58 50 42 32 65 5 28 54 15 40 46 12 4 75 59 39 37 41 74 70 8 16 66 62 49 45 51
20

**************************************************************

Step Counter = 3; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: F8 82 2A 54 E0 7C 50 15 57 CD 98 51 69 D4 AC AB

**************************************************************

T_ID 2 - Submode insertion (8 bits):  F8  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  82  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  2A  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  54  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  E0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  7C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  50  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  15  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  57  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  CD  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  98  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  51  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  69  -  Range = 3 (decimal); hr1 = 1 (decimal)

Event 1 is full. step_ctr = 51, seq_ctr = 48, time = 4995
```

*Sample Data*

```
T_ID 2 - Submode insertion (8 bits):  D4  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  AB  -  Range = 3 (decimal); hr1 = 2 (decimal)

************************************************************

Step Counter = 74; Transaction ID = 2; Transaction Counter = 0;

New DRBG octets: 52 ED C6 80 FC 49 2C 49 97 71 6C 16 C0 27 FB 31

************************************************************

T_ID 2 - Submode insertion (8 bits):  52  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  ED  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  C6  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  80  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  FC  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  49  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  2C  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  49  -  Range = 3 (decimal); hr1 = 0 (decimal)

Event 2 is full. step_ctr = 52, seq_ctr = 94, time = 4998

T_ID 2 - Submode insertion (8 bits):  97  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  71  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  6C  -  Range = 3 (decimal); hr1 = 1 (decimal)

T_ID 2 - Submode insertion (8 bits):  16  -  Range = 3 (decimal); hr1 = 0 (decimal)

T_ID 2 - Submode insertion (8 bits):  C0  -  Range = 3 (decimal); hr1 = 2 (decimal)

T_ID 2 - Submode insertion (8 bits):  27  -  Range = 3 (decimal); hr1 = 0 (decimal)
```

*Sample Data*

```
steps per event: 29, time = 2913

Event 1 channels:

11 7 76 36 32 9 52 67 63 46 66 75 71 5 13 58 40 4 29 76 43 17 34 33 55 47 12 2 59
51 21 41 6 64 35 31 10 27 37 53 49 60 39 70 45 7 15 11 61 57 73

Event 2 channels:

67 50 66 11 57 73 19 42 3 50 69 68 65 28 20 54 55 57 60 72 43 6 14 10 2 76 18 31 68
64 48 52 3 22 61 7 67 38 30 44 73 56 34 26 17 69 36 58 50 42 32 65

Event 3 channels:

61 59 55 50 42 65 5 28 54 15 40 46 12 4 75 59 39 37 41 74 70 8 16 66 62 49 45 51 20
```

Low Energy Controller
Part D

# MESSAGE SEQUENCE CHARTS

*Examples of message sequence charts showing the interactions of the Host Controller Interface with the Link Layer.*

# CONTENTS

**1 Introduction ..... 3590**
1.1 Notation ..... 3590
1.2 Control flow ..... 3590
1.3 Example MSC ..... 3591
1.4 Forward compatibility ..... 3591

**2 Standby state ..... 3593**
2.1 Initial setup ..... 3593
2.2 Random Device address ..... 3595
2.3 Filter Accept List ..... 3595
2.4 Adding IRK to resolving list ..... 3596
2.5 Default data length ..... 3596
2.6 Periodic Advertiser List ..... 3597

**3 Advertising state ..... 3598**
3.1 Undirected advertising ..... 3598
3.2 Directed advertising ..... 3599
3.3 Advertising using ADV_EXT_IND ..... 3601
3.4 Scan request notifications ..... 3602
3.5 Advertising duration ended ..... 3603
3.6 Periodic advertising ..... 3604
3.7 Connectionless Constant Tone Extension transmission ..... 3605
3.8 Isochronous Broadcasting State ..... 3606
3.8.1 Create a Broadcast Isochronous Group ..... 3606
3.8.2 Terminate a Broadcast Isochronous Group ..... 3607
3.9 Periodic advertising with responses (PAwR) ..... 3607
3.10 Transmitting PAwR subevents ..... 3608
3.11 Using a response slot in PAwR ..... 3608
3.12 Connecting from PAwR ..... 3610
3.13 Failed connection attempts from PAwR ..... 3610

**4 Scanning state ..... 3612**
4.1 Passive scanning ..... 3612
4.2 Active scanning ..... 3613
4.3 Passive scanning for directed advertisements with Privacy ..... 3614
4.4 Active scanning with Privacy ..... 3615
4.5 Active scanning with Privacy and Controller based resolvable private address generation ..... 3616
4.6 Active scanning on the secondary advertising Physical channel ..... 3617

4.7 Scan timeout ..... 3618
4.8 Scanning for periodic advertisements ..... 3619
4.9 Cancel scanning for periodic advertisements ..... 3620
4.10 Periodic advertising synchronization timeout ..... 3621
4.11 Terminate reception of periodic advertising ..... 3622
4.12 Connectionless Constant Tone Extension reception ..... 3623
4.13 Synchronization with separate enable of reports ..... 3624

**5 Initiating state ..... 3625**
5.1 Initiating a connection ..... 3625
5.2 Canceling an initiation ..... 3626
5.3 Initiating a connection using undirected advertising with Privacy ..... 3627
5.4 Initiating a connection using directed advertising with Privacy . 3628
5.5 Initiating a connection that fails to establish ..... 3629
5.6 Initiating a connection on the secondary advertising physical channel ..... 3630
5.7 Initiating a Channel Selection algorithm #2 connection ..... 3631
5.8 Initiating a connection using an advertising set ..... 3632

**6 Connection state ..... 3633**
6.1 Sending data ..... 3633
6.2 Connection update ..... 3634
6.3 Channel map update ..... 3634
6.4 Features exchange ..... 3635
6.5 Version exchange ..... 3639
6.6 Start encryption ..... 3641
6.7 Start encryption without Long Term Key ..... 3642
6.8 Start encryption with event masked ..... 3643
6.9 Start encryption without Peripheral supporting encryption ..... 3644
6.10 Restart encryption ..... 3645
6.11 Disconnect ..... 3646
6.12 Connection parameters request ..... 3647
6.13 LE Ping ..... 3652
6.14 Data length update ..... 3654
6.15 PHY update ..... 3655
6.16 Minimum number of used channels request ..... 3660
6.17 LL procedure collision ..... 3661
6.18 Constant Tone Extension Request ..... 3662
6.19 Connected Isochronous Group Setup ..... 3664
6.20 Host Rejects Connected Isochronous Stream ..... 3666
6.21 Link Layer Rejects Connected Isochronous Stream ..... 3668
6.22 Link Layer Rejects Connected Isochronous Stream ..... 3670

6.23 Host A Terminates Connected Isochronous Stream ..... 3670
6.24 ACL disconnected ..... 3672
6.25 Host A Removes Connected Isochronous Group ..... 3673
6.26 Request Sleep Clock Accuracy ..... 3675
6.27 Power Control ..... 3676
6.28 Data path setup for a music stream over a CIS ..... 3683
6.29 Data path setup for bi-directional voice over a CIS ..... 3685
6.30 [This section is no longer used] ..... 3686
6.31 Modifying the subrate of a connection ..... 3686
6.32 Channel Classification Enable ..... 3688
6.33 Channel Classification Reporting ..... 3689
6.34 Channel Sounding setup phase ..... 3690
6.35 Channel Sounding started by Central in initiator role ..... 3692
6.36 Channel Sounding started by Peripheral in reflector role ..... 3694
6.37 Channel Sounding started by Central, rejected by Peripheral .. 3696
6.38 Channel Sounding configuration removal during an active CS procedure ..... 3696
6.39 Frame Space Update ..... 3697
6.40 Modifying the Connection Rate ..... 3699

**7 Periodic advertising sync transfer ..... 3702**
7.1 Transfer by scanner, reports initially disabled ..... 3702
7.2 Transfer by scanner, reports initially enabled ..... 3703
7.3 Transfer by the advertiser ..... 3704

**8 Synchronization state ..... 3705**
8.1 Synchronizing with a Broadcast Isochronous Group ..... 3705
8.2 Terminate Synchronization with a BIG ..... 3706
8.3 New Channel Map for Broadcast Isochronous Group ..... 3706
8.4 Lost Synchronization with a Broadcast Isochronous Group ..... 3707
8.5 Data path setup for a BIS ..... 3707

# 1 INTRODUCTION

This section shows typical interactions between Host Controller interface (HCI) commands and events and the Link Layer (LL). It focuses on the message sequence charts (MSCs) for the procedures specified in “Bluetooth Host Controller Interface Functional Specification” with regard to Link Layer control procedures from “Link Layer”. This section illustrates only the most useful scenarios; it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or Host interface. In all message sequence charts it is assumed that all events are not masked, so the Controller will not filter out any events.

The sequence of messages in these message sequence charts is for illustrative purposes. The messages may be sent in a different order where allowed by the Link Layer or HCI. If any of these charts differ with text in the Link Layer or HCI Parts, the text in those Parts overrides these charts.

## 1.1 Notation

The notation used in the message sequence charts (MSCs) consists of ovals, elongated hexagons, boxes, lines, and arrows. The vertical lines terminated on the top by a shadow box and at the bottom by solid oval indicate a protocol entity that resides in a device. MSCs describe interactions between these entities and states those entities may be in.

The following symbols represent interactions and states:

| | |
|---|---|
| **Oval** | Defines the context for the message sequence chart |
| **Hexagon** | Indicates a condition needed to start the transactions below this hexagon. The location and width of the Hexagon indicates which entity or entities make this decision. |
| **Box** | Replaces a group of transactions. May indicate a user action, or a procedure in the Link Layer. |
| **Dashed Box** | Optional group of transactions. |
| **Solid Arrow** | Represents a message, signal or transaction. Can be used to show Link Layer and HCI traffic. |
| **Dashed Arrow** | Represents an optional message, signal or transaction. Can be used to show Link Layer and HCI traffic. |

## 1.2 Control flow

Some message sequences are split into several charts. In these charts, numbers indicate normal or required ordering and letters represent alternative paths. For example, Step 4 is after Step 3, and Step 5a could be executed instead of Step 5b.

## 1.3 Example MSC

The protocol entities represented in the example shown in Figure 1.1 illustrate the interactions of two devices named A and B; each device includes a Host and a LL entity in this example. Other MSCs in this section may show the interactions of more than two devices.

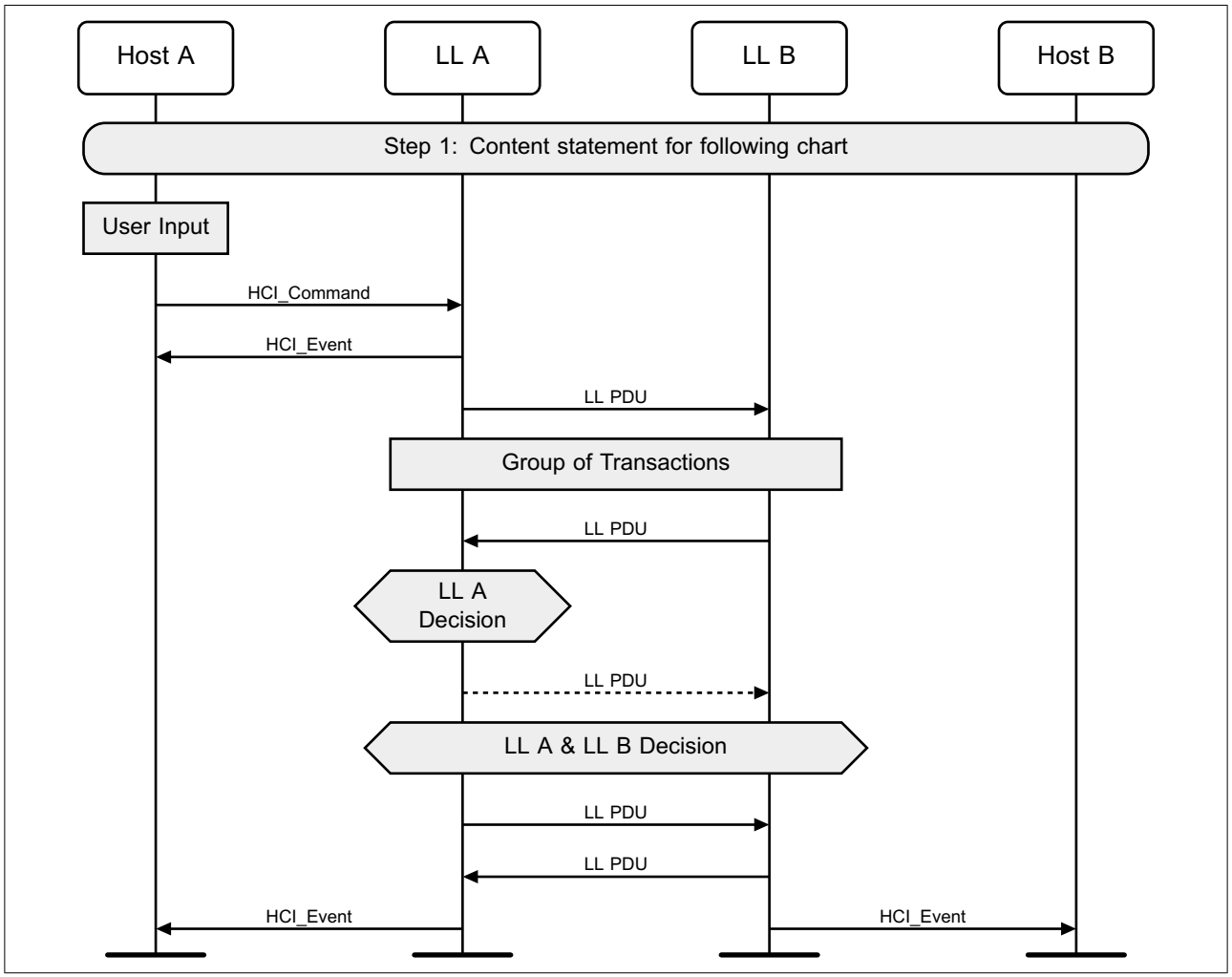



*Figure 1.1: Example MSC*

## 1.4 Forward compatibility

Many of the message sequences in this Part use HCI commands or events that have enhanced or extended variants that were added to the specification later than the relevant sequence. Such variants can be related commands or events with different names (e.g,. HCI_LE_Advertising_Report and HCI_LE_Extended_Advertising_Report events) or commands or events with multiple versions (e.g., HCI_LE_Generate_DHKey command). In some instances (for example, see [Vol 4] Part E, Section 3.1.1), a Host is

required to use the new variant rather than the one shown in the MSC. Even when this is not a requirement, Host implementers may prefer to use the newer variants.

In these circumstances, the MSCs have not been rewritten to use newer features but have been left unchanged. In general, the new commands and events will directly replace the old ones, but this is not always the case and readers should not assume it.

# 2 STANDBY STATE

## 2.1 Initial setup

To perform initial setup of a LE Controller, the following sequence of actions may be required.

First, the Host would wait for the Controller to indicate the number of HCI Command packets the Host is currently allowed to send using a Command Complete event on a No OPeration command opcode. Then it would reset the Controller to a known state. Then it needs to read the local supported features to check that low energy is supported on this Controller. It would then set the event mask and LE event mask to enable the events that it needs the Controller to generate to the Host. Next, it will check the buffers that are available for data flow, using the Read Buffer Size and LE Read Buffer Size commands. Then it would read the locally supported LE features and select the features

that it may use. Finally, it will read the public device address if the Controller has one (see Figure 2.1).

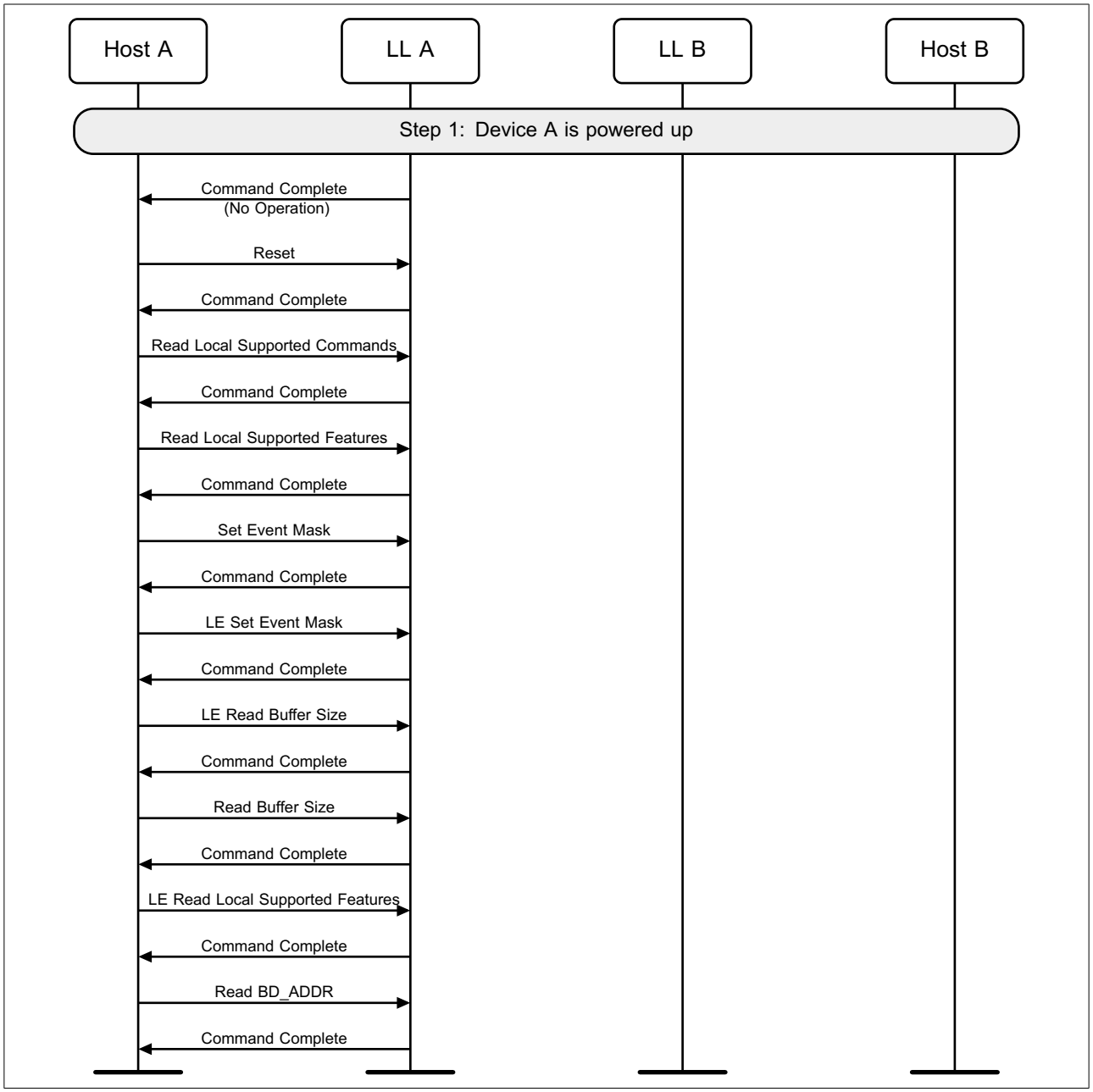



*Figure 2.1: Initial setup*

## 2.2 Random Device address

A device may use a random device address, but this address has to be configured before being used during advertising, scanning or initiating (see Figure 2.2).

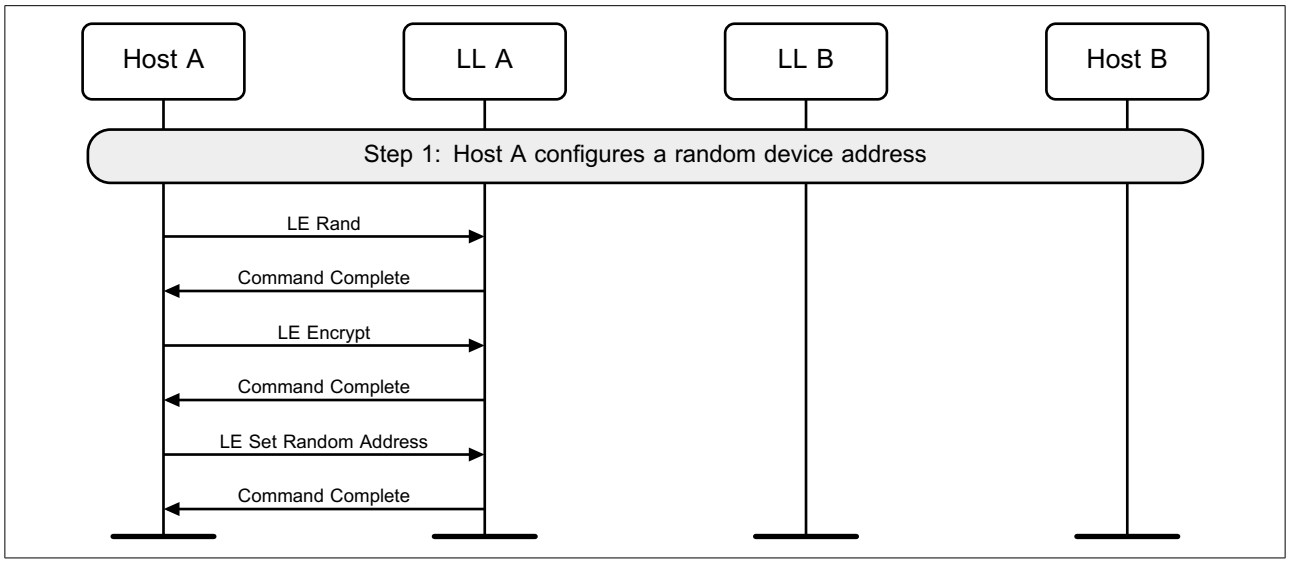



*Figure 2.2: Random device address*

## 2.3 Filter Accept List

Before advertising, scanning or initiating can use a Filter Accept List, the Filter Accept List may be cleared and devices added in as required (see Figure 2.3).

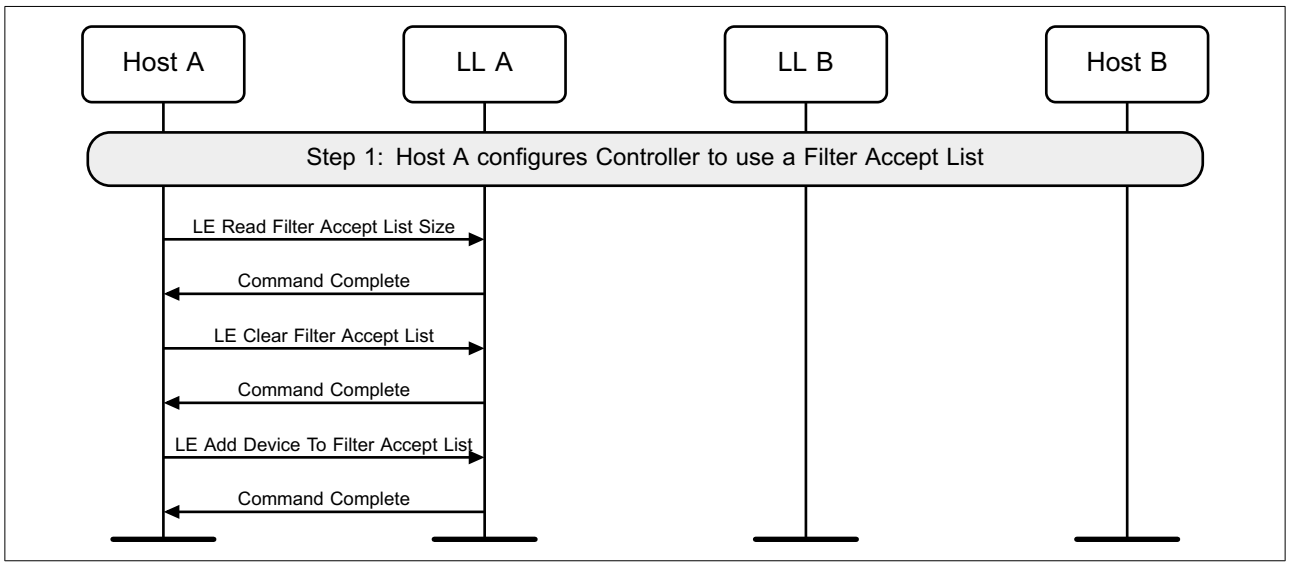



*Figure 2.3: Filter Accept List*

## 2.4 Adding IRK to resolving list

Before advertising, scanning or initiating can use resolving lists, the resolving list may be cleared and devices added in as required (see Figure 2.4).

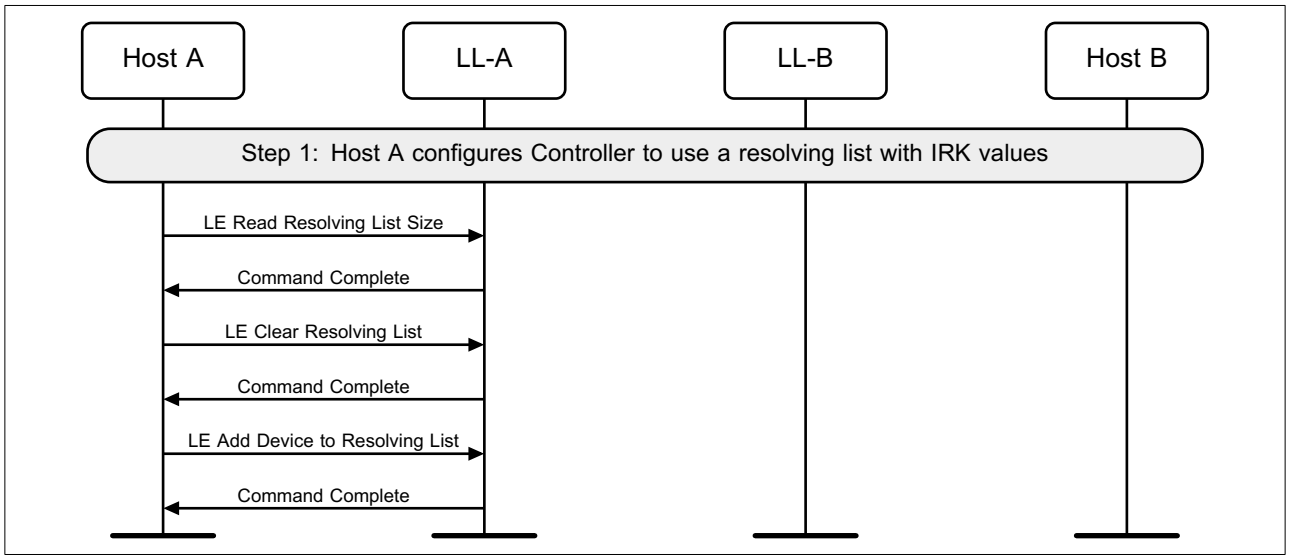



*Figure 2.4: Resolving lists*

## 2.5 Default data length

Before creating a connection, the Host may specify its preferred values for the Controller's maximum transmission packet size and maximum packet transmission time to be used for new connections. This may be done on either the Central or the Peripheral.

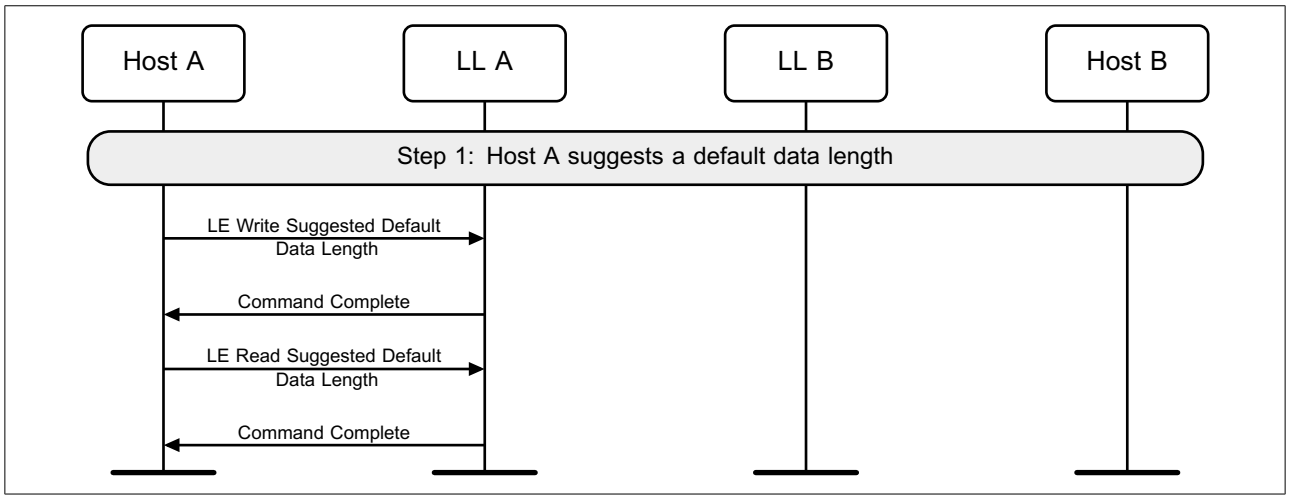



*Figure 2.5: Default data length*

## 2.6 Periodic Advertiser List

The Periodic Advertiser List may be cleared and entries added as required, before it is made use of (see Figure 2.6).

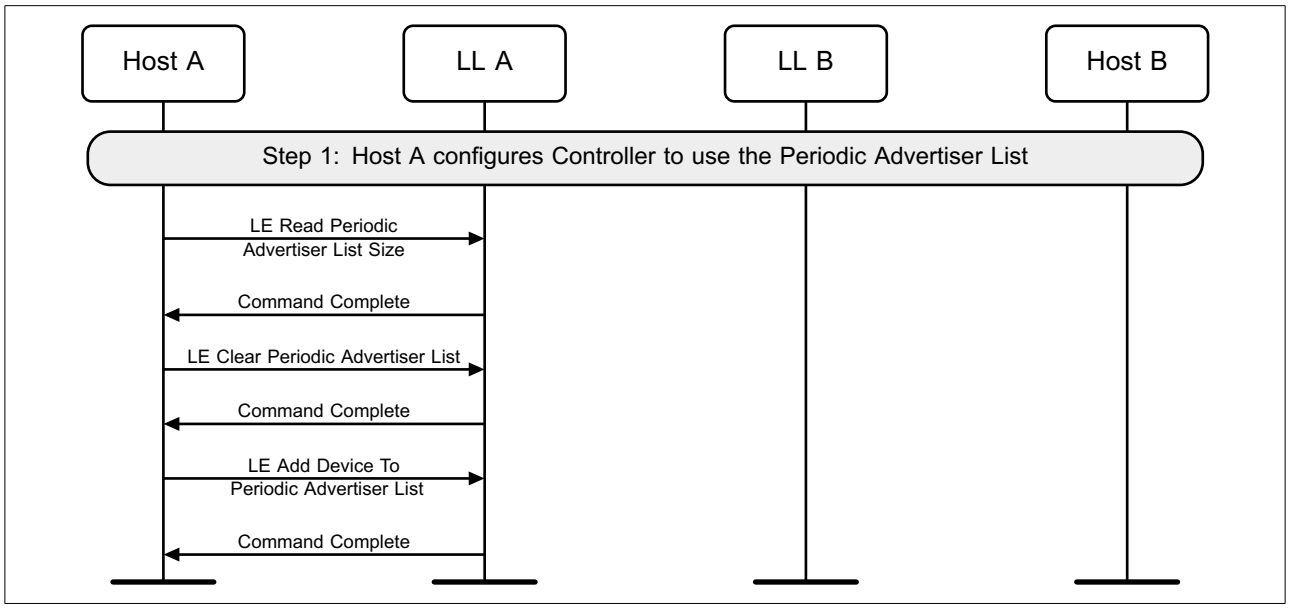



*Figure 2.6: Periodic Advertiser List*

# 3 ADVERTISING STATE

## 3.1 Undirected advertising

A device may enter the Advertising state by enabling advertising. It should also configure the advertising parameters before doing this (see Figure 3.1).

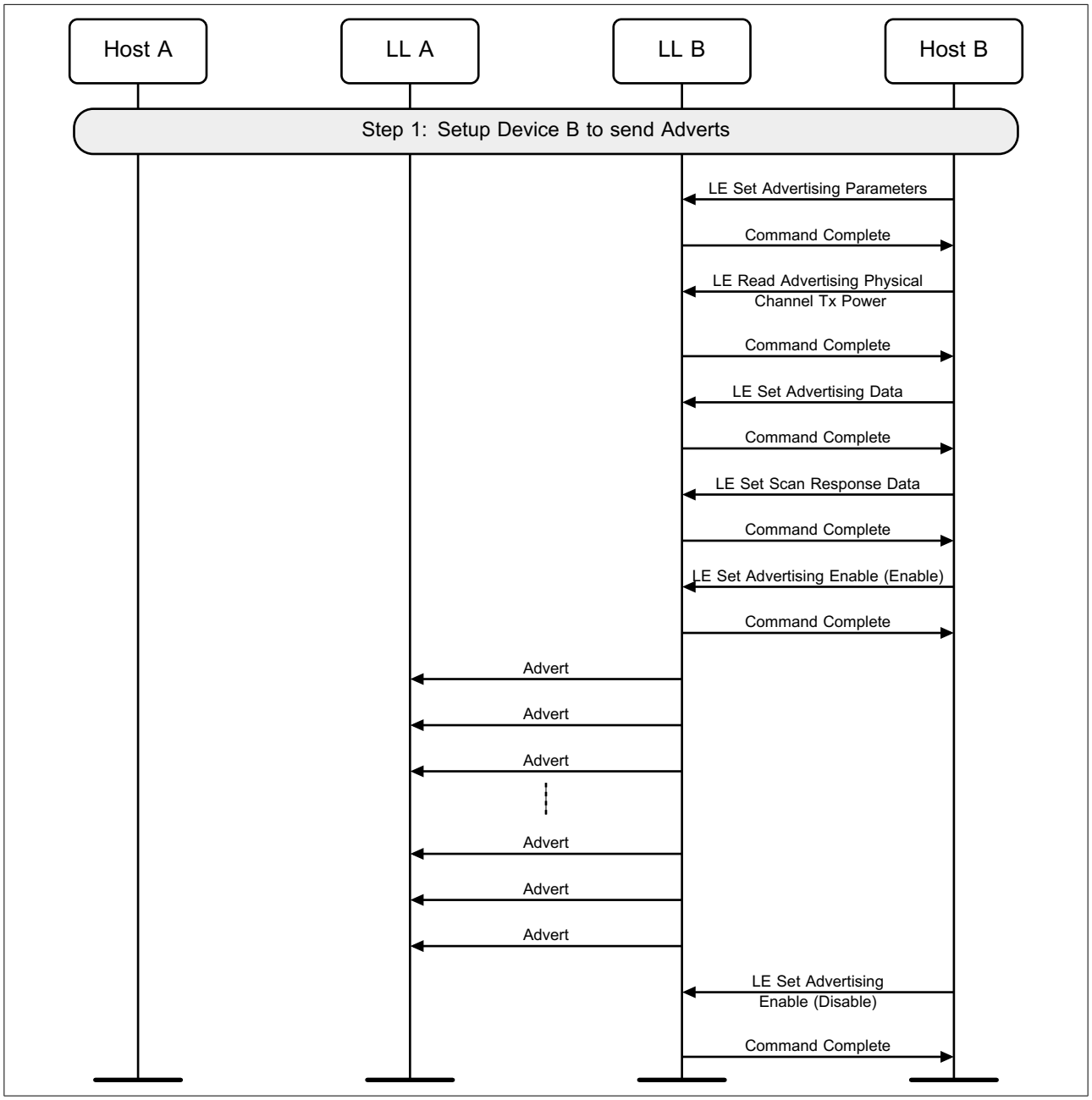



*Figure 3.1: Undirected advertising*

## 3.2 Directed advertising

A device may use directed advertising to allow an initiator to connect to it. High duty cycle directed advertising is time limited in the Controller and therefore this may fail before a connection is created. This example only shows the failure case (see Figure 3.2).

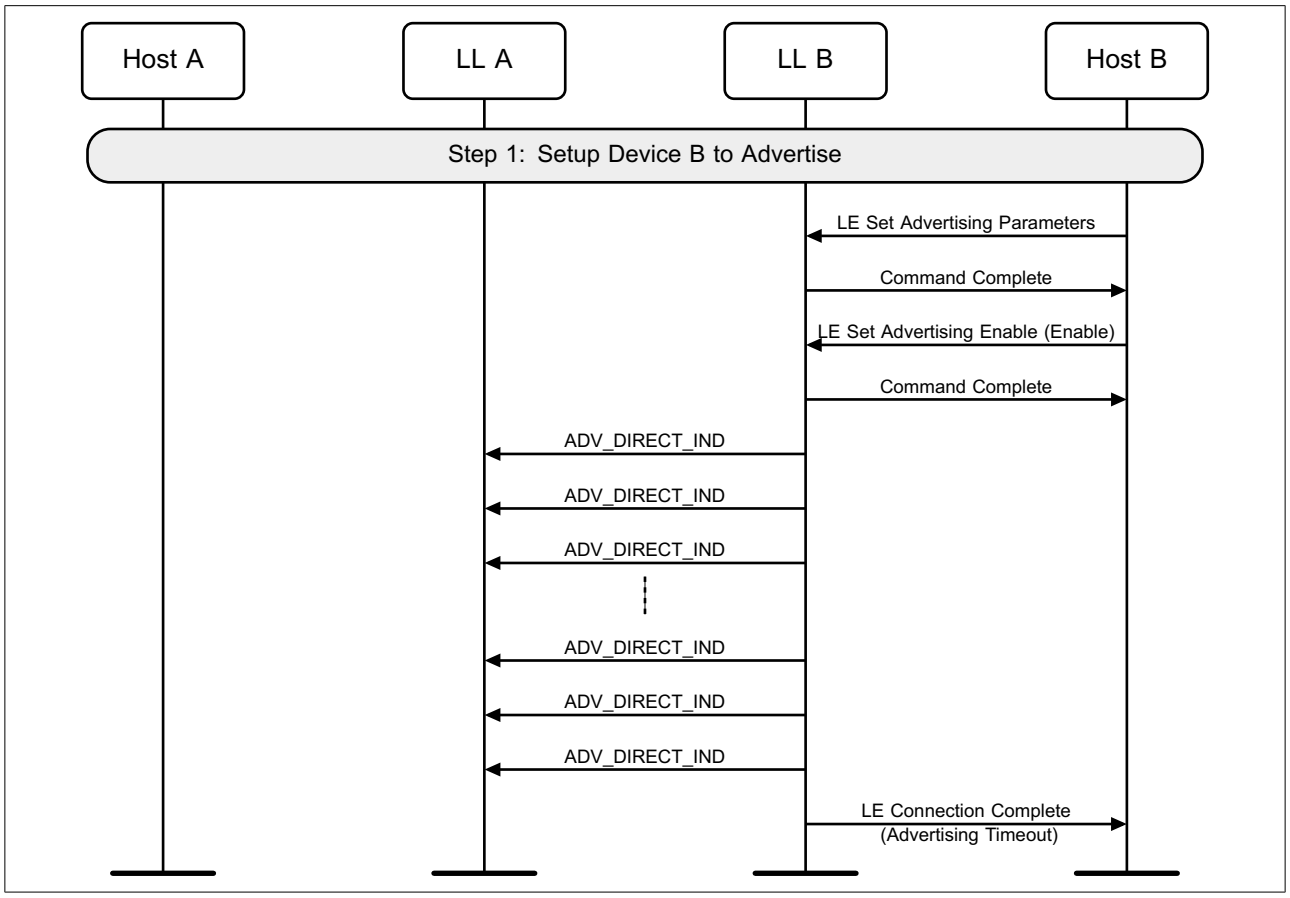



*Figure 3.2: High duty cycle directed advertising showing failure case*

Low duty cycle directed advertising is not time-limited. This example shows the case where no connection is made. A device should also configure the advertising parameters before doing this (see Figure 3.3).

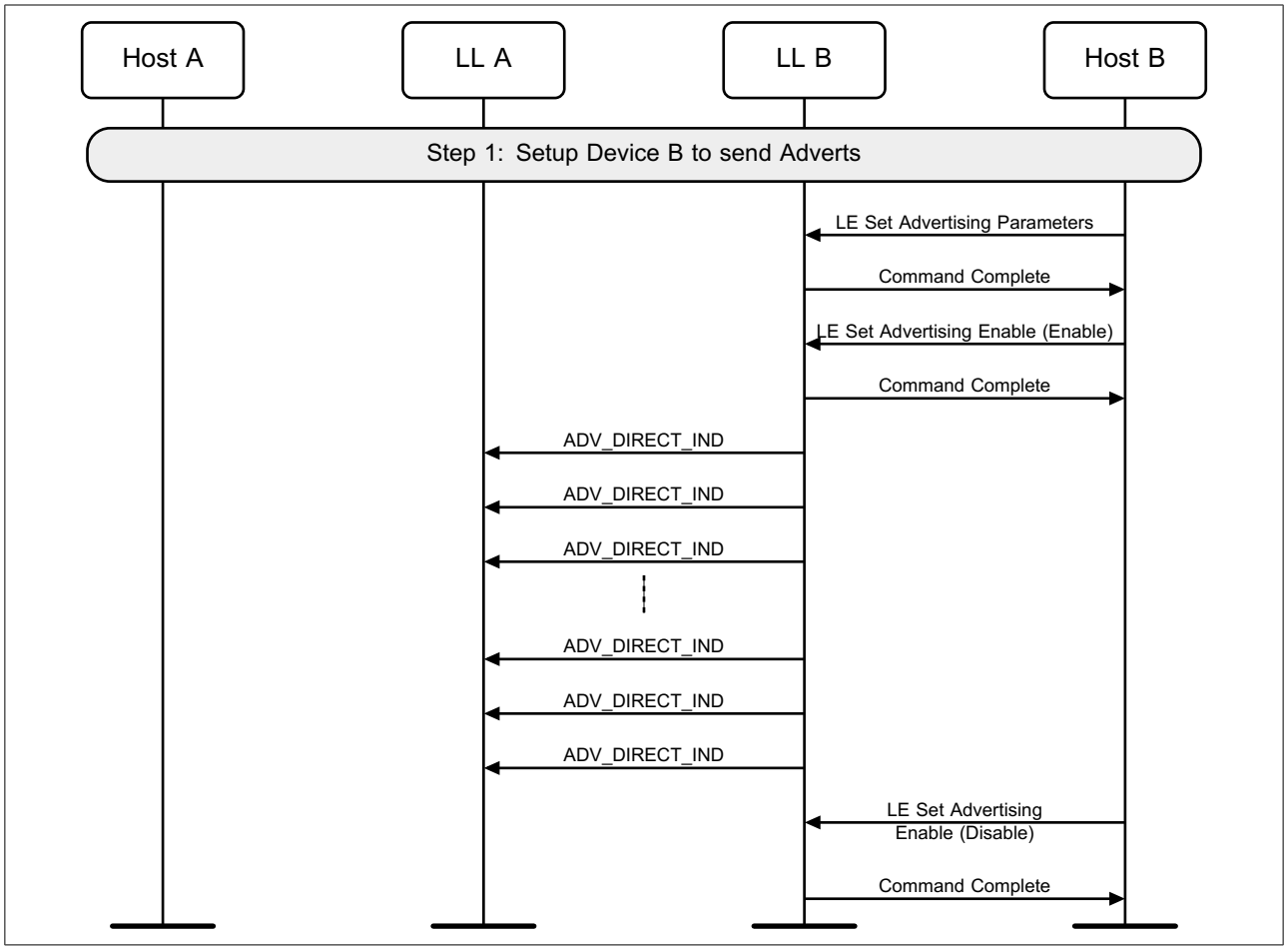



*Figure 3.3: Low duty cycle directed advertising*

## 3.3 Advertising using ADV_EXT_IND

A device may enter the Advertising state by enabling advertising a set. It should also configure the advertising set parameters before doing this (see Figure 3.4).

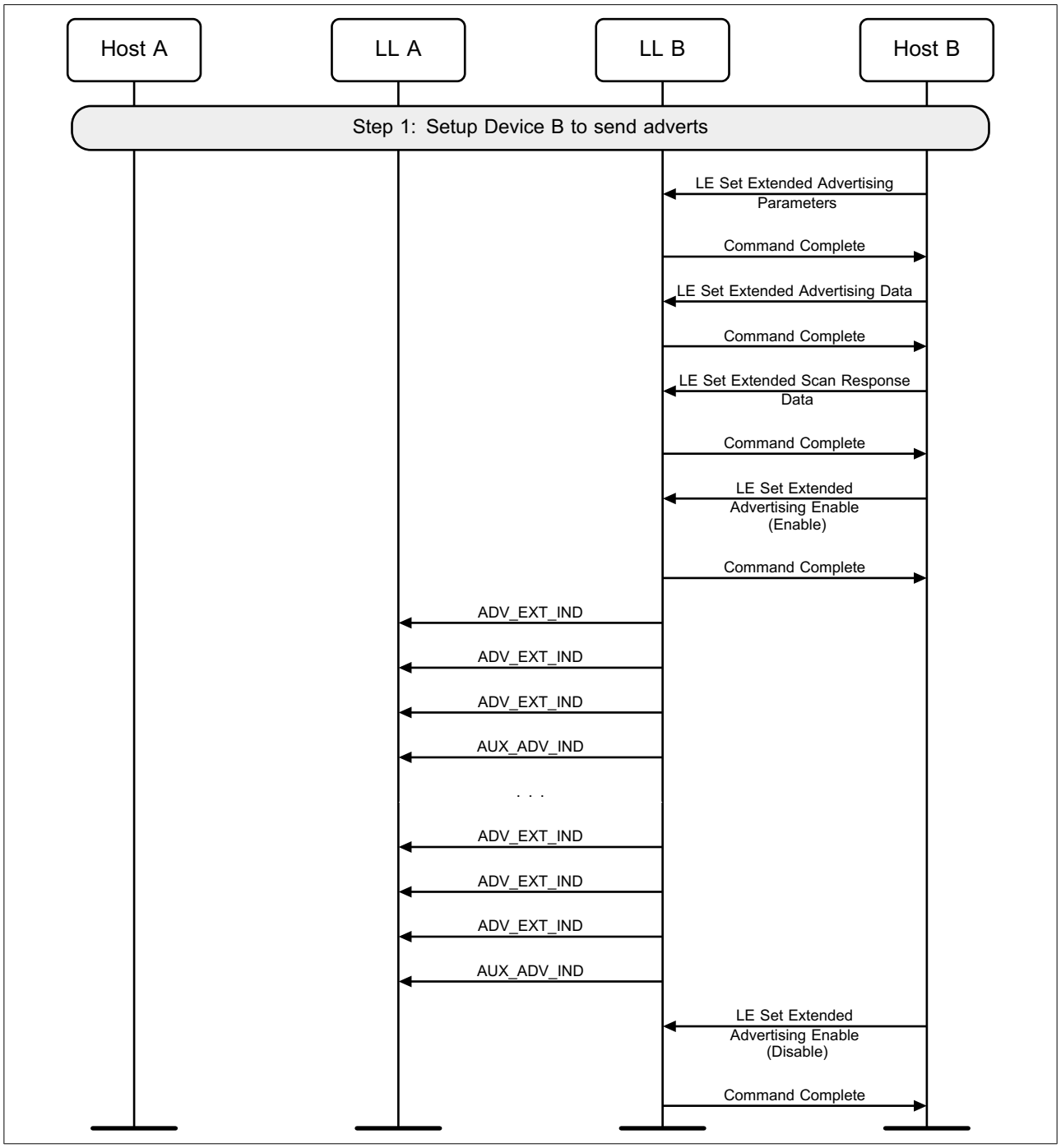



*Figure 3.4: Advertising using ADV_EXT_IND*

## 3.4 Scan request notifications

A device may enable scan request notifications in an advertising set (see Figure 3.5).

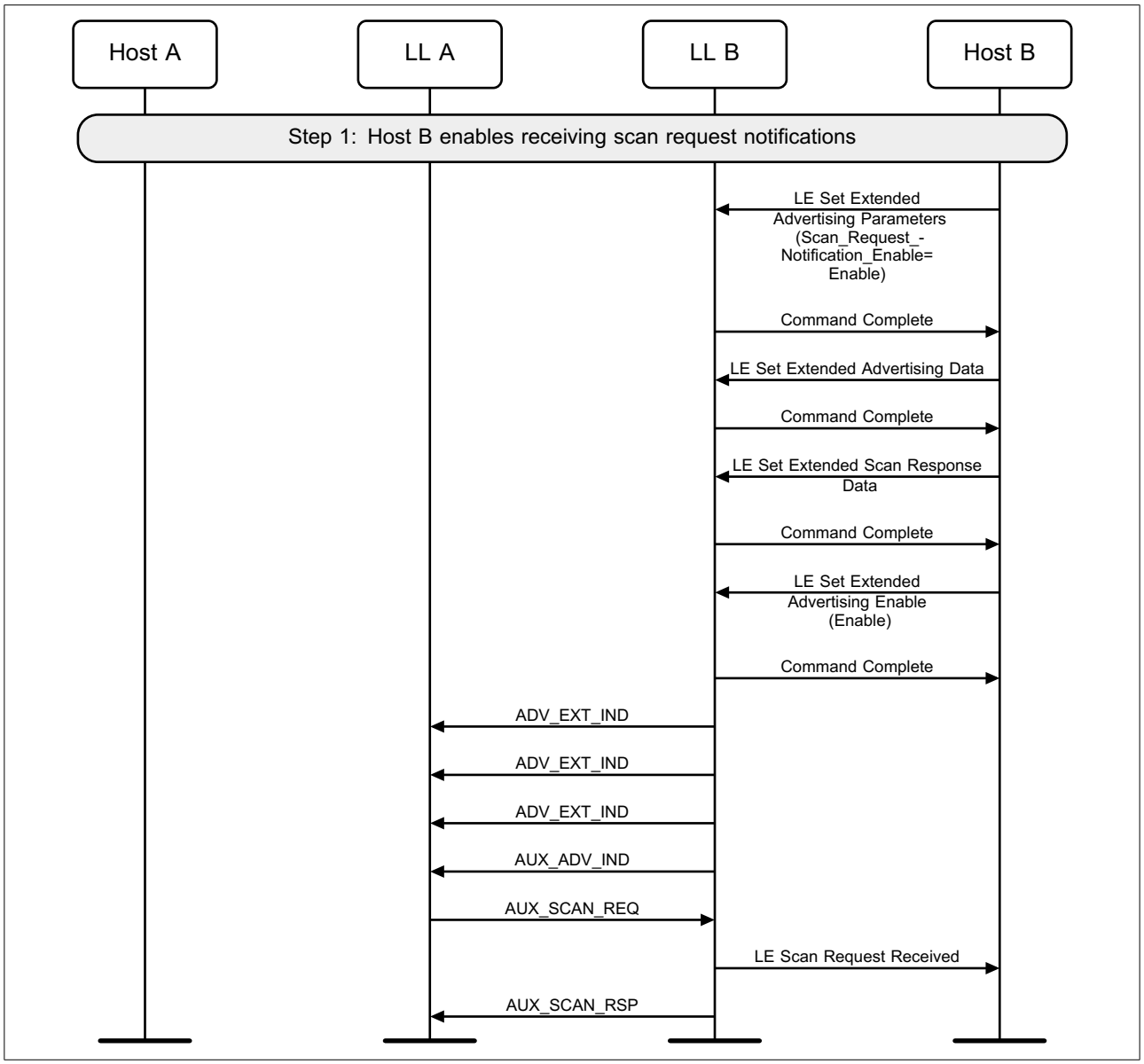



*Figure 3.5: Scan request notifications*

## 3.5 Advertising duration ended

A device may enter the Advertising State by enabling advertising a set for a limited duration of time (see Figure 3.6).

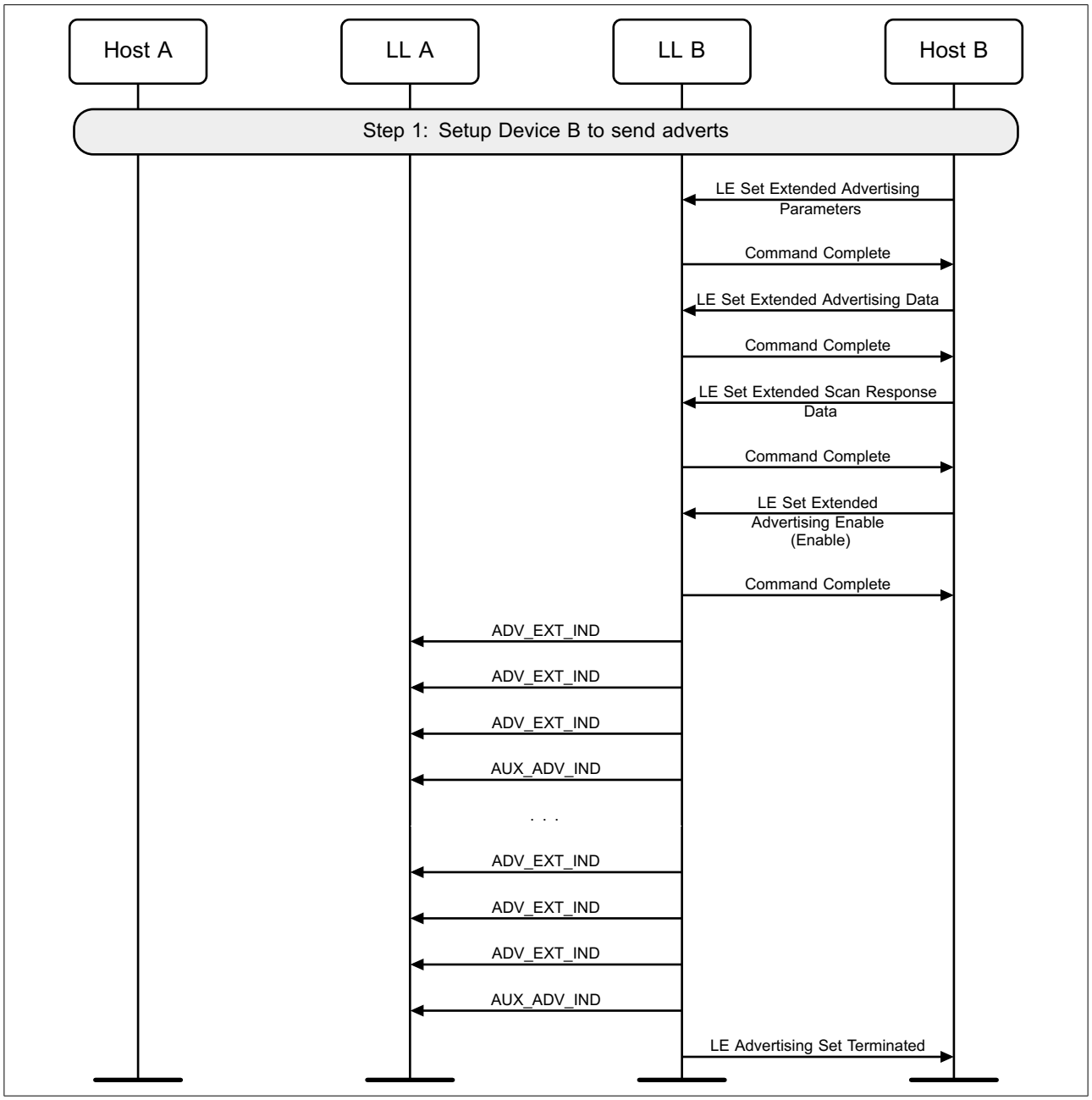



*Figure 3.6: Advertising duration ended*

## 3.6 Periodic advertising

A device may enter the Advertising State by enabling periodic advertising in a set. It should also configure the advertising set parameters before doing this (see Figure 3.7).

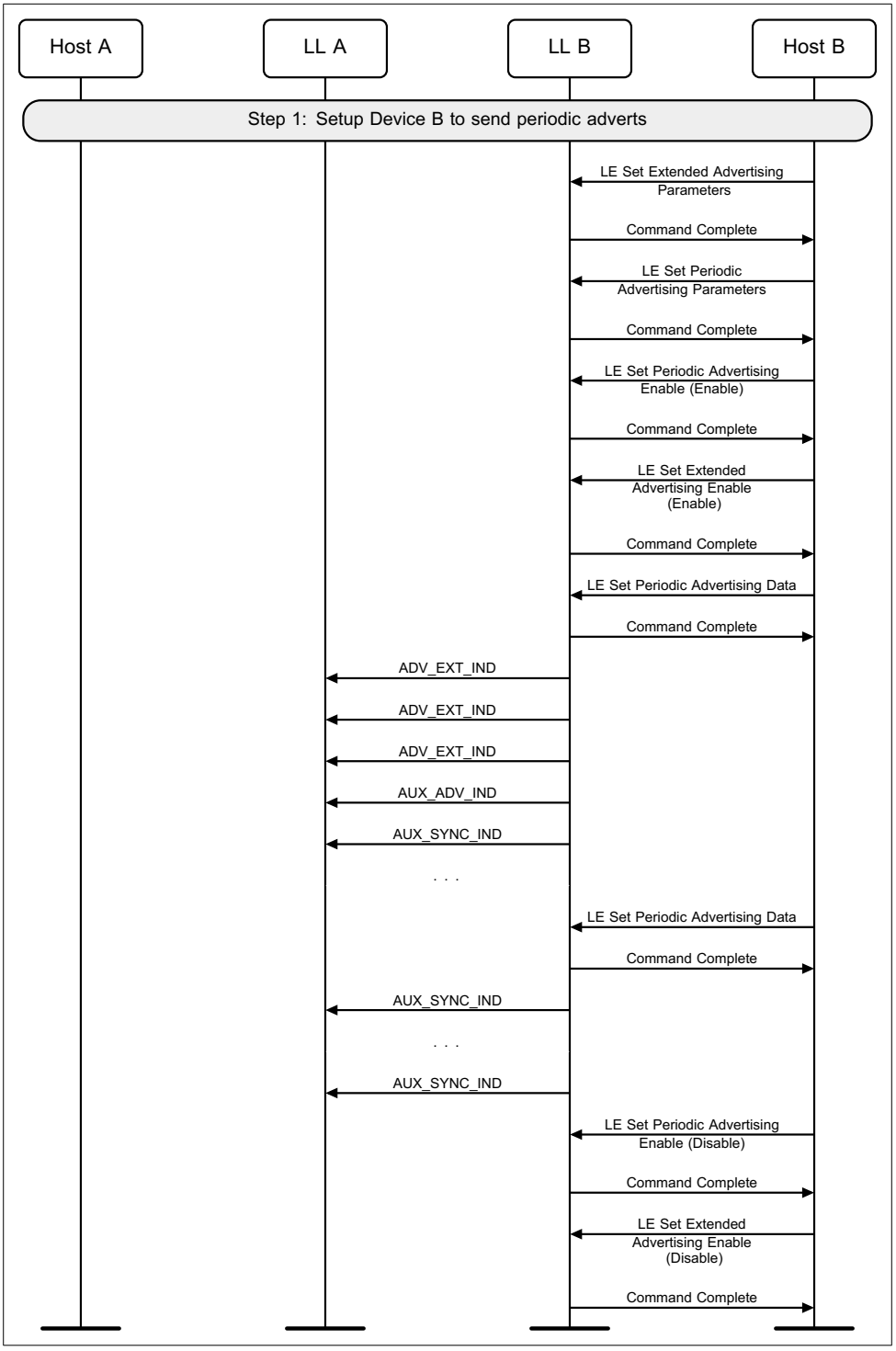



*Figure 3.7: Periodic advertising*

## 3.7 Connectionless Constant Tone Extension transmission

A device may send periodic advertising packets containing a Constant Tone Extension (see Figure 3.8).

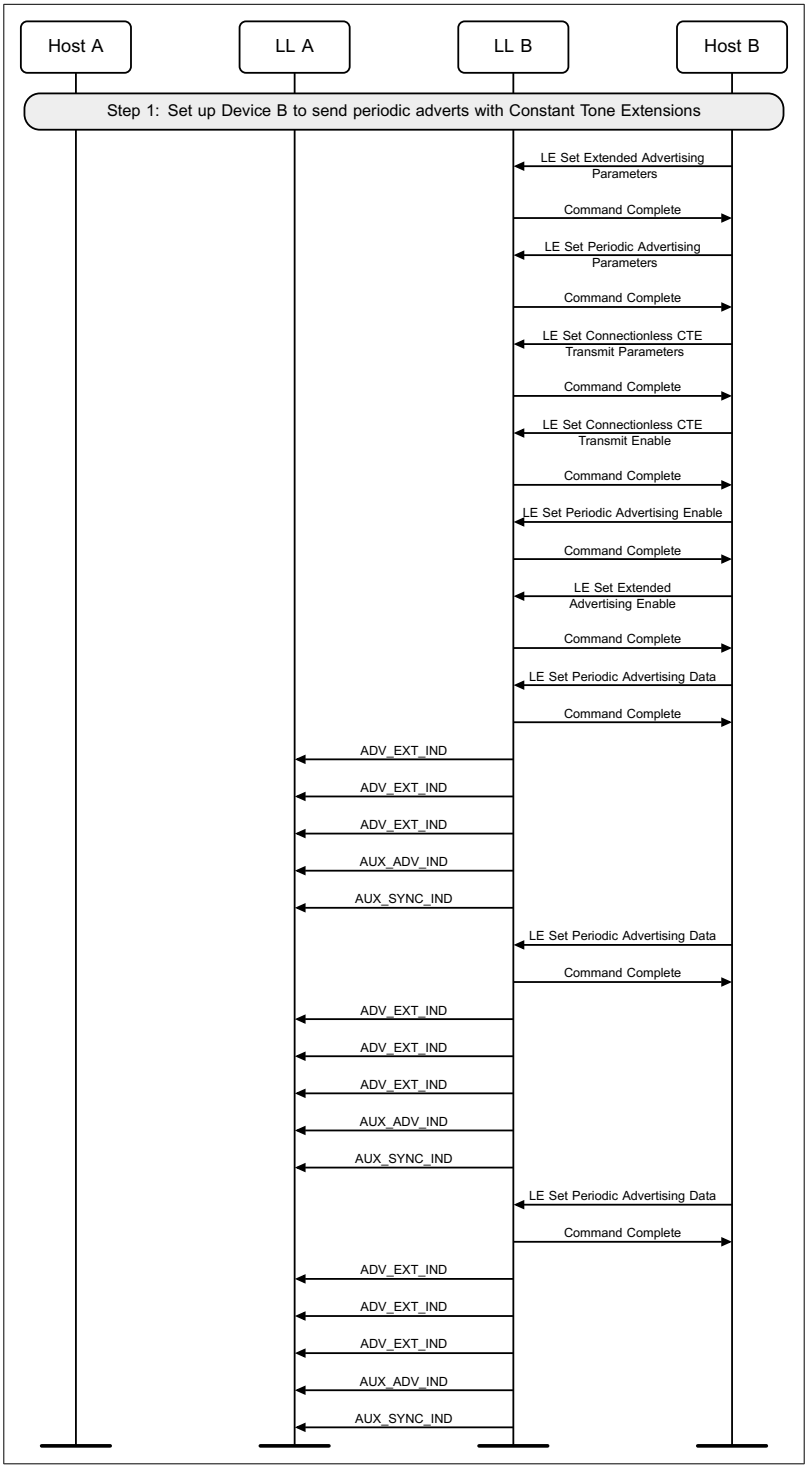



*Figure 3.8: Connectionless CTE transmission*

## 3.8 Isochronous Broadcasting State

### 3.8.1 Create a Broadcast Isochronous Group

A device enters the Isochronous Broadcasting State and enables the periodic advertising associated with the BIG. The device then creates the BIG and sends isochronous data using the isochronous data paths (see Figure 3.9).

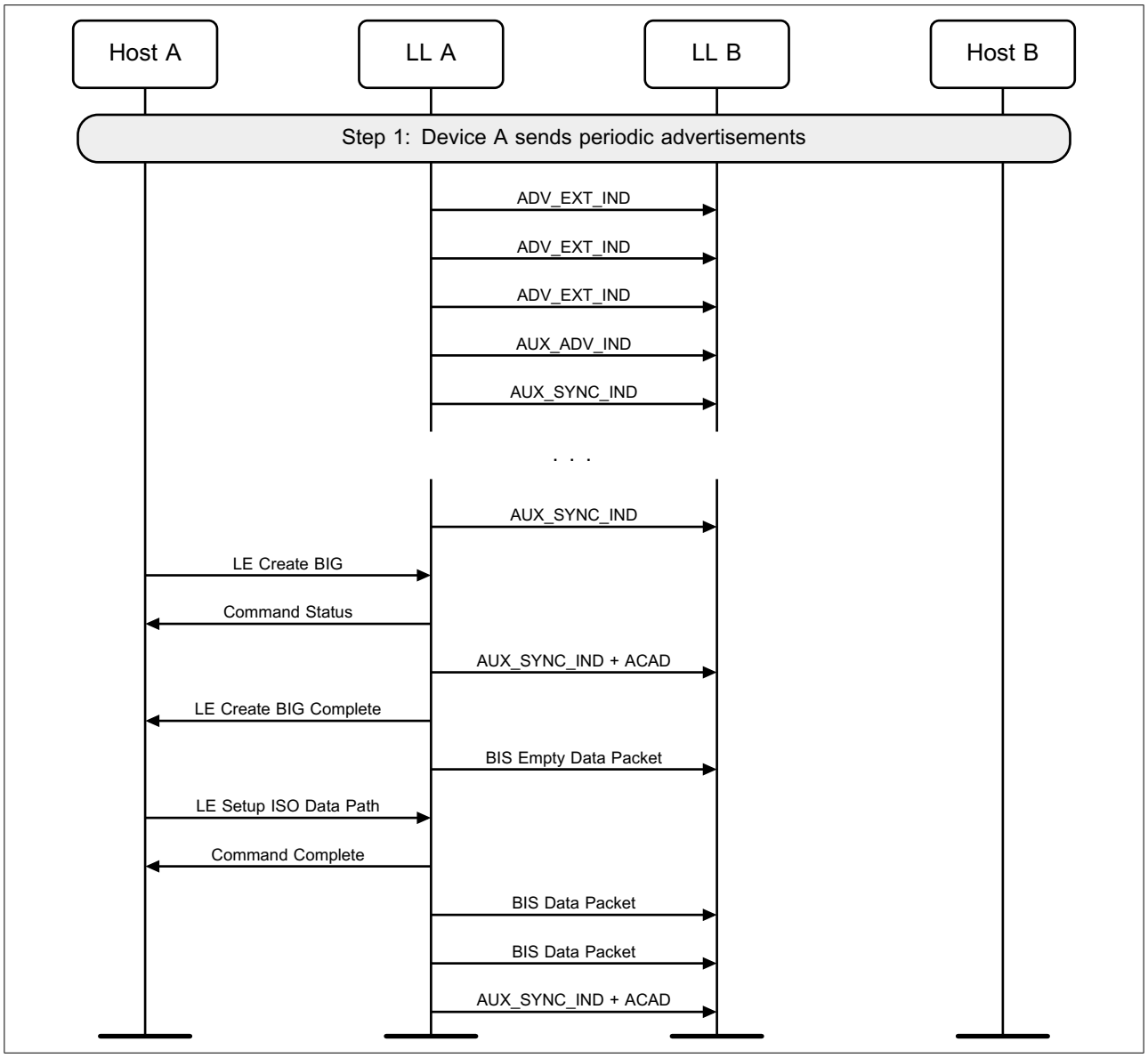



*Figure 3.9: Device A creates a BIG*

### 3.8.2 Terminate a Broadcast Isochronous Group

A device terminates a BIG (see Figure 3.10).

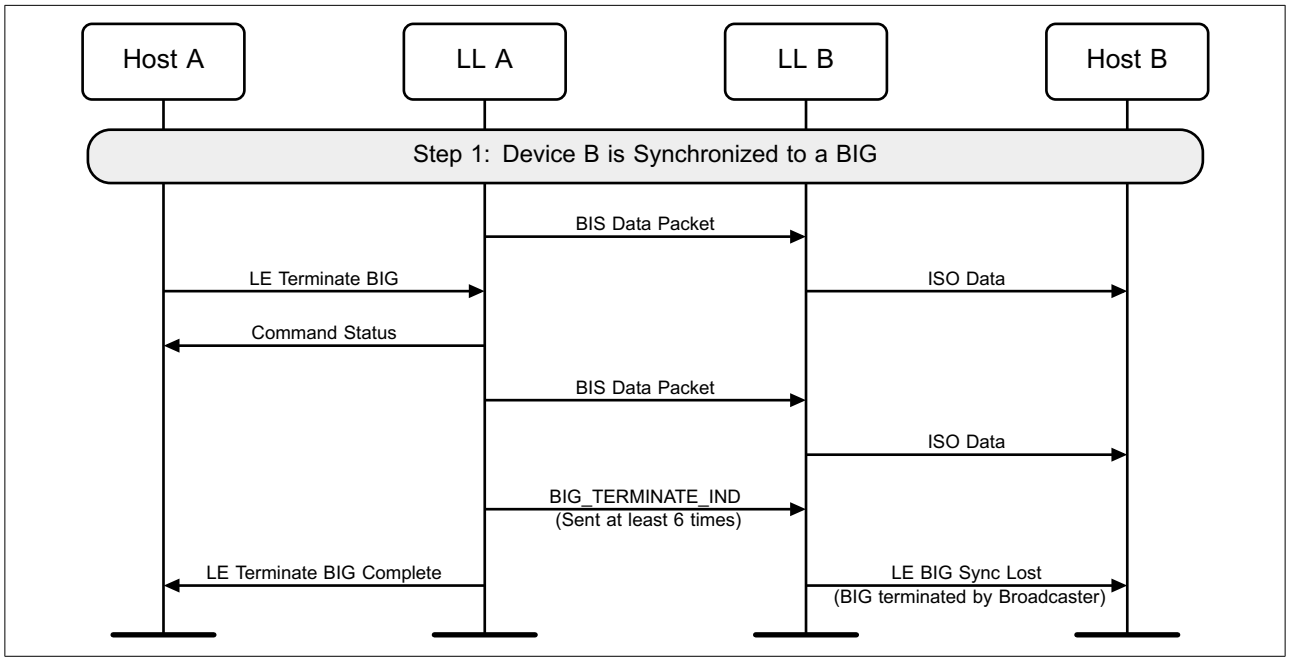



*Figure 3.10: Device A terminates a BIG*

## 3.9 Periodic advertising with responses (PAwR)

A device may enter the Advertising state by enabling PAwR in a set. The device should configure the advertising set parameters before doing this (see Figure 3.11).

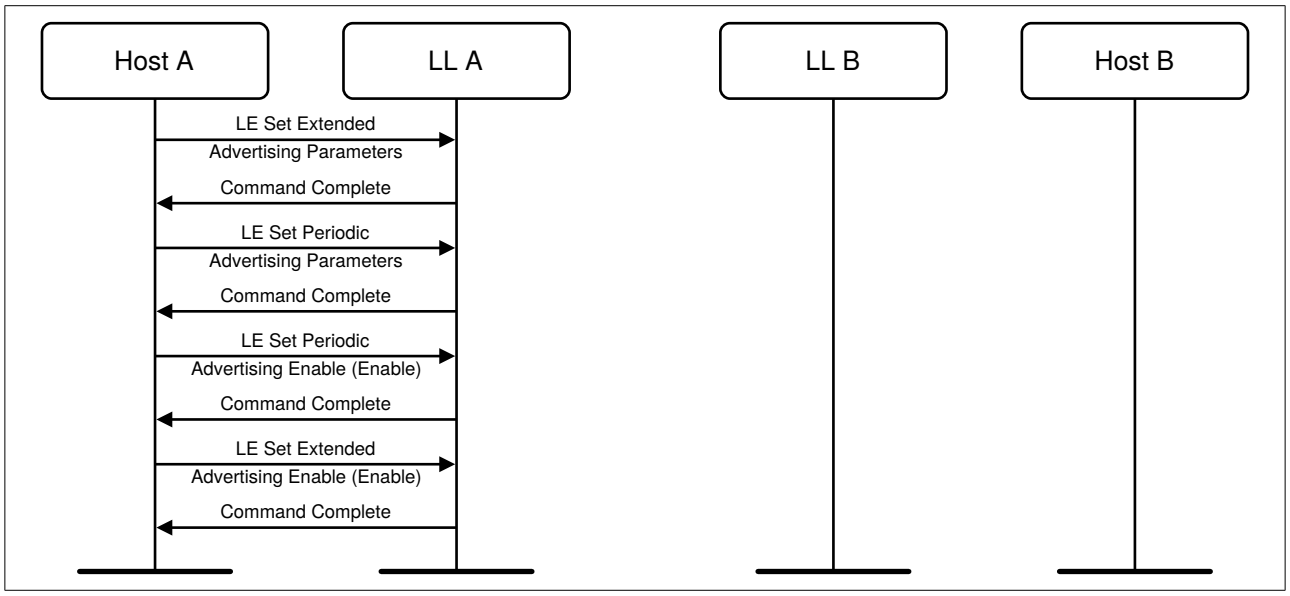



*Figure 3.11: Configuring PAwR*

## 3.10 Transmitting PAwR subevents

A Controller may request data to be sent in one or more subevents of a PAwR advertising set. The Host can send this data to the Controller ahead of when the subevents are scheduled.

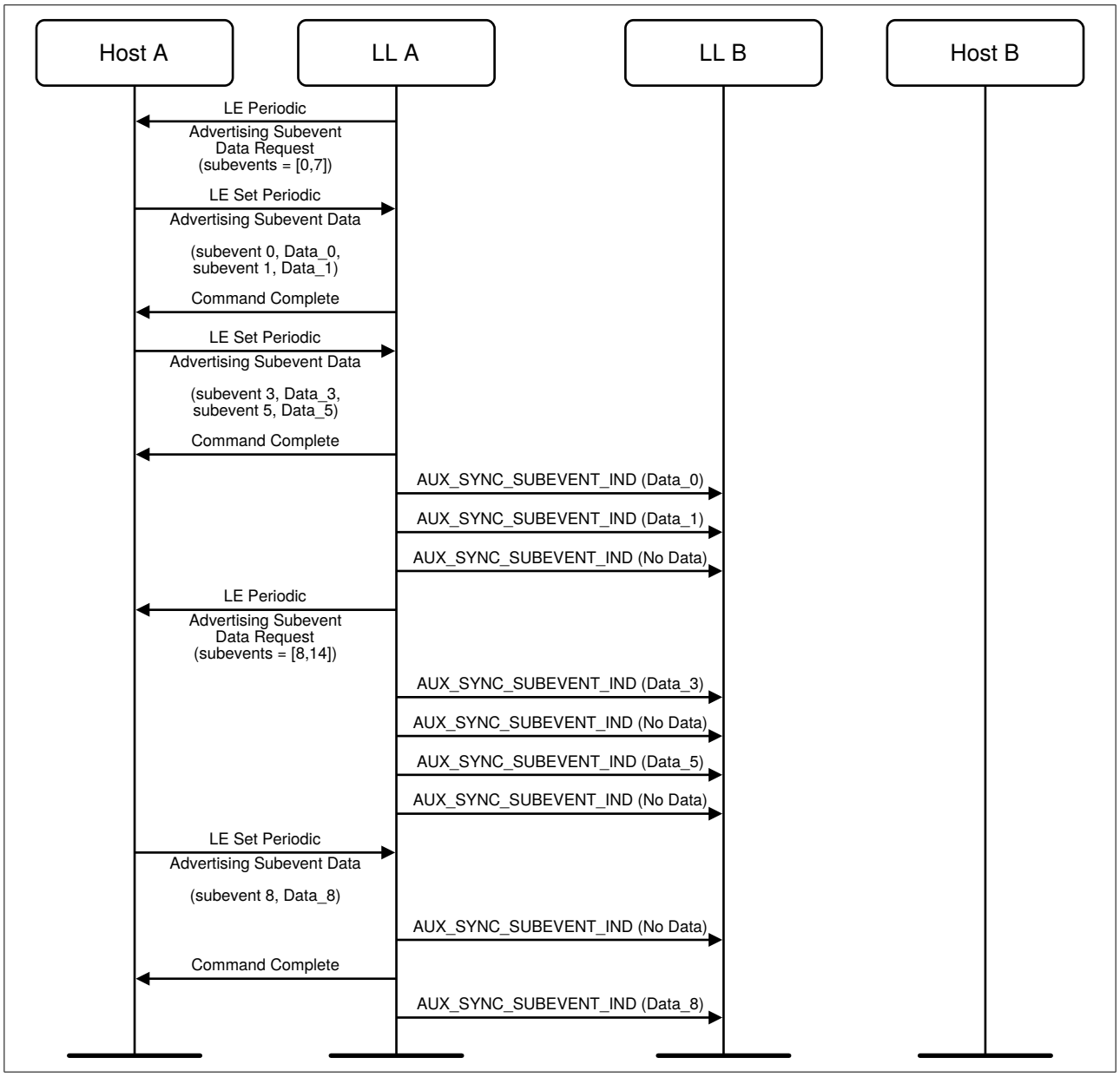



*Figure 3.12: Transmitting PAwR subevents*

## 3.11 Using a response slot in PAwR

A device can send a response to a PAwR advertisement packet. The timing of the response is determined by the Host using information from the advertisement packet.

The Host can send a response by indicating the response slot and data to be sent in that response slot.

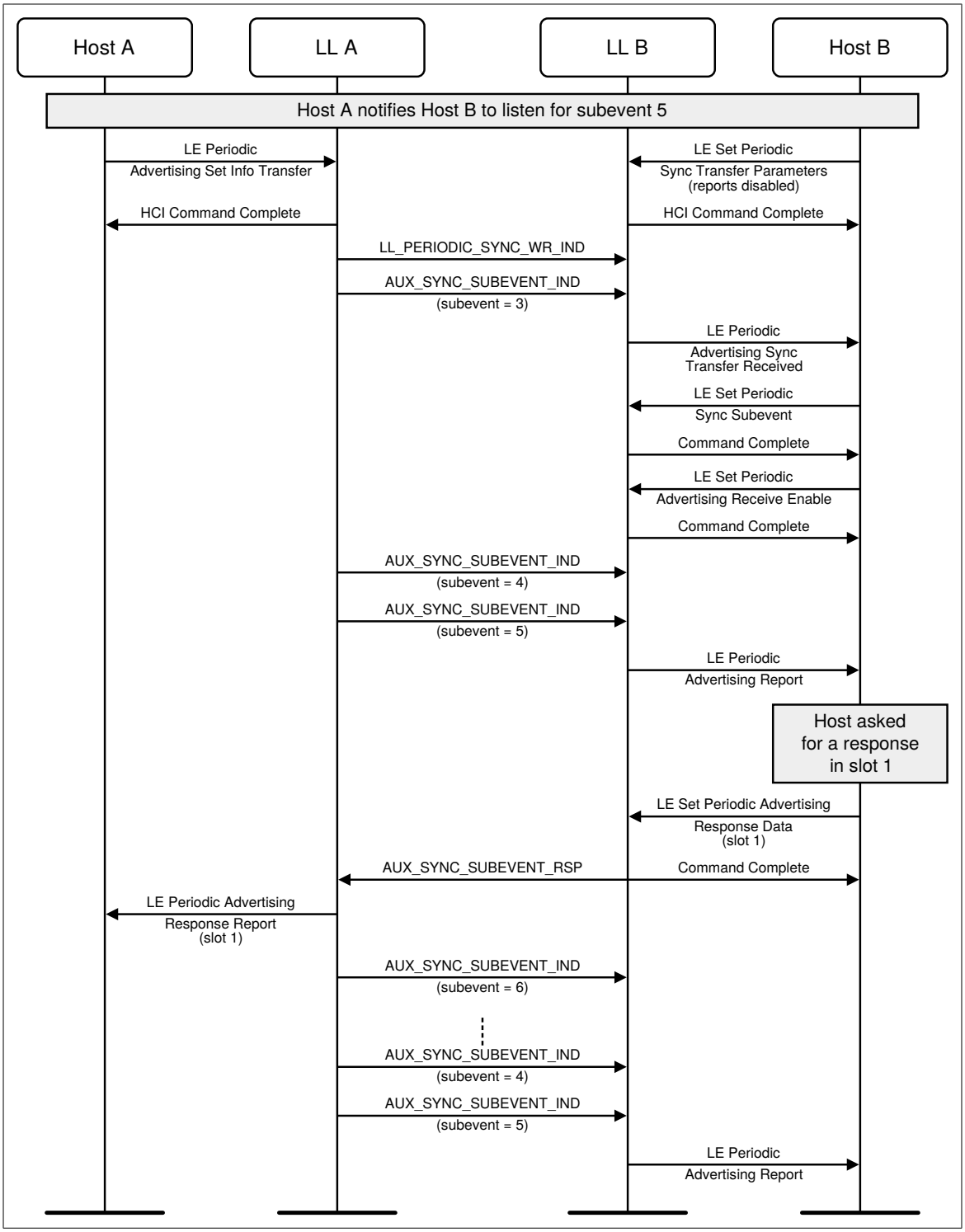



*Figure 3.13: Using a response slot in PAwR*

## 3.12 Connecting from PAwR

A device may initiate a connection with a synchronized device by using the LE Extended Create Connection command which indicates the subevent and BD_ADDR of the peer device. This initiates a connection by sending an AUX_CONNECT_REQ PDU in that subevent.

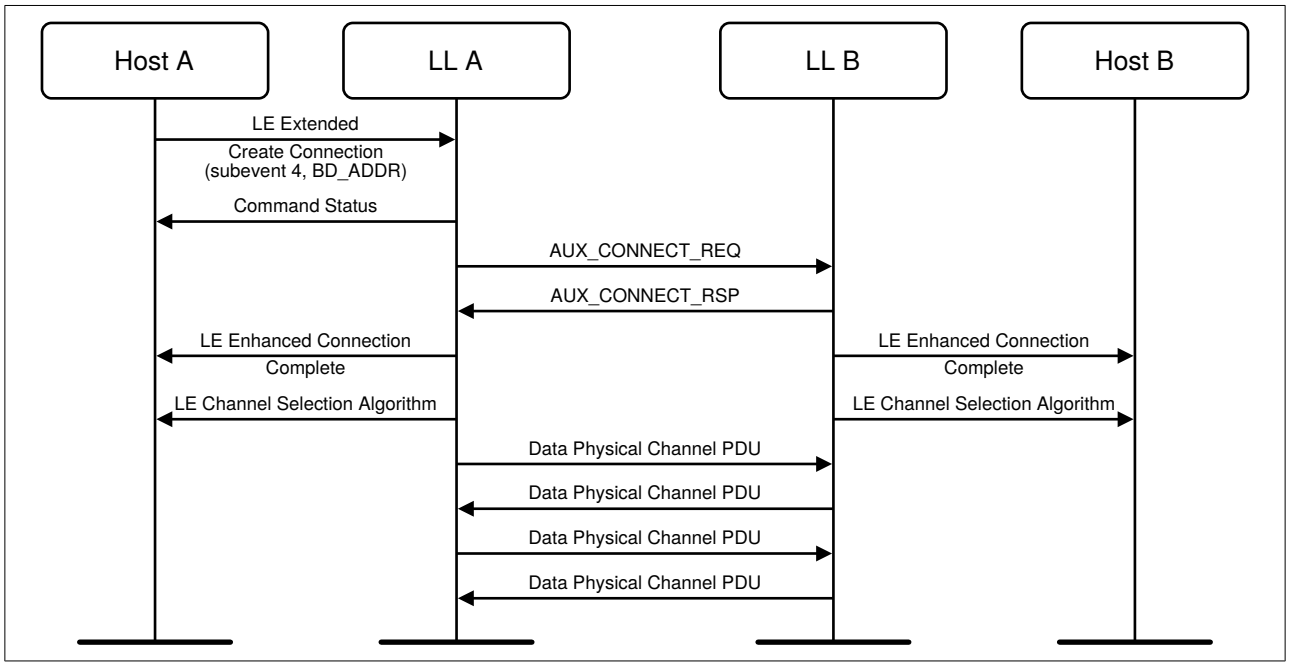



*Figure 3.14: Connecting from PAwR*

## 3.13 Failed connection attempts from PAwR

A synchronized device can miss an AUX_CONNECT_REQ PDU sent by the periodic advertiser.

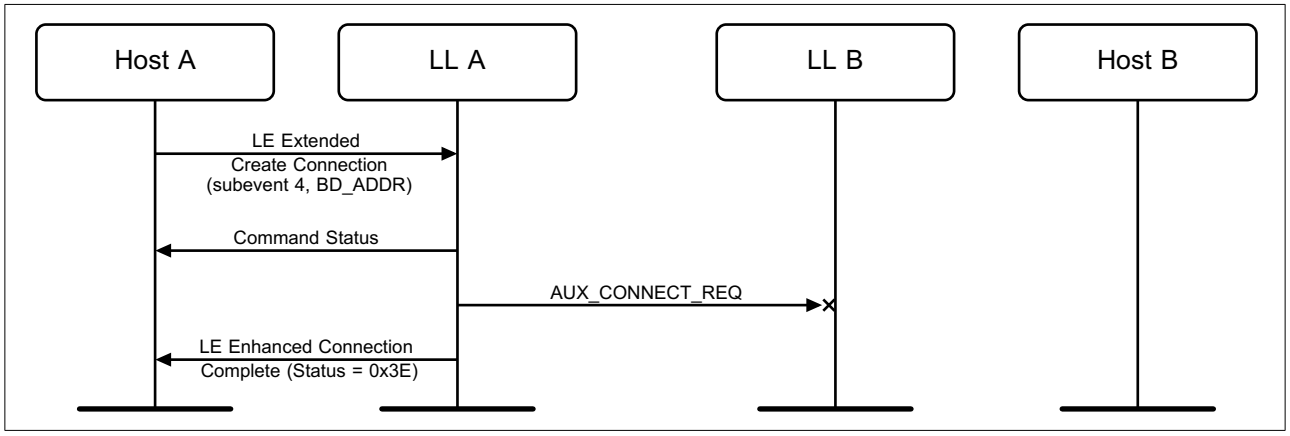



*Figure 3.15: Synchronized device misses the AUX_CONNECT_REQ PDU*

The periodic advertiser can miss an AUX_CONNECT_RSP PDU sent by a synchronized device.

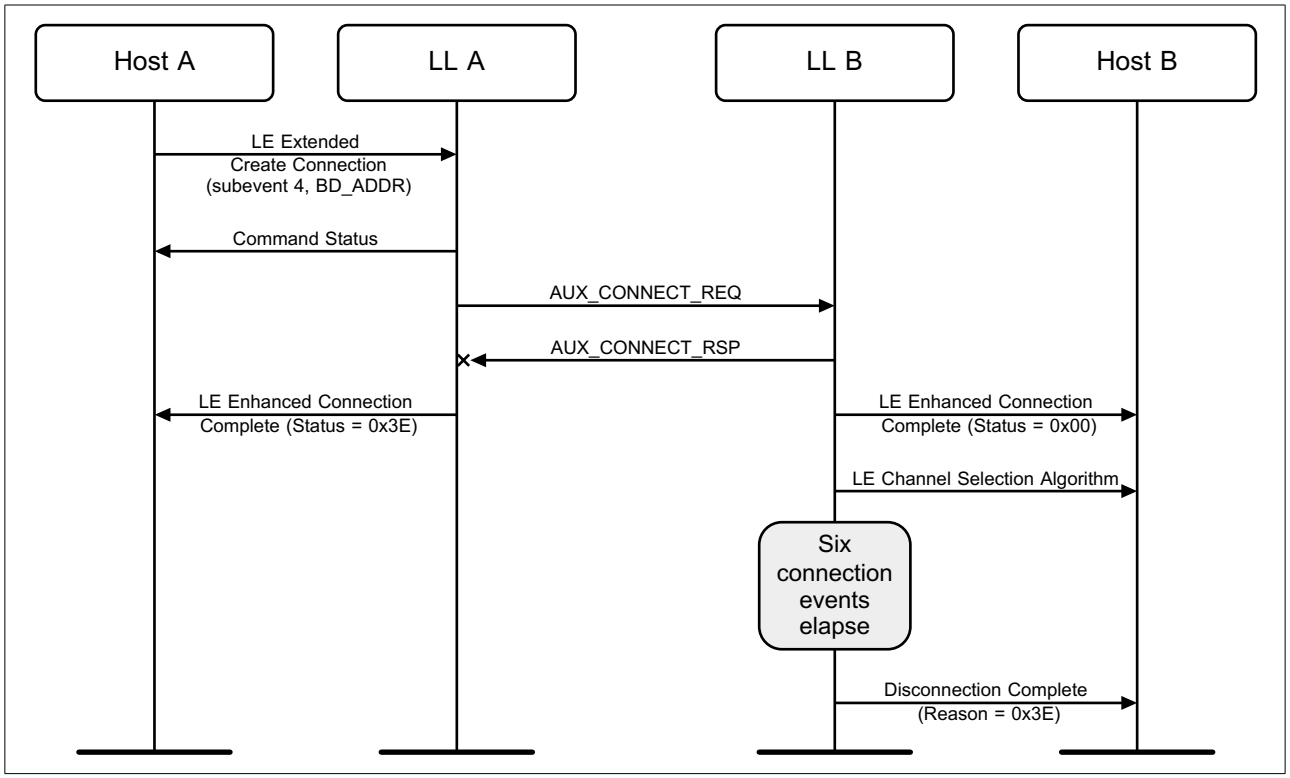



*Figure 3.16: Periodic advertiser misses the AUX_CONNECT_RSP PDU*

# 4 SCANNING STATE

## 4.1 Passive scanning

A device can use passive scanning to find advertising devices in the area. This would receive advertising packets from peer devices and report these to the Host (see Figure 4.1).

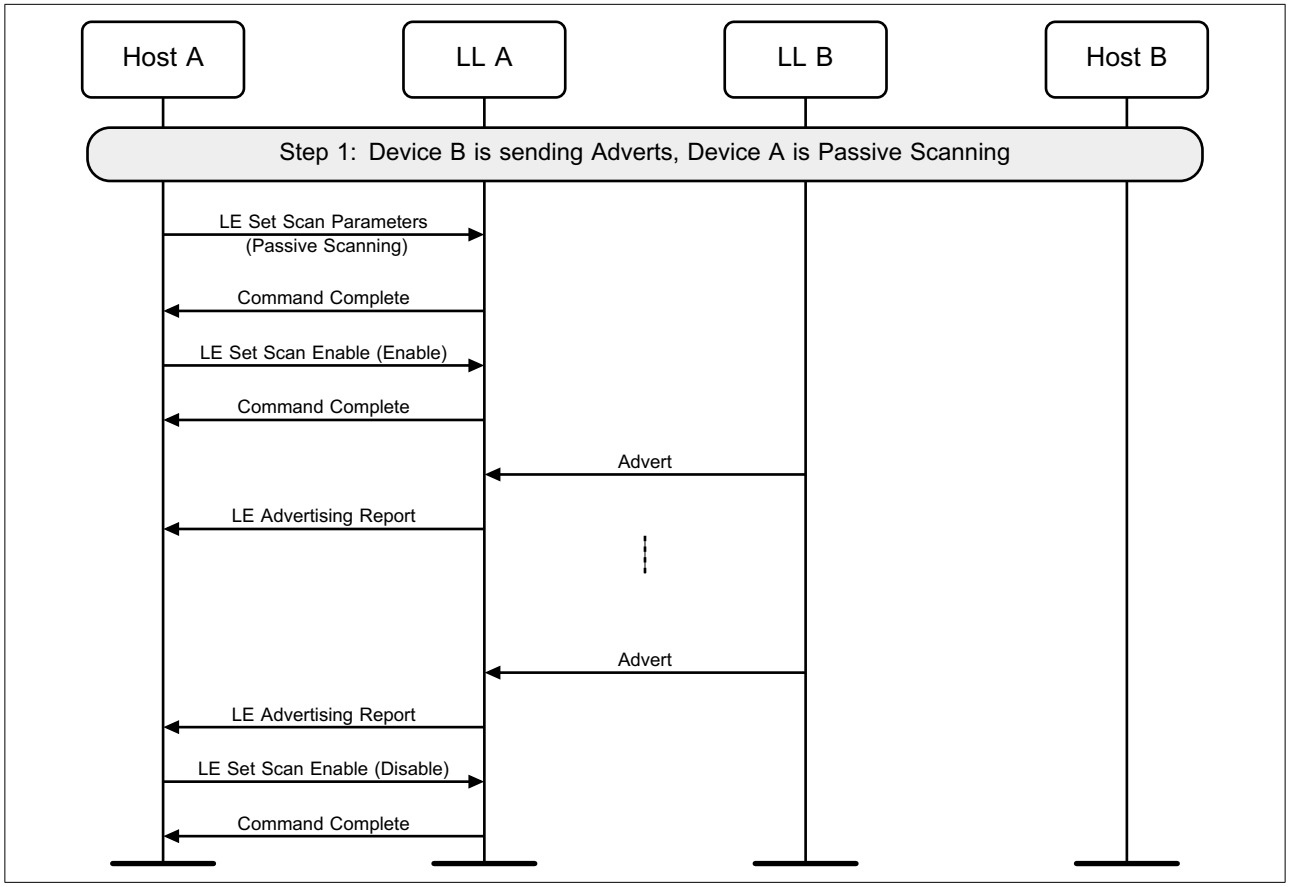



*Figure 4.1: Passive scanning*

## 4.2 Active scanning

A device may use active scanning to obtain more information about devices that may be useful to populate a user interface. Active scanning involves more Link Layer advertising messages (see Figure 4.2).

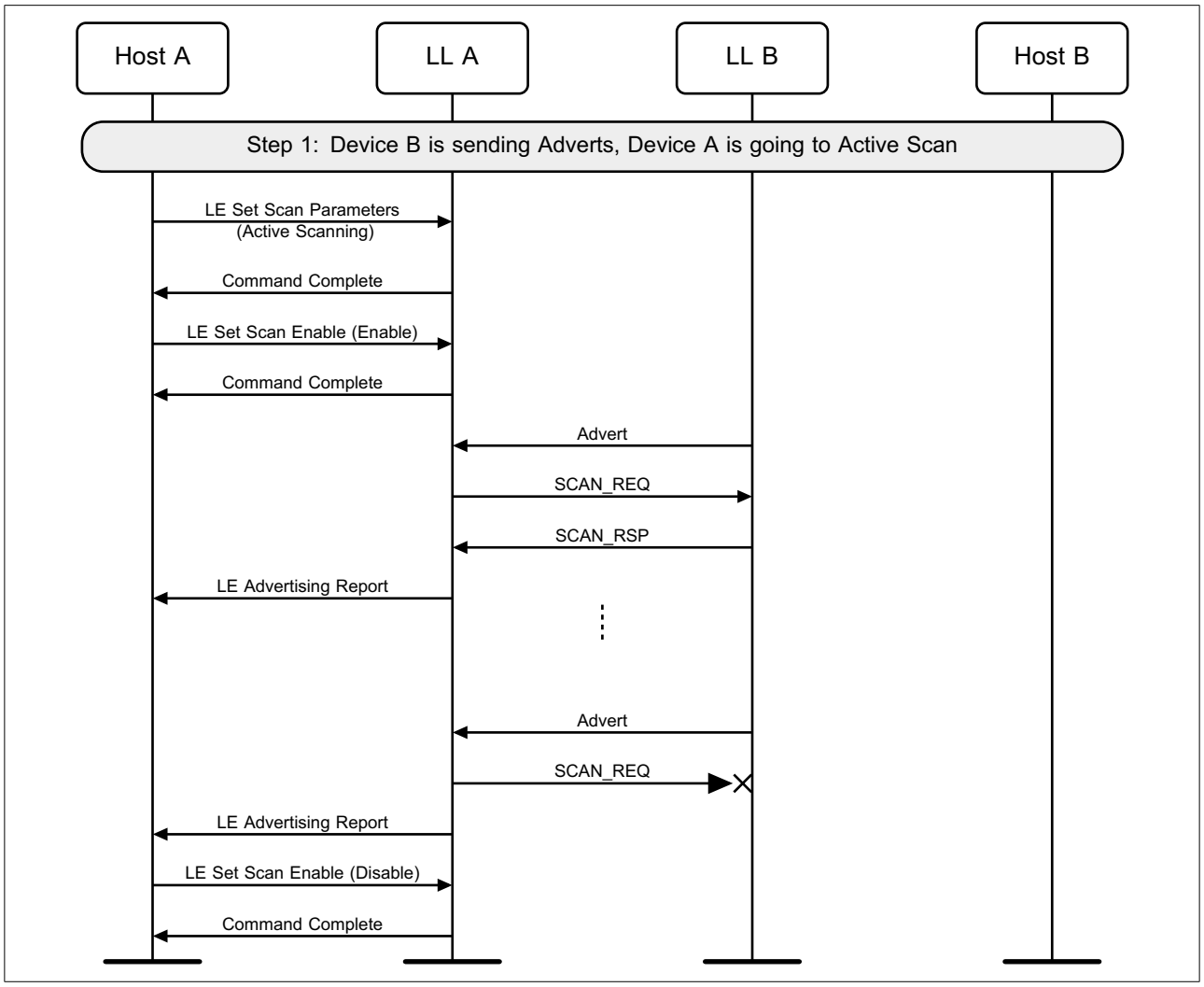



*Figure 4.2: Active scanning*

## 4.3 Passive scanning for directed advertisements with Privacy

If a device does not support Privacy in the Controller, it may choose to forward LE Directed Advertising Report events from devices supporting Privacy without requiring filtering through the Controller Resolving List.

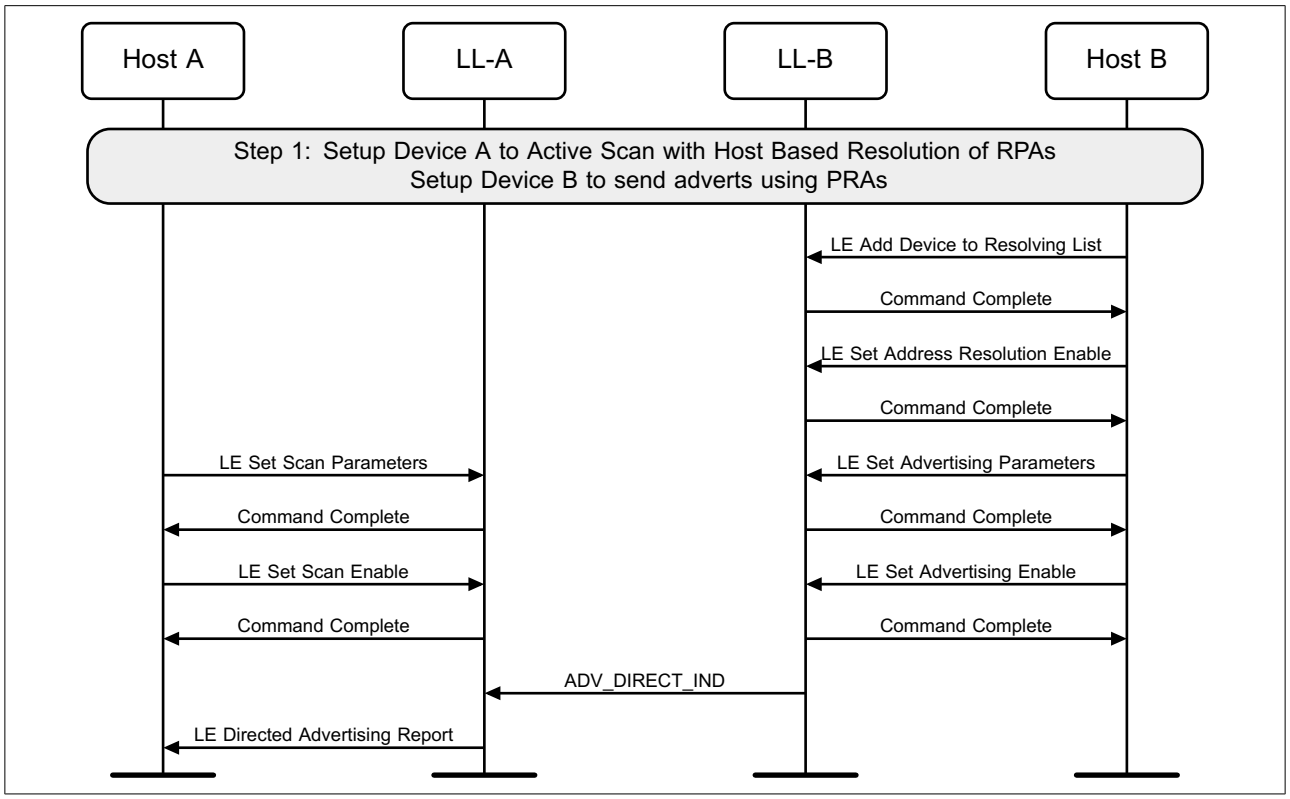



*Figure 4.3: Directed advertising with Privacy*

## 4.4 Active scanning with Privacy

A device may use active scanning to obtain more information about devices that may be useful to populate a user interface. Privacy may be used during active scanning to make it more difficult to track either device during active scanning (see Figure 4.4).

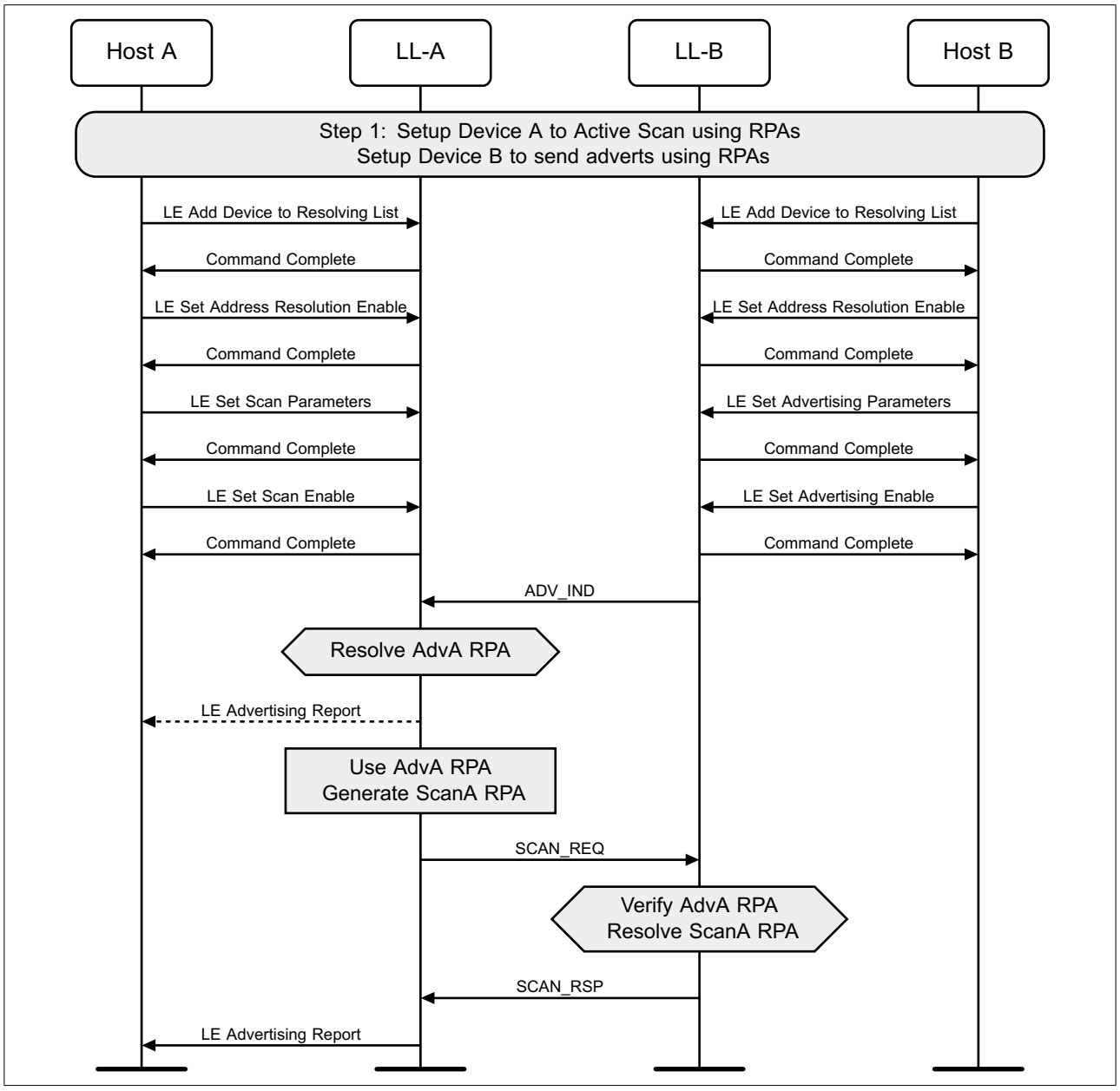



*Figure 4.4: Active scanning with Privacy*

## 4.5 Active scanning with Privacy and Controller based resolvable private address generation

A Controller will periodically update the resolvable private addresses used on both devices if the devices use active scanning and advertising with Privacy. A Host may at anytime retrieve the read from the Controller the current addresses being used (see Figure 4.5).

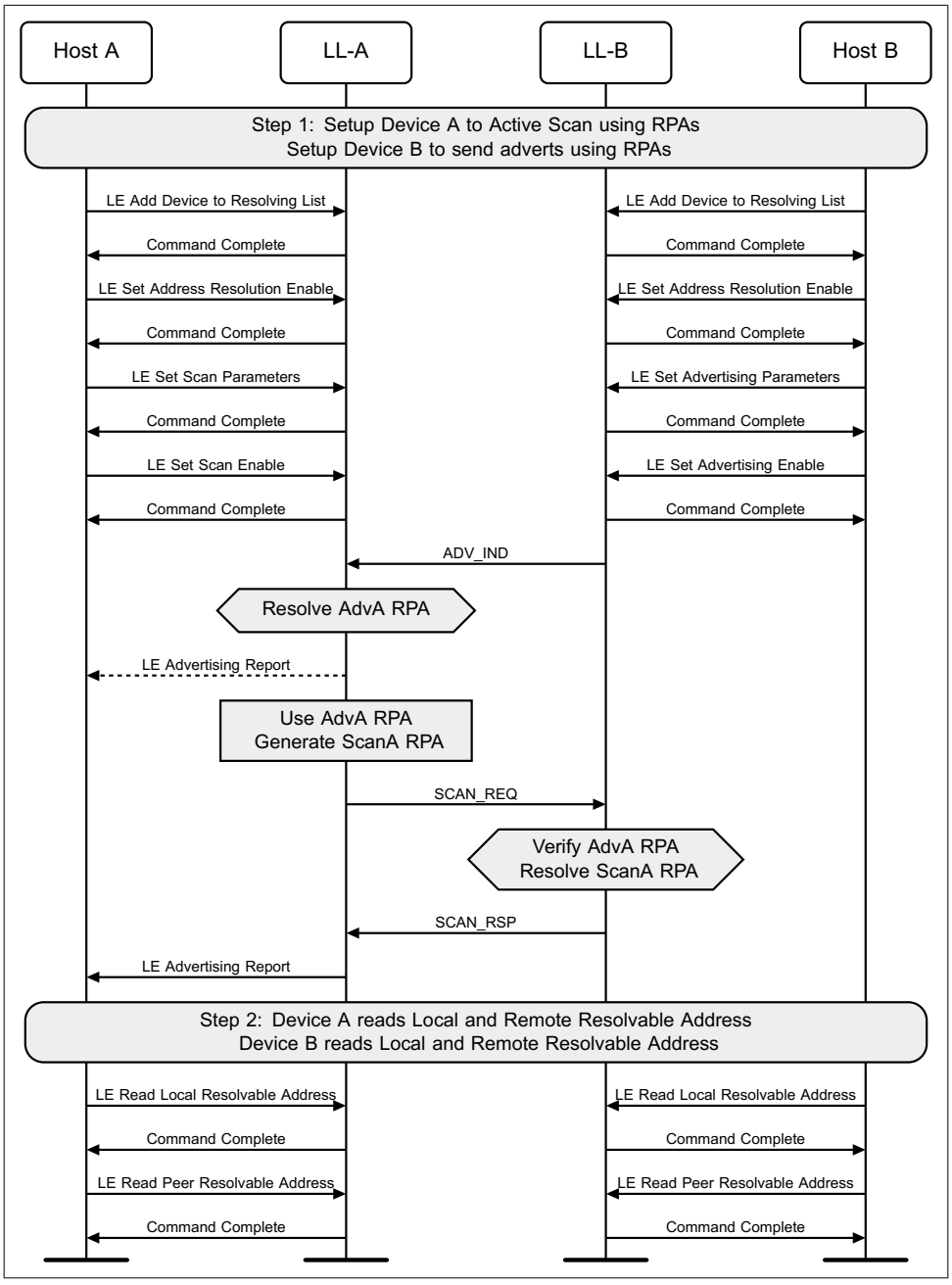



*Figure 4.5: Retrieving local and remote resolvable address updates from the Controller*

## 4.6 Active scanning on the secondary advertising Physical channel

A device may use active scanning on the secondary advertising physical channel in order to obtain more information about devices that may be useful to populate a user interface (see Figure 4.6).

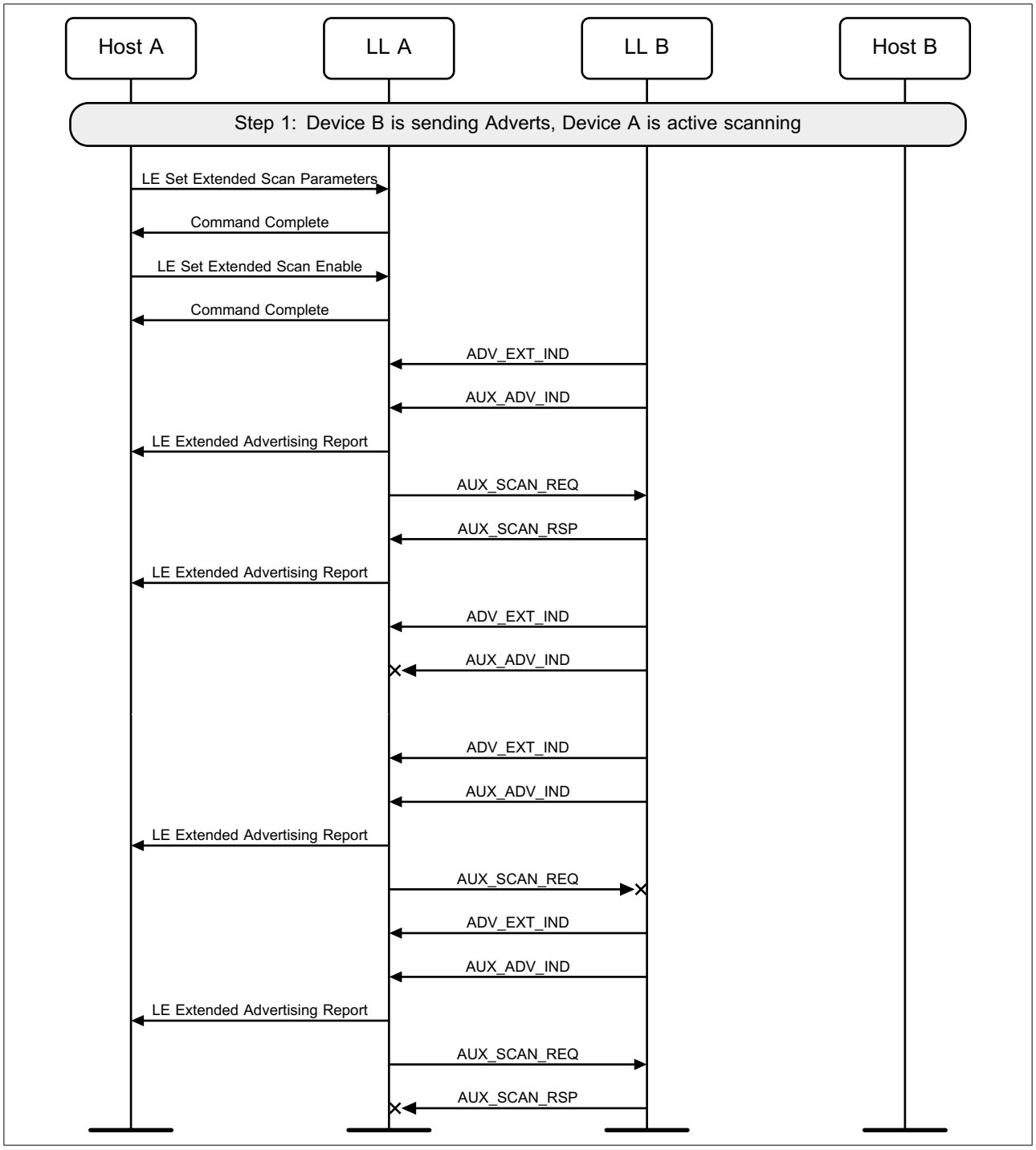



*Figure 4.6: Extended active scanning on the secondary advertising physical channel*

## 4.7 Scan timeout

A device may scan for a limited duration of time (see Figure 4.7).

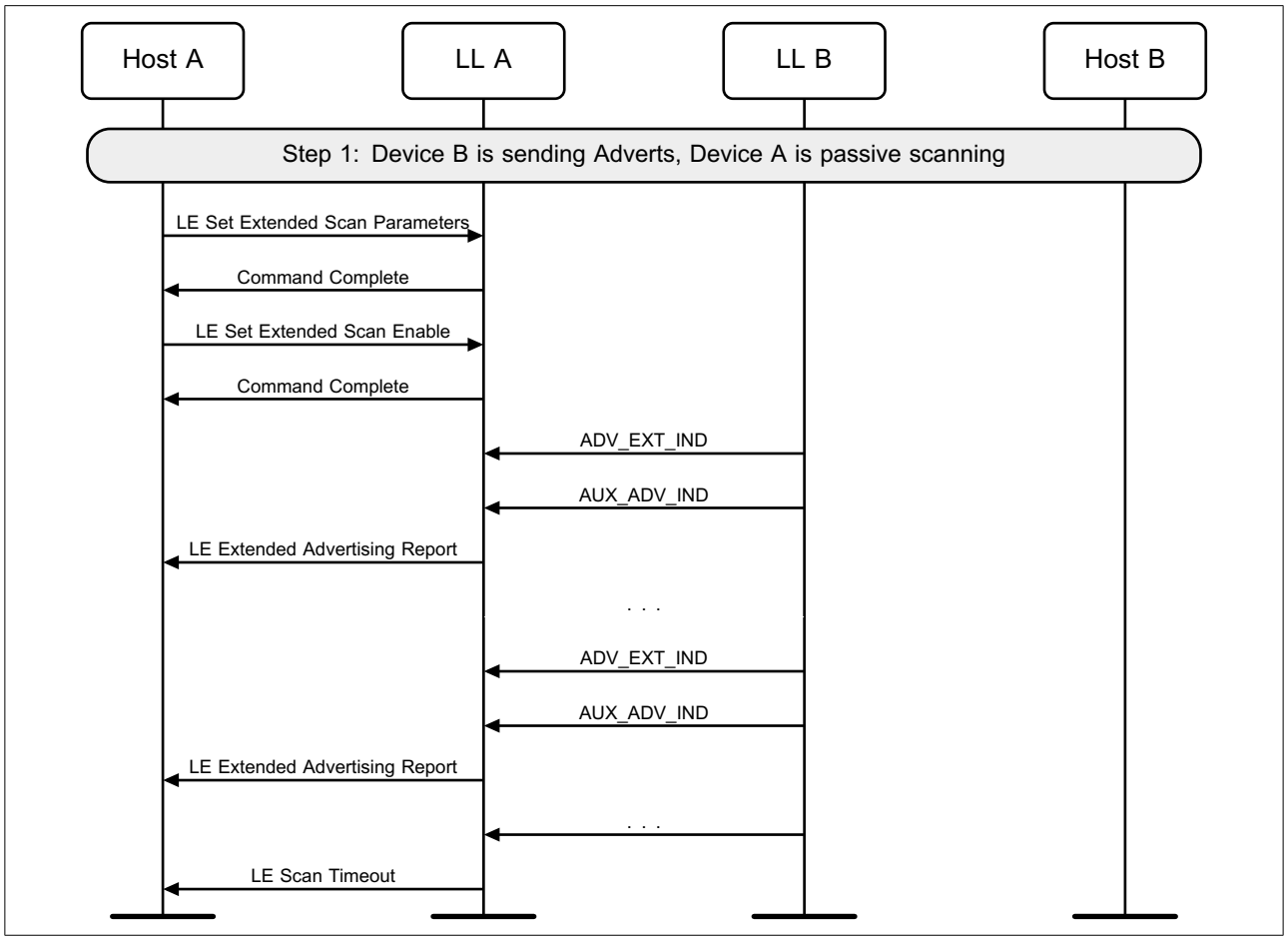



*Figure 4.7: Scan timeout*

## 4.8 Scanning for periodic advertisements

A device may establish synchronization with a periodic advertiser and report periodic advertising packets to the Host (see Figure 4.8).

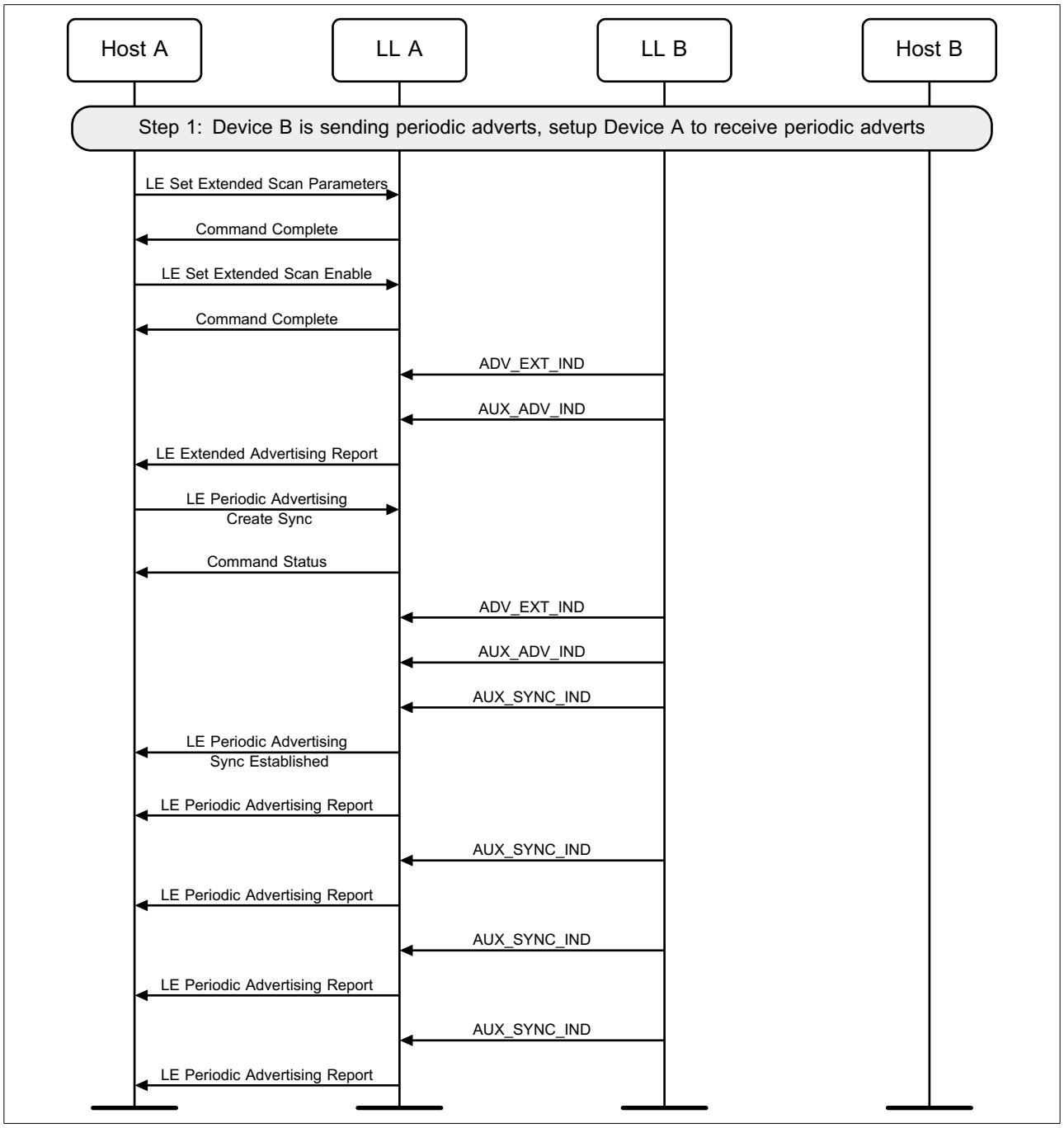



*Figure 4.8: Periodic scanning*

## 4.9 Cancel scanning for periodic advertisements

A device may cancel a pending request to establish synchronization with a periodic advertiser. This example shows an unsuccessful synchronization, followed by cancellation of the synchronization (see Figure 4.9).

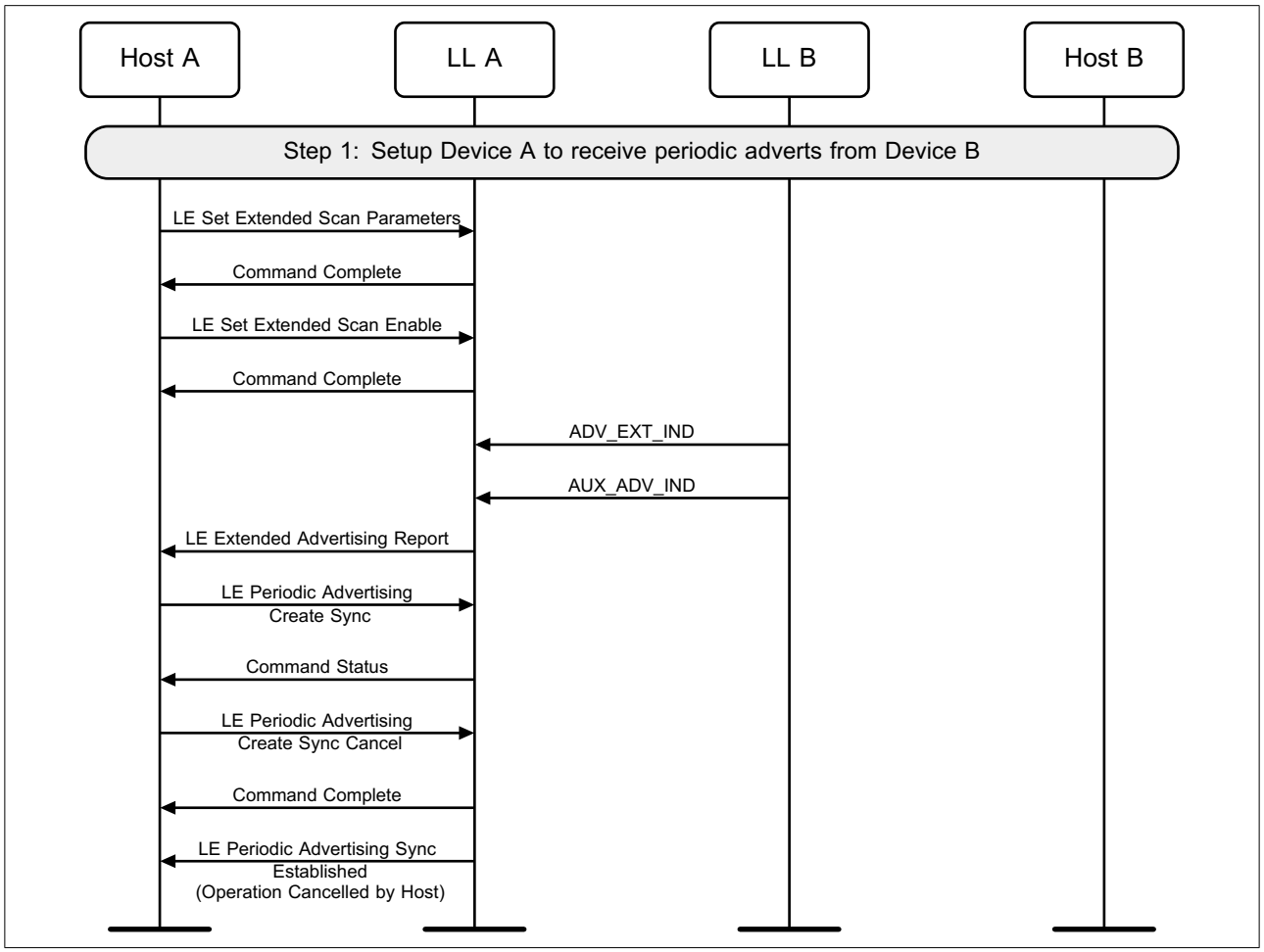



*Figure 4.9: Periodic scanning cancel*

## 4.10 Periodic advertising synchronization timeout

A device may lose synchronization with a periodic advertiser (see Figure 4.10).

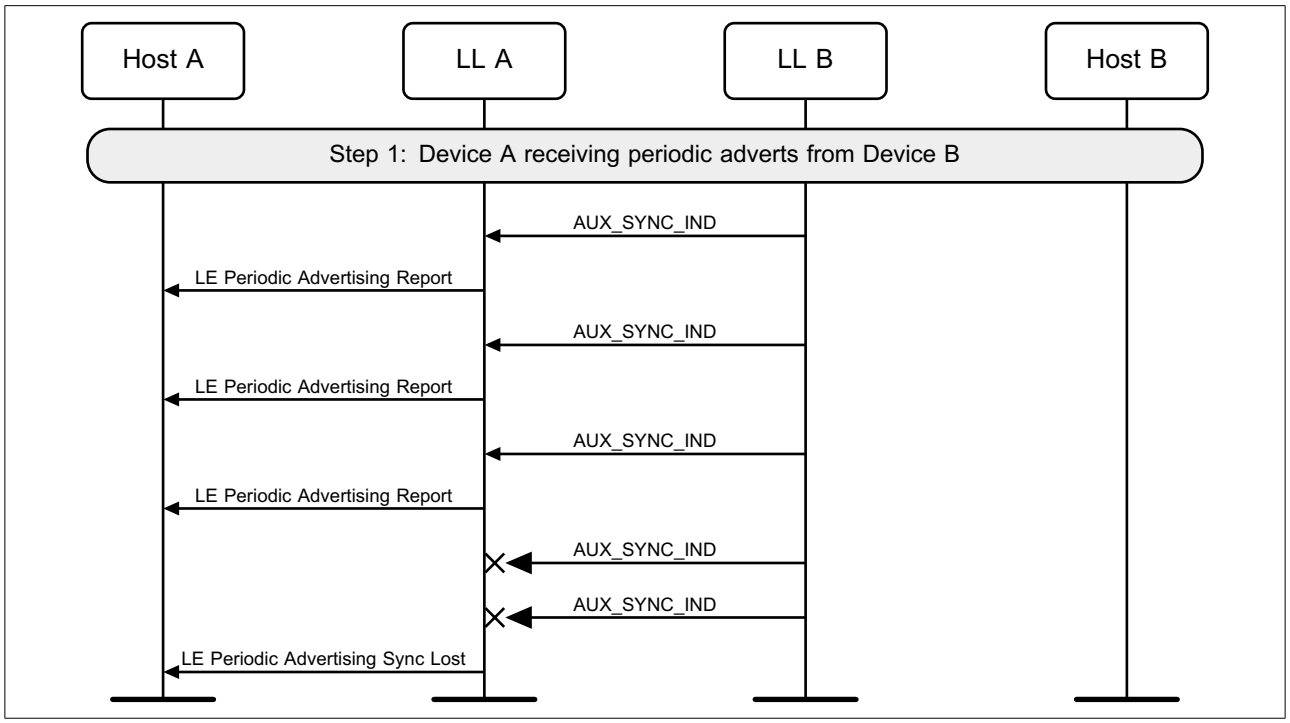



*Figure 4.10: Periodic scanning timeout*

## 4.11 Terminate reception of periodic advertising

Once synchronized with a periodic advertiser, the Host can terminate the synchronization (see Figure 4.11).

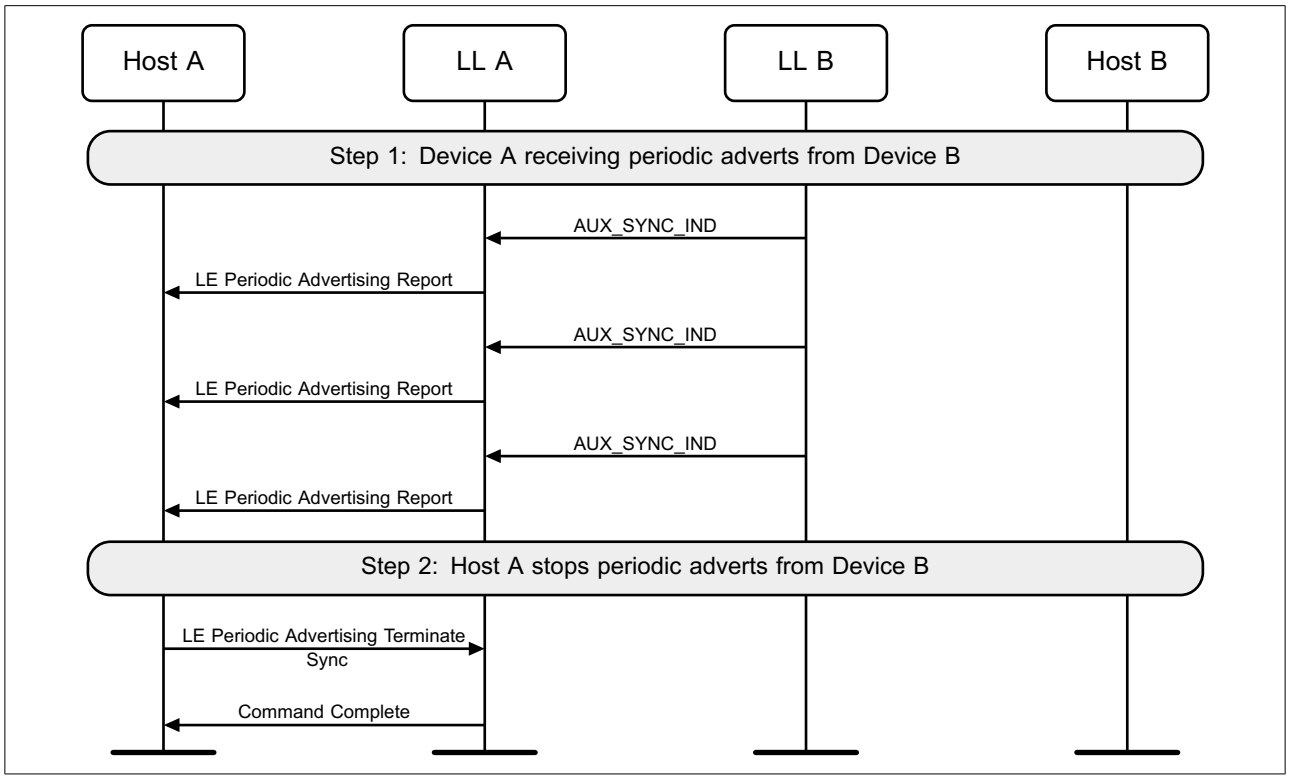



*Figure 4.11: Periodic scanning terminate*

## 4.12 Connectionless Constant Tone Extension reception

A device may receive periodic advertising packets containing a Constant Tone Extension and send IQ samples to the Host (see Figure 4.12).

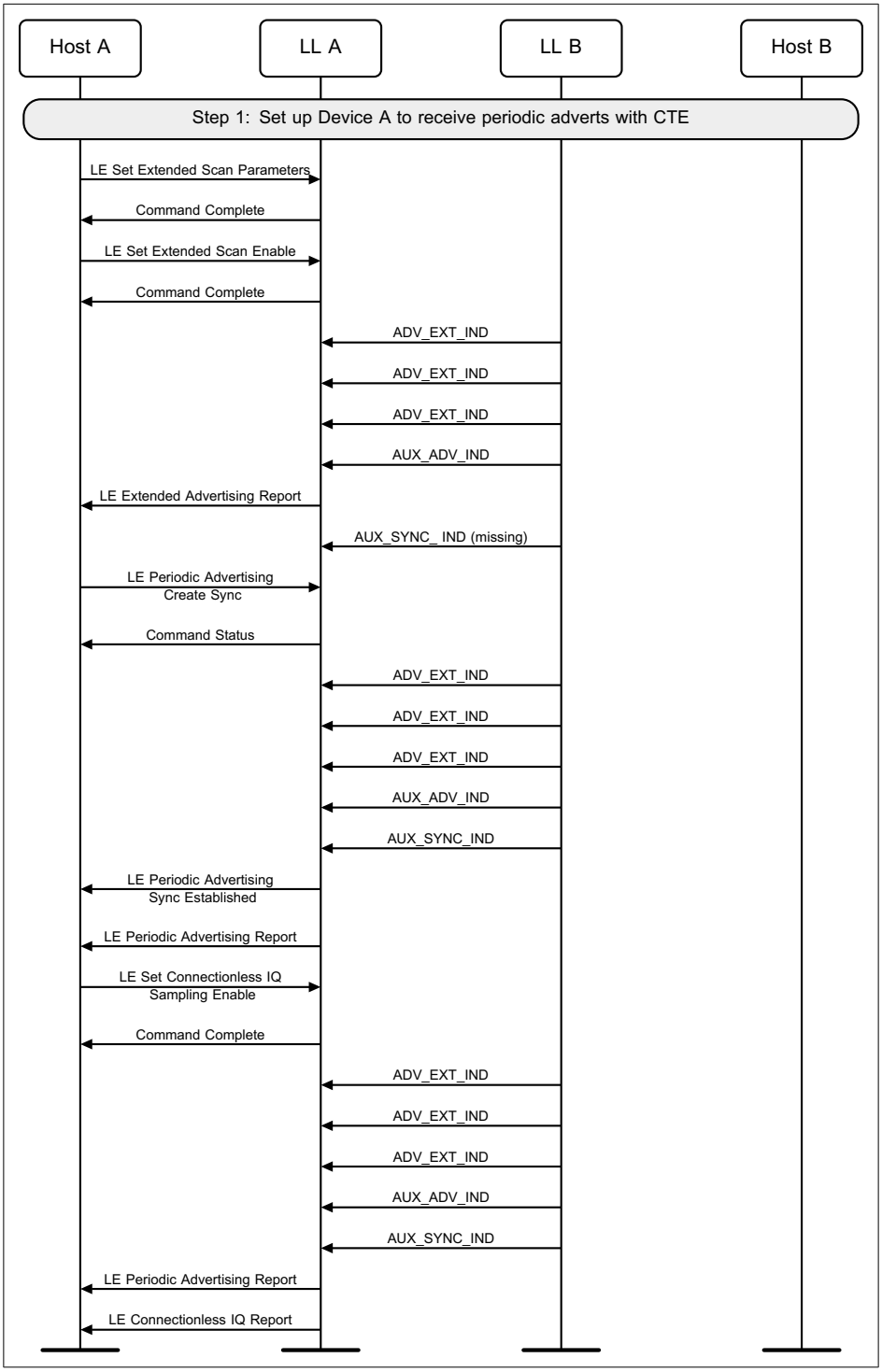



*Figure 4.12: Connectionless Constant Tone Extension reception*

## 4.13 Synchronization with separate enable of reports

A device may enable or disable reports after establishing synchronization with a periodic advertising train.

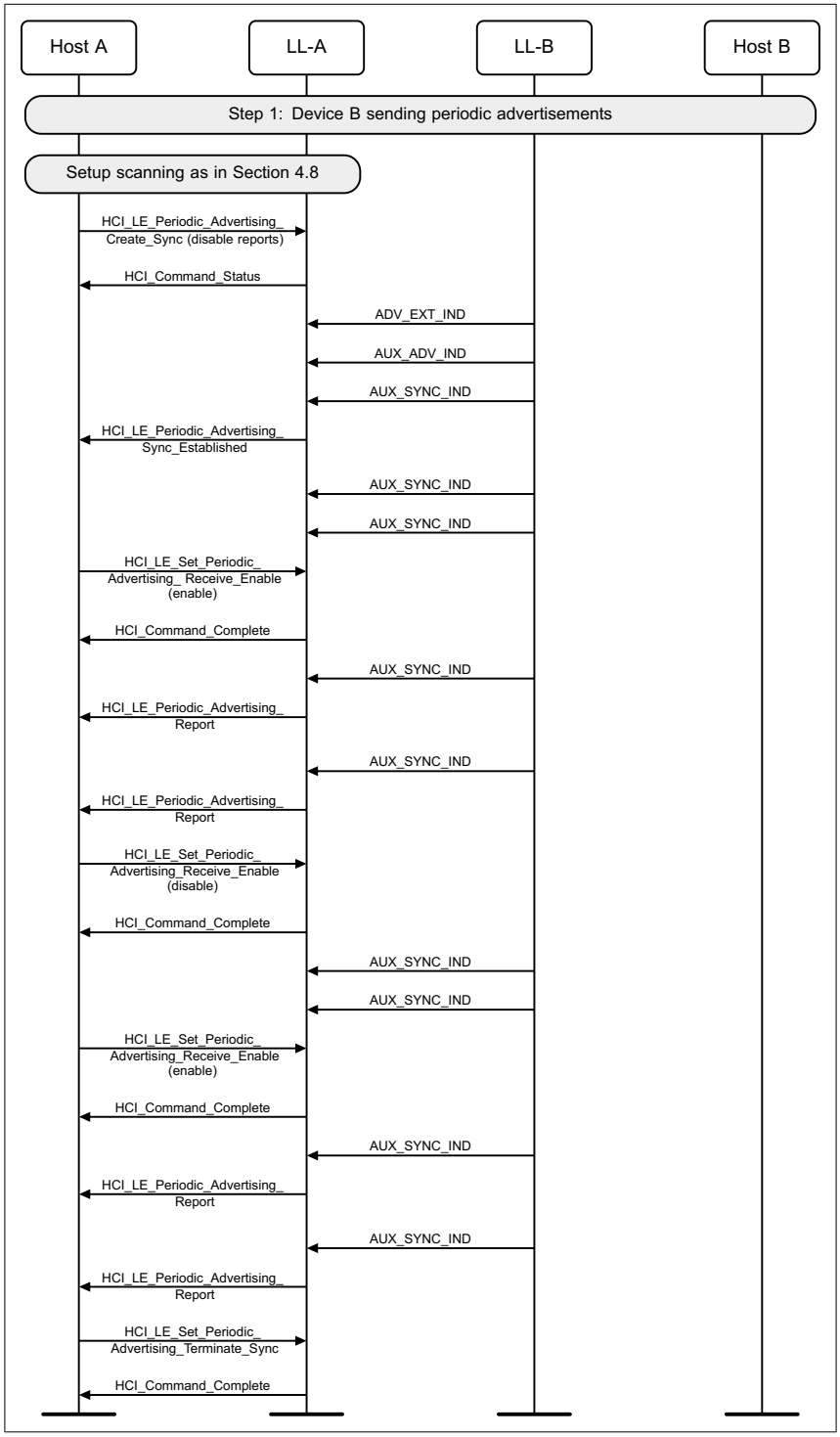



*Figure 4.13: Periodic scanning with separate enable*

# 5 INITIATING STATE

## 5.1 Initiating a connection

A device can initiate a connection to an advertiser. This example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.1).

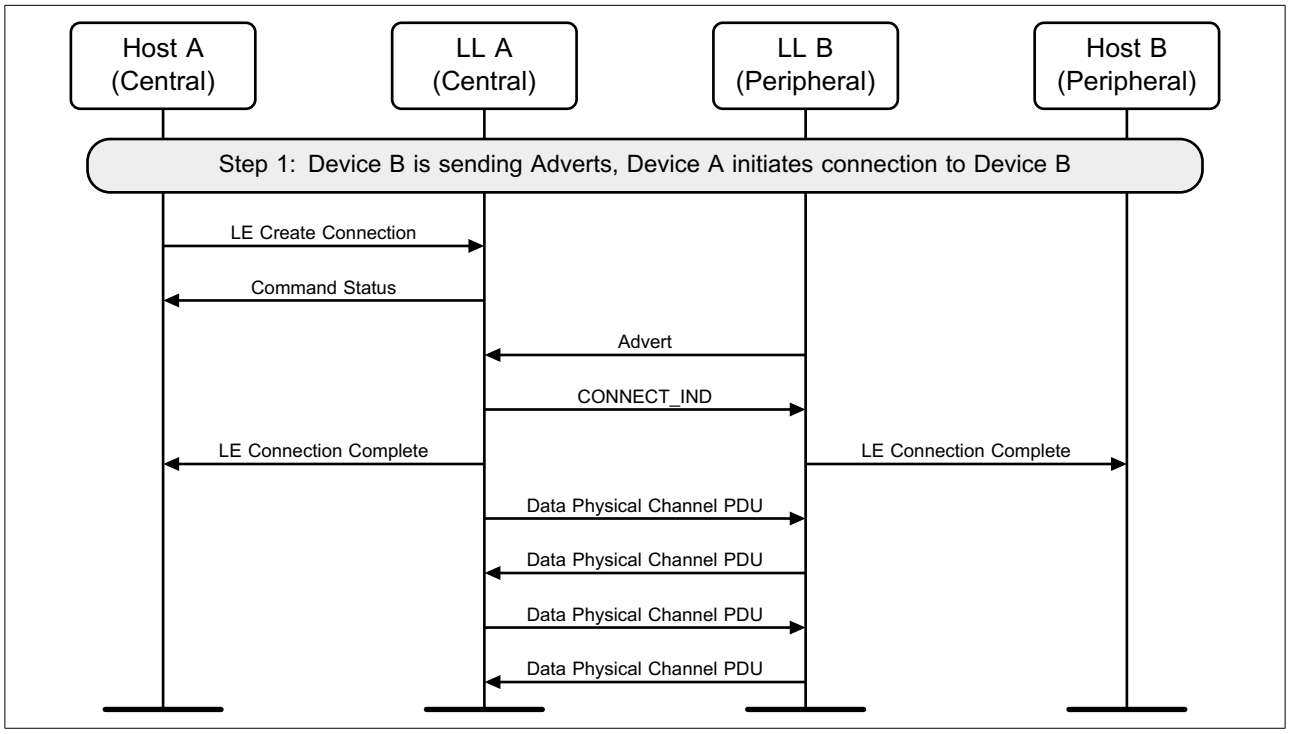



*Figure 5.1: Initiating a connection*

## 5.2 Canceling an initiation

A device can cancel a pending connection creation. This example shows an unsuccessful initiation, followed by a cancellation of the initiation (see Figure 5.2).

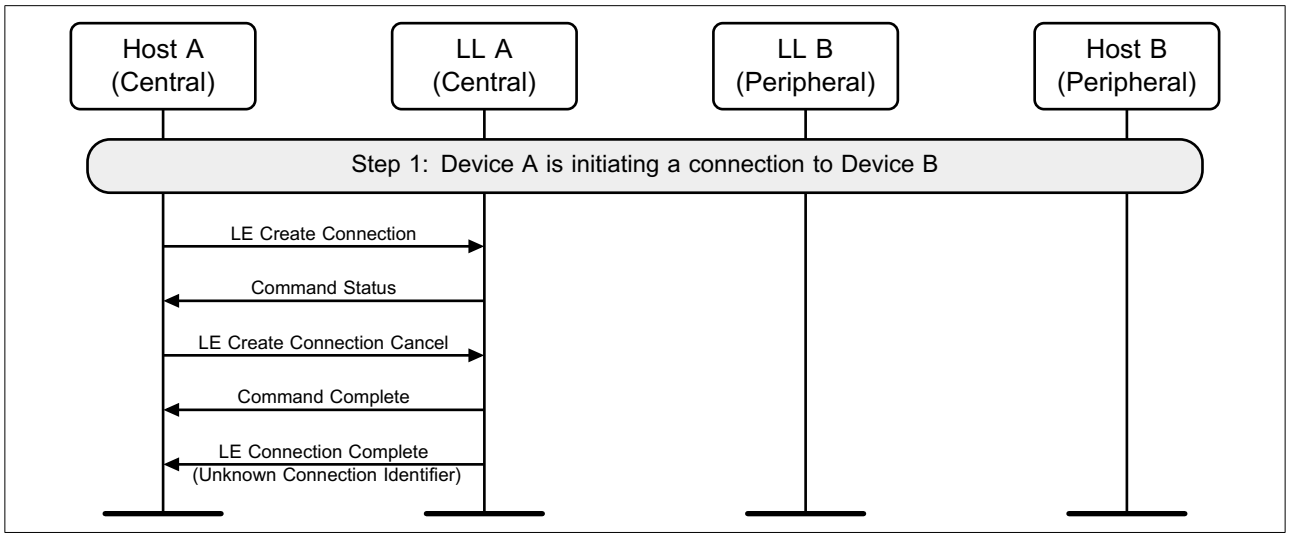



*Figure 5.2: Canceling an initiation*

## 5.3 Initiating a connection using undirected advertising with Privacy

A device can initiate a connection to an advertiser. Privacy may be used during connection initiation to make it more difficult to track either device during connection setup. The example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.3).

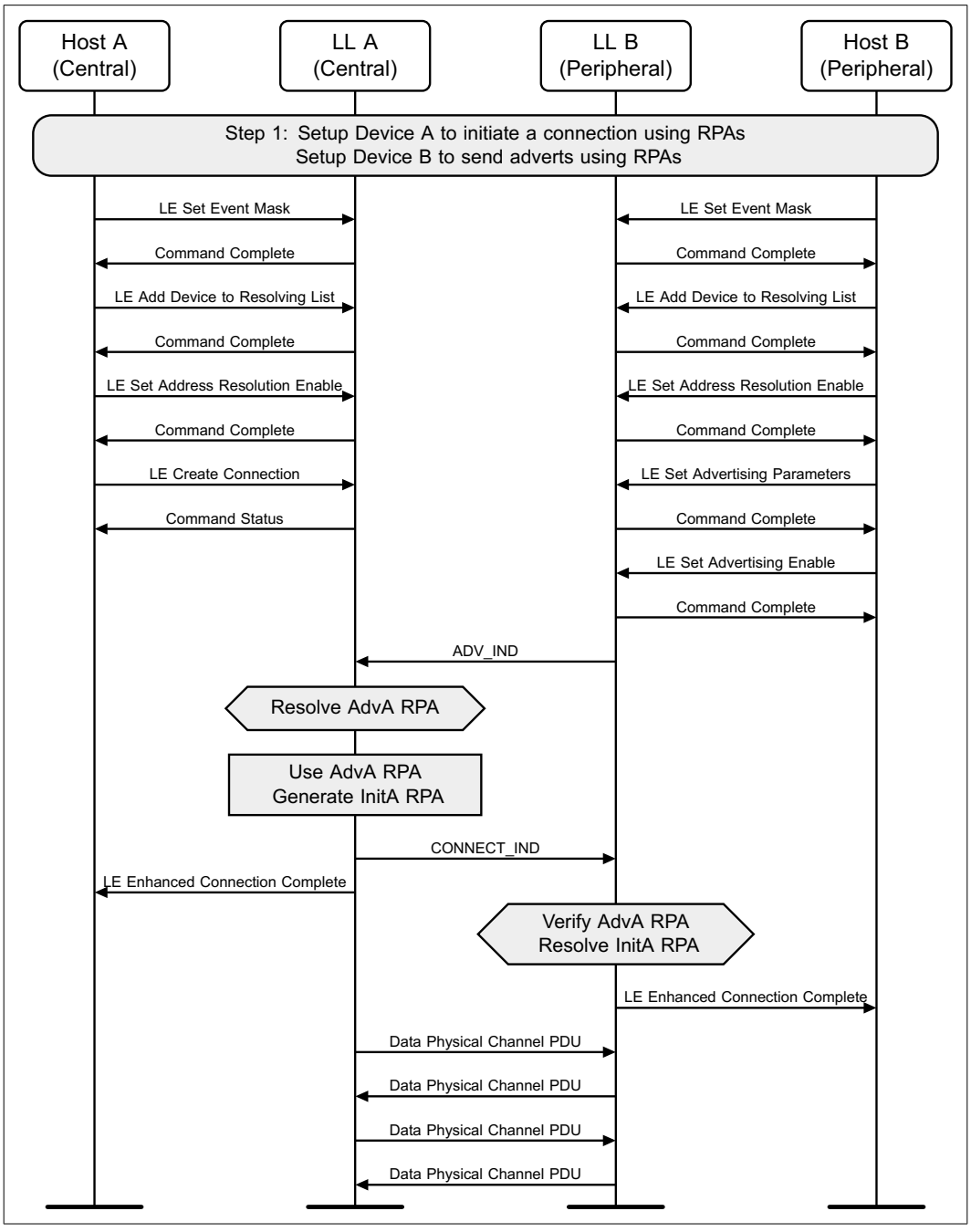



*Figure 5.3: Initiating a connection using undirected advertising with Privacy*

## 5.4 Initiating a connection using directed advertising with Privacy

A device can initiate a connection to an advertiser who is using Directed Advertising. Privacy may be used during connection initiation to make it more difficult to track either device during connection setup as well as target a single initiator. The example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.4).

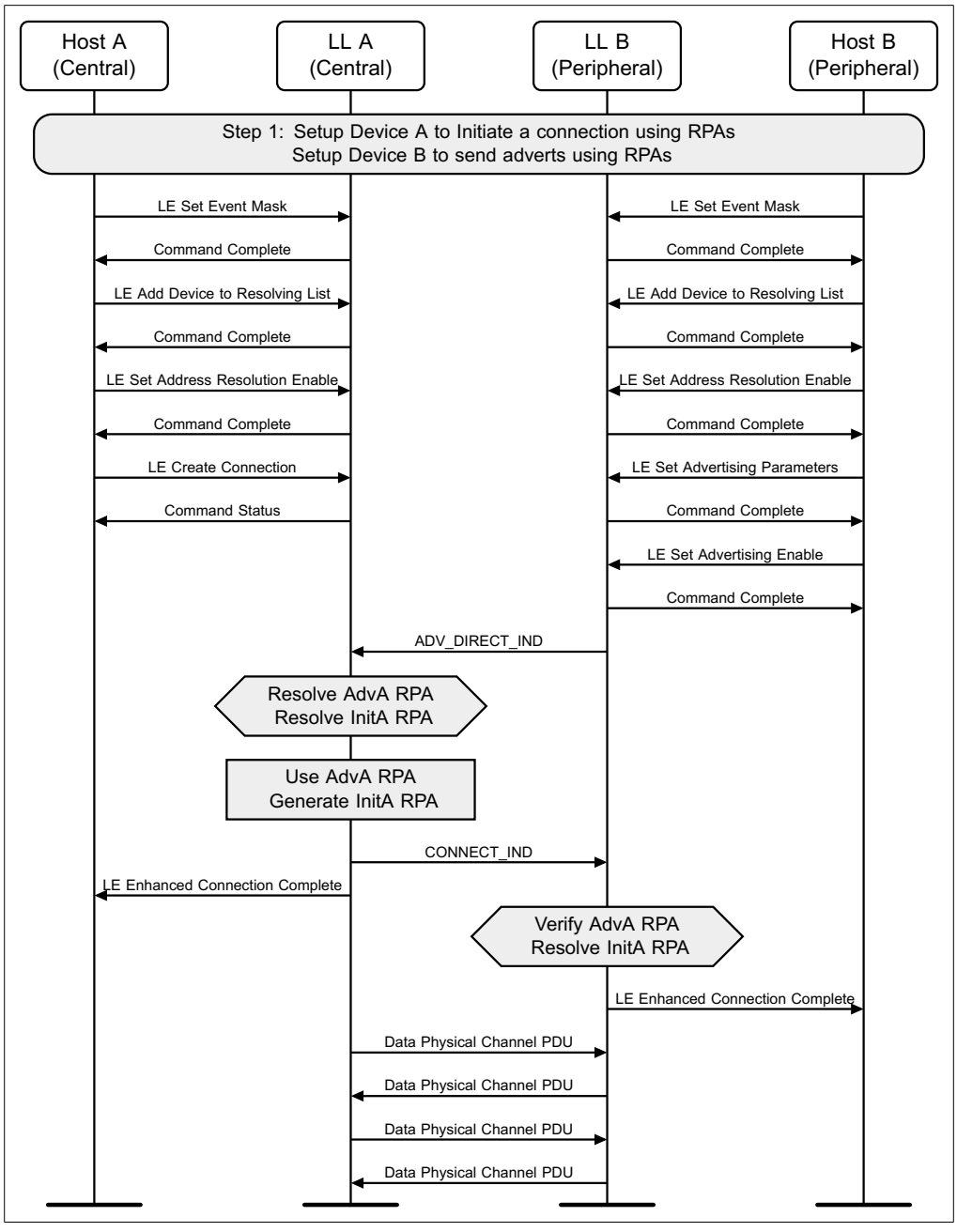



*Figure 5.4: Initiating a connection using directed advertising with Privacy*

## 5.5 Initiating a connection that fails to establish

This example shows an initiation that fails to establish because Device B (the advertiser) fails to respond to the Data Physical Channel PDUs sent by Device A.

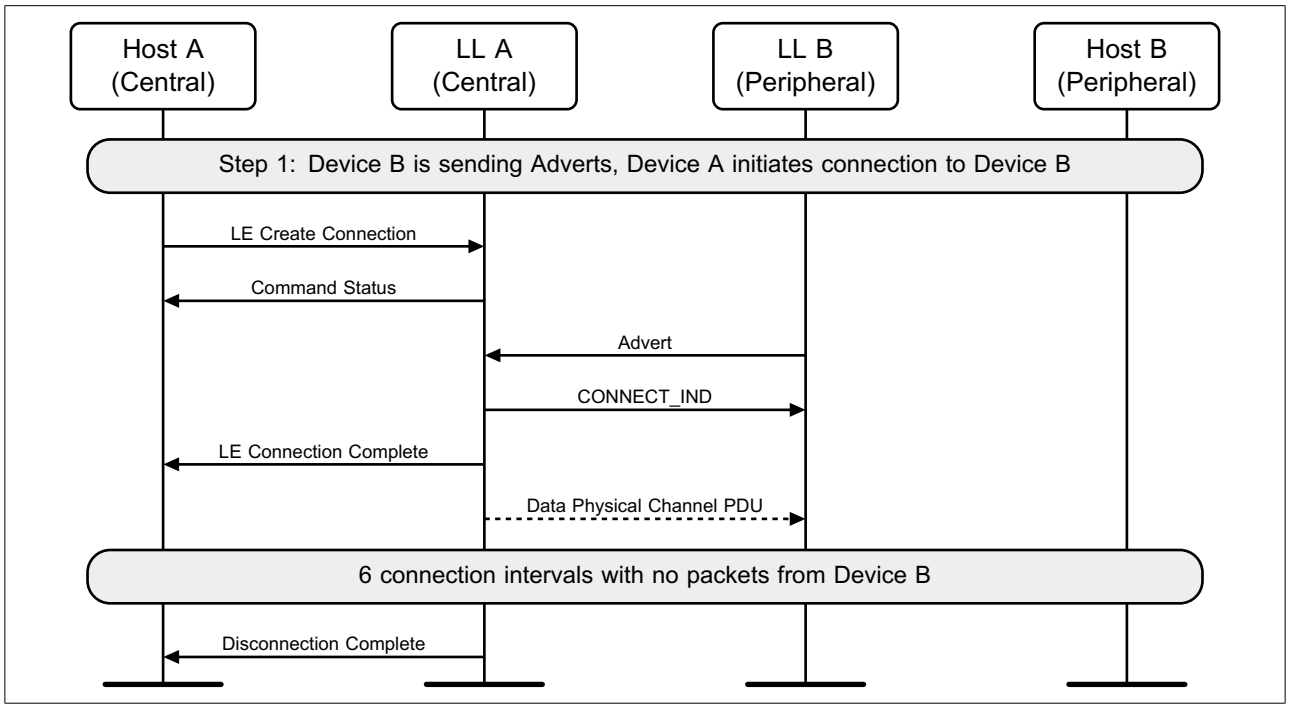



*Figure 5.5: Initiating a connection that fails to establish*

Device A is not required to send Data Channel PDUs in the 6 connection intervals before establishment fails. However, if it does not do so, Device B is unable to respond.

## 5.6 Initiating a connection on the secondary advertising physical channel

A device can initiate a connection to an advertiser on the secondary channel. This example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.6).

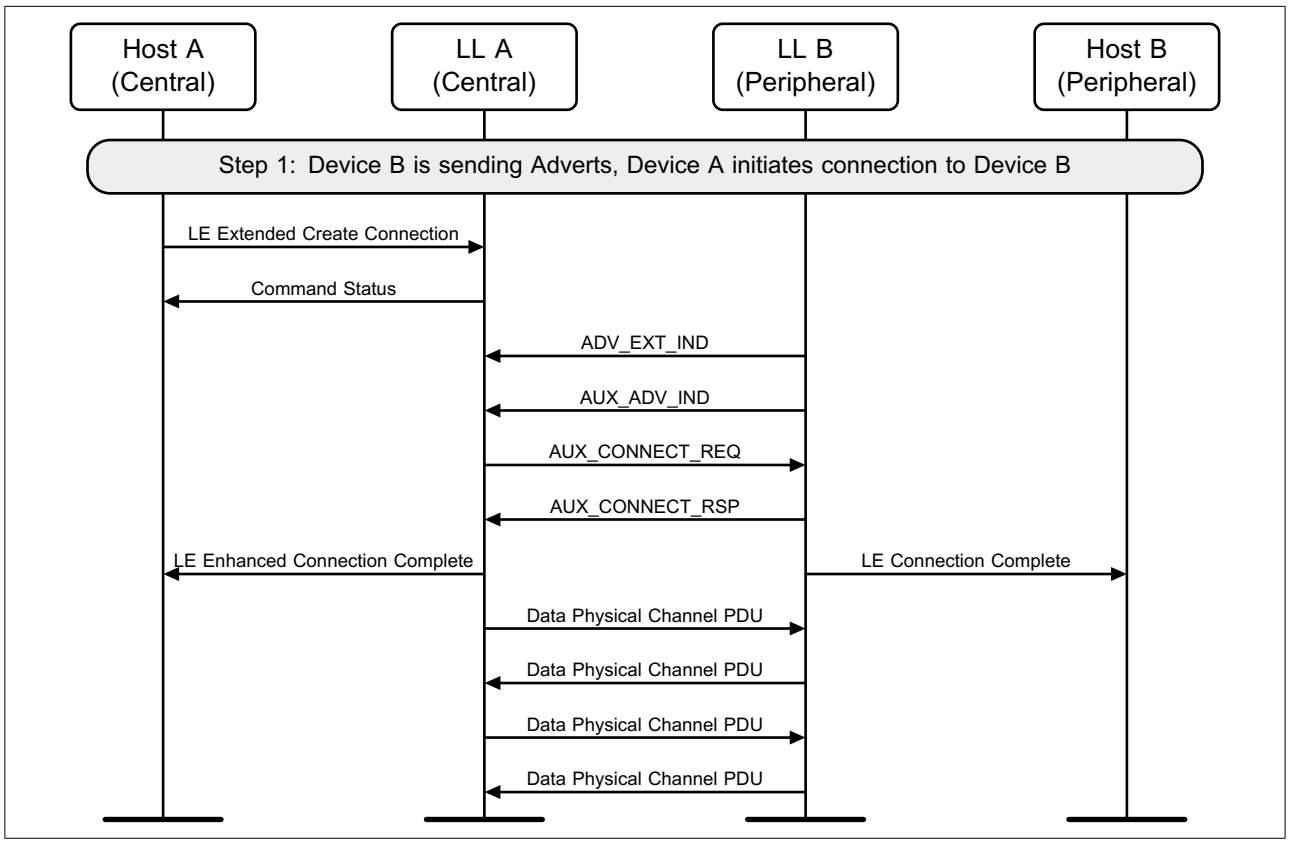



*Figure 5.6: Initiating a connection on the secondary advertising physical channel*

## 5.7 Initiating a Channel Selection algorithm #2 connection

Where a device supports the Channel Selection Algorithm #2 feature, it can initiate a connection which will use Channel Selection Algorithm #2 to an advertiser who has the ChSel field of the advertising physical channel PDU set to 1. The example shows a successful initiation, resulting in the connection using Channel Selection Algorithm #2.

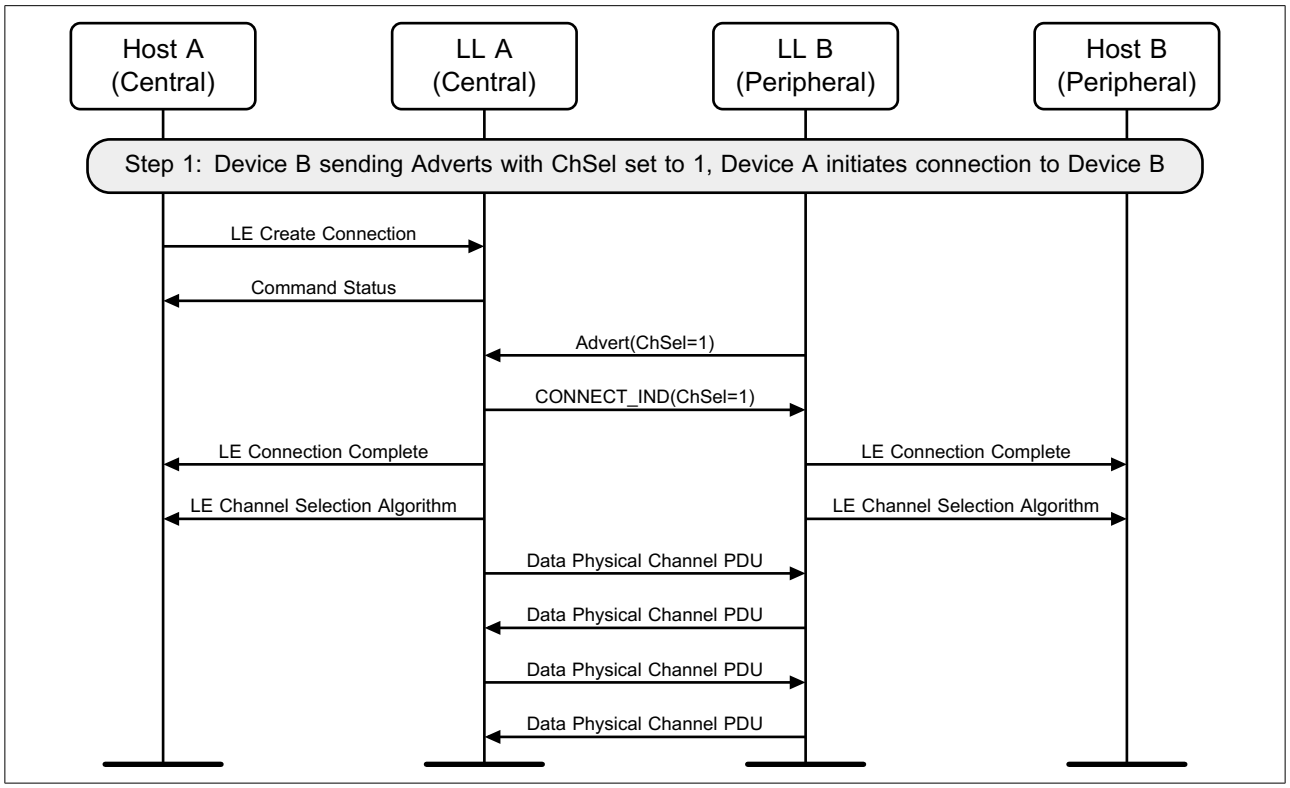



*Figure 5.7: Initiating a Channel Selection algorithm #2 connection*

## 5.8 Initiating a connection using an advertising set

A device can initiate a connection to an advertiser using an Extended Advertising advertising set. This causes additional events to be generated (see Figure 5.8). The LE Channel Selection Algorithm event can be generated irrespective of which algorithm is used by the connection.

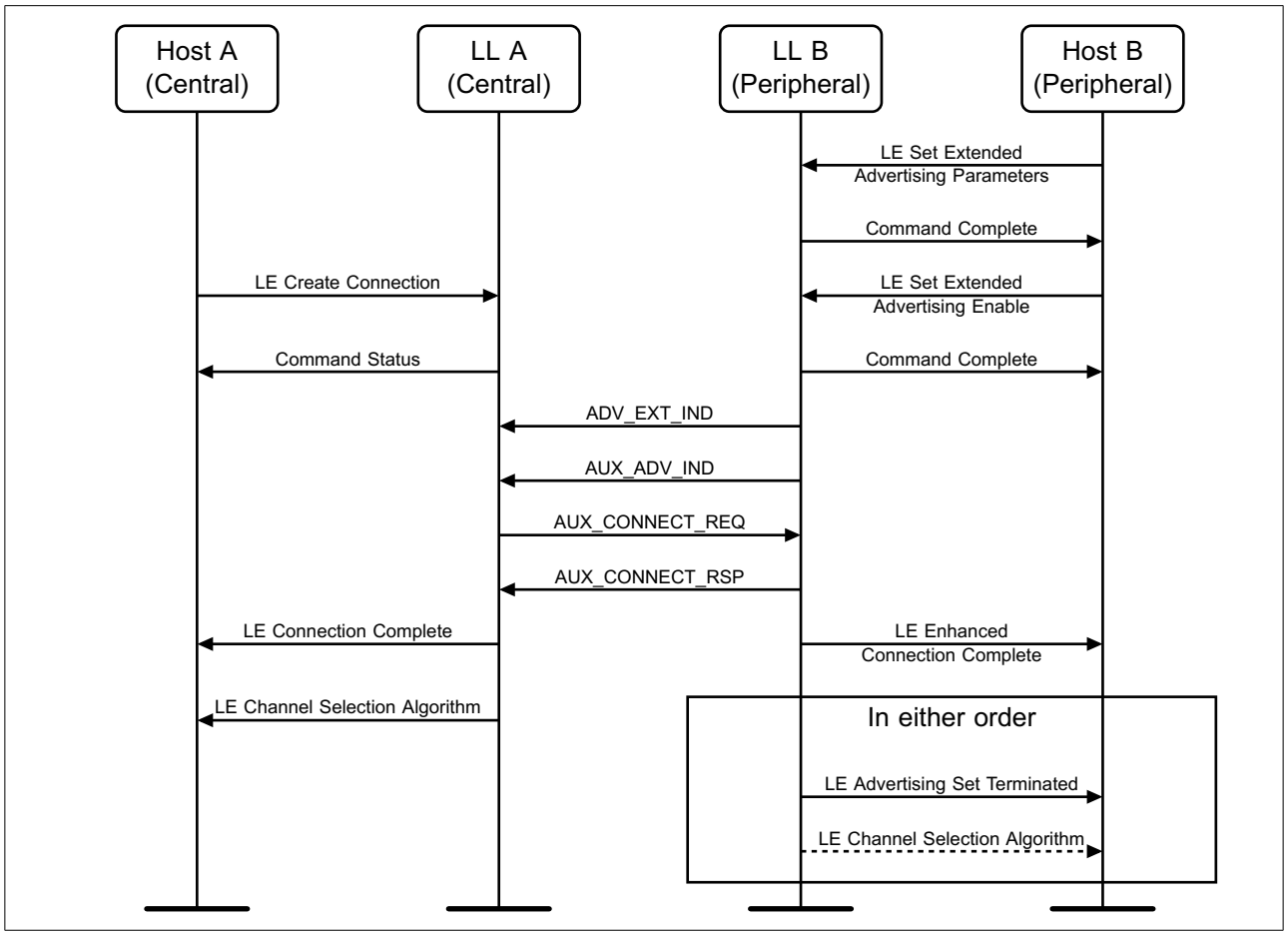



*Figure 5.8: Initiating a connection using an advertising set*

# 6 CONNECTION STATE

## 6.1 Sending data

Once two devices are in a connection, either device can send data. This example shows both devices sending data, for example when the Attribute Protocol does a read request and a read response is returned (see Figure 6.1).

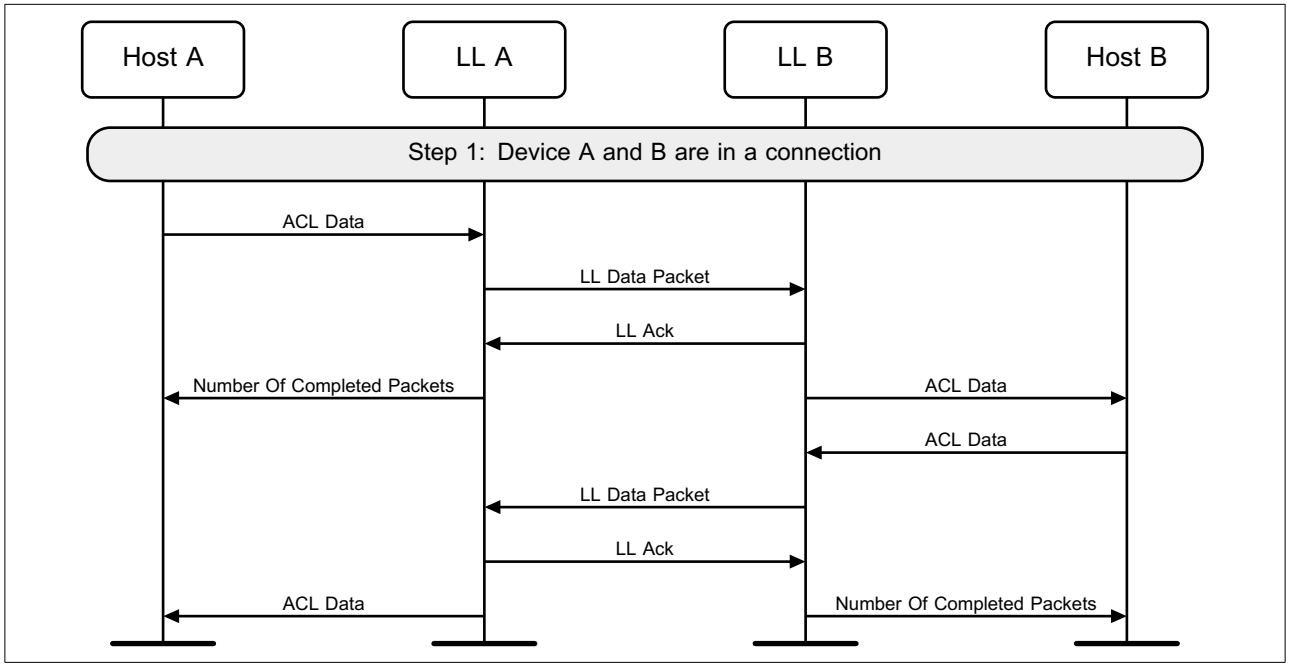



*Figure 6.1: Sending data*

## 6.2 Connection update

The Central of the connection may request a connection update using a Link Layer control procedure (see Figure 6.2).

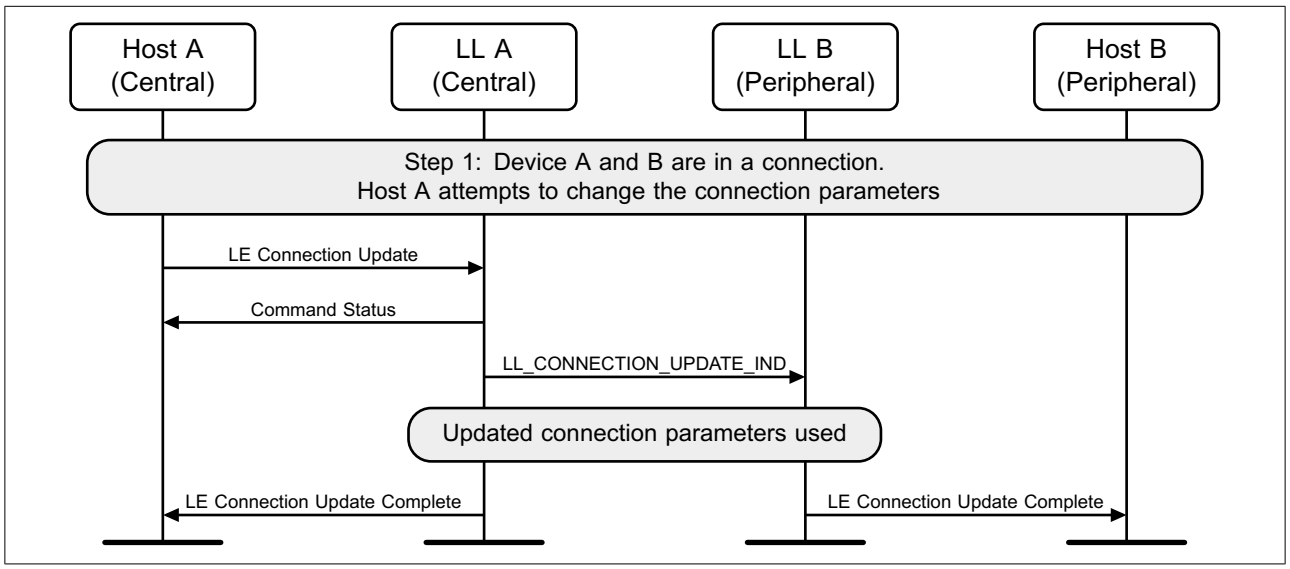



*Figure 6.2: Connection update*

## 6.3 Channel map update

The Controller of the Central may receive some channel classification data from the Host and then perform the Channel Update Link Layer Control procedure (see Figure 6.3).

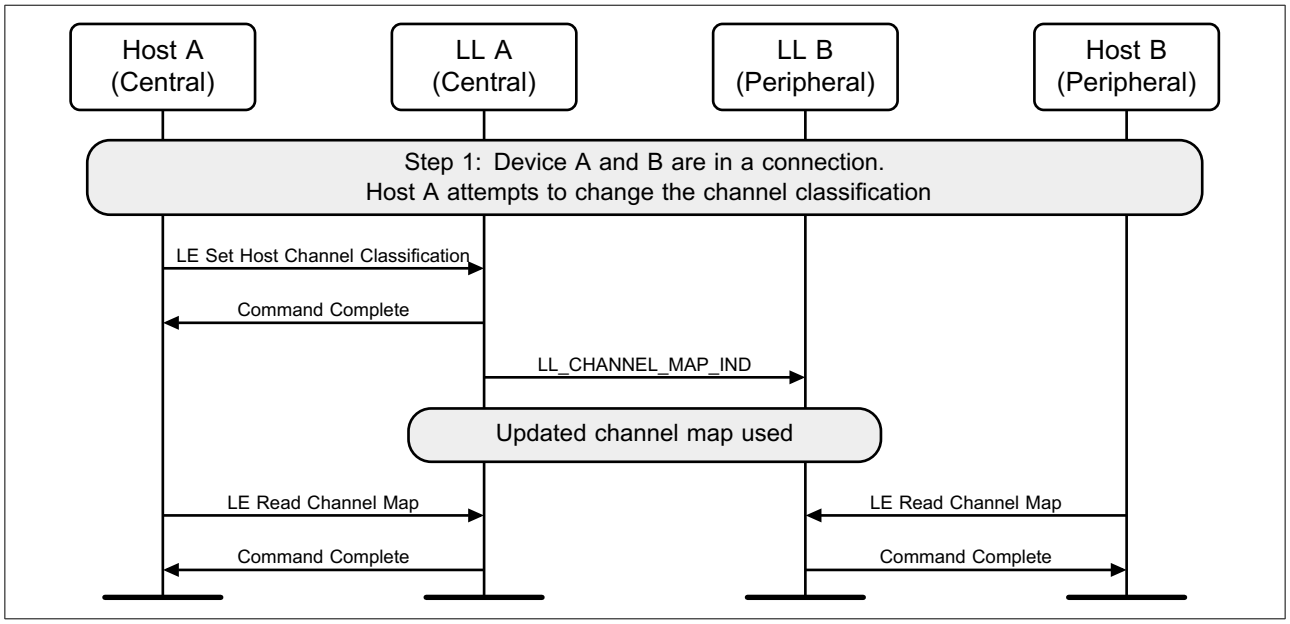



*Figure 6.3: Channel map update*

## 6.4 Features exchange

Both the Central and Peripheral can discover the set of features available on the remote device. To just fetch page 0, the Feature Exchange Link Layer Control procedure is used (see Figure 6.4 and Figure 6.5).

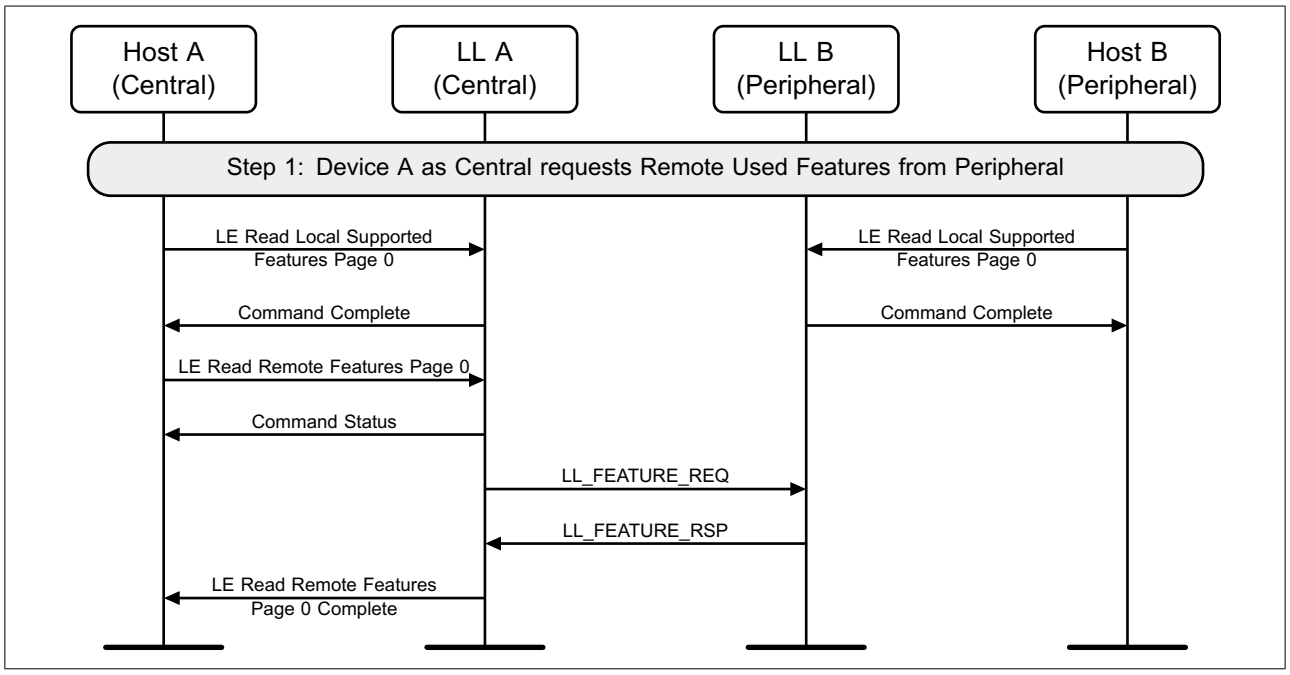



*Figure 6.4: Central-initiated features exchange*

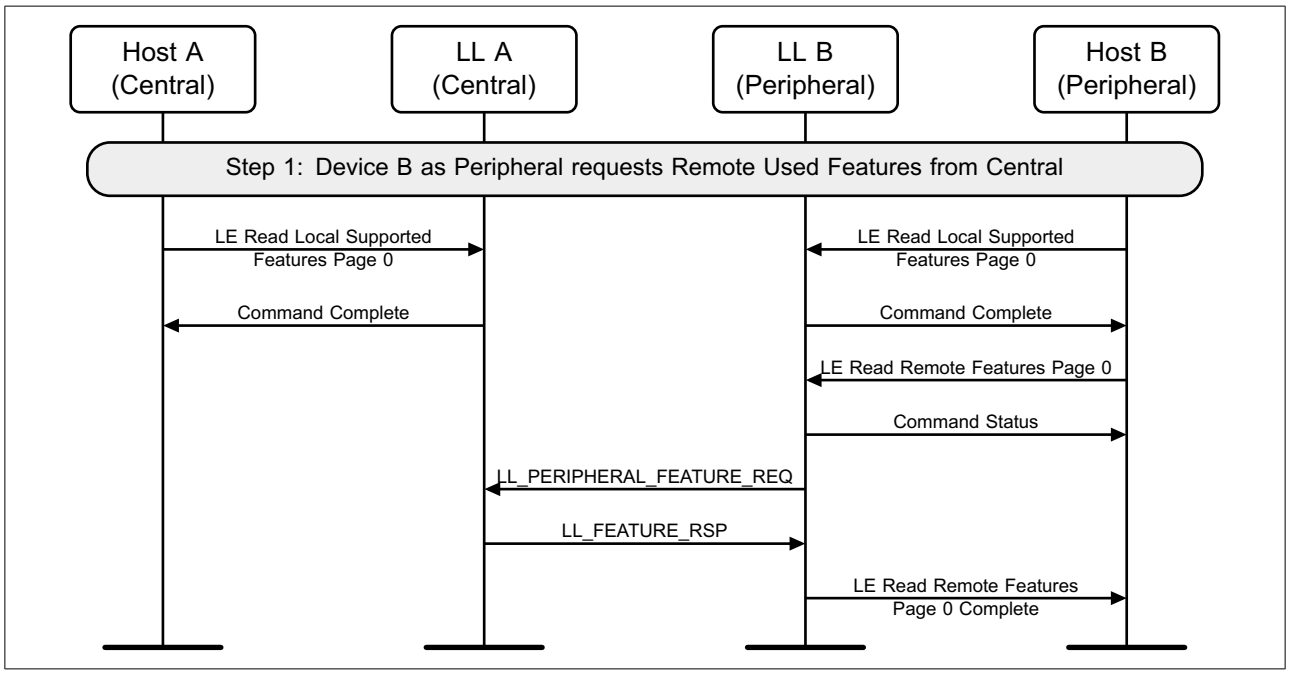



*Figure 6.5: Peripheral-initiated features exchange*

*Message Sequence Charts*

To fetch the entire feature list, this procedure is combined with 10 uses of the Feature Page Exchange procedure (see Figure 6.6).

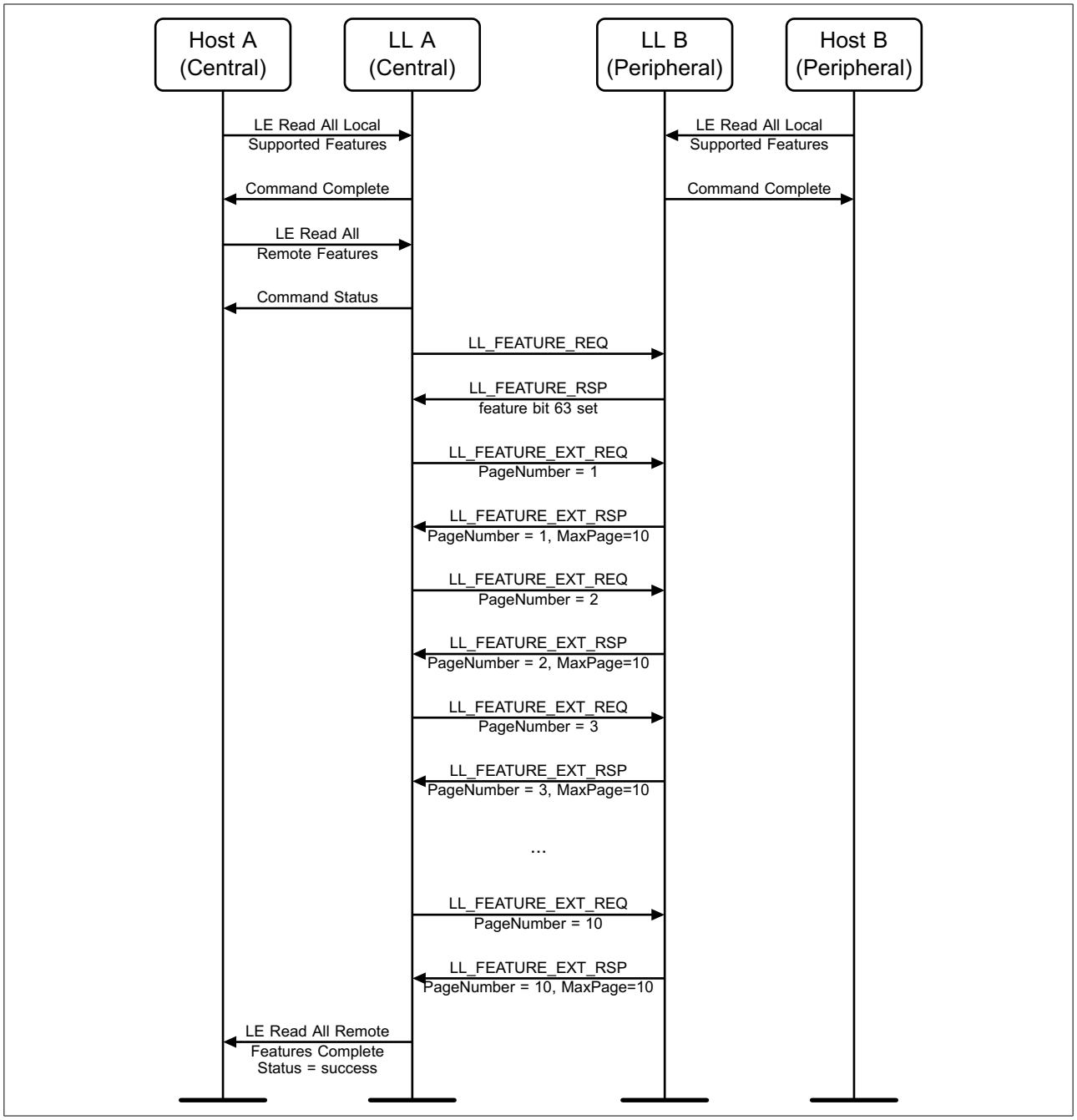



*Figure 6.6: Complete feature exchange*

If the peer only supports features in pages 0 to 2, then the Feature Page Exchange procedure need only be used twice (see Figure 6.7).

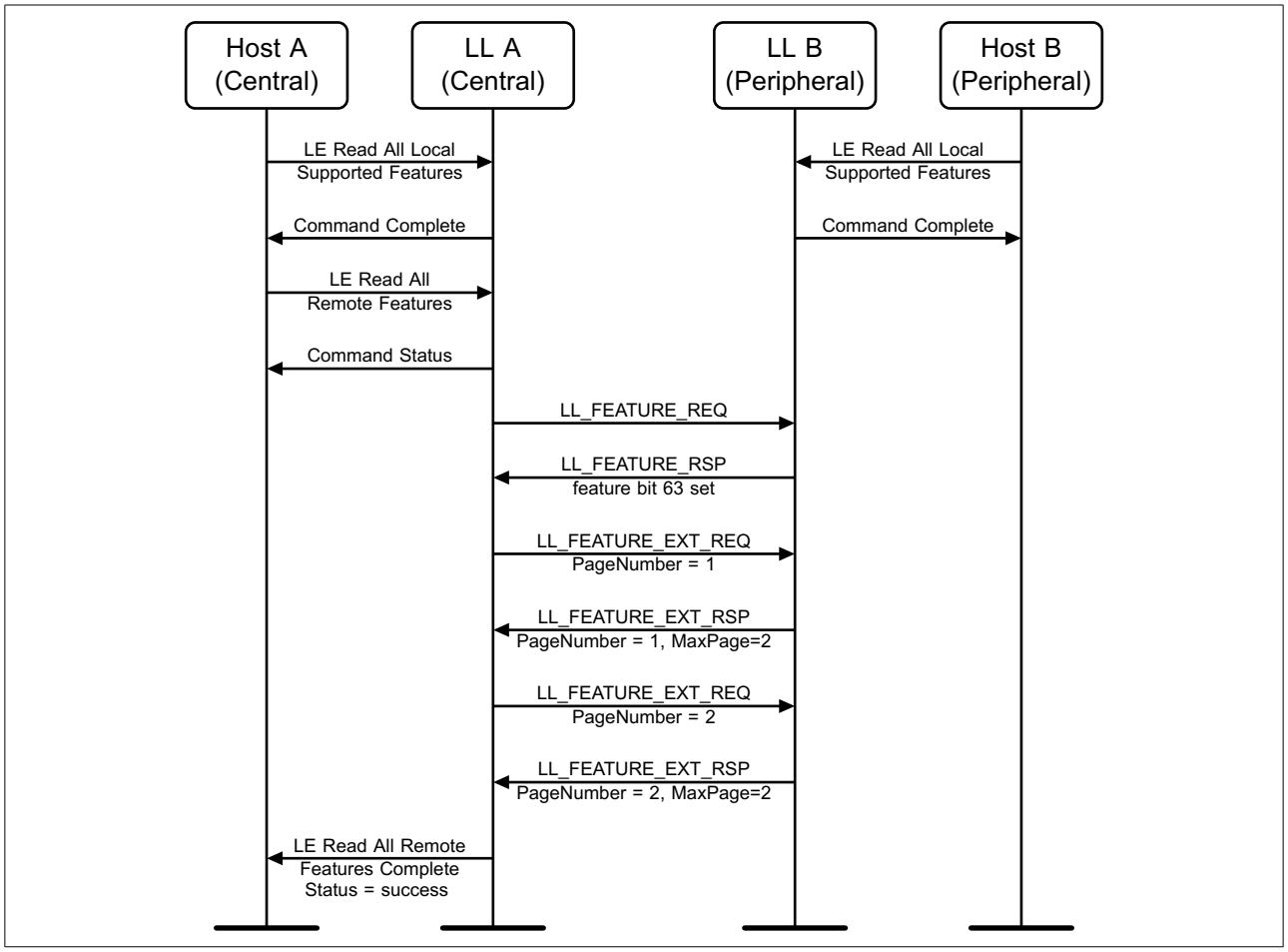



*Figure 6.7: Complete feature exchange where only pages 0 to 2 are used*

If the Peripheral does not support the Feature Page Exchange procedure, then the command has finished successfully as soon as it determines that it does not support the new procedure (see Figure 6.8).

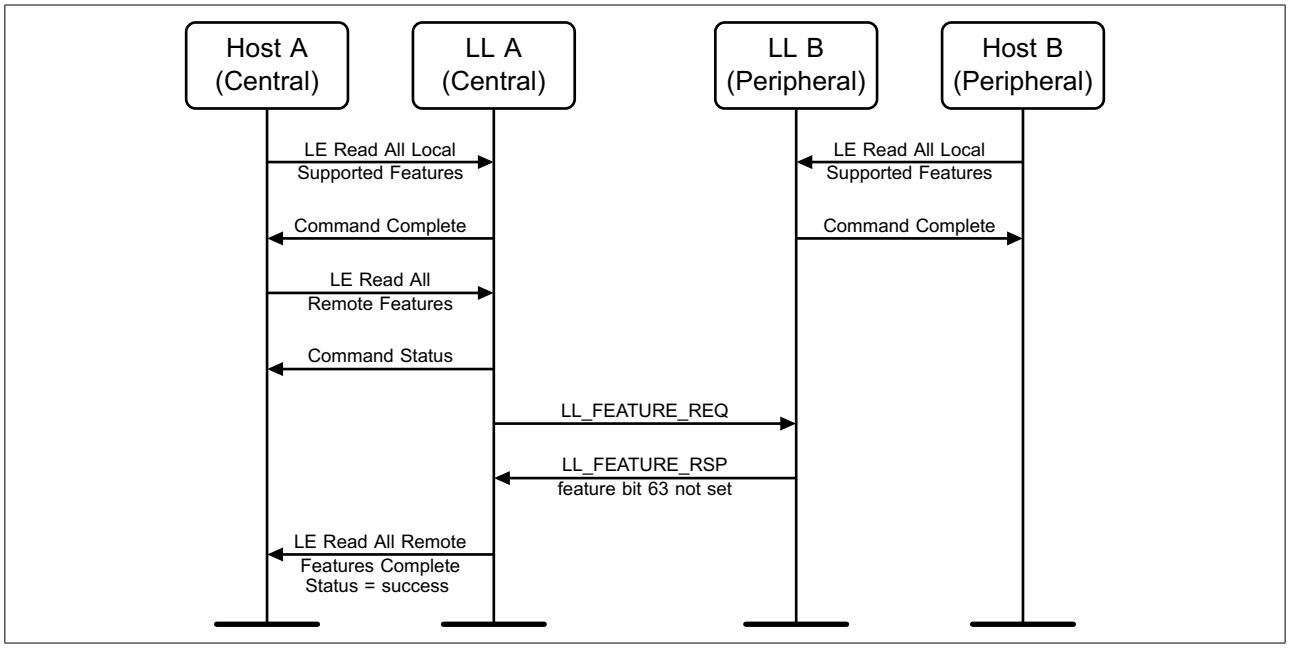



*Figure 6.8: Complete feature exchange where the Peripheral does not support the new procedure*

## 6.5 Version exchange

Either device may perform a Version Exchange procedure (see Figure 6.9 and Figure 6.10).

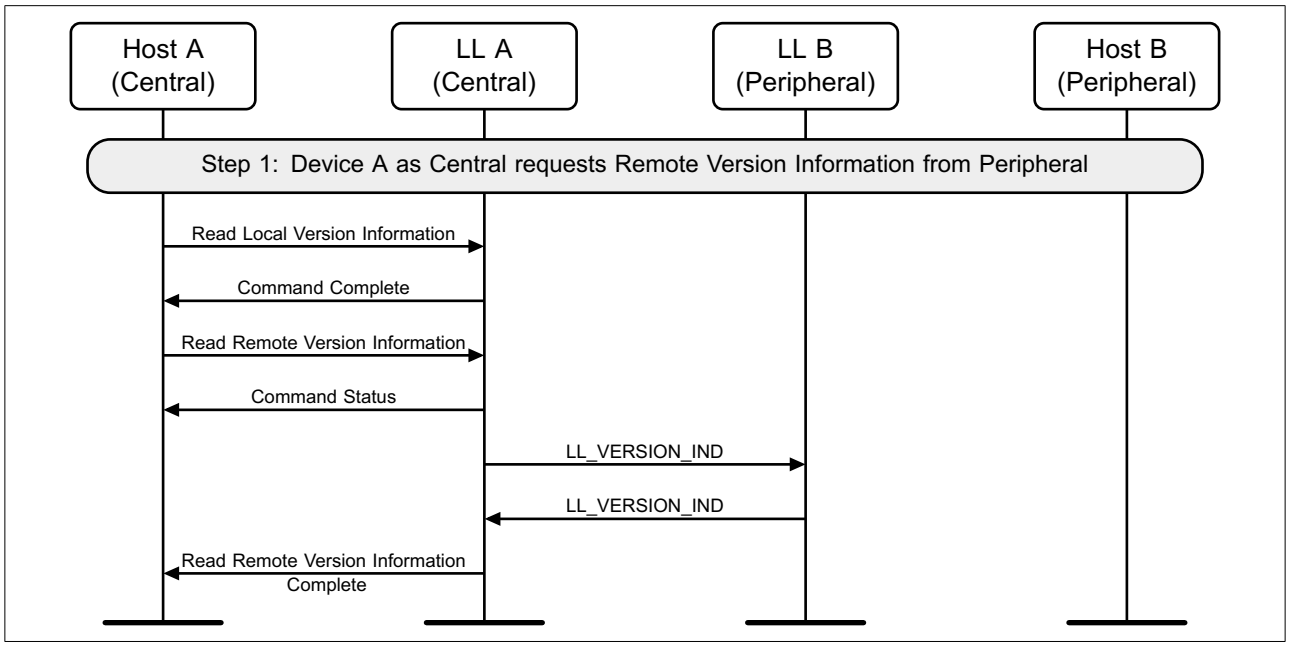



*Figure 6.9: Version exchange from Central*

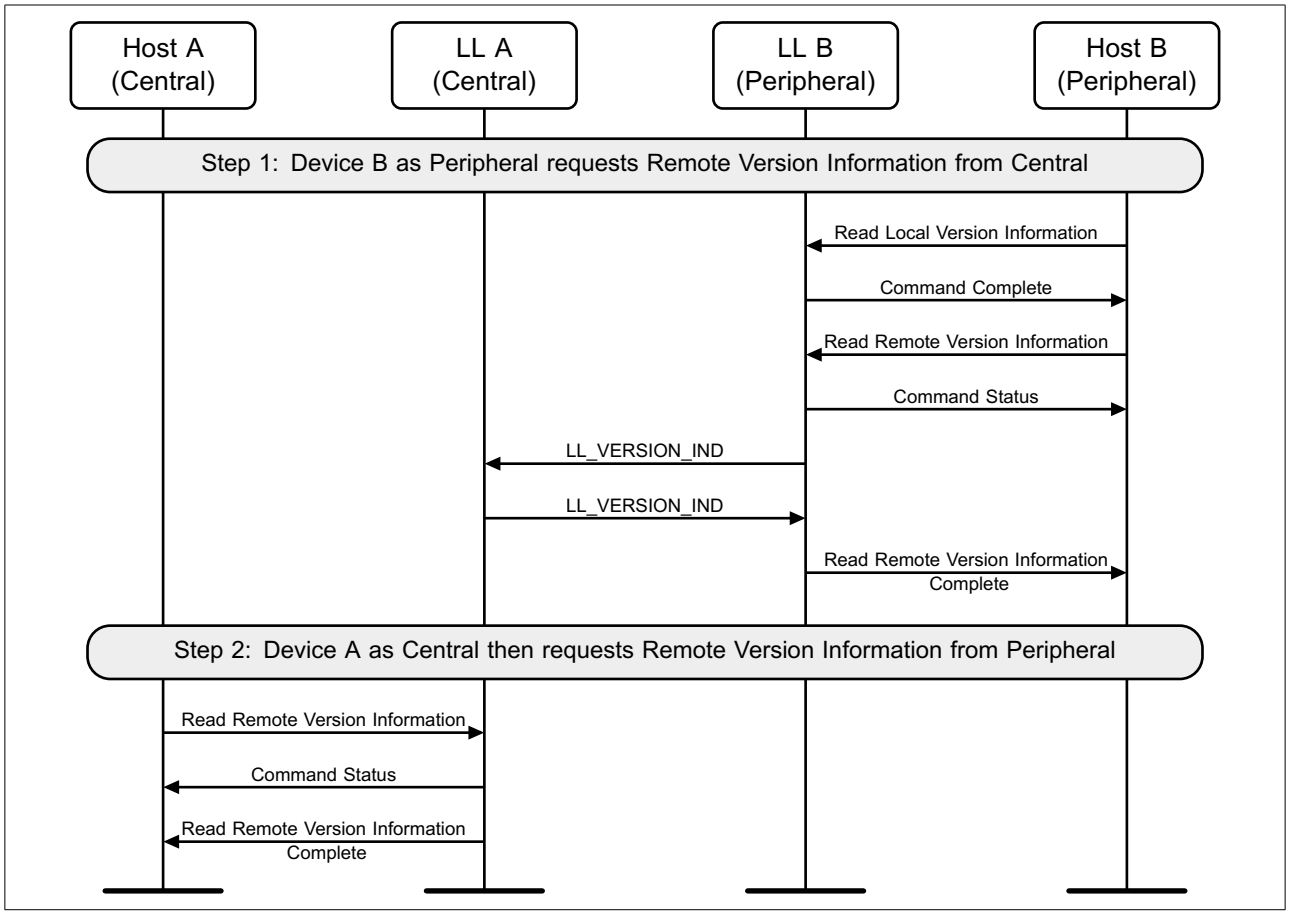



*Figure 6.10: Version exchange from Peripheral*

## 6.6 Start encryption

If encryption has not been started on a connection, it may be started by the Central (see Figure 6.11).

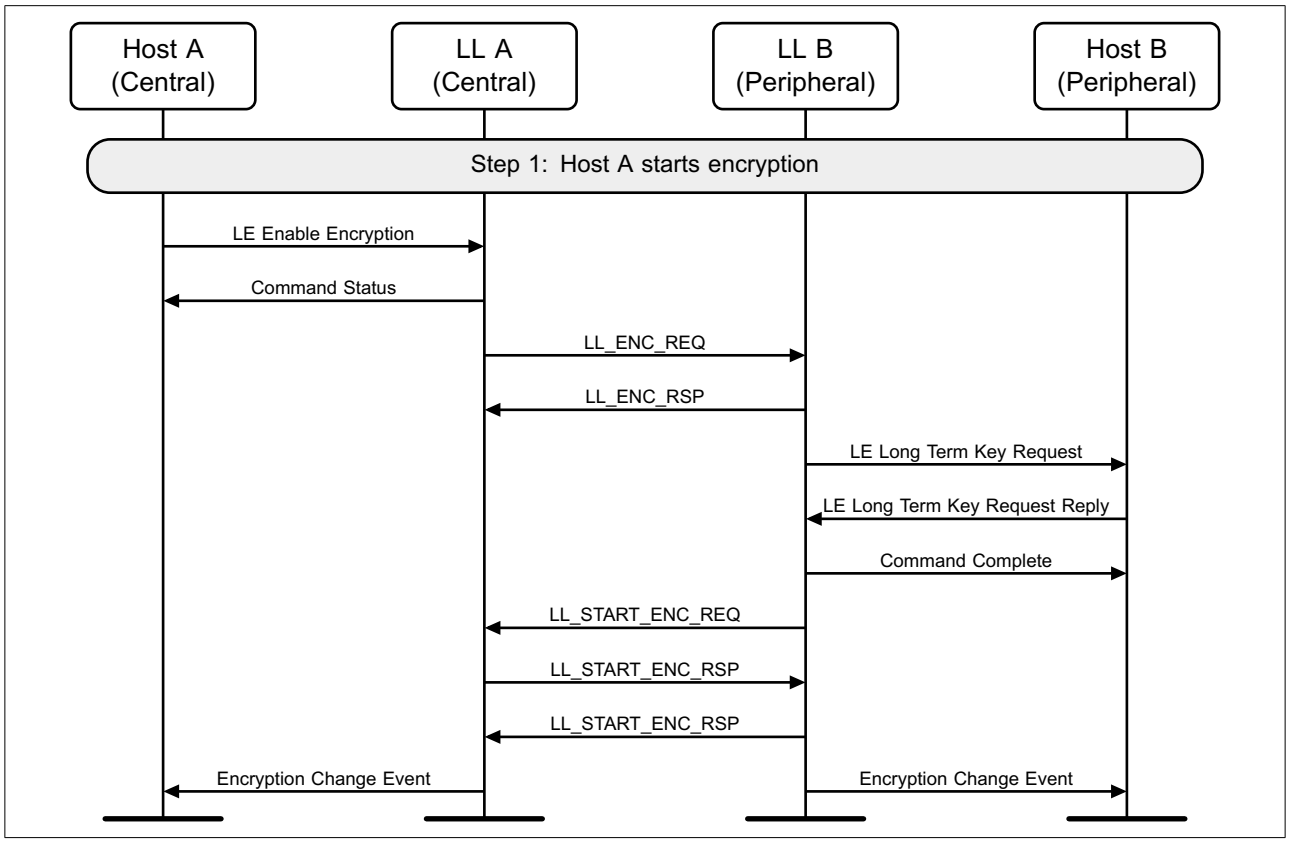



*Figure 6.11: Start encryption*

## 6.7 Start encryption without Long Term Key

If encryption has not been started on a connection, it may be started by the Central. Figure 6.12 shows the failure case of the Peripheral not having the long term key for the Central.

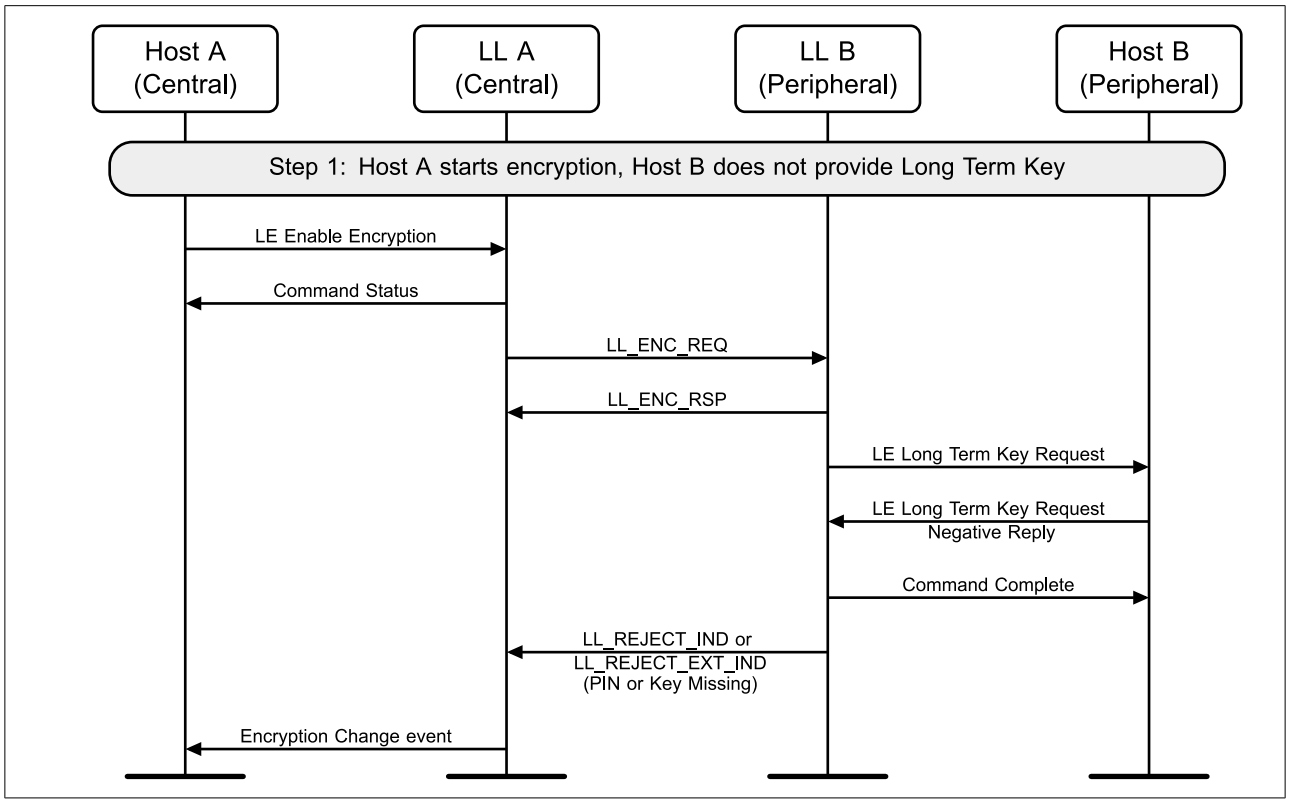



*Figure 6.12: Start encryption without Long Term Key*

## 6.8 Start encryption with event masked

If encryption has not been started on a connection, it may be started by the Central. Figure 6.13 shows the failure case when the Peripheral has masked out the HCI_LE_Long_Term_Key_Request event.

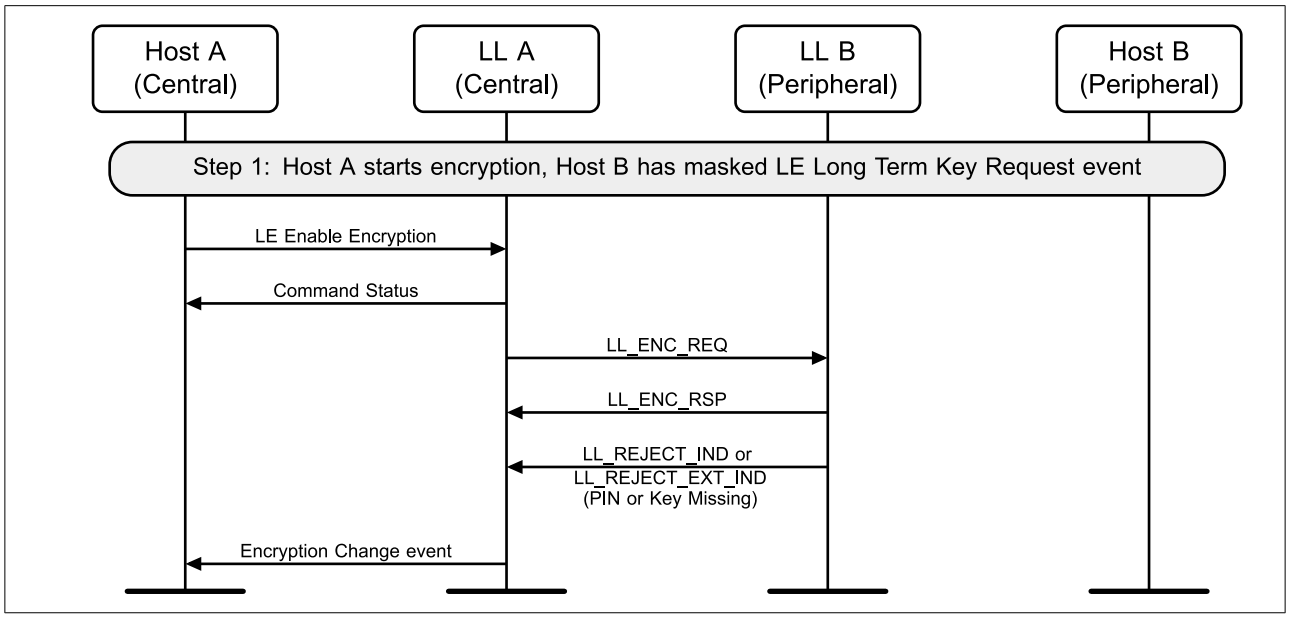



*Figure 6.13: Start encryption with Peripheral masking out event*

## 6.9 Start encryption without Peripheral supporting encryption

If encryption has not been started on a connection, it may be started by the Central. Figure 6.14 shows the failure case of the Peripheral that does not support the encryption feature.

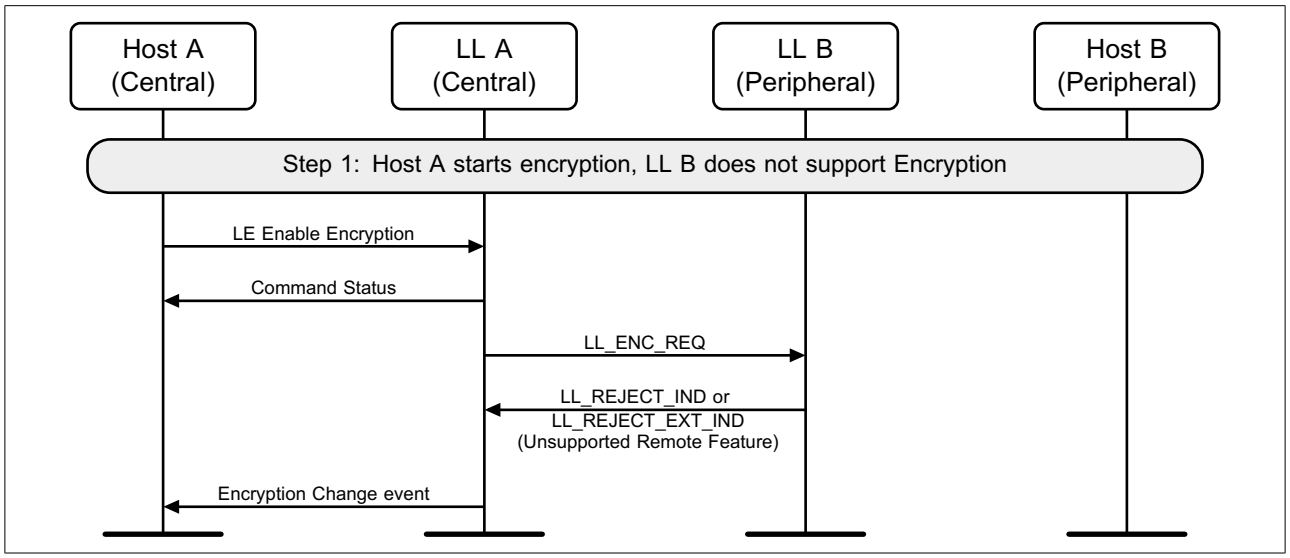



*Figure 6.14: Start encryption failure when Peripheral does not support encryption*

## 6.10 Restart encryption

If encryption has already been started on a connection, it may be restarted by the Central. This may be required to use a stronger encryption as negotiated by the Security Manager Protocol (see Figure 6.15).

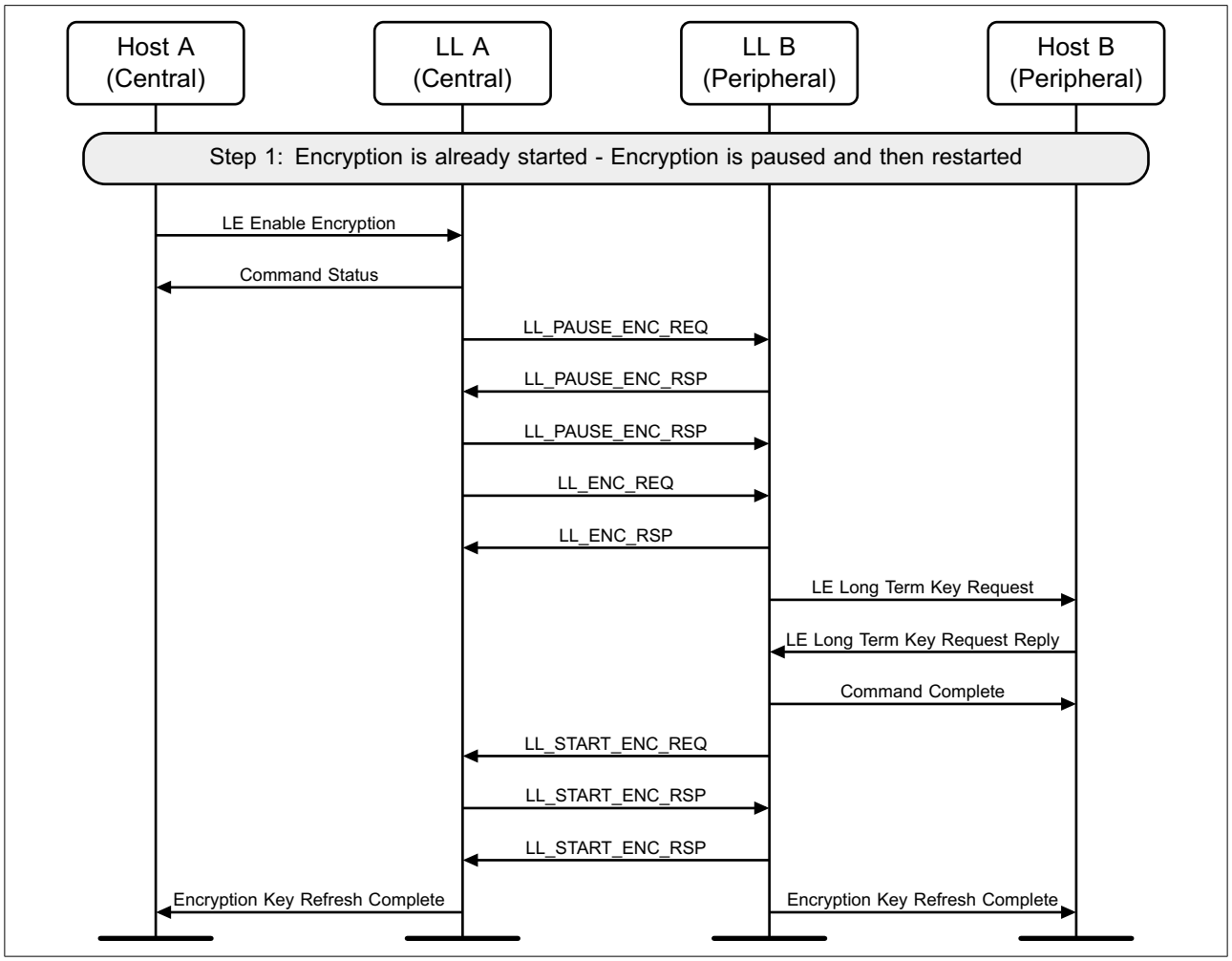



*Figure 6.15: Restart encryption*

## 6.11 Disconnect

Once a connection has no need to be kept active, the Host can disconnect it. This can be done by either device (see Figure 6.16 and Figure 6.17).

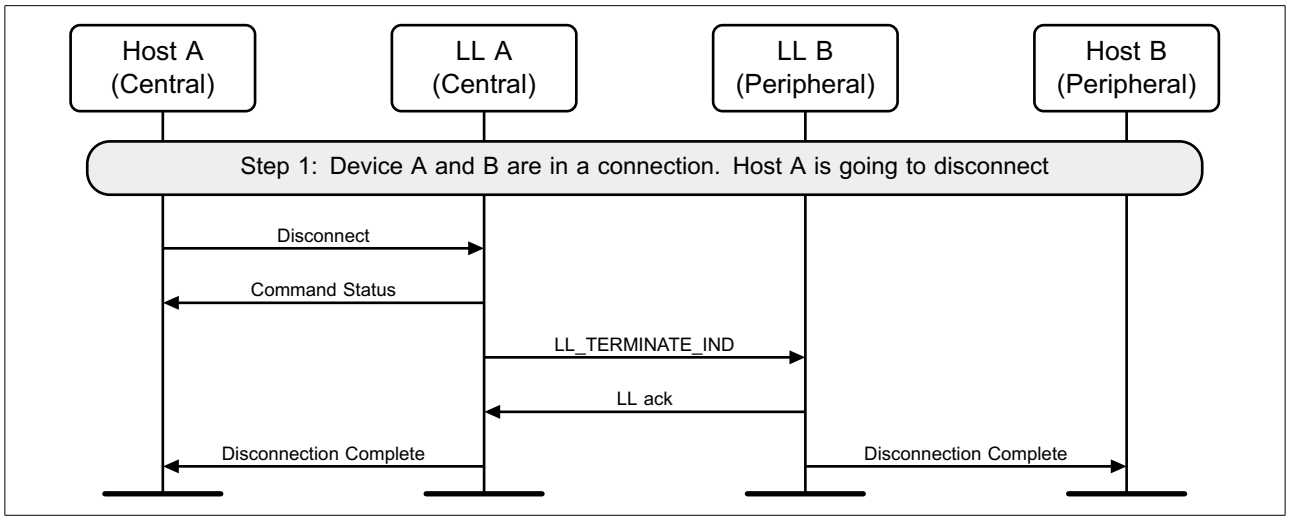



*Figure 6.16: Disconnect from Central*

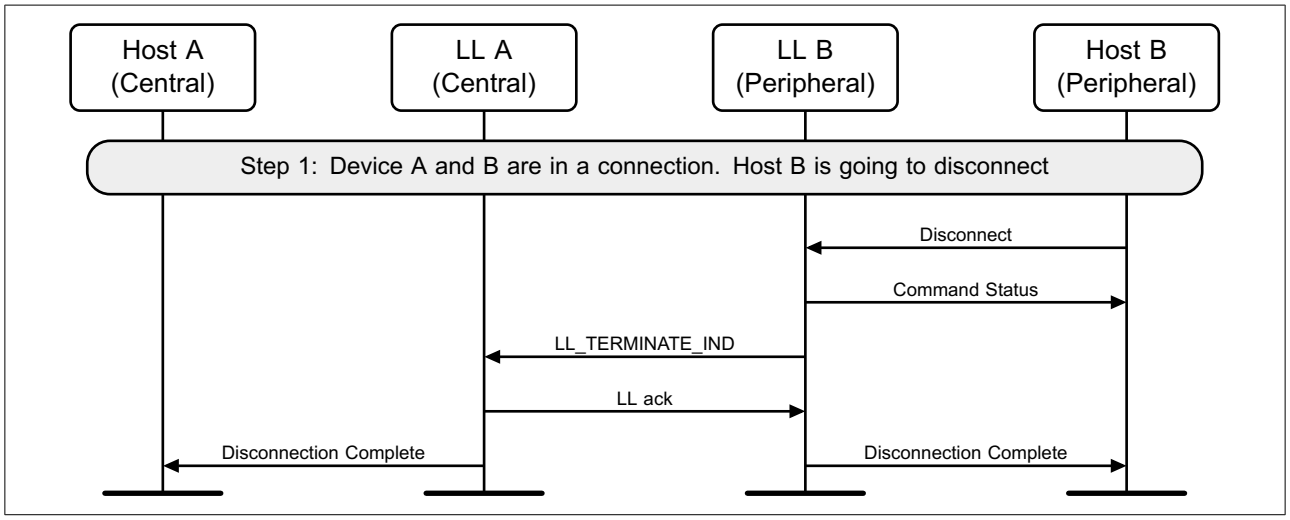



*Figure 6.17: Disconnect from Peripheral*

## 6.12 Connection parameters request

The Central or the Peripheral of the connection may request change in connection parameters using a Link Layer control procedure (see Figure 6.18 to Figure 6.25).

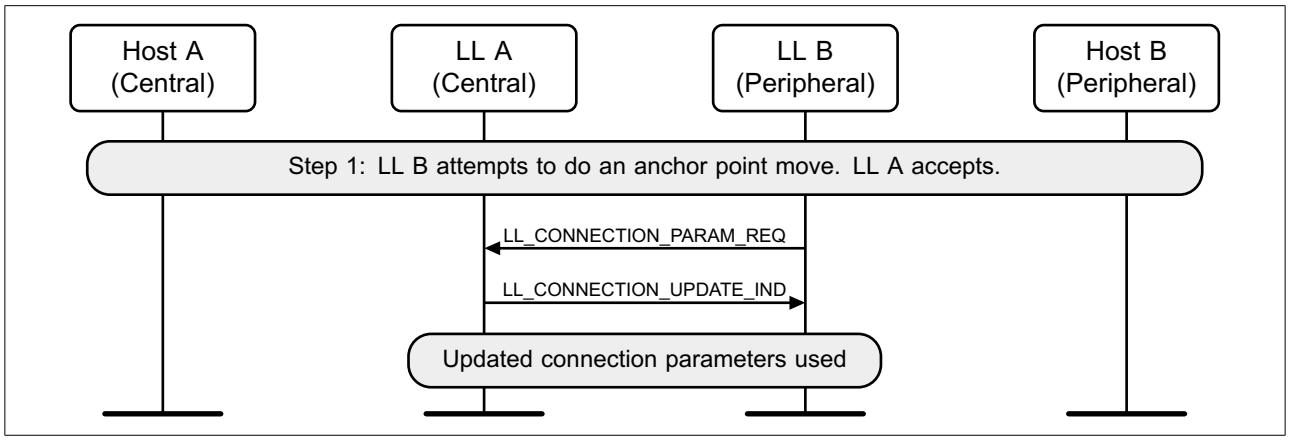



*Figure 6.18: Peripheral-initiated Connection Parameters Request procedure - Peripheral requests a change in anchor points, Central accepts*

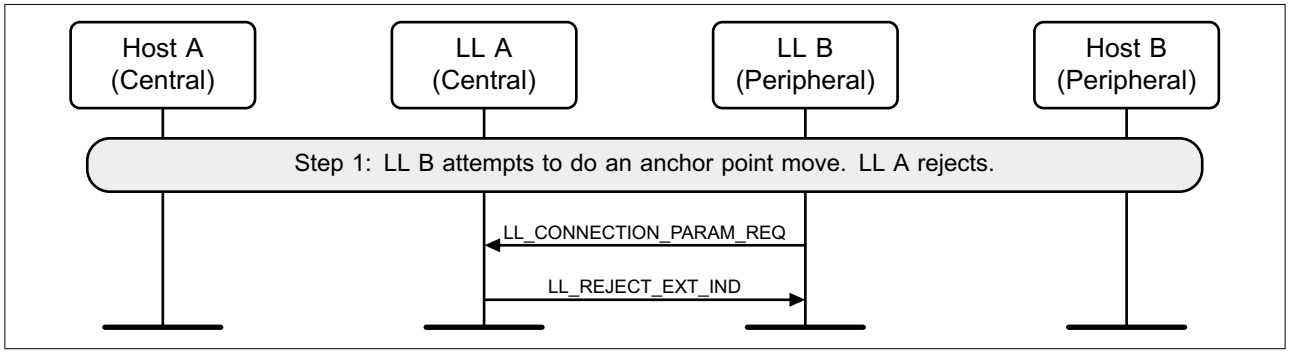



*Figure 6.19: Peripheral-initiated Connection Parameters Request procedure – Peripheral requests a change in anchor points, Central rejects*

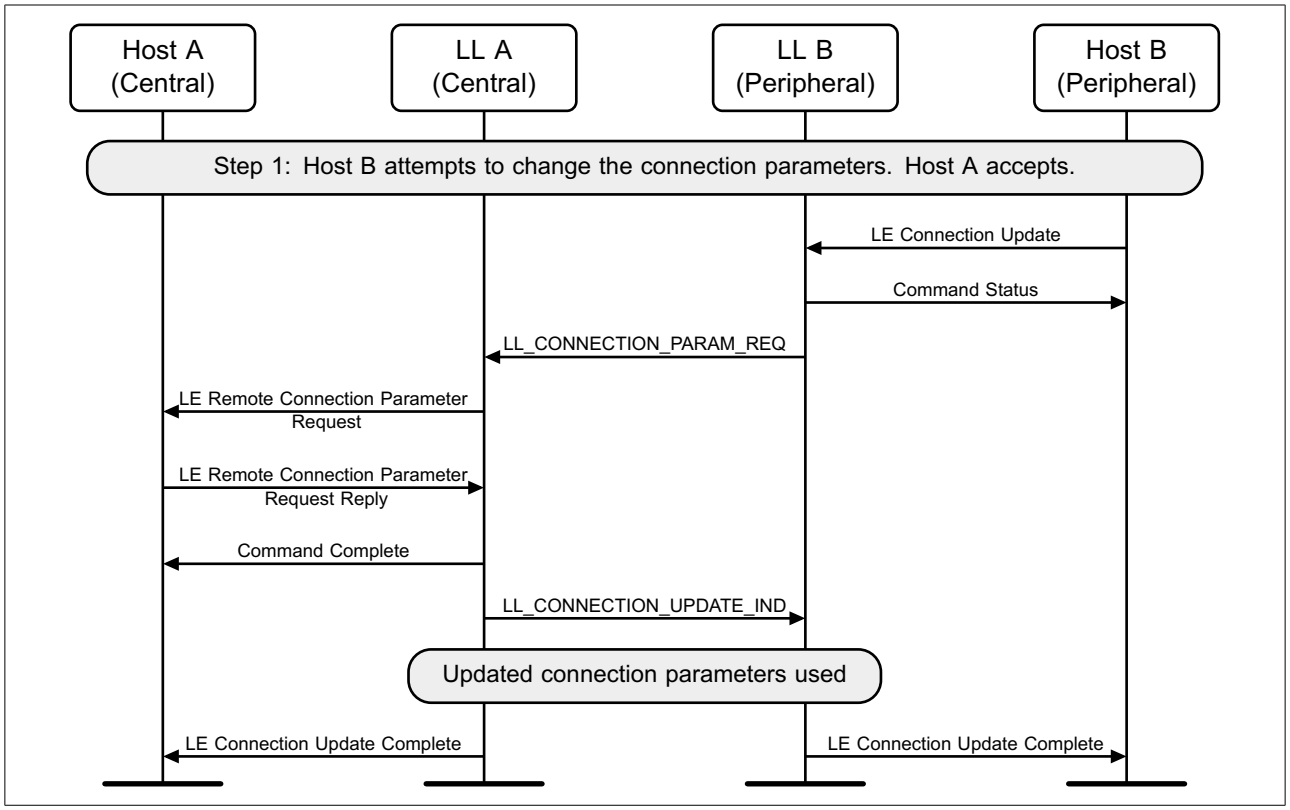



*Figure 6.20: Peripheral-initiated Connection Parameters Request procedure – Peripheral requests change in LE connection parameters, Central's Host accepts*

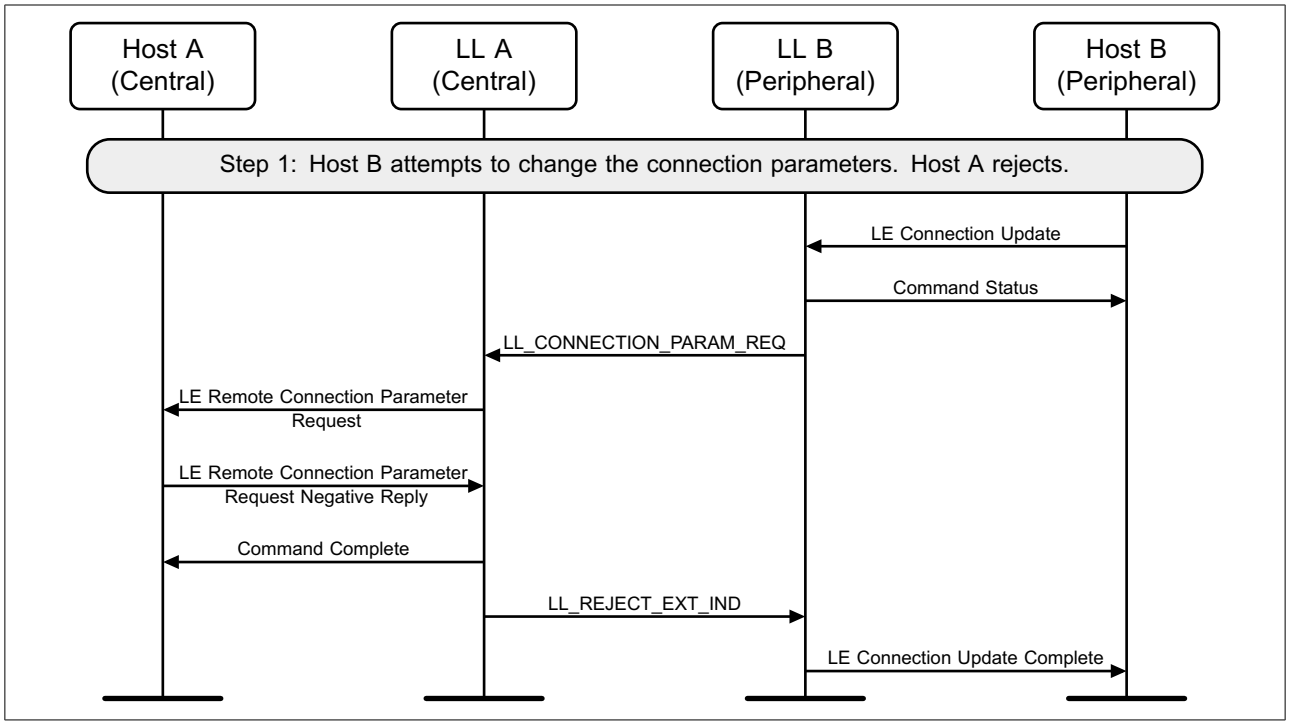



*Figure 6.21: Peripheral-initiated Connection Parameters Request procedure – Peripheral requests change in LE connection parameters, Central's Host rejects*

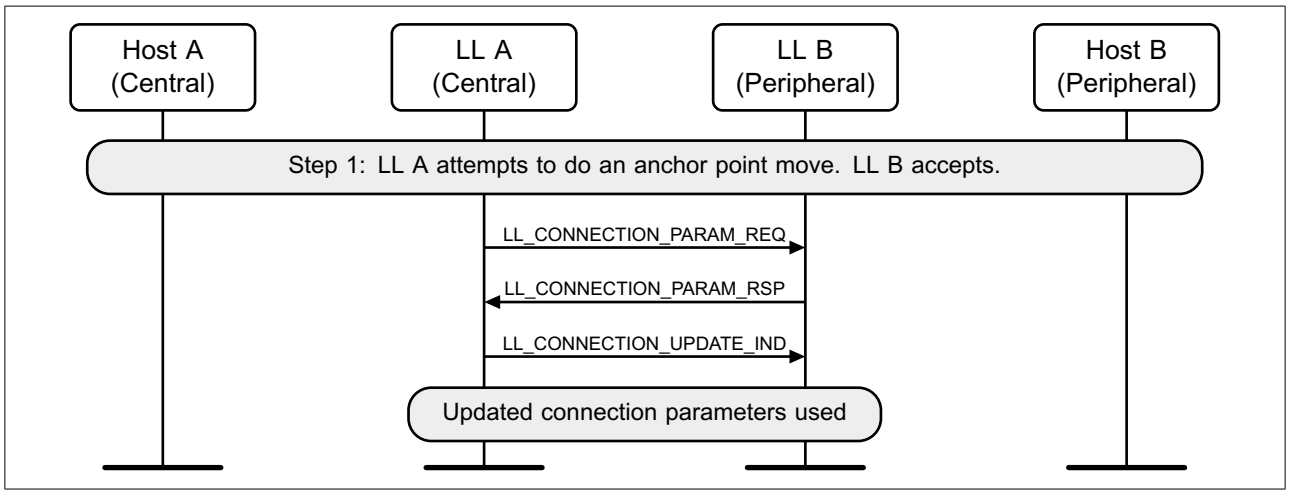



*Figure 6.22: Central-initiated Connection Parameters Request procedure – Central requests a change in anchor points, Peripheral accepts*

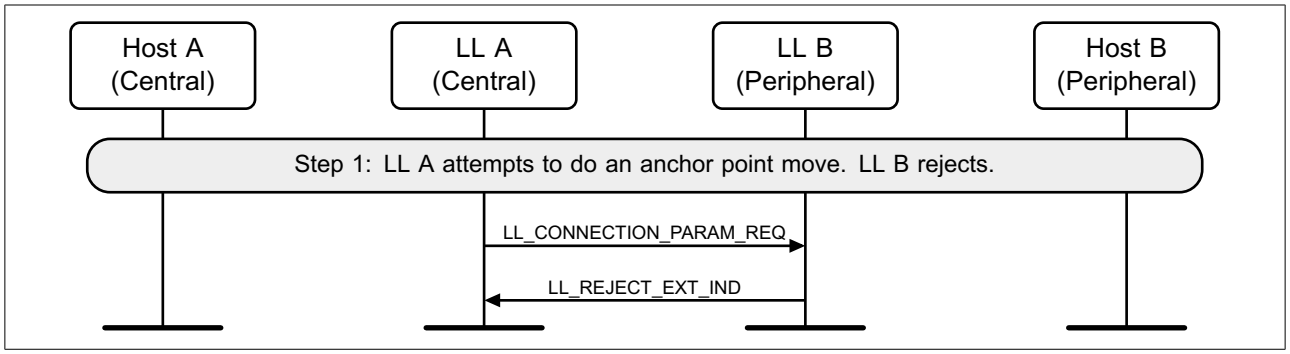



*Figure 6.23: Central-initiated Connection Parameters Request procedure – Central requests a change in anchor points, Peripheral rejects*

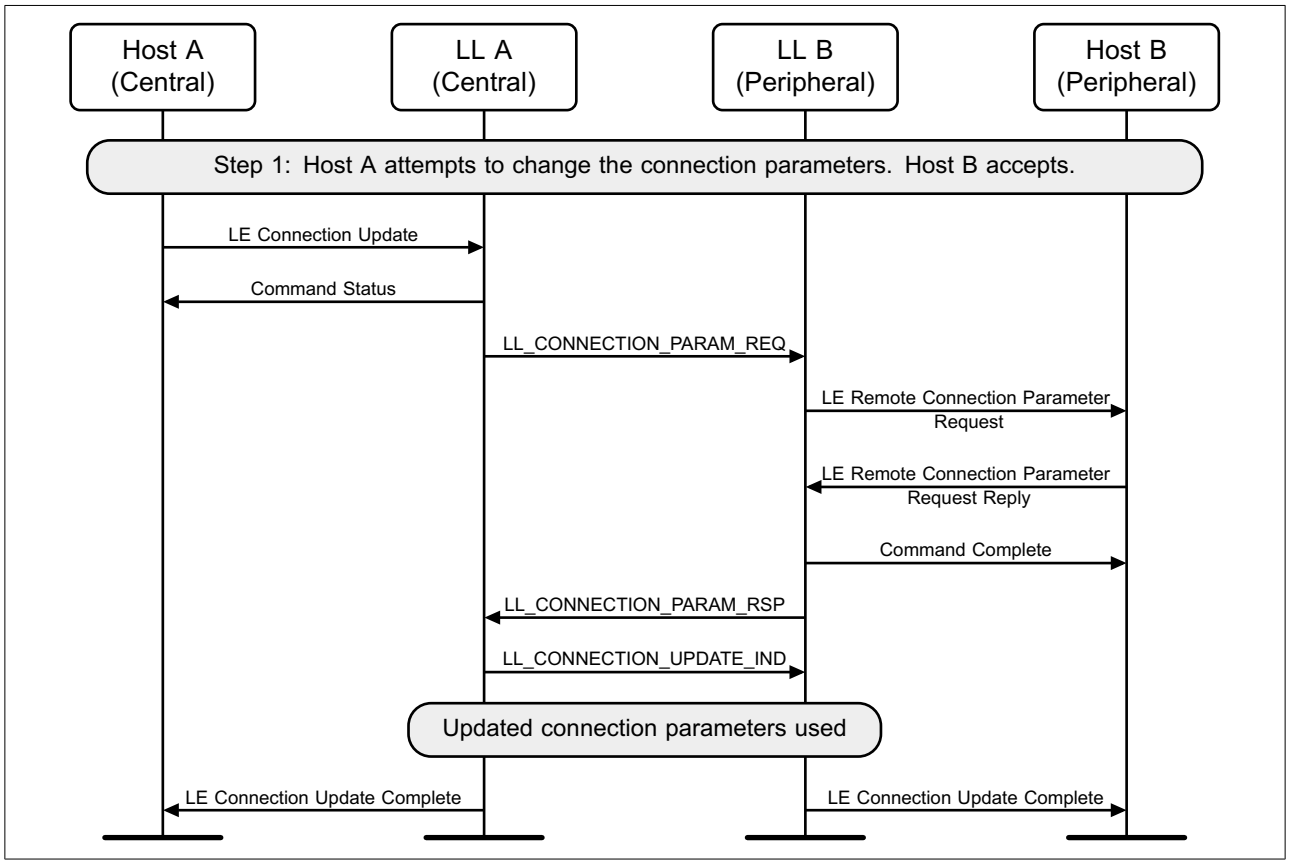



*Figure 6.24: Central-initiated Connection Parameters Request procedure – Central requests change in LE connection parameters, Peripheral's Host accepts*

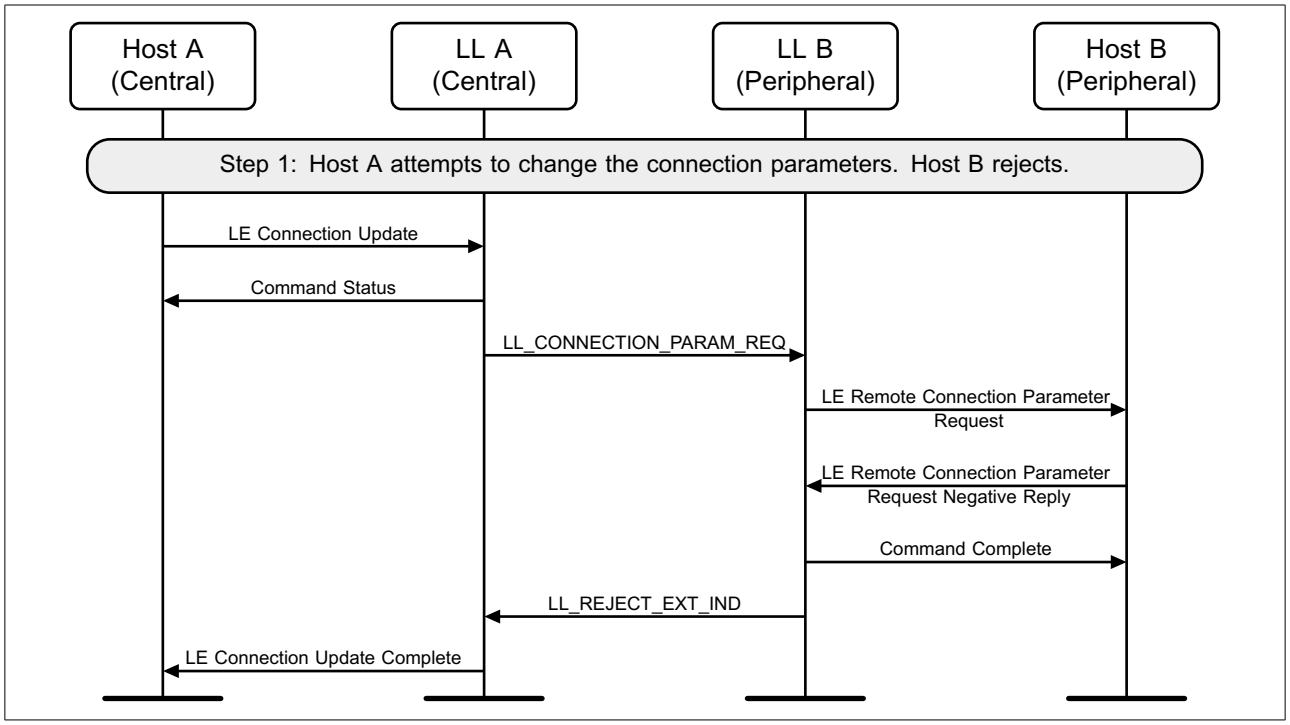



*Figure 6.25: Central-initiated Connection Parameters Request procedure – Central requests change in LE connection parameters, Peripheral's Host rejects*

## 6.13 LE Ping

A Host may use the HCI_Write_Authenticated_Payload_Timeout command to change the maximum interval between packets containing a valid MIC that the Link Layer will enforce when encryption is used.

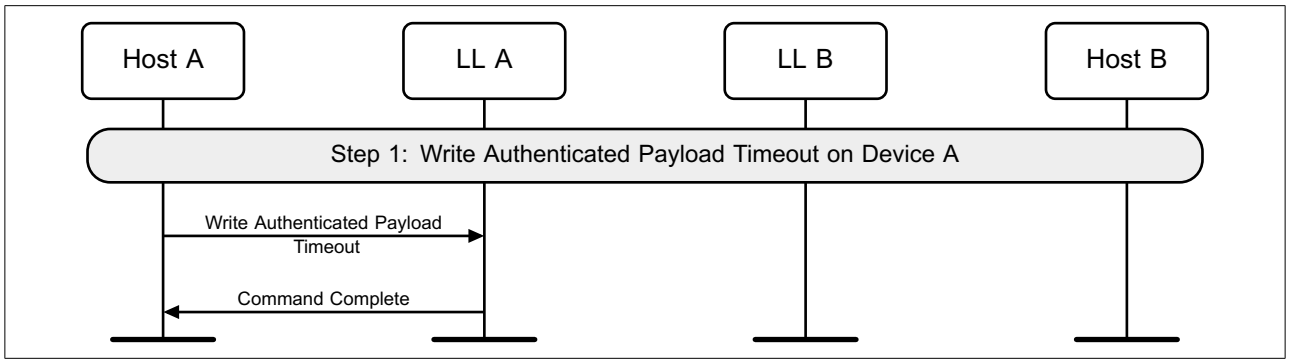



*Figure 6.26: Set LE authenticated payload timeout*

Either Link Layer can authenticate the remote device using the LE Ping procedure even if the remote device does not support the LE Ping feature. This procedure can also be used for soliciting a packet from the remote device containing a valid MIC. LL A may be a Central or a Peripheral.

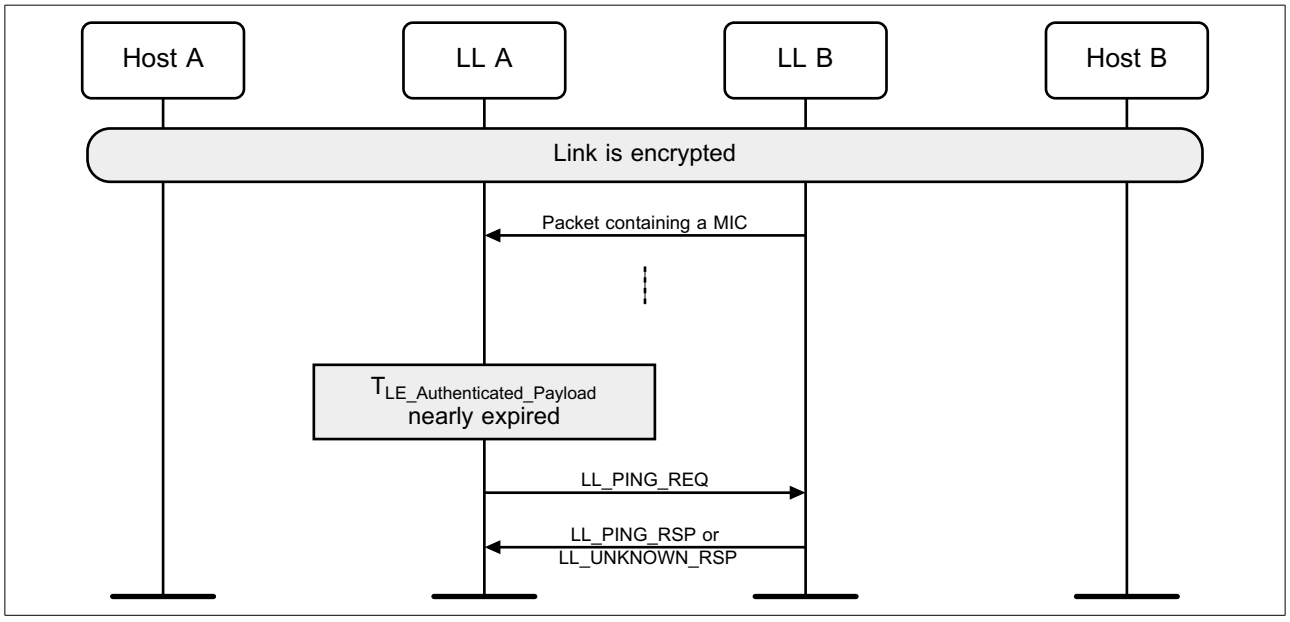



*Figure 6.27: Successful LE Ping*

When a packet with a valid MIC has not been received within the LE Authenticated Payload Timeout, the Host is notified that the timer has expired.

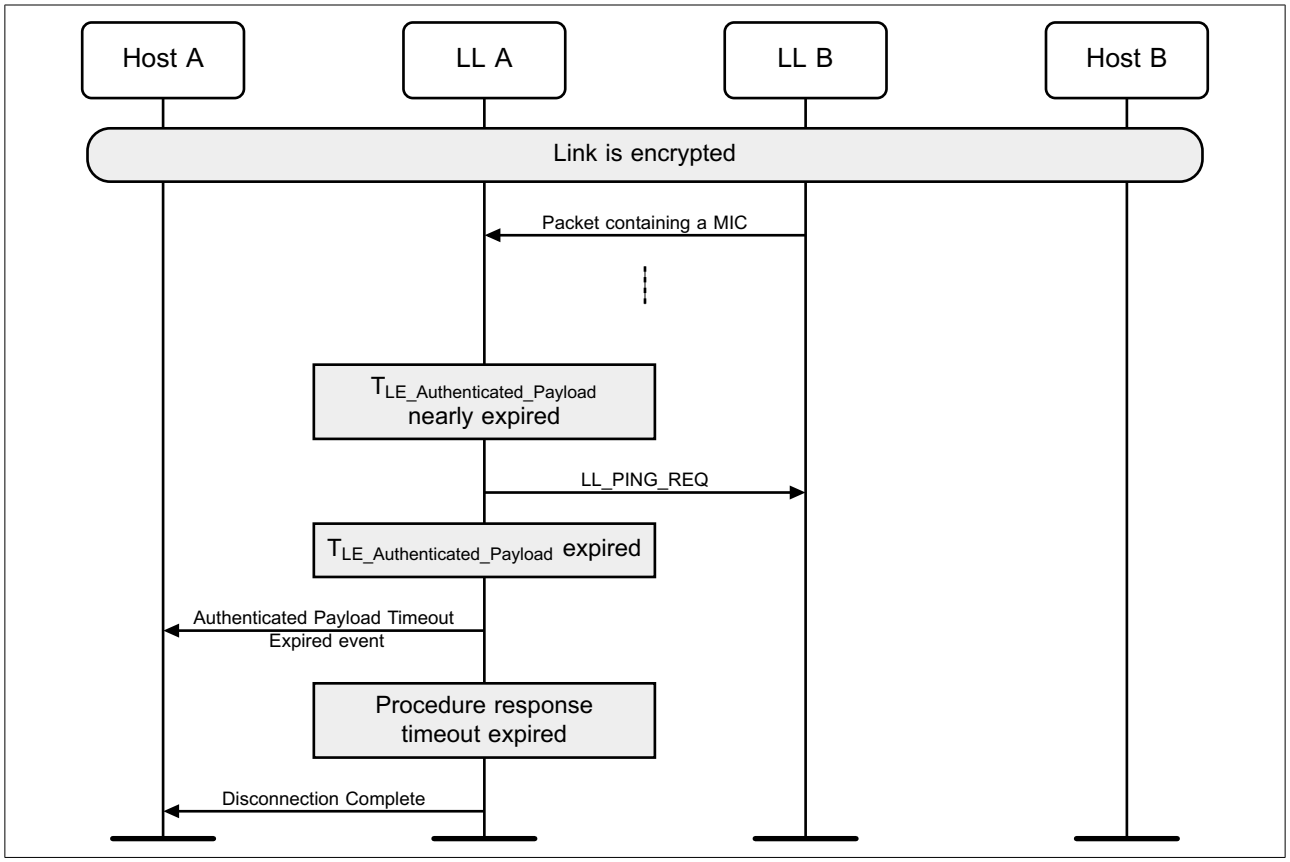



*Figure 6.28: Unsuccessful LE Ping*

The $T_{LE_Authenticated_Payload}$ Timer gets reset when the Host sets the Authenticated Payload Timeout.

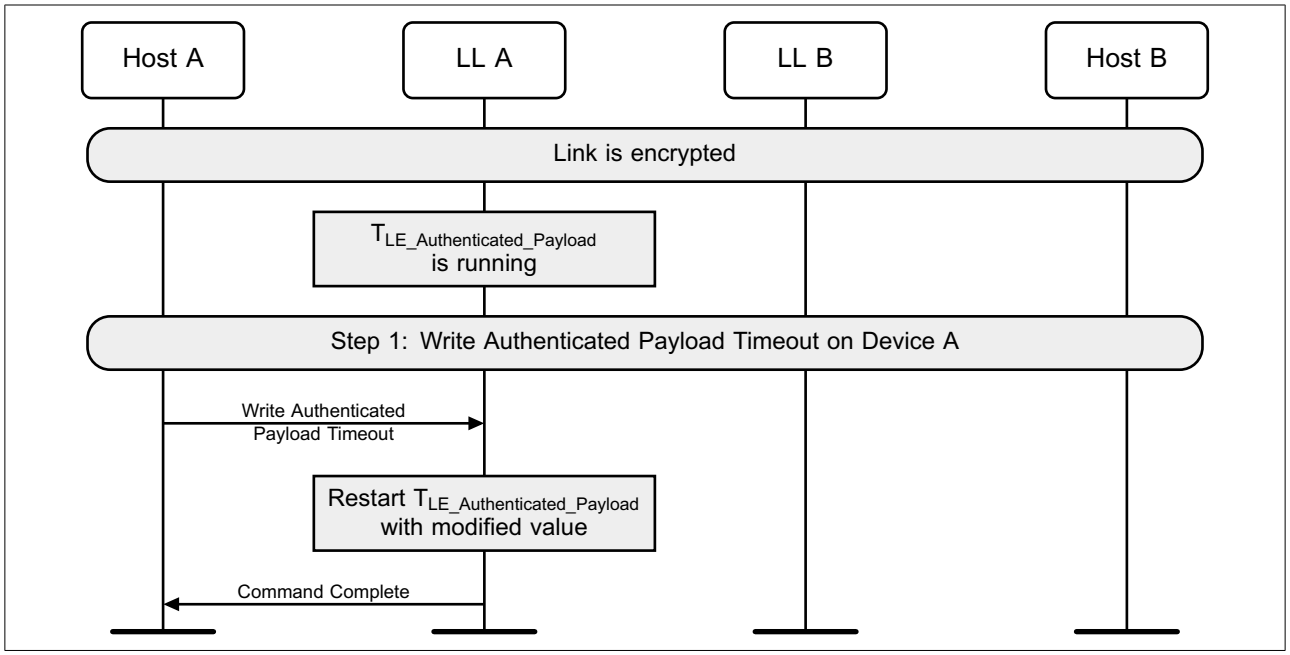



*Figure 6.29: $T_{LE_Authenticated_Payload}$ Timer reset*

## 6.14 Data length update

Once a connection has been created, the Host may suggest maximum transmission packet size and maximum packet transmission time to be used for the connection. This may be done on either the Central or the Peripheral.

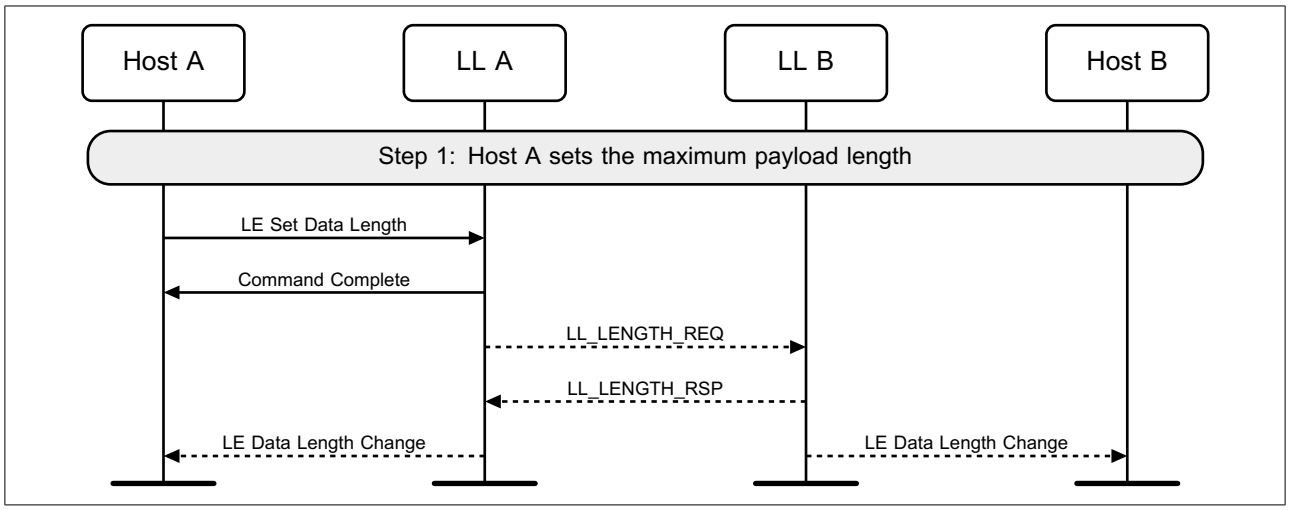



*Figure 6.30: Data length update*

## 6.15 PHY update

The Central or Peripheral of the connection may request a change in the PHY using a Link Layer control procedure (see Figure 6.31 to Figure 6.39).

The sequence of events shown in Figure 6.33 and Figure 6.36 can only happen before feature exchange and can only happen once per connection, because a Link Layer must not use a procedure that it knows the peer does not support as required by [Vol 6] Part B, Section 4.6.

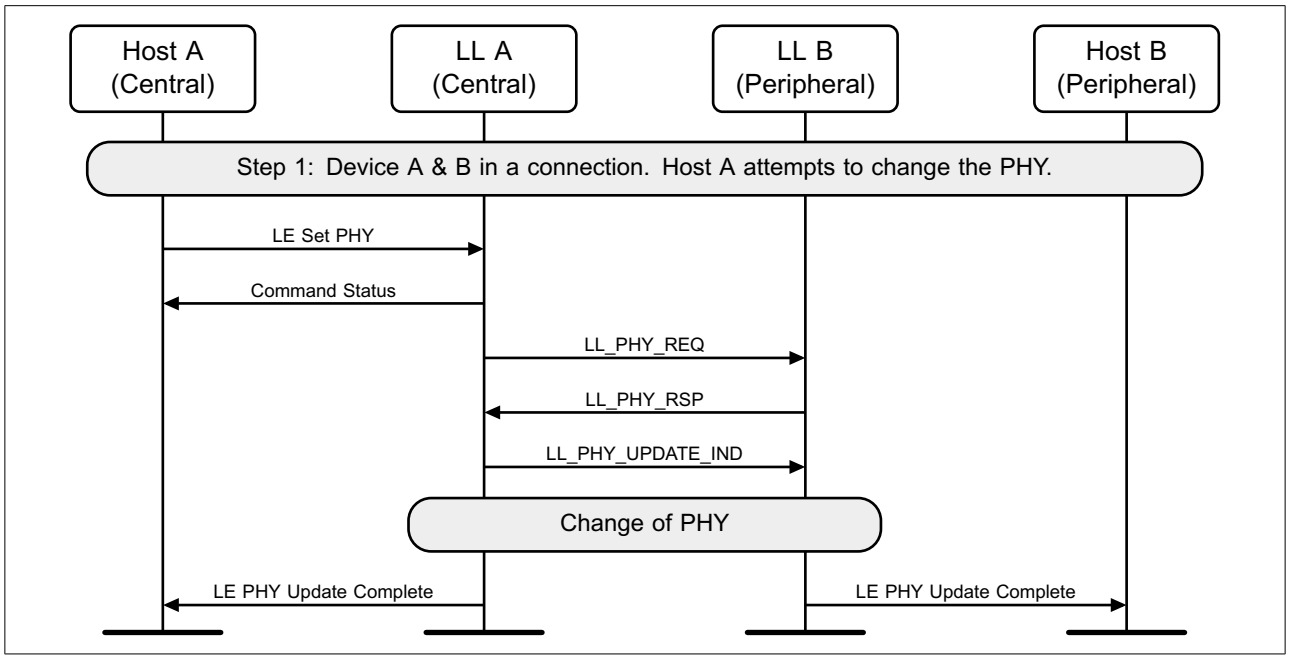



*Figure 6.31: Central-initiated PHY Update procedure – Central requests a change of PHY, PHY changed in at least one direction*

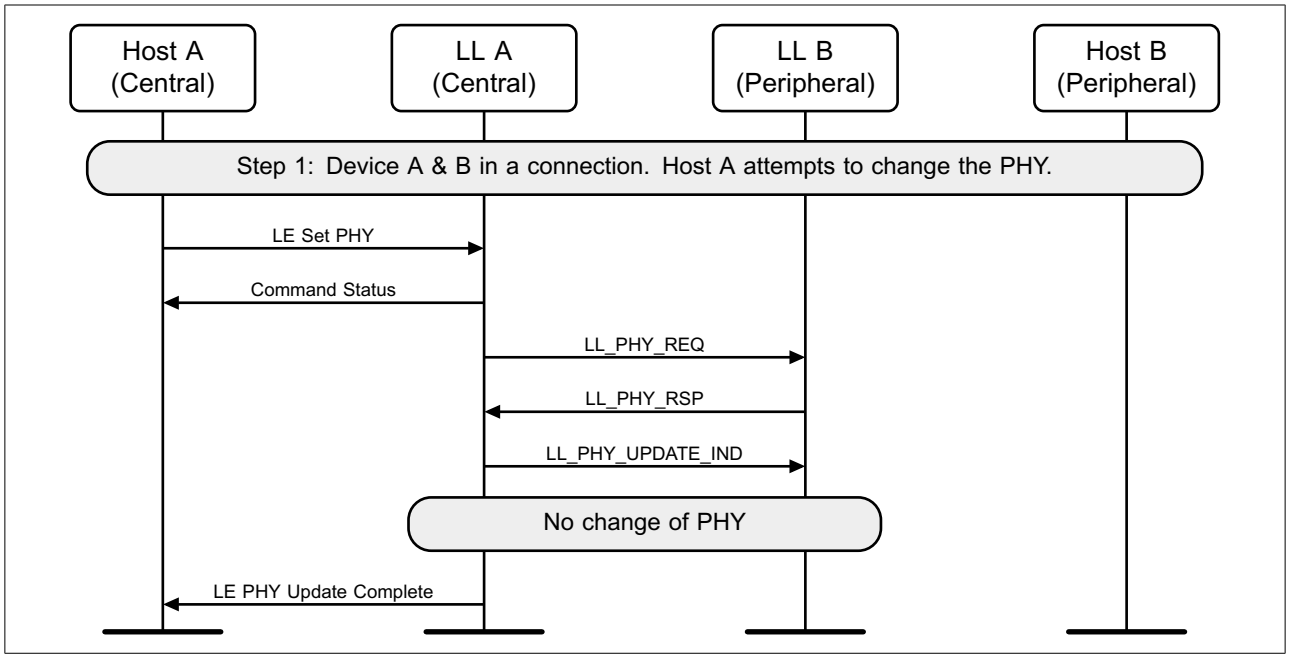



*Figure 6.32: Central-initiated PHY Update procedure – PHY not changed (either because Peripheral doesn't specify PHYs that the Central prefers, or because the Central concludes that the current PHYs are still best)*

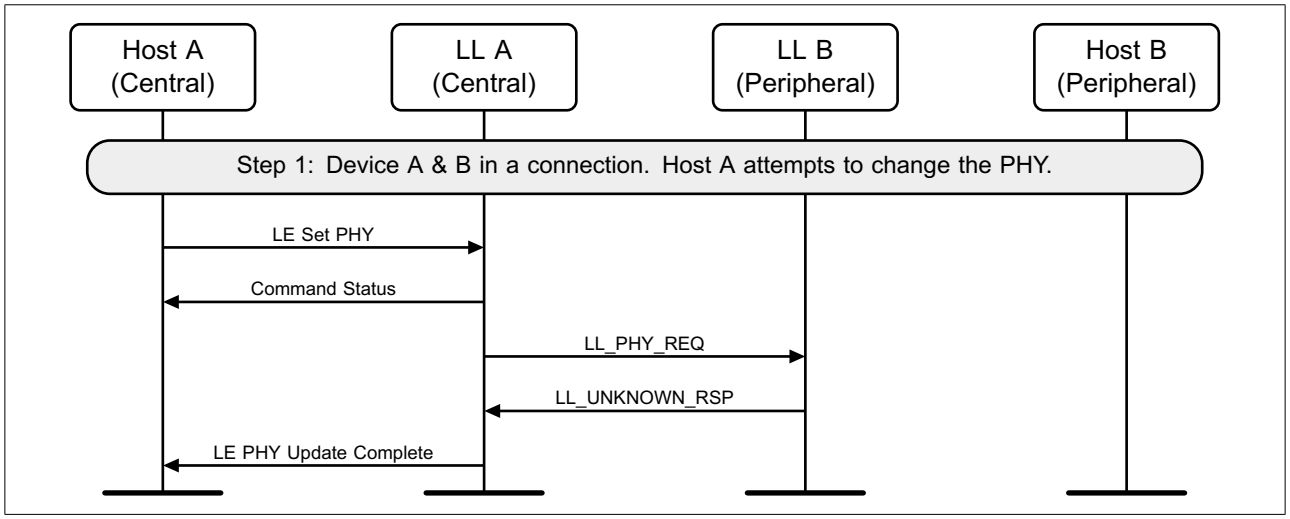



*Figure 6.33: Central-initiated PHY Update procedure – Central requests a change of PHY, Peripheral does not support the feature*

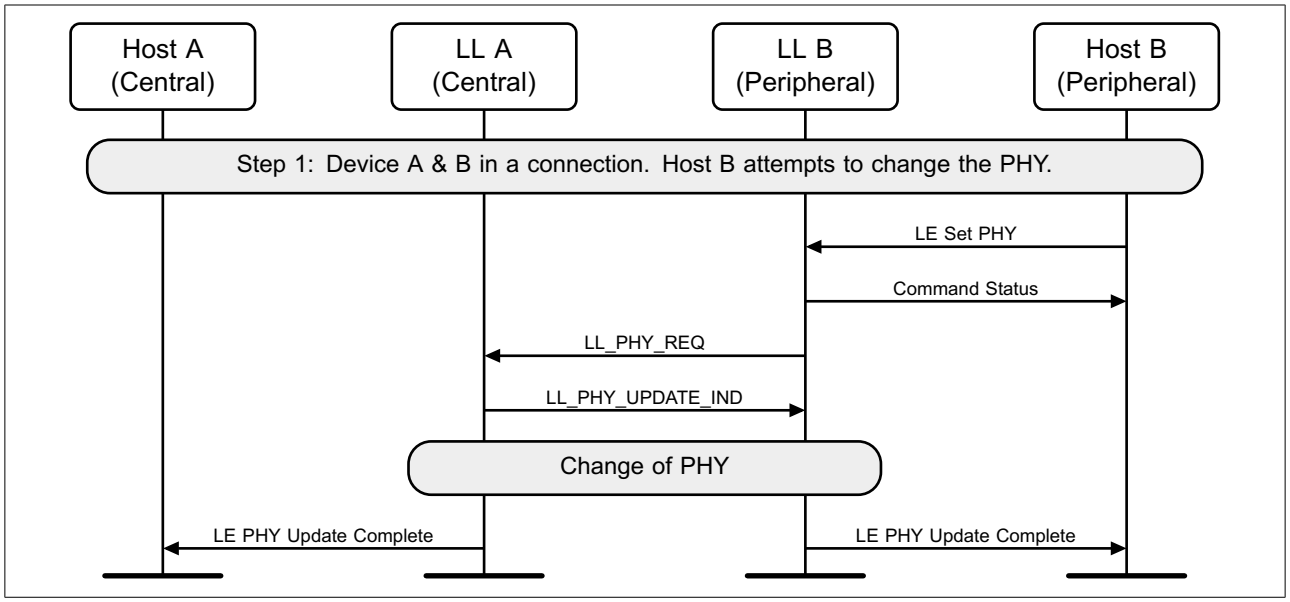



*Figure 6.34: Peripheral-initiated PHY Update procedure – Peripheral requests a change of PHY, PHY changed in at least one direction*

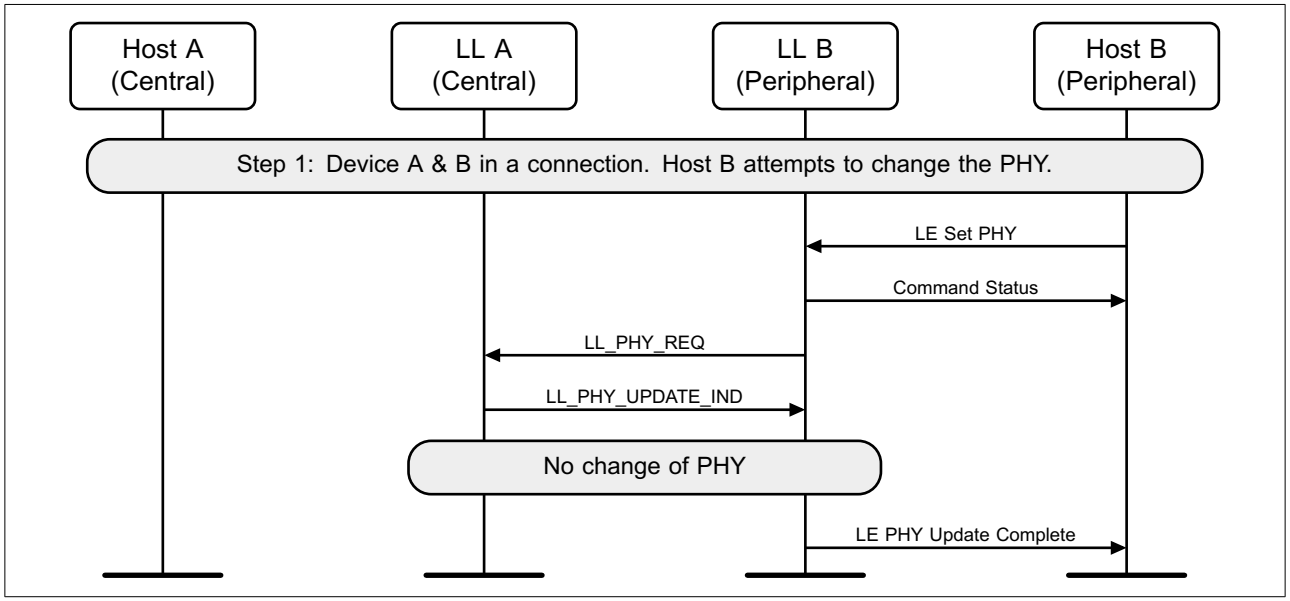



*Figure 6.35: Peripheral-initiated PHY Update procedure – PHY not changed (either because Peripheral doesn't specify PHYs that the Central prefers, or because the Central concludes that the current PHYs are still best)*

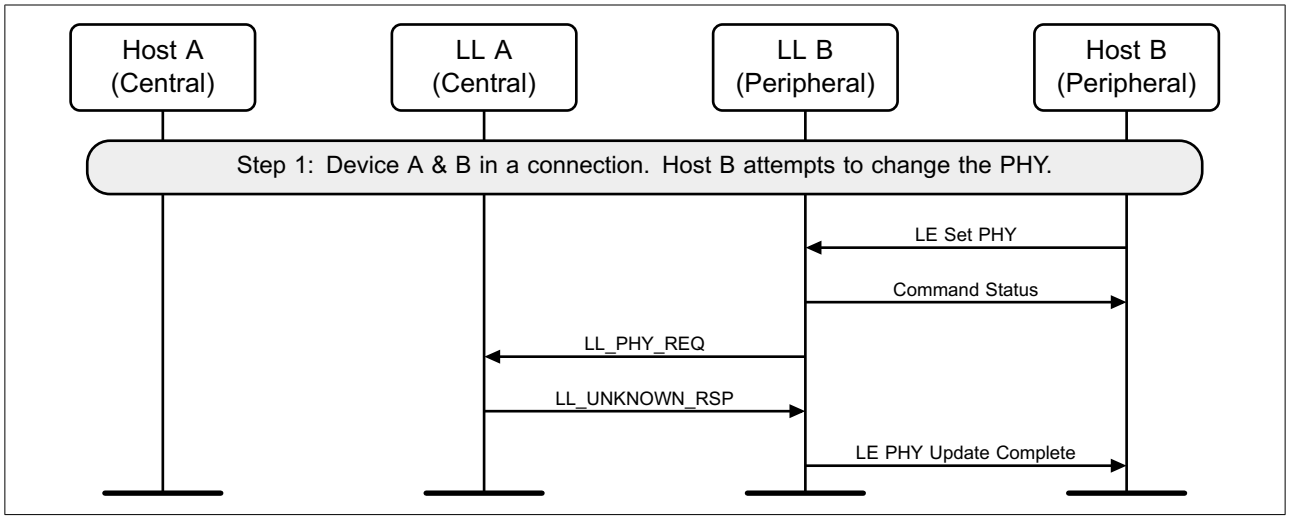



*Figure 6.36: Peripheral-initiated PHY Update procedure – Peripheral requests a change of PHY, Central does not support the feature*

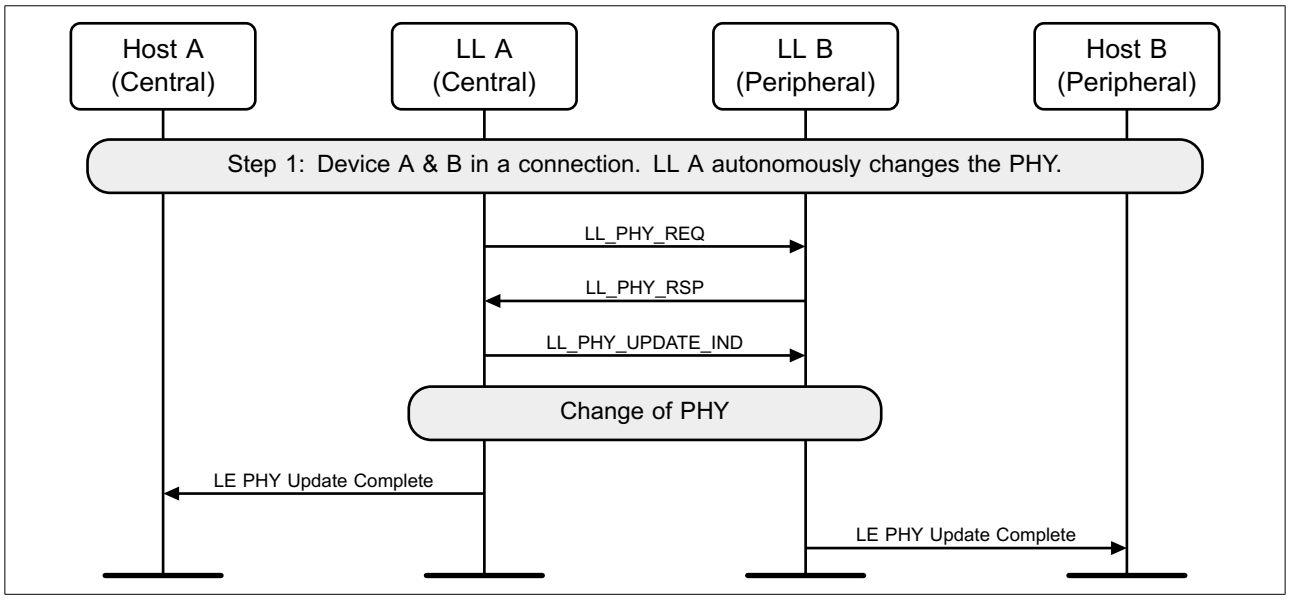



*Figure 6.37: Autonomous Central-initiated PHY Update procedure – Central requests a change of PHY, Peripheral accepts, PHY changed in at least one direction*

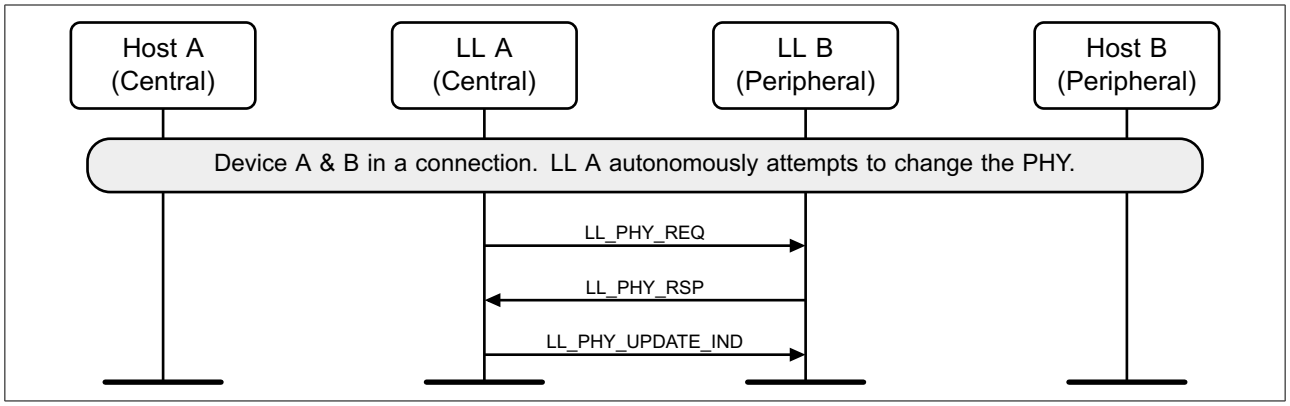



*Figure 6.38: Autonomous Central-initiated PHY Update procedure – Central requests a change of PHY, PHY not changed (either because Peripheral doesn't specify PHYs that the Central prefers, or because the Central concludes that the current PHYs are still best)*

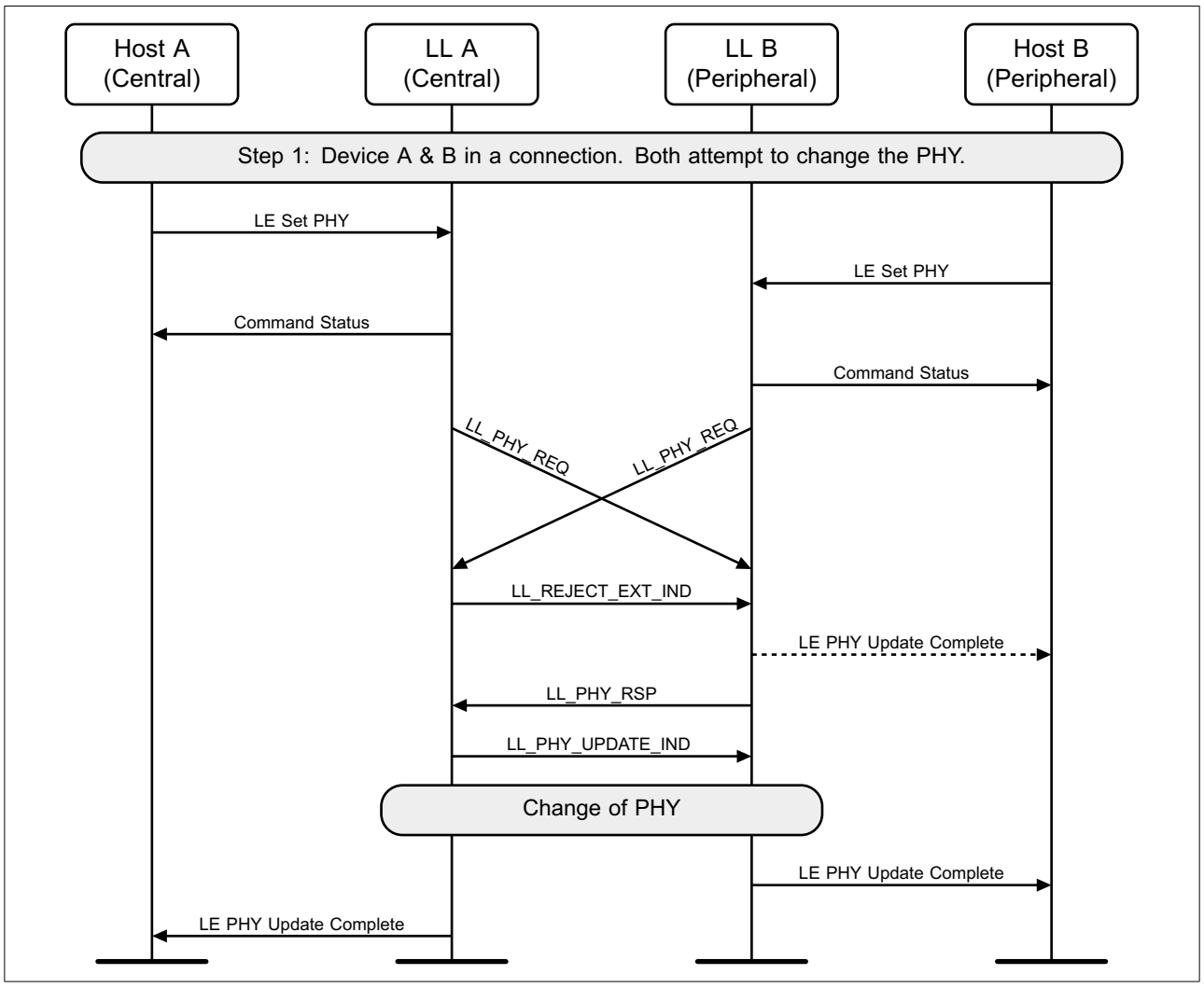



*Figure 6.39: Central and Peripheral crossover PHY Update procedure – Central and Peripheral request a change of PHY concurrently*

## 6.16 Minimum number of used channels request

Where a Peripheral supports the Minimum Number of Used Channels procedure, it can request that a certain minimum number of channels be used on the indicated PHY. The example shows a successful request, resulting in a channel map update with the requested minimum number of channels used for the connection.

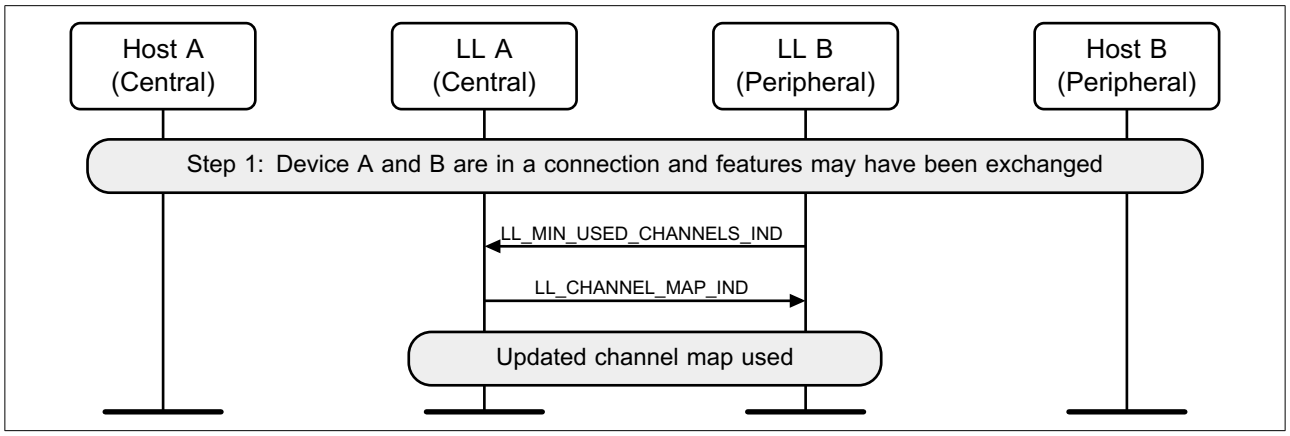



*Figure 6.40: Requesting minimum number of used channels*

## 6.17 LL procedure collision

The Link Layers of both the Central and Peripheral may initiate the same LL procedure at the same time.

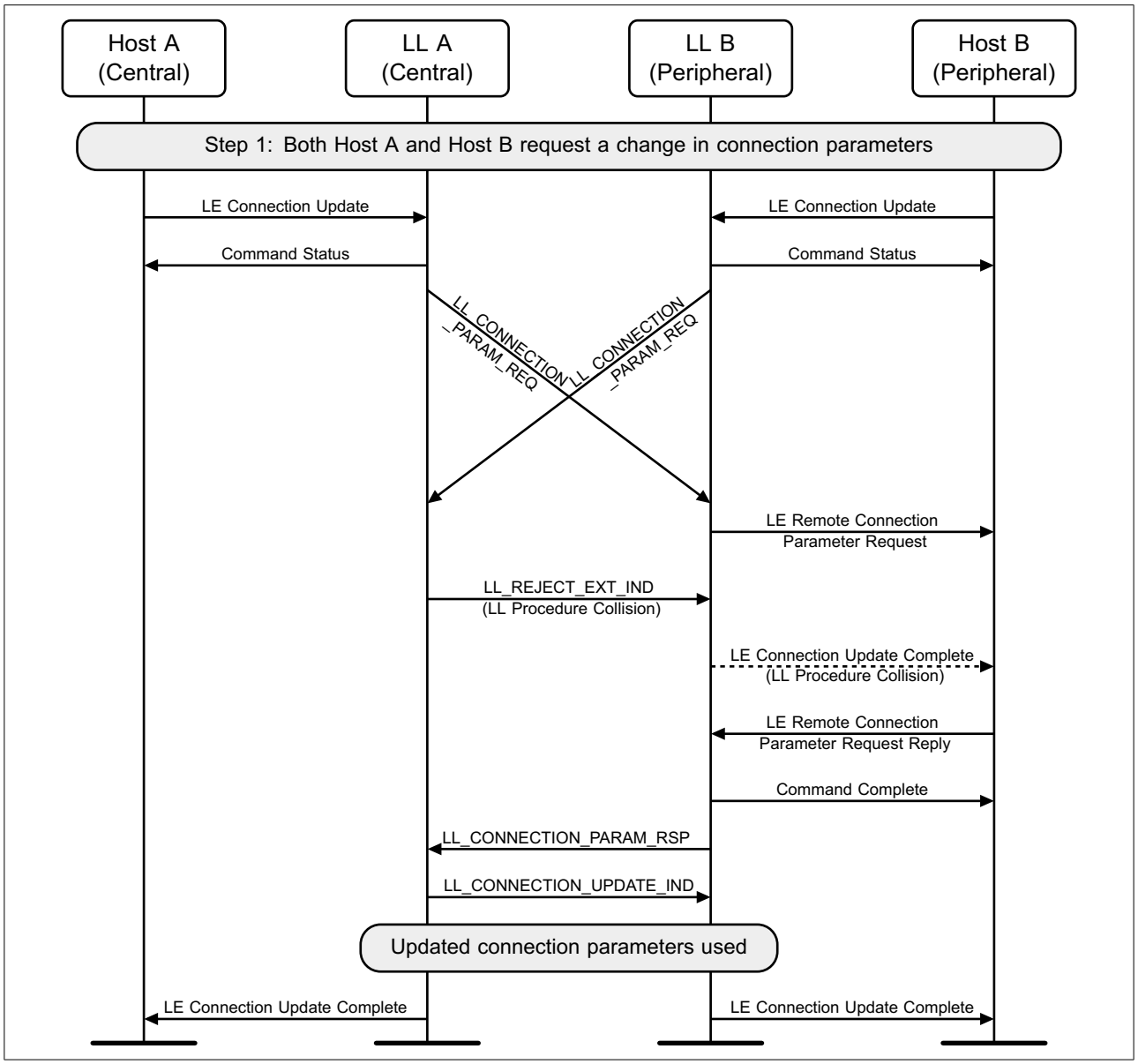



*Figure 6.41: LL procedure collision*

## 6.18 Constant Tone Extension Request

The Central or Peripheral of the connection may request the remote device to send an LL_CTE_RSP PDU with a Constant Tone Extension (see Figure 6.42 to Figure 6.44).

The sequence of events shown in figure 6.40 can only happen before feature exchange and can only happen once per connection, because a Link Layer must not use a procedure that it knows the peer does not support as required by [Vol 6] Part B, Section 4.6.

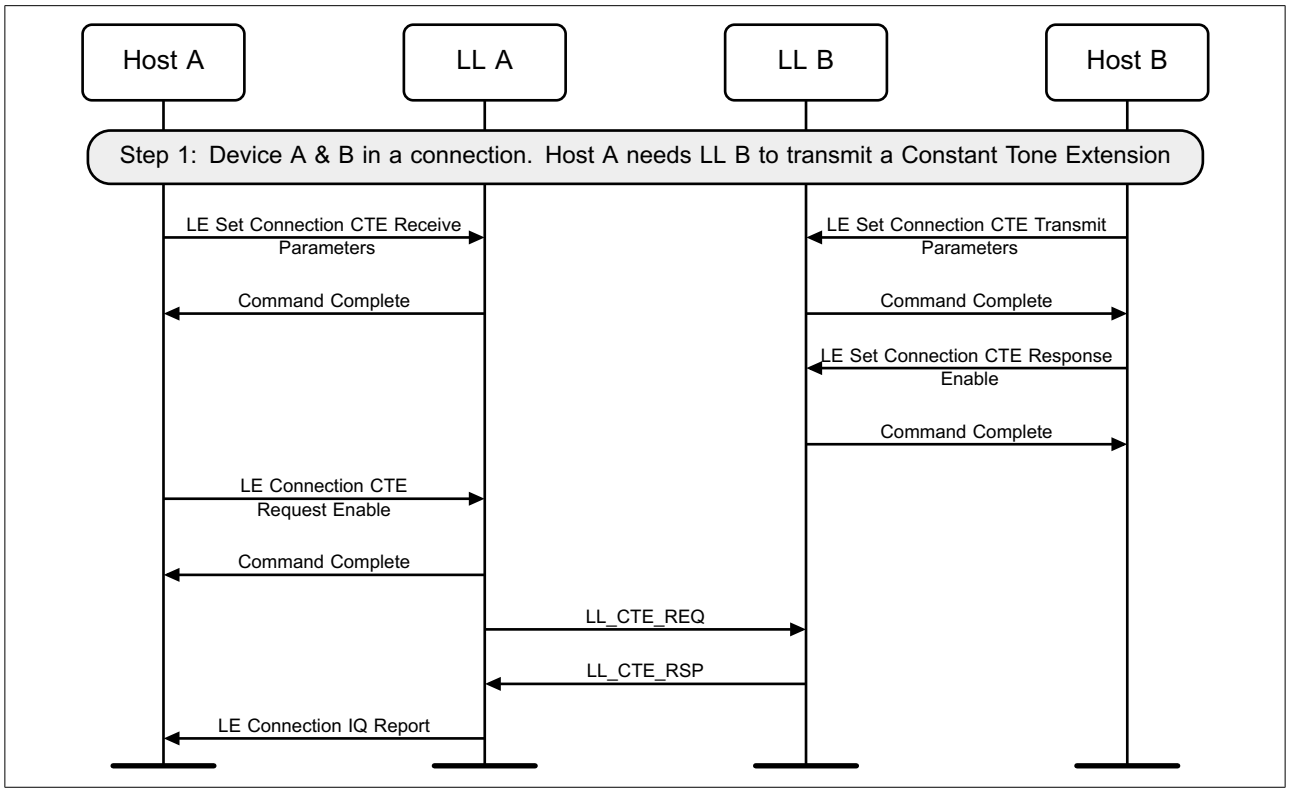



*Figure 6.42: Central or Peripheral-initiated Constant Tone Extension Request procedure – remote device responds successfully*

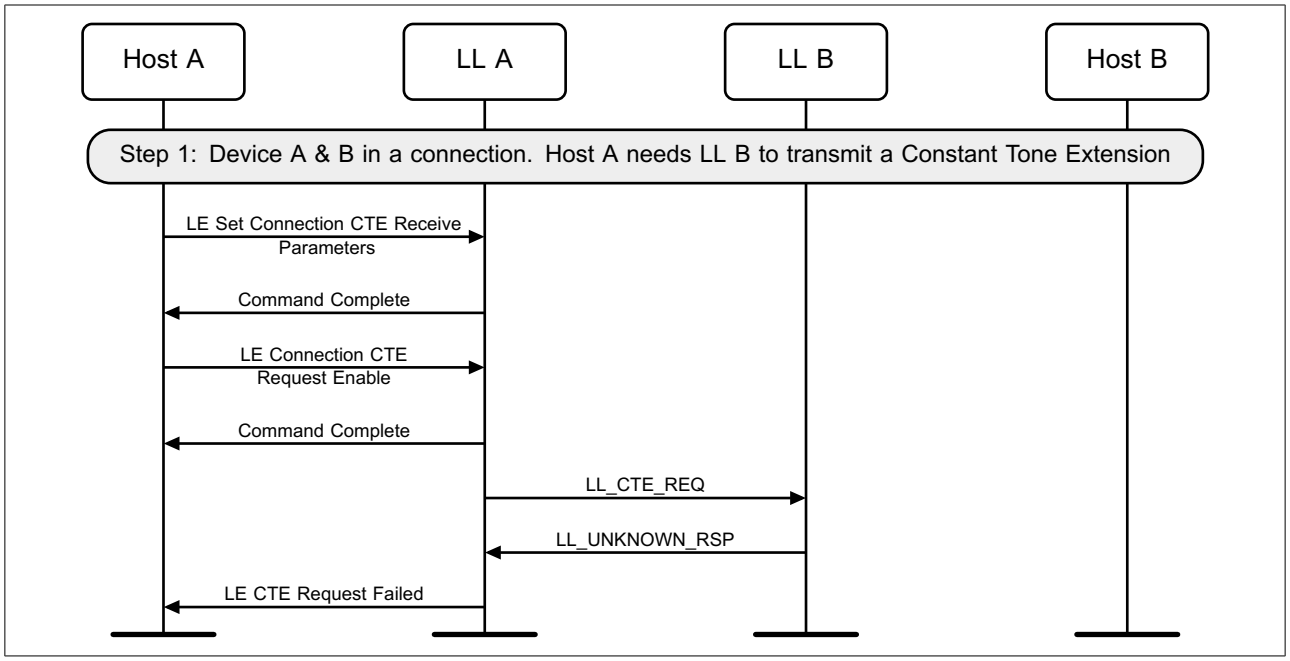



*Figure 6.43: Central or Peripheral-initiated Constant Tone Extension Request procedure – remote device does not support the feature*

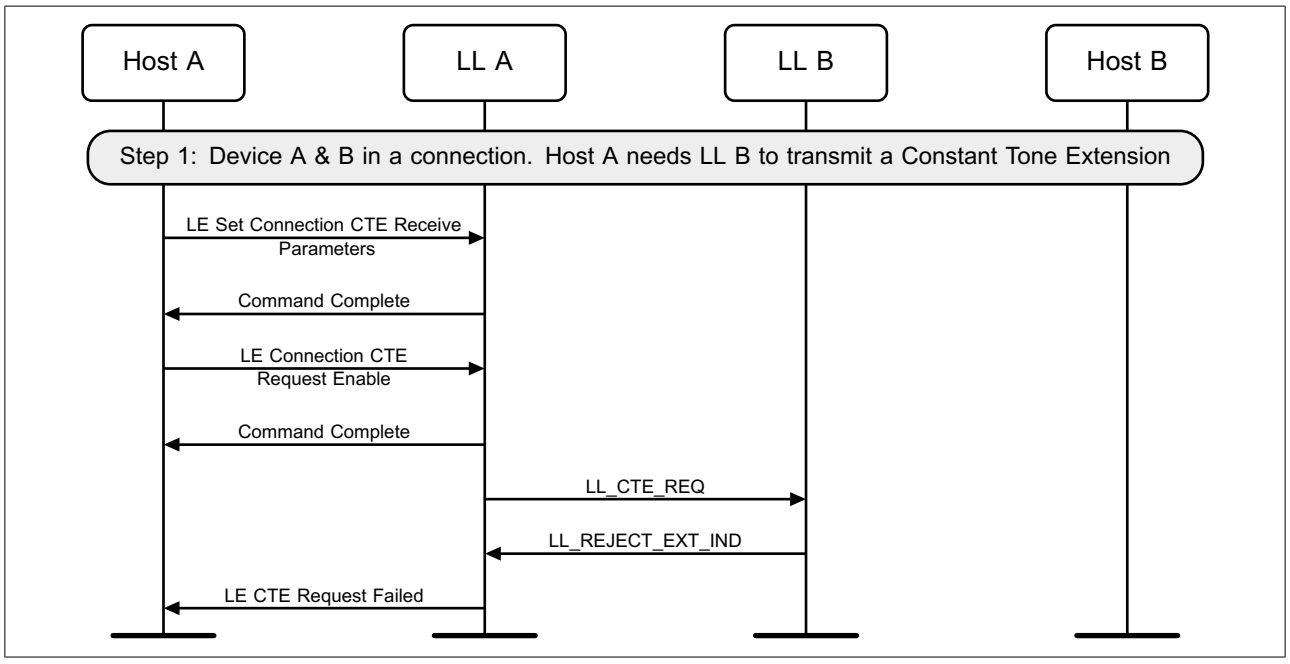



*Figure 6.44: Central or Peripheral-initiated Constant Tone Extension Request procedure – remote device rejects*

## 6.19 Connected Isochronous Group Setup

A Central sets up a CIG with parameters for one or more CISes, then establishes a CIS with a Peripheral (see Figure 6.45).

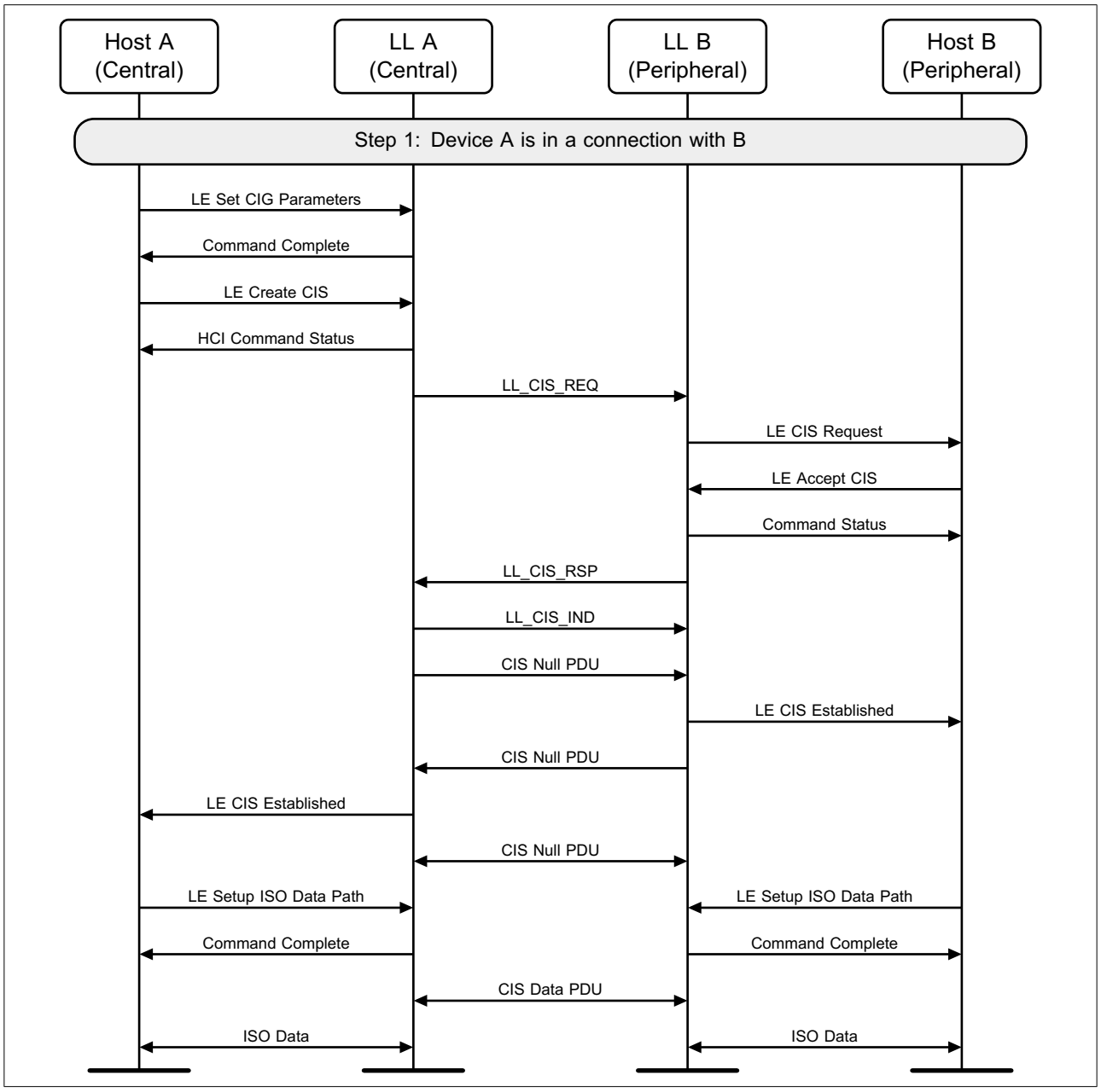



*Figure 6.45: Device A establishes a CIS with a Peripheral*

A Central sets up CIG parameters and establishes a CIS with two Peripherals (see Figure 6.46).

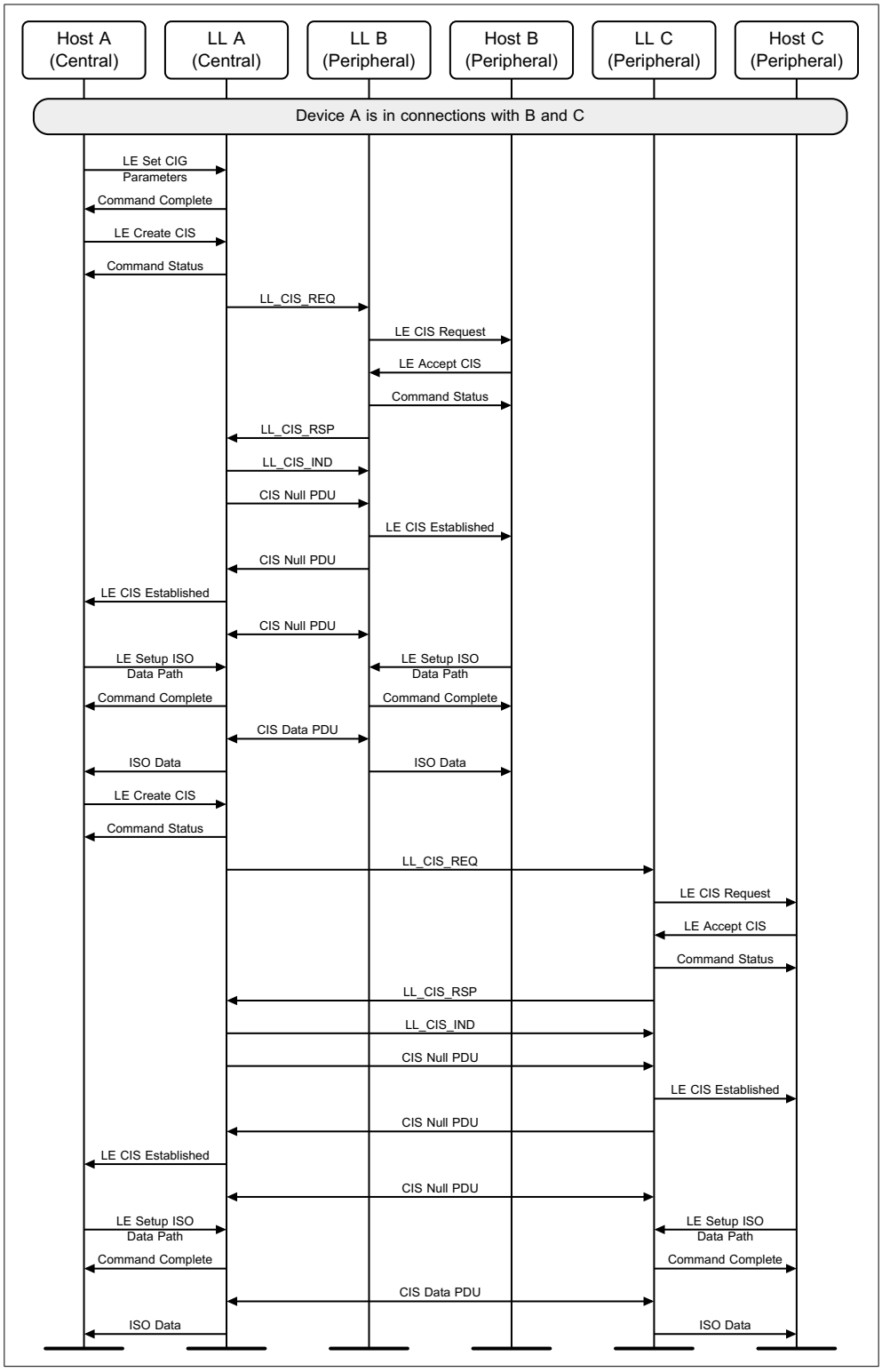



*Figure 6.46: Device A establishes CISes with two Peripherals*

## 6.20 Host Rejects Connected Isochronous Stream

The Peripheral's Host rejects the request to establish a CIS with the Central (see Figure 6.47).

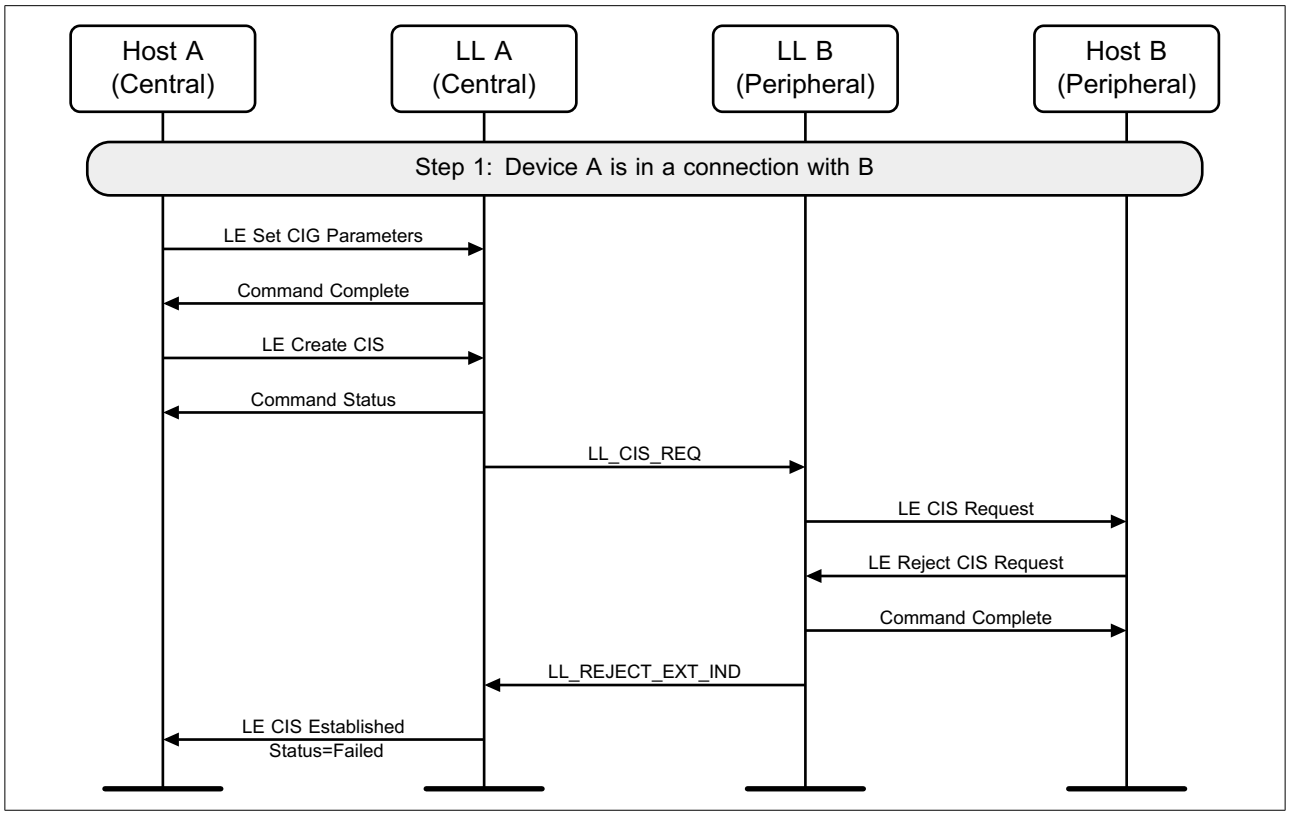



*Figure 6.47: The Host in Device B rejects a CIS from Device A*

While setting up CISes with two Peripherals, the Host of one of the Peripherals rejects the request to establish a CIS with the Central (see Figure 6.48).

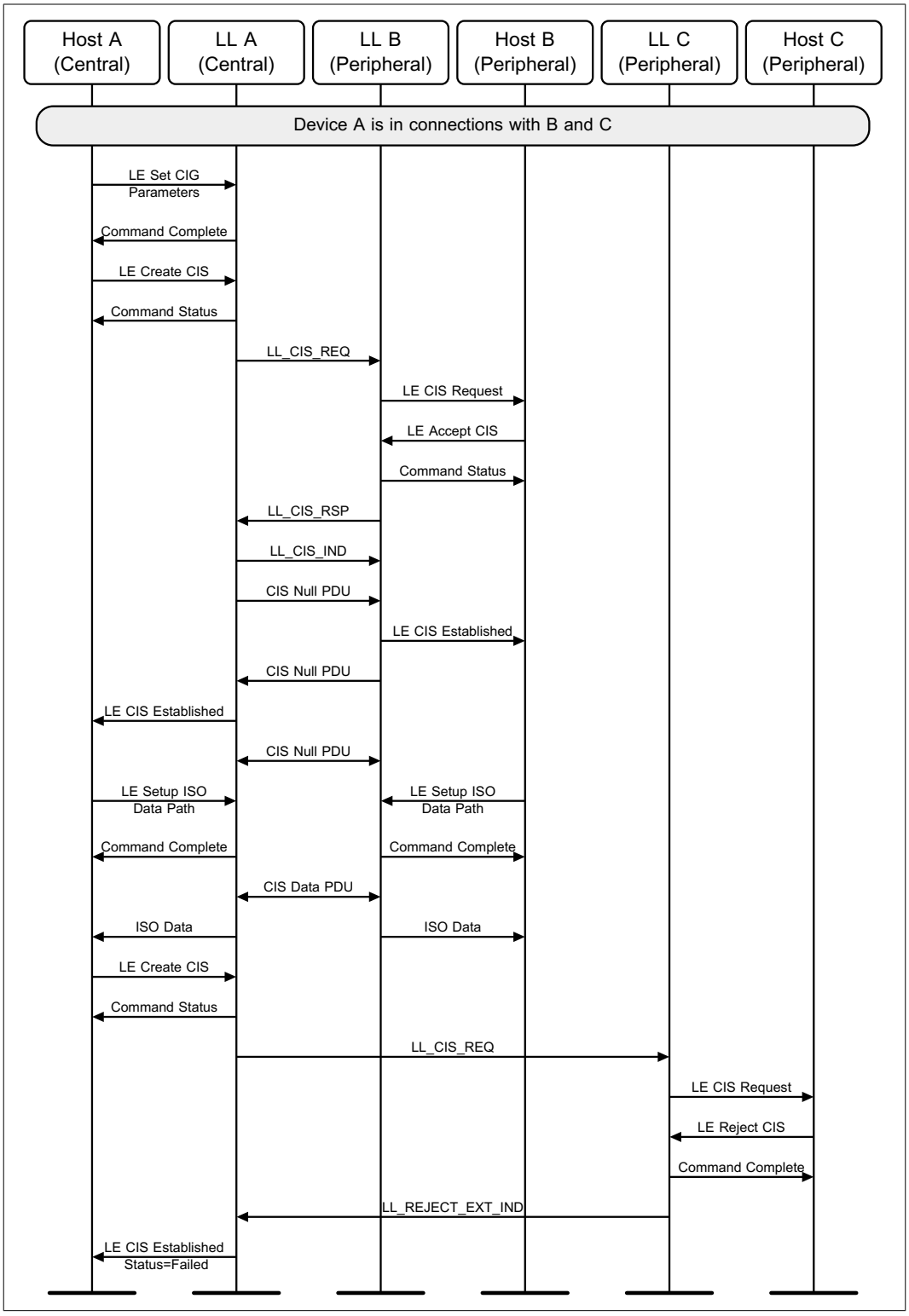



*Figure 6.48: The Host in Device C rejects a CIS from Device A*

## 6.21 Link Layer Rejects Connected Isochronous Stream

The Link Layer of the Peripheral rejects the request to establish a CIS with a Central (see Figure 6.49).

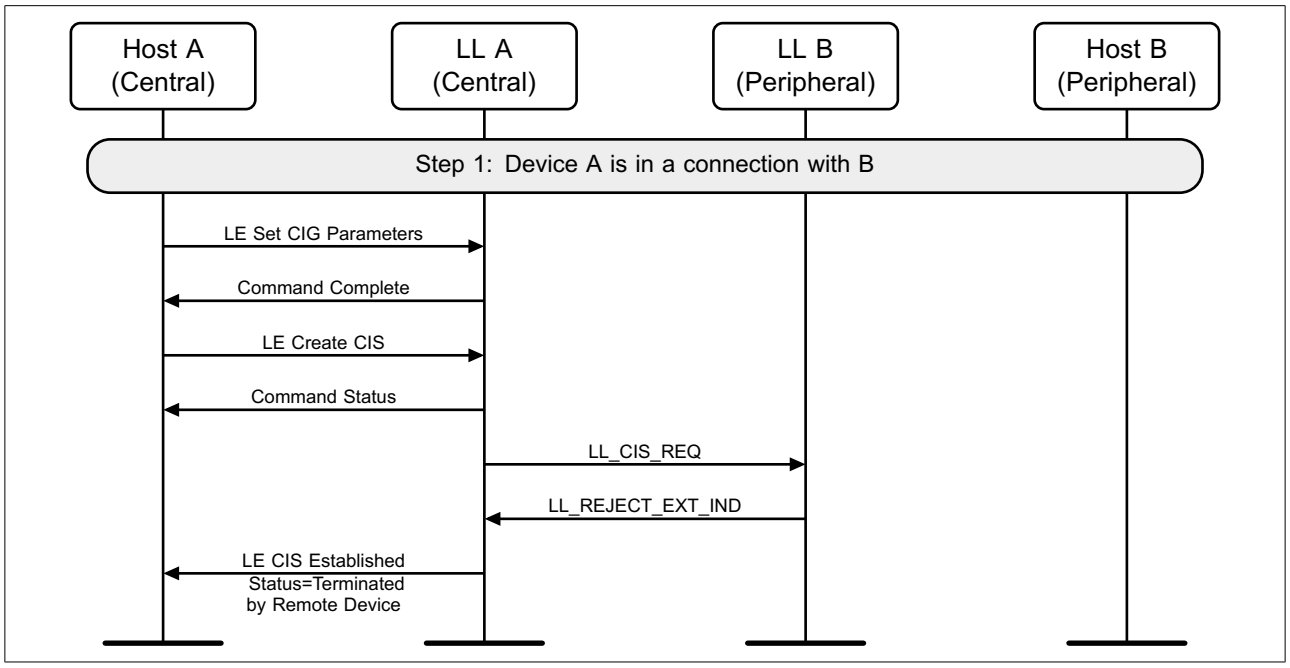



*Figure 6.49: The Link Layer in Device B rejects a CIS request from Device A*

While setting up CISes with two Peripherals, the Link Layer of one of the Peripherals rejects the request to establish a CIS with the Central (see Figure 6.50).

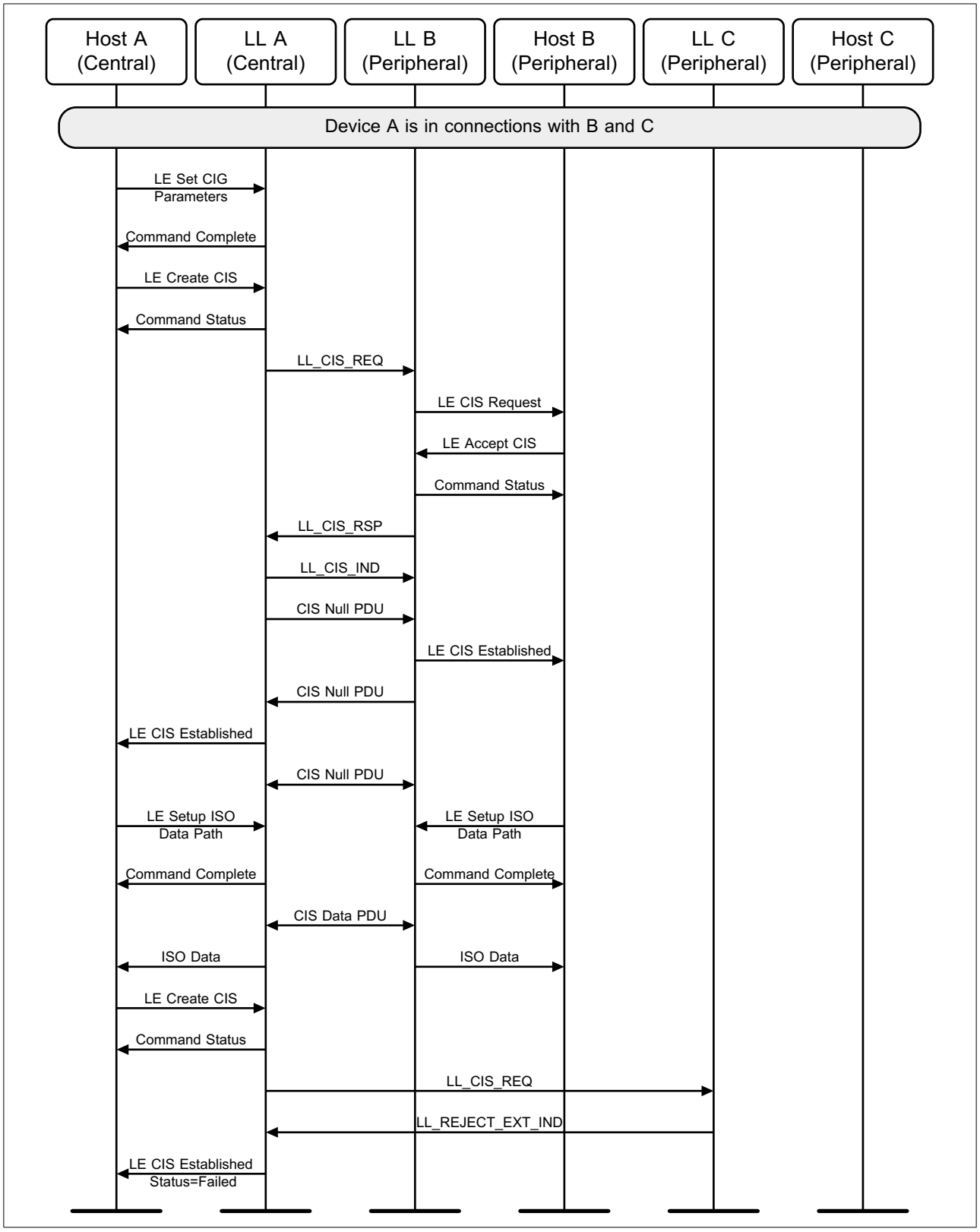



*Figure 6.50: The Link Layer in Device C rejects a CIS from Device A*

## 6.22 Link Layer Rejects Connected Isochronous Stream

The Link Layer of the Central rejects the request to create a CIS (see Figure 6.51).

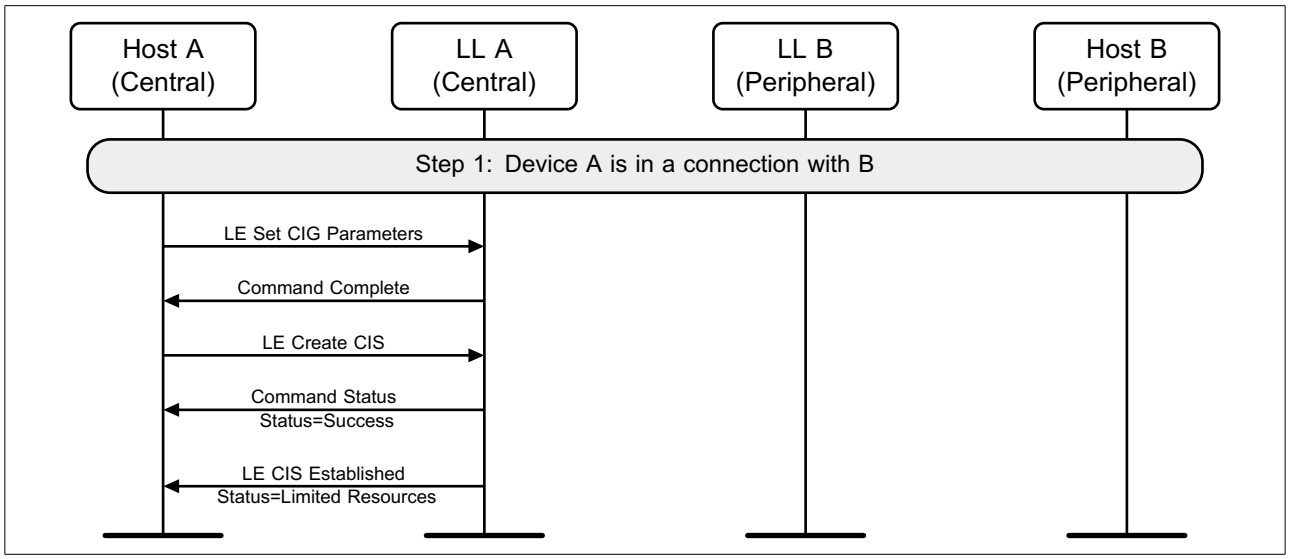



*Figure 6.51: Link Layer in Device A rejects a request to create a CIS*

## 6.23 Host A Terminates Connected Isochronous Stream

The Host of the Central terminates a Connected Isochronous Stream (see Figure 6.52). Either A or B can be the Central.

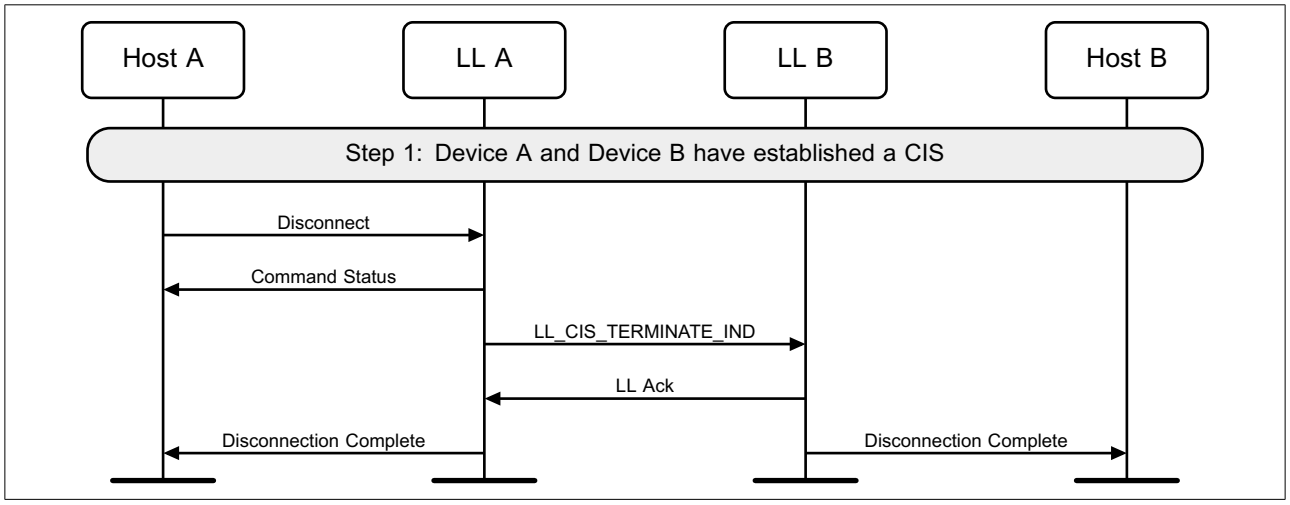



*Figure 6.52: Device A terminates an established CIS*

The Host of device A, which is the Central, terminates a Connected Isochronous Stream either before the Link Layer starts creating that CIS (see Figure 6.53) or during the creation process (see Figure 6.54).

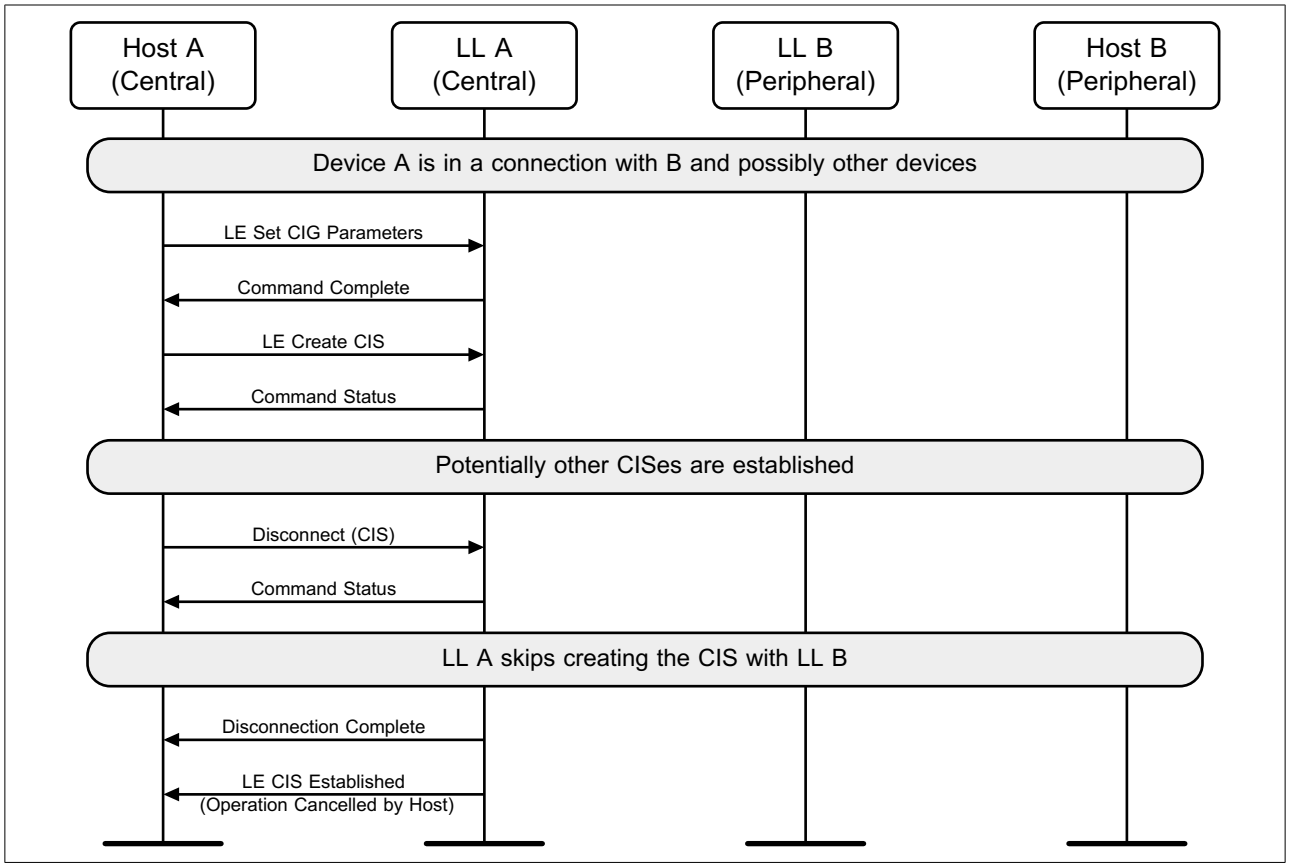



*Figure 6.53: Device A terminates a CIS before the Link Layer starts creating it*

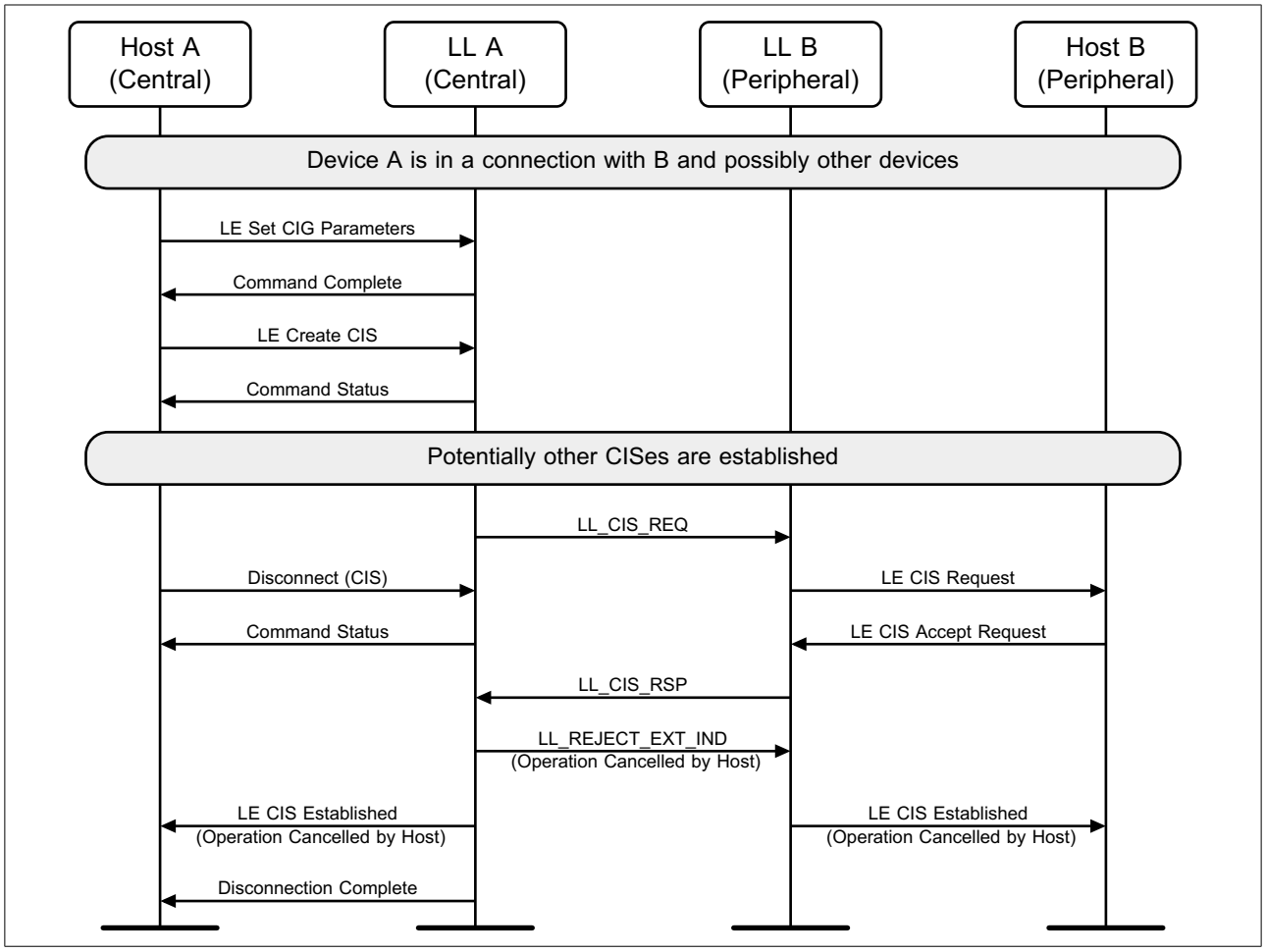



*Figure 6.54: Device A terminates a CIS during the creation process*

## 6.24 ACL disconnected

The disconnection of the ACL causes the disconnections of CISes (see Figure 6.55).

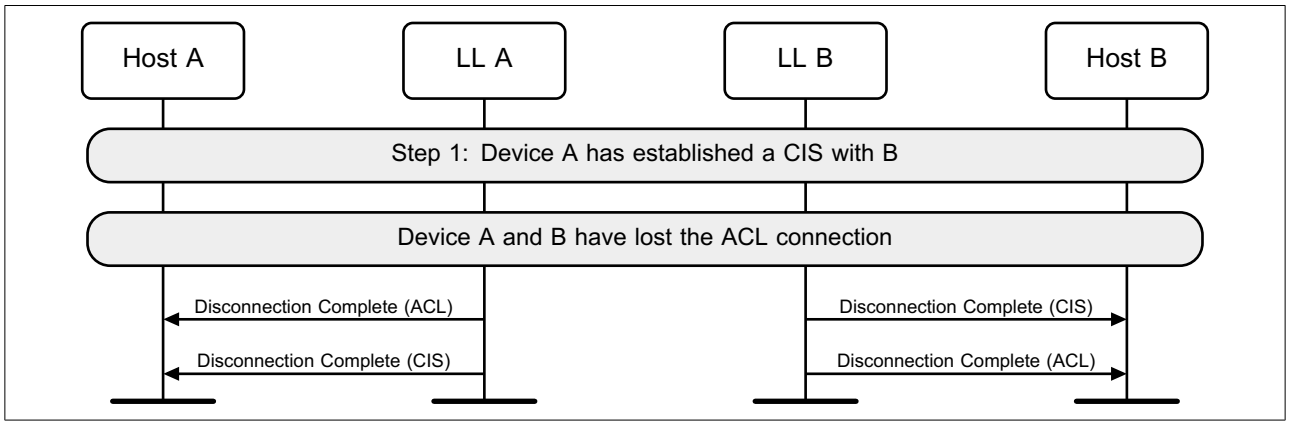



*Figure 6.55: ACL connection Terminated*

## 6.25 Host A Removes Connected Isochronous Group

A Host of the Central terminates the CIS and removes the CIG from the Link Layer (see Figure 6.56).

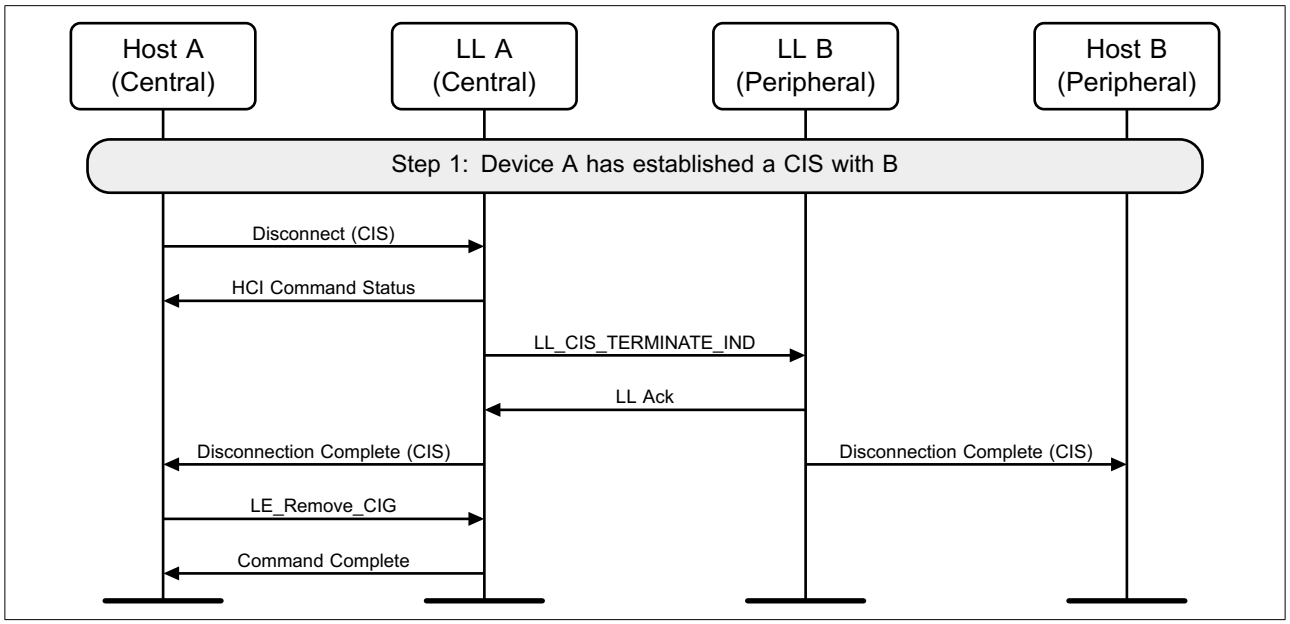



*Figure 6.56: Device A terminates a CIS and removes a CIG*

The Host of the Central terminates both the CISes in the CIG and removes the CIG from the Controller (see Figure 6.57).

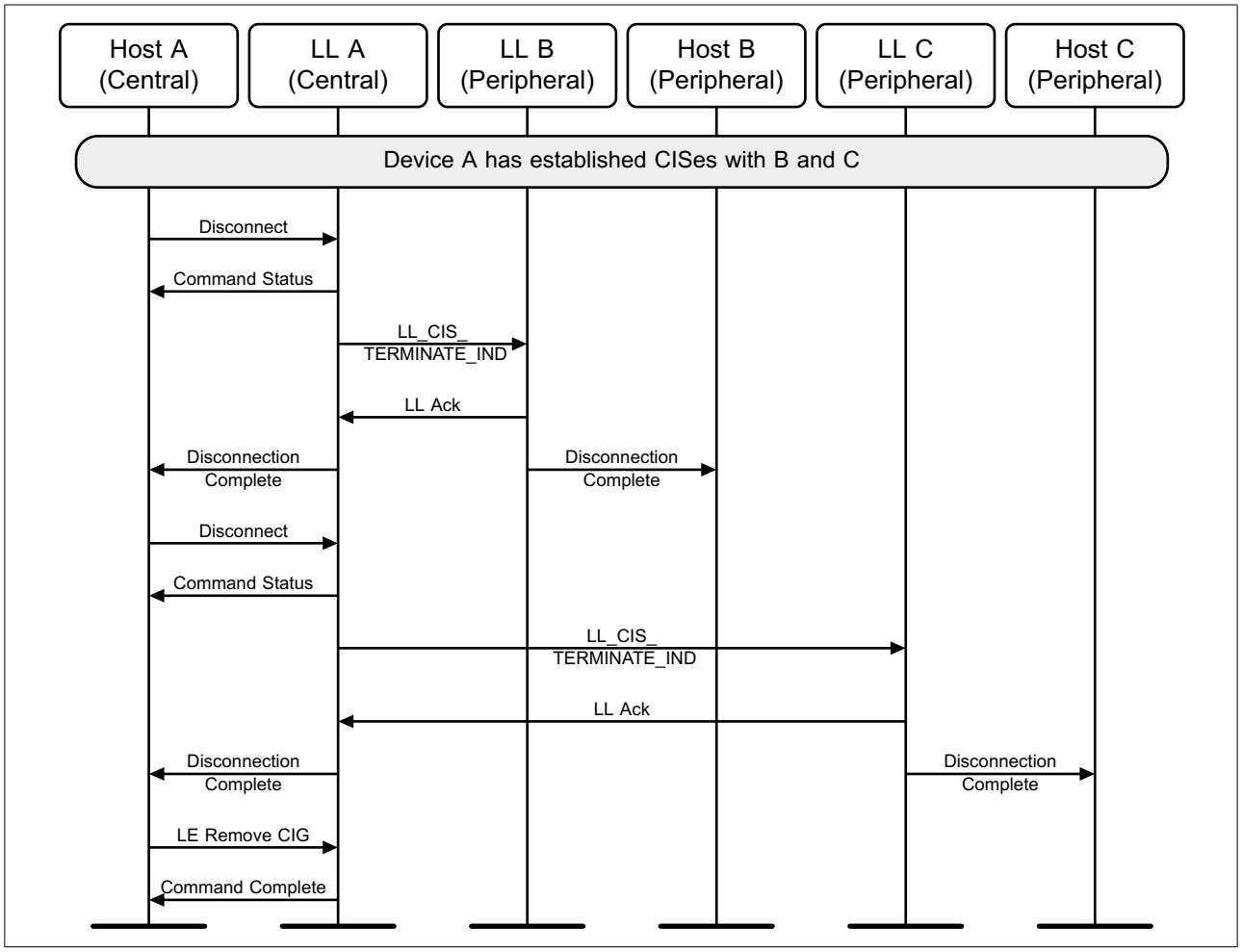



*Figure 6.57: Device A terminates both CISes and removes the CIG*

## 6.26 Request Sleep Clock Accuracy

Either device can initiate a Sleep Clock Accuracy Update procedure to query the sleep clock accuracy of the peer device (see Figure 6.58).

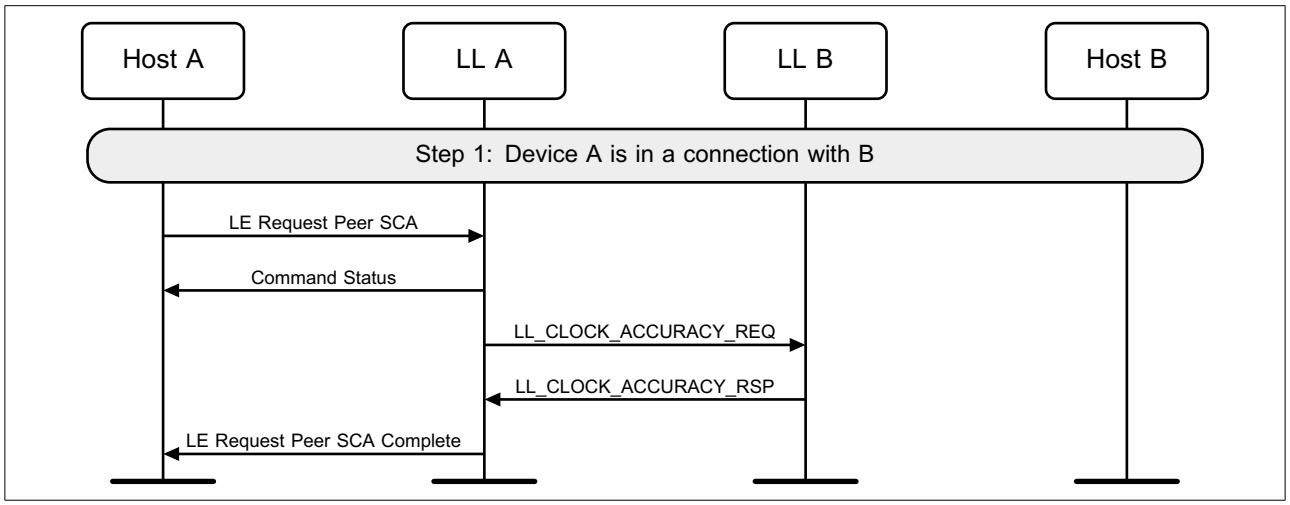



*Figure 6.58: Device A requests SCA from Device B*

## 6.27 Power Control

Either device can initiate a Power Control Request procedure to request the peer device to adjust its transmit power level (see Figure 6.59).

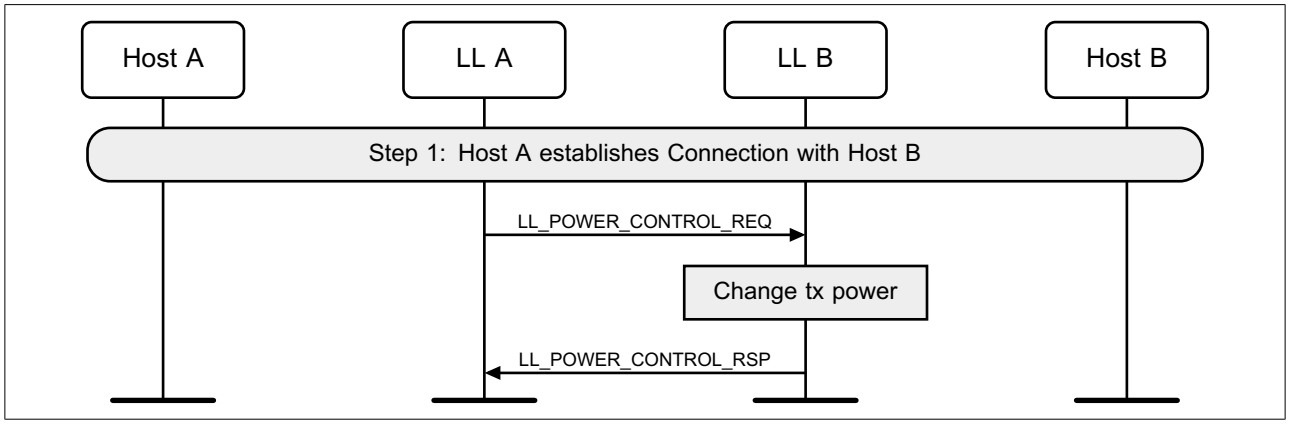



*Figure 6.59: Power Control Request procedure to request remote transmit power update*

Either device can use the Power Control Request procedure to query the Acceptable Power Reduction (APR) value from the peer device and adjust its transmit power based on the response (see Figure 6.60).

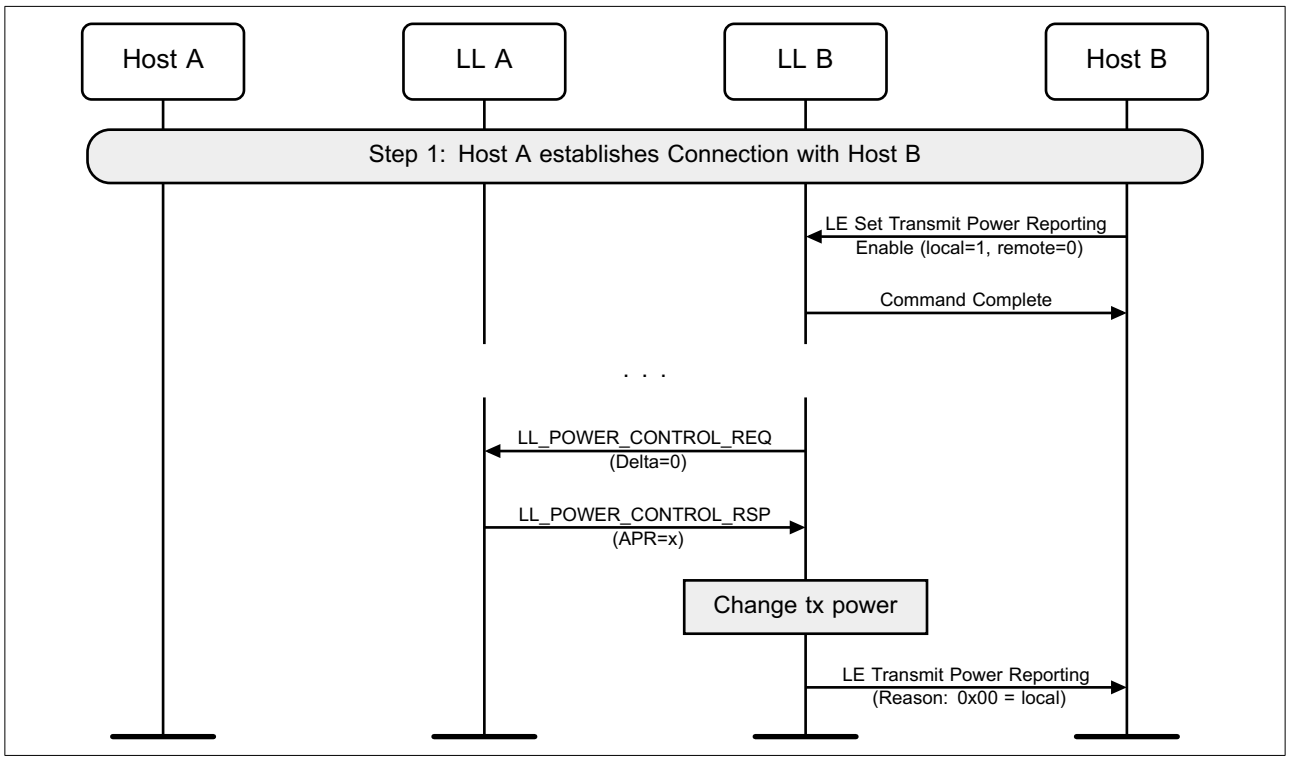



*Figure 6.60: Power Control Request procedure to query Acceptable Power Reduction (APR) and transmitter Update*

When the Host issues a command to enable reporting of remote power level changes, the Controller may initiate a Power Control Request procedure to request the remote device to start power level management (see [Vol 6] Part B, Section 4.5.15). When the remote Controller changes its transmit power level, it sends an indication to the peer device (see Figure 6.61). If reporting of remote transmit power level changes is enabled, the Controller sends an LE Transmit Power Reporting event to the Host.

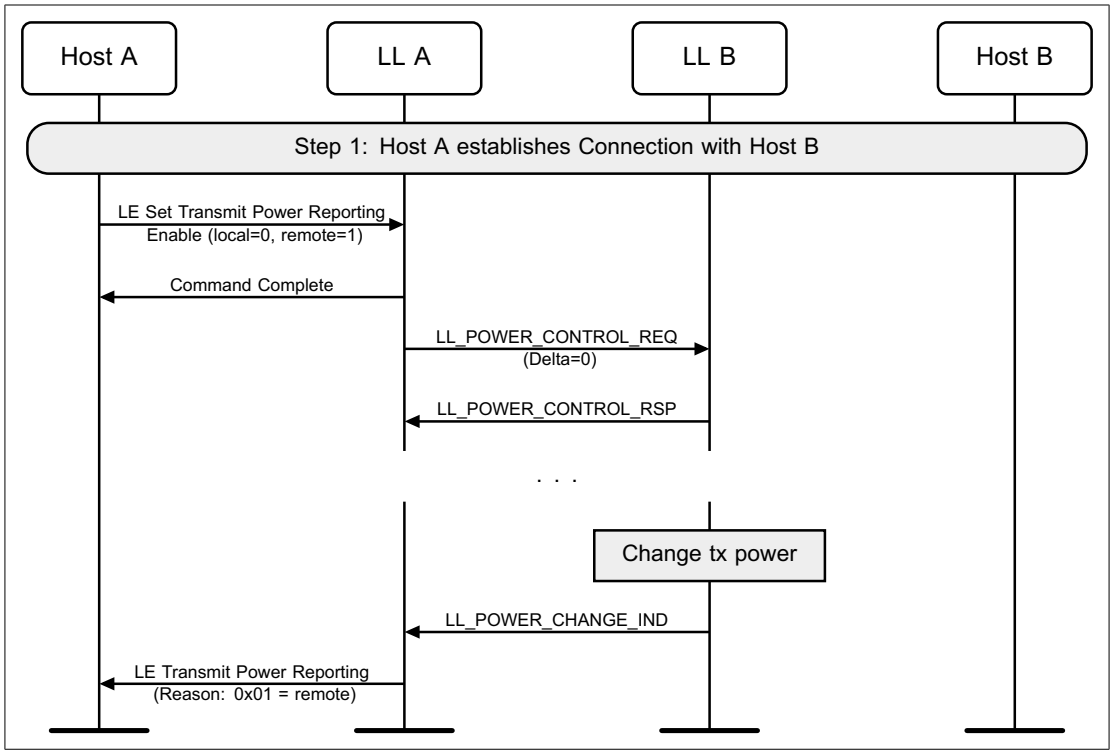



*Figure 6.61: LL Power Change Indication procedure when device adjusts TX power autonomously*

When a Host issues a command to read the transmit power level of a remote device, the Controller can initiate an LL Power Control Request procedure to query the transmit power level of the remote device (see Figure 6.62).

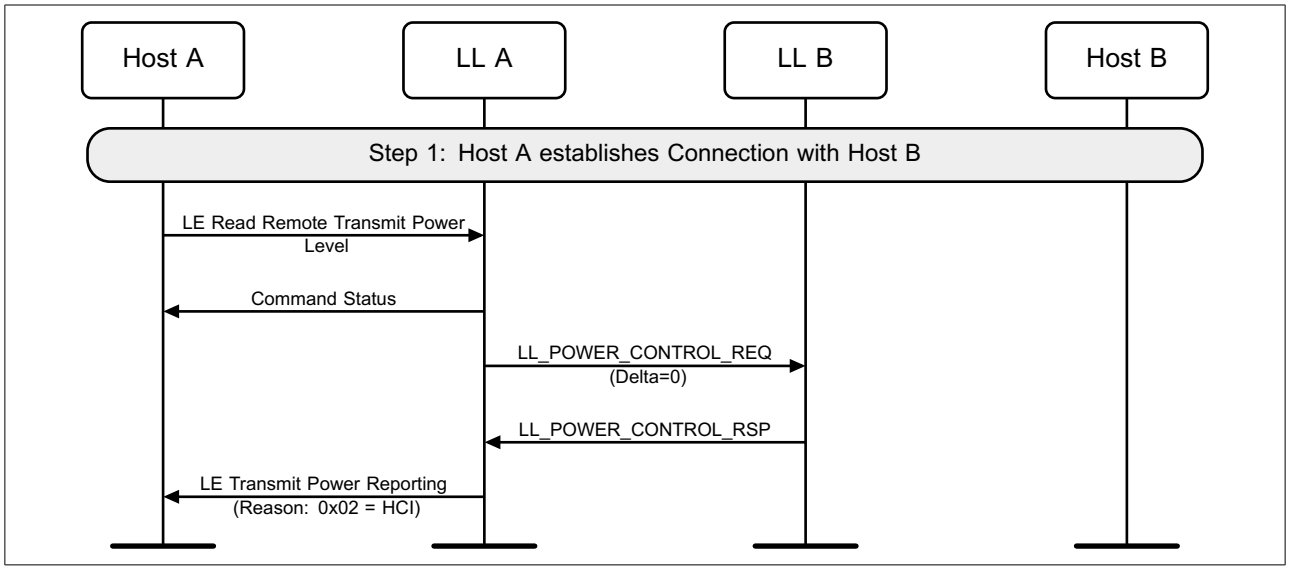



*Figure 6.62: HCI command to read remote transmit power level*

Before a PHY Update procedure is performed, the Link Layer may request a preferred transmit power level for the new PHY to be used by the remote device (see Figure 6.63).

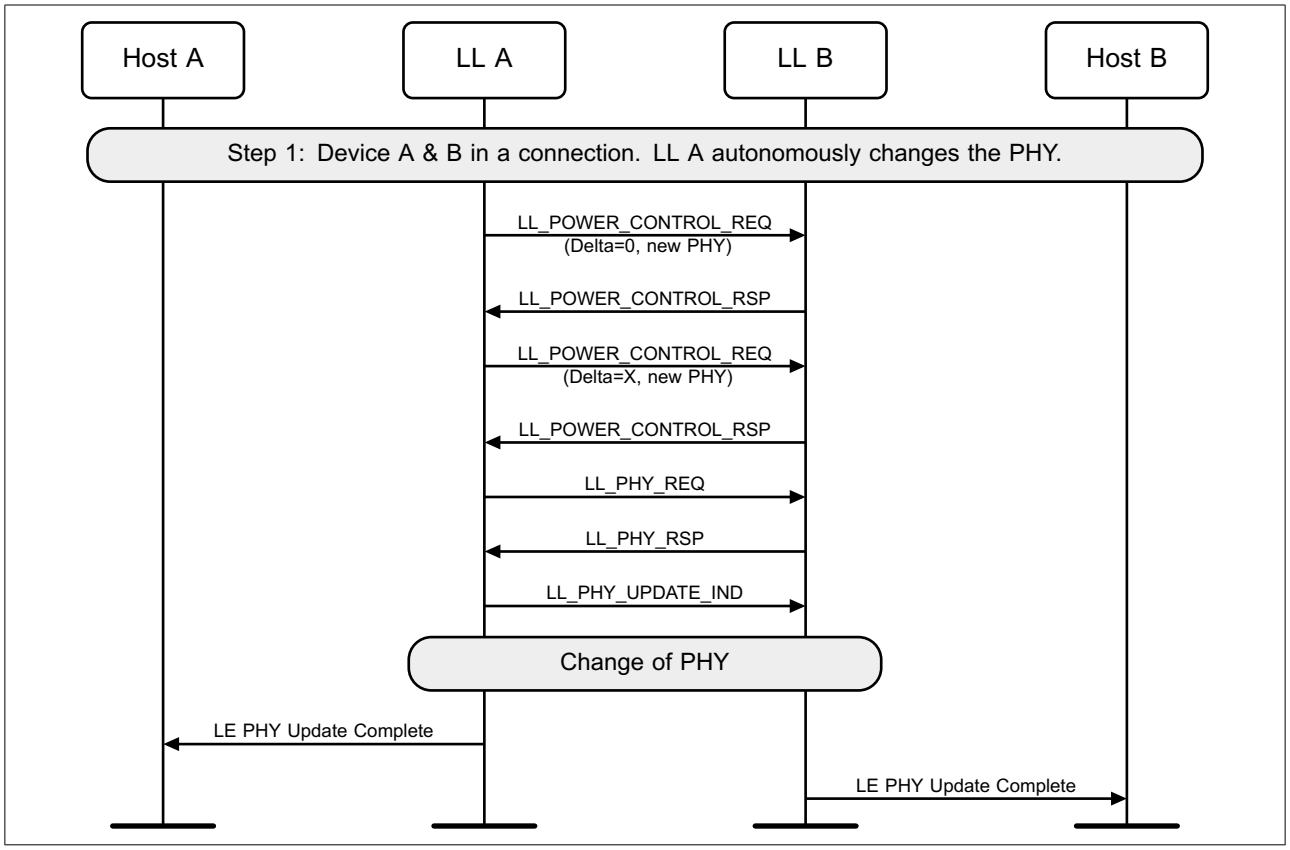



*Figure 6.63: Autonomous Central-initiated PHY Update procedure with preferred transmit power level. PHY changed in both directions.*

An implementation may choose to manage transmit power levels only on active PHYs and could reject a request for preferred transmit power level on the new PHY before a PHY Update procedure is performed (see Figure 6.64).

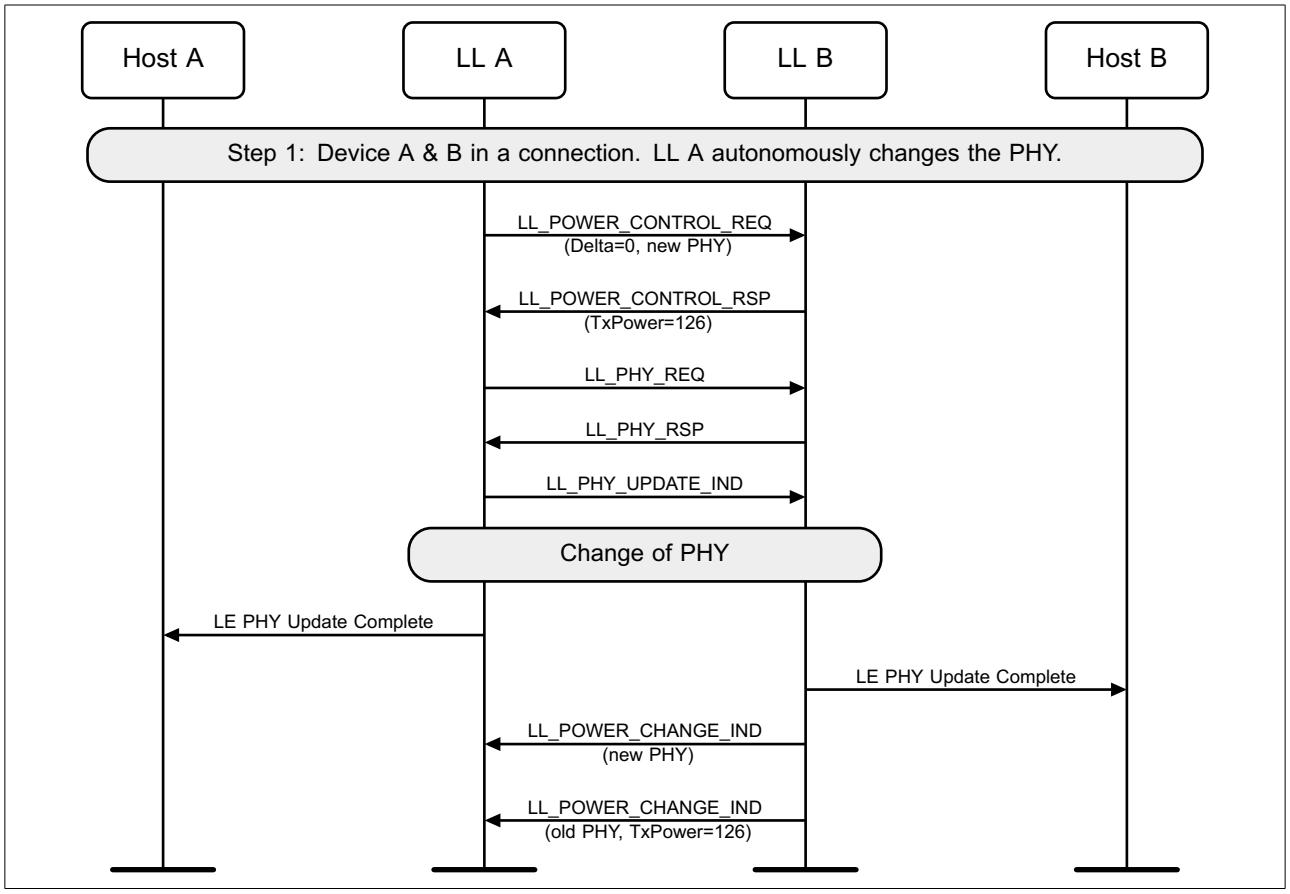



*Figure 6.64: Autonomous Central-initiated PHY Update procedure with remote device maintaining only a single transmit power level. PHY changed in both directions.*

When a PHY Update procedure is performed to switch to the LE Coded PHY, where the local and remote devices can use both S=8 and S=2 coding, the Controller treats them as separate PHYs for the purpose of Power Control (see Figure 6.65).

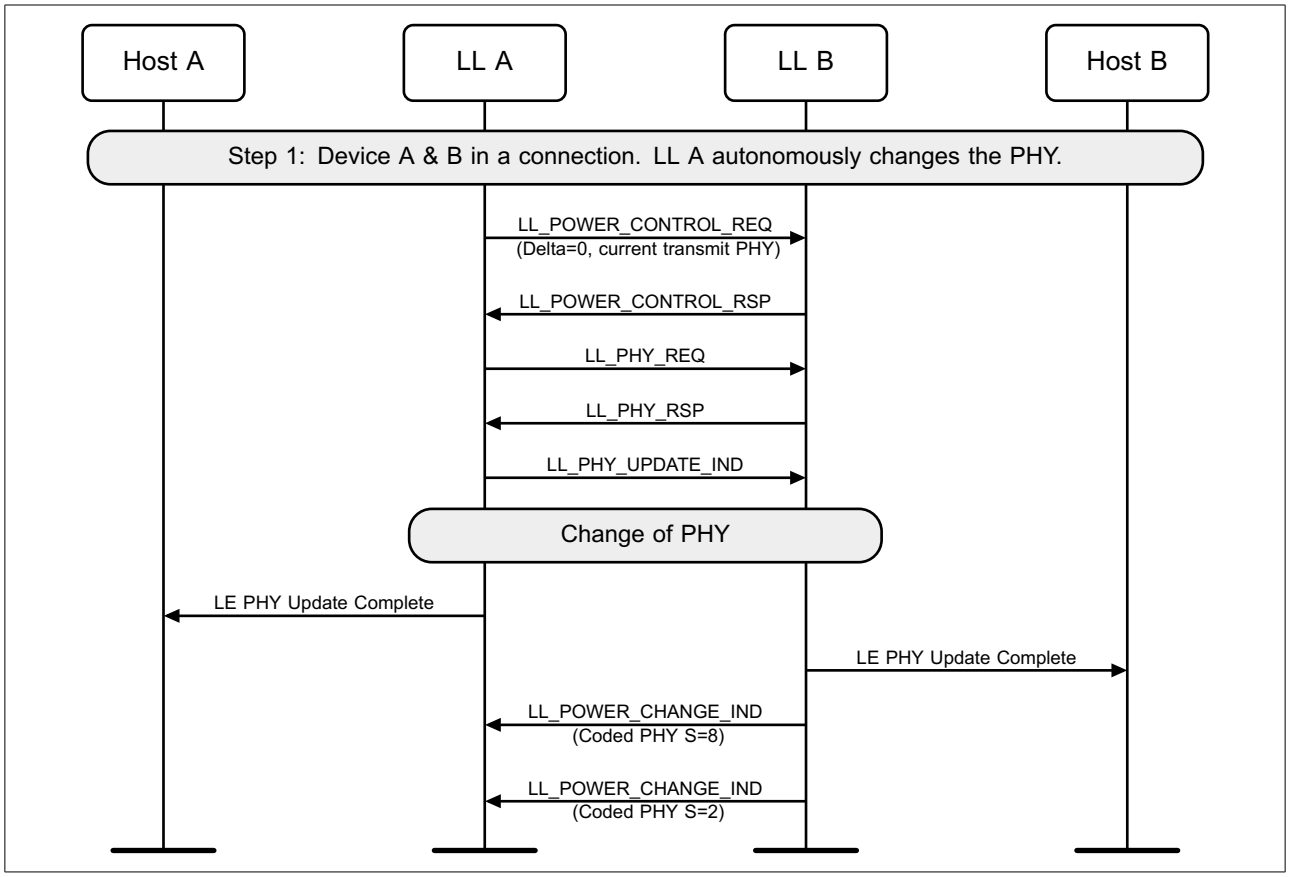



*Figure 6.65: Autonomous Central-initiated PHY Update procedure to change to the LE Coded PHY (S=8 and S=2). PHY changed in both directions.*

When an associated CIS is established on a PHY different from the one used by ACL, the Power Change Indication procedure may be used to indicate the transmit power level used on the new PHY. When the associated CIS is disconnected, the implementations may choose to stop managing the PHY used on the CIS (see Figure 6.66).

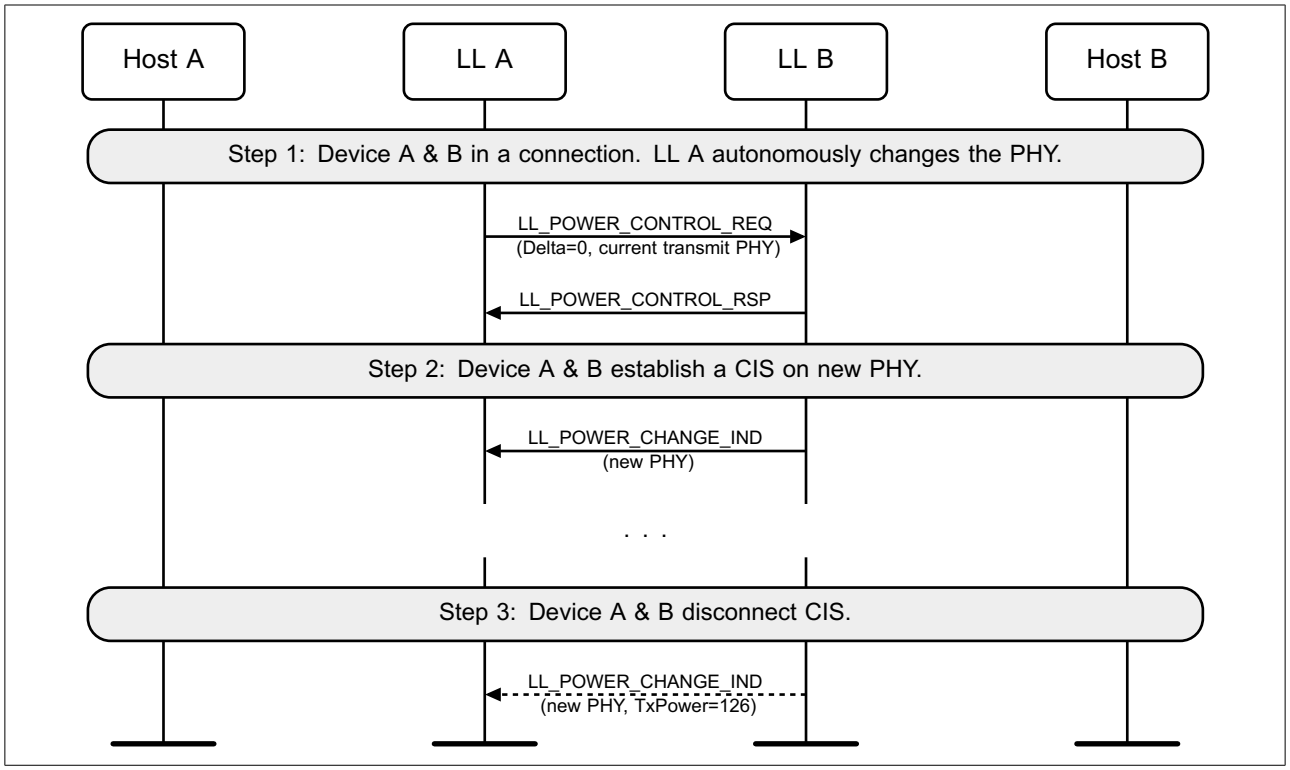



*Figure 6.66: ACL with associated CIS established on a new PHY different from one on ACL*

When a Host enables path loss monitoring, the Controller may initiate the Power Control Request procedure to query the remote transmit power level. As the remote device moves around, the Controller monitors the path loss on the connection and sends events to the Host as appropriate (see Figure 6.67).

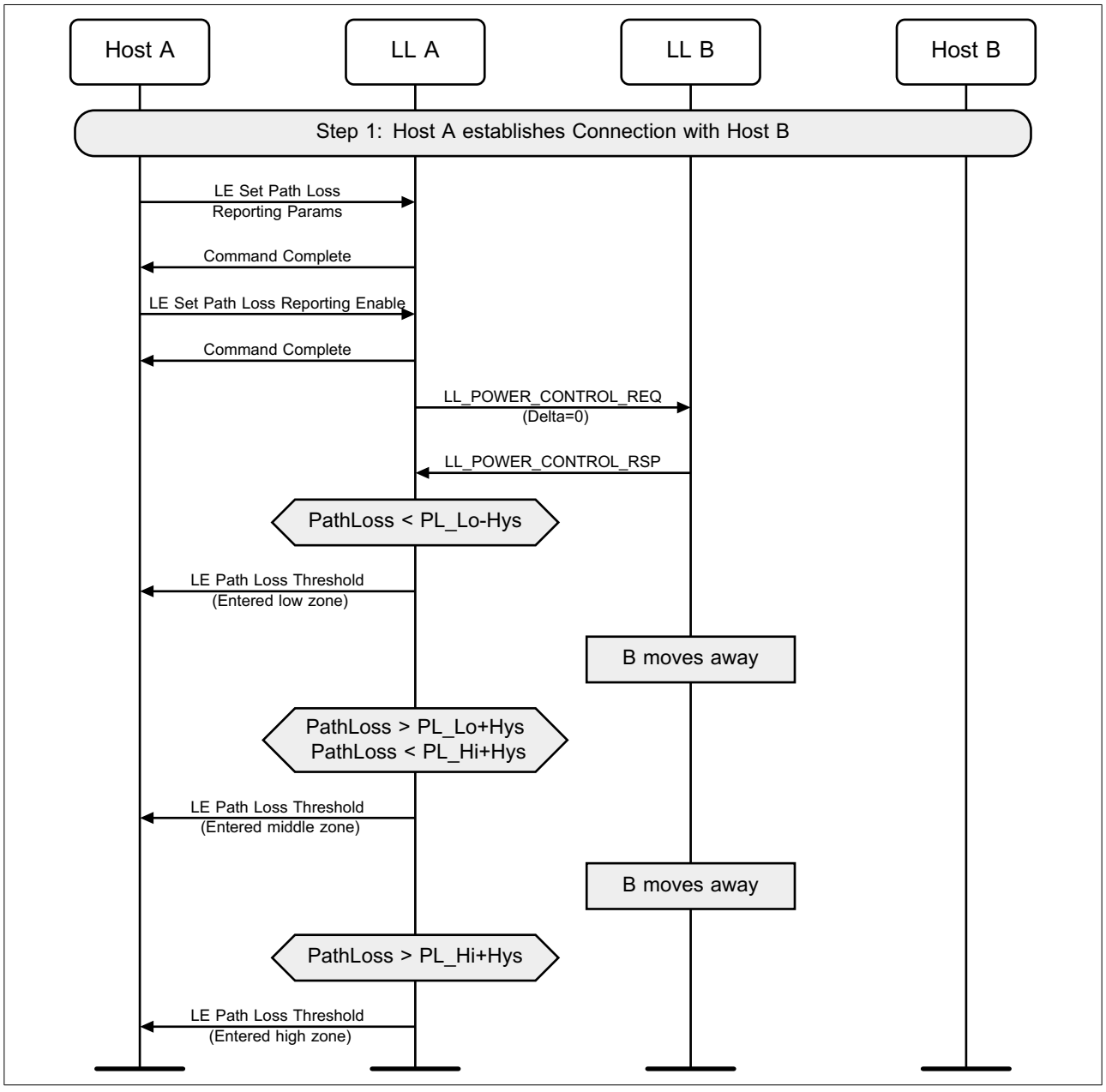



*Figure 6.67: Power Control Request procedure in context of path loss monitoring*

## 6.28 Data path setup for a music stream over a CIS

Figure 6.68 shows an example of unidirectional music streaming from Host A to Host B. Host A does LC3 codec processing in the Host and sends data over HCI to the Controller, which transmits the PDUs over the air without further processing. Host B

configures an output data path to the speaker(s) and enables LC3 codec processing in the Controller.

The notifications between the Hosts are specified by the relevant profile(s).

If the audio data is stereo, then how the data sent to the output data path is split into the separate audio signals is outside the scope of this specification.

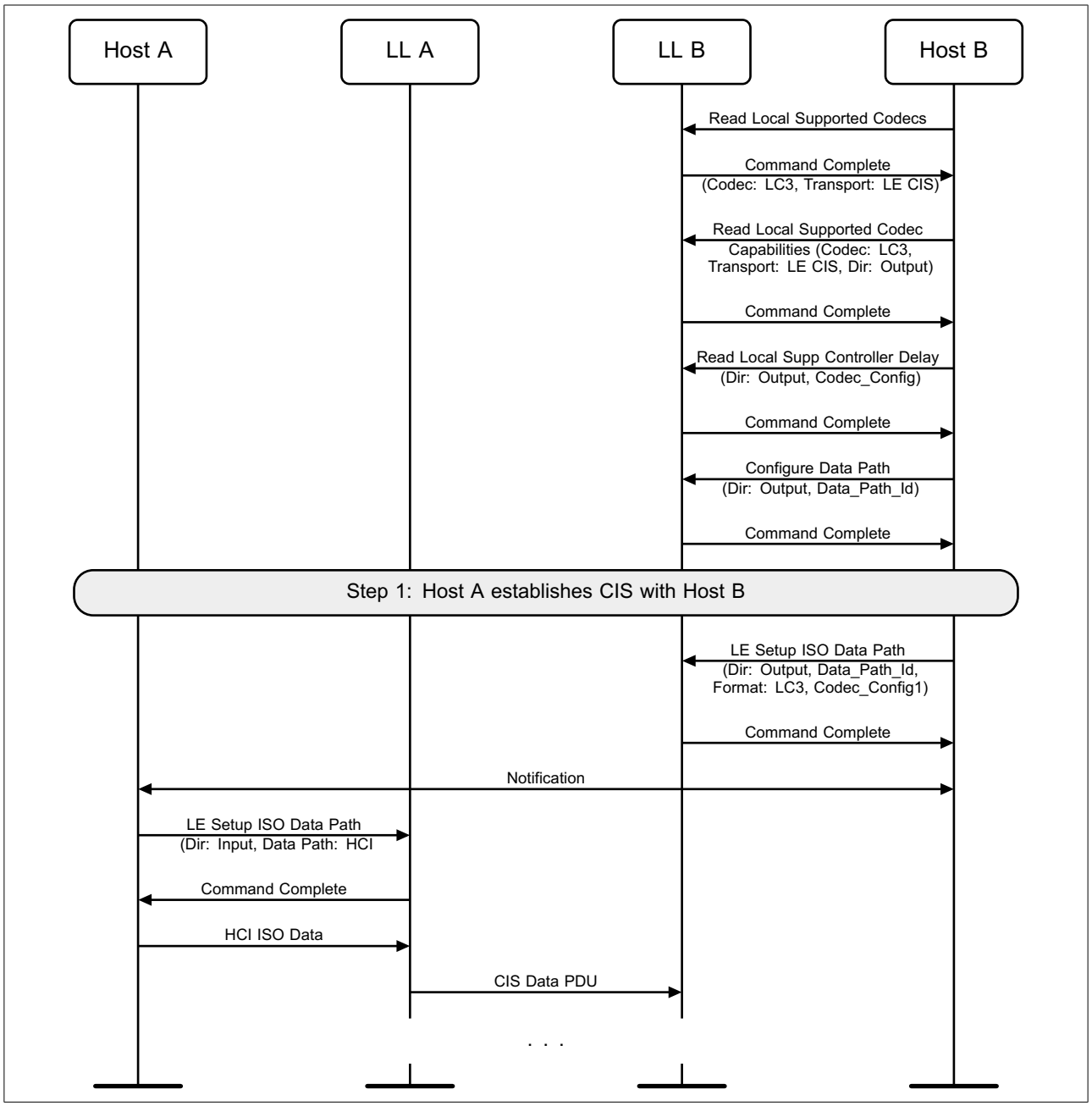



*Figure 6.68: Host A with codec in the Host streams music to Host B which has a codec in the Controller*

## 6.29 Data path setup for bi-directional voice over a CIS

Figure 6.69 shows an example of a bidirectional voice call between Host A and Host B. Device A contains proprietary audio interface hardware with an embedded LC3 codec. Hence the Host sets up the ISO data path with Coding_Format set to transparent mode. Device B supports LC3 codec processing in the Controller and hence Host B sets up the ISO data path with Coding_Format set to LC3 along with the codec configuration data defined by the relevant profile(s).

The notifications between the Hosts are specified by the relevant profile(s).

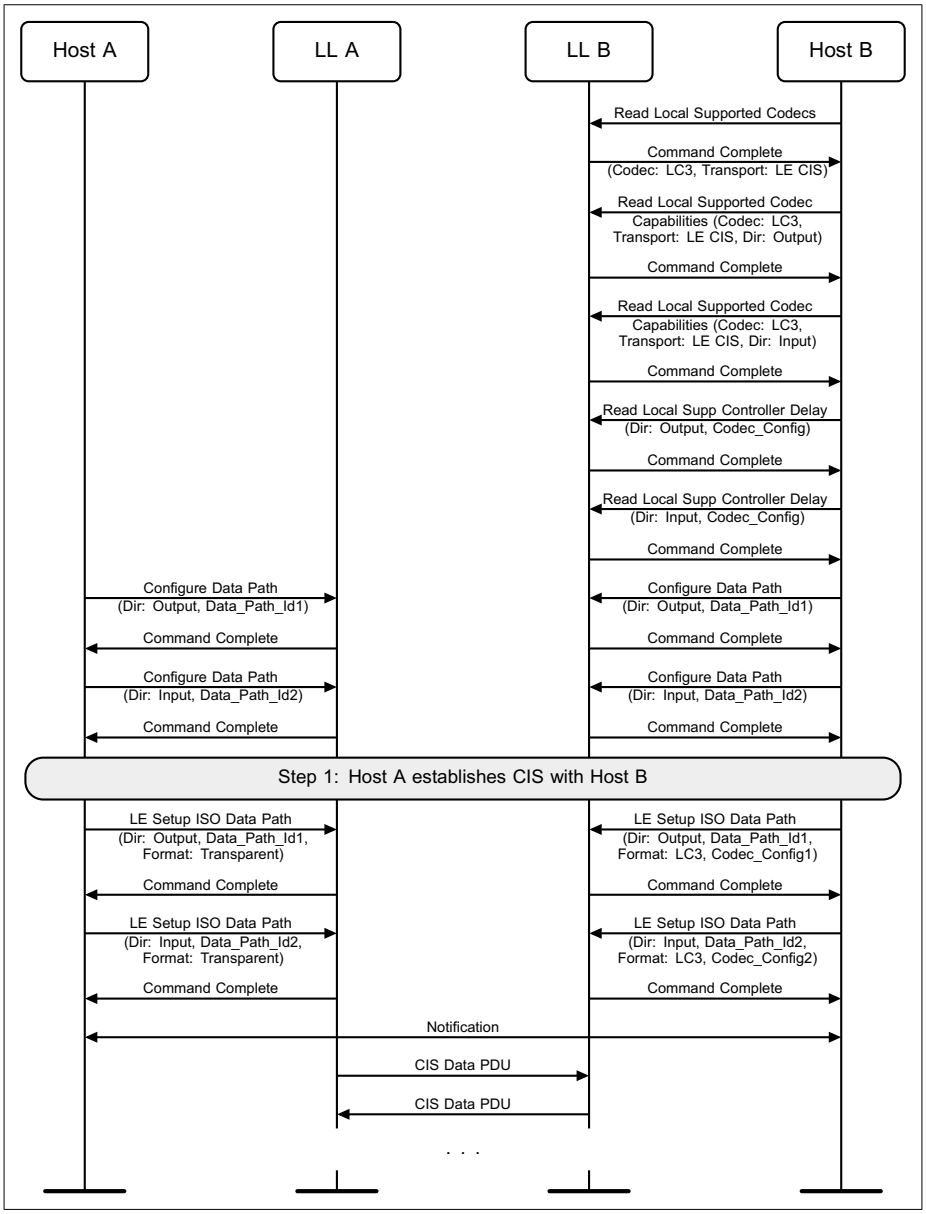



*Figure 6.69: Host A with codec in the data path establishes a bi-directional voice call with Host B with codec in the Controller*

## 6.30 [This section is no longer used]

The contents of this section are now in Section 8.5.

## 6.31 Modifying the subrate of a connection

A Central A in a connection with a Peripheral B modifies the subrating of the connection. The first two PDUs are not received by device B (see Figure 6.70).

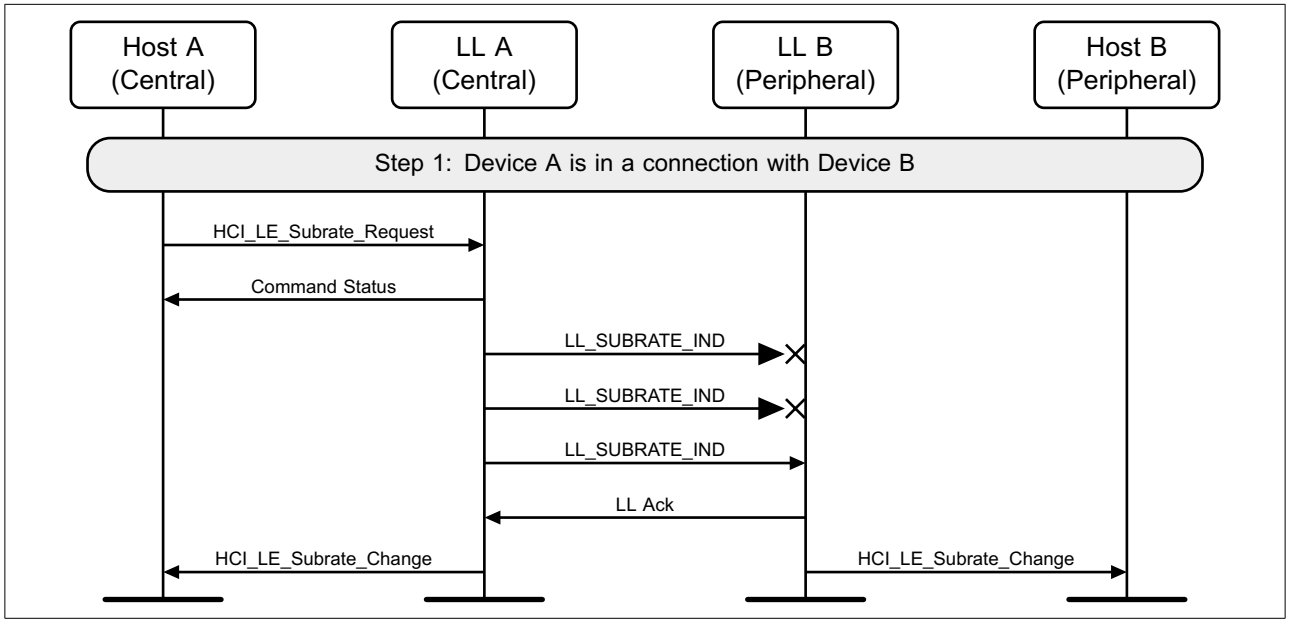



*Figure 6.70: Central A modifies the connection subrating*

Peripheral B requests a new subrating for its connection with device A and makes the change (see Figure 6.71).

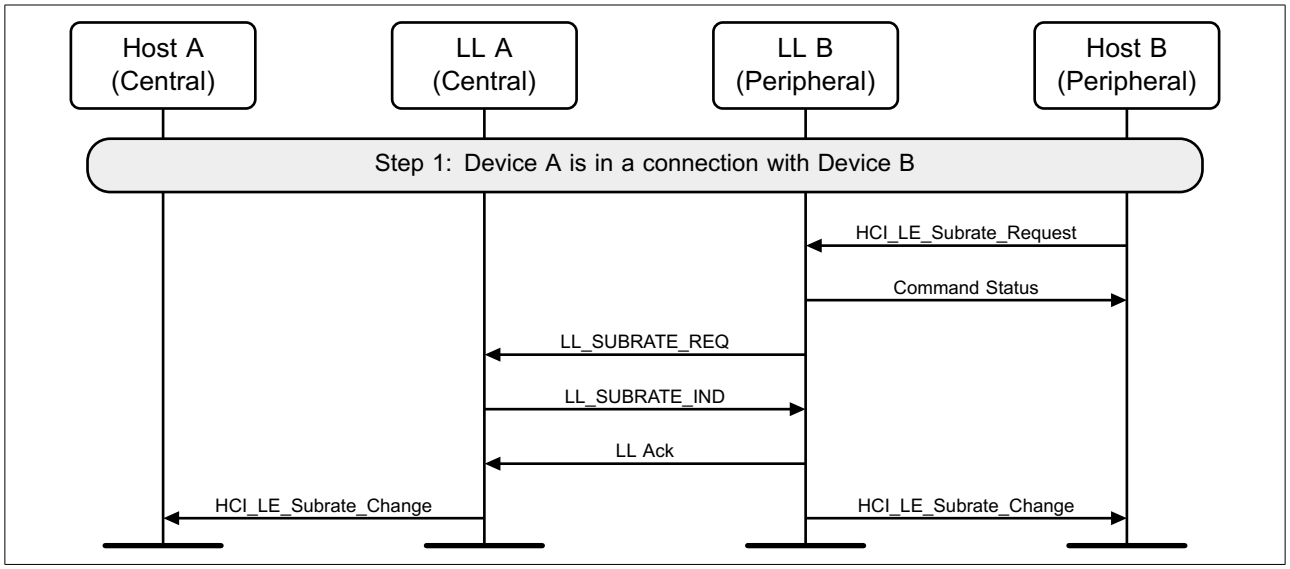



*Figure 6.71: Peripheral B requests a change to the connection subrate which is accepted*

Peripheral B requests a new Subrating for its connection with device A but device A rejects the change (see Figure 6.72).

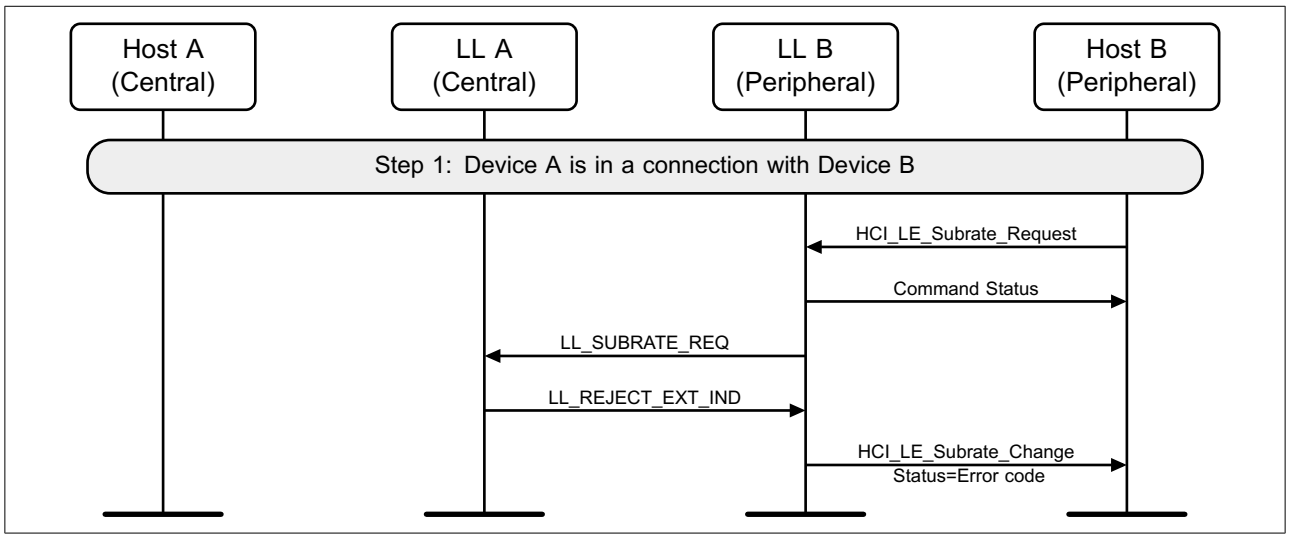



*Figure 6.72: Peripheral B requests a change to the connection subrate which is rejected*

## 6.32 Channel Classification Enable

The Central can enable reporting of channel classification information on the Peripheral.

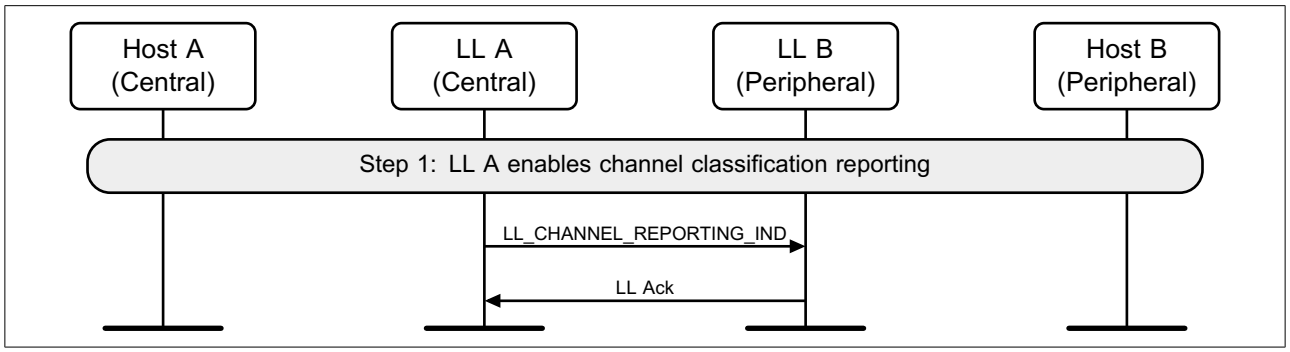



*Figure 6.73: Central requests the Peripheral to enable reporting of channel classification information*

## 6.33 Channel Classification Reporting

The reporting is enabled, the Peripheral can provide channel classification information to the Central.

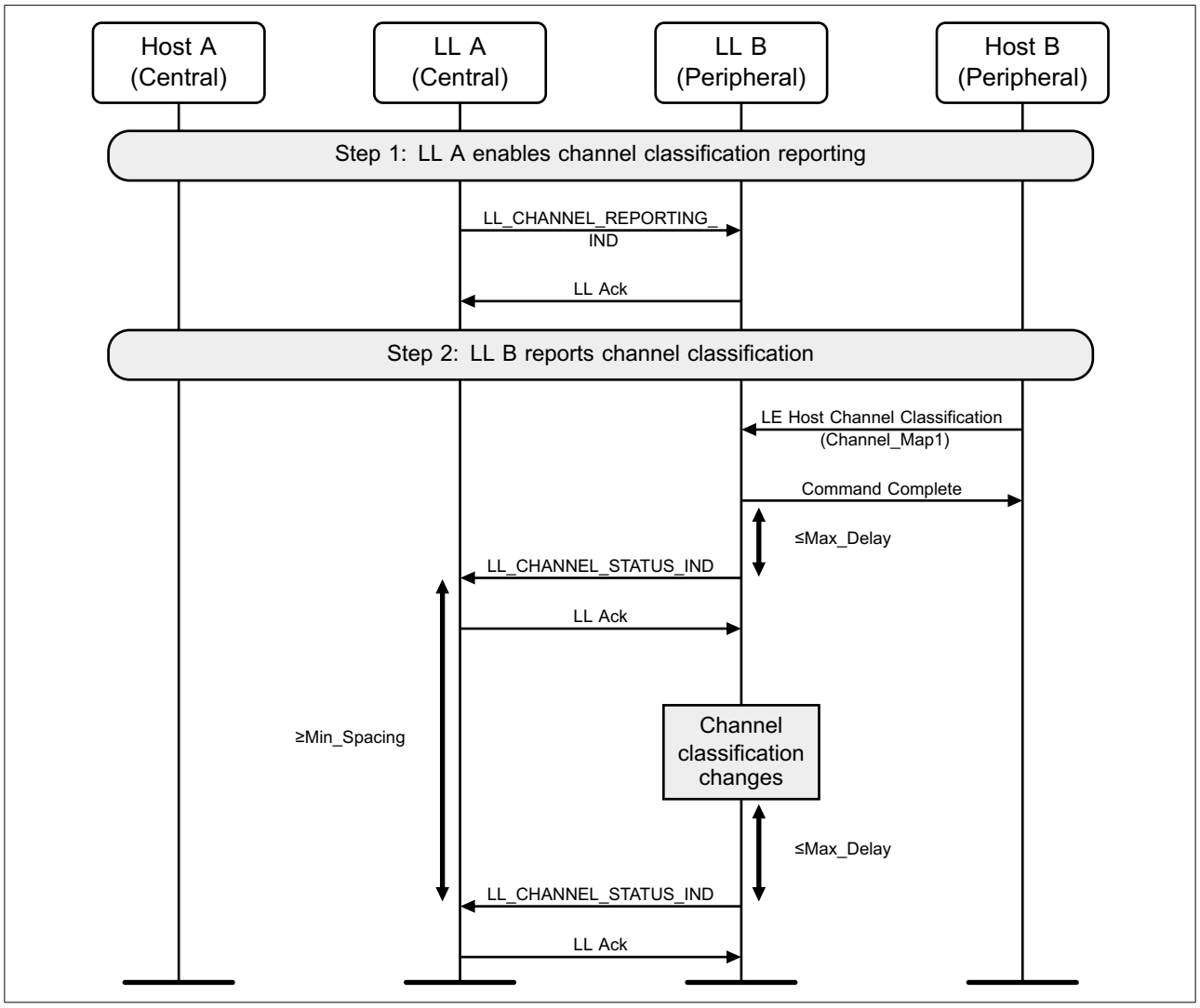



*Figure 6.74: Peripheral provides channel classification information to the Central*

## 6.34 Channel Sounding setup phase

The example flow in Figure 6.75 shows the CS setup phase, in which CS capabilities are read and roles are enabled at both ends of a connection. An encrypted ACL connection is established before exchanging CS capabilities and security parameters.

A distance measurement application in a Host can issue HCI commands to start the setup phase with the Channel Sounding Capability Exchange procedure. The local and remote Host can then set the default settings for Channel Sounding. Based on the capability support for a non-zero Frequency Actuation Error, the application can issue appropriate commands to read or write the mode-0 Frequency Actuation Error table. The application can create CS configurations using the Channel Sounding Configuration procedure. The Controller in the Central role can then start the Channel Sounding Security Start procedure. After the Channel Sounding Security Start procedure completes, the application can consider the setup phase to be completed.

After the first time the setup phase is completed with a specific remote device, an application may cache the remote capabilities, remote FAE table, and CS configuration parameters used. During subsequent connections, the Host may avoid LL control procedure exchanges as described in Figure 6.75.

*Message Sequence Charts*

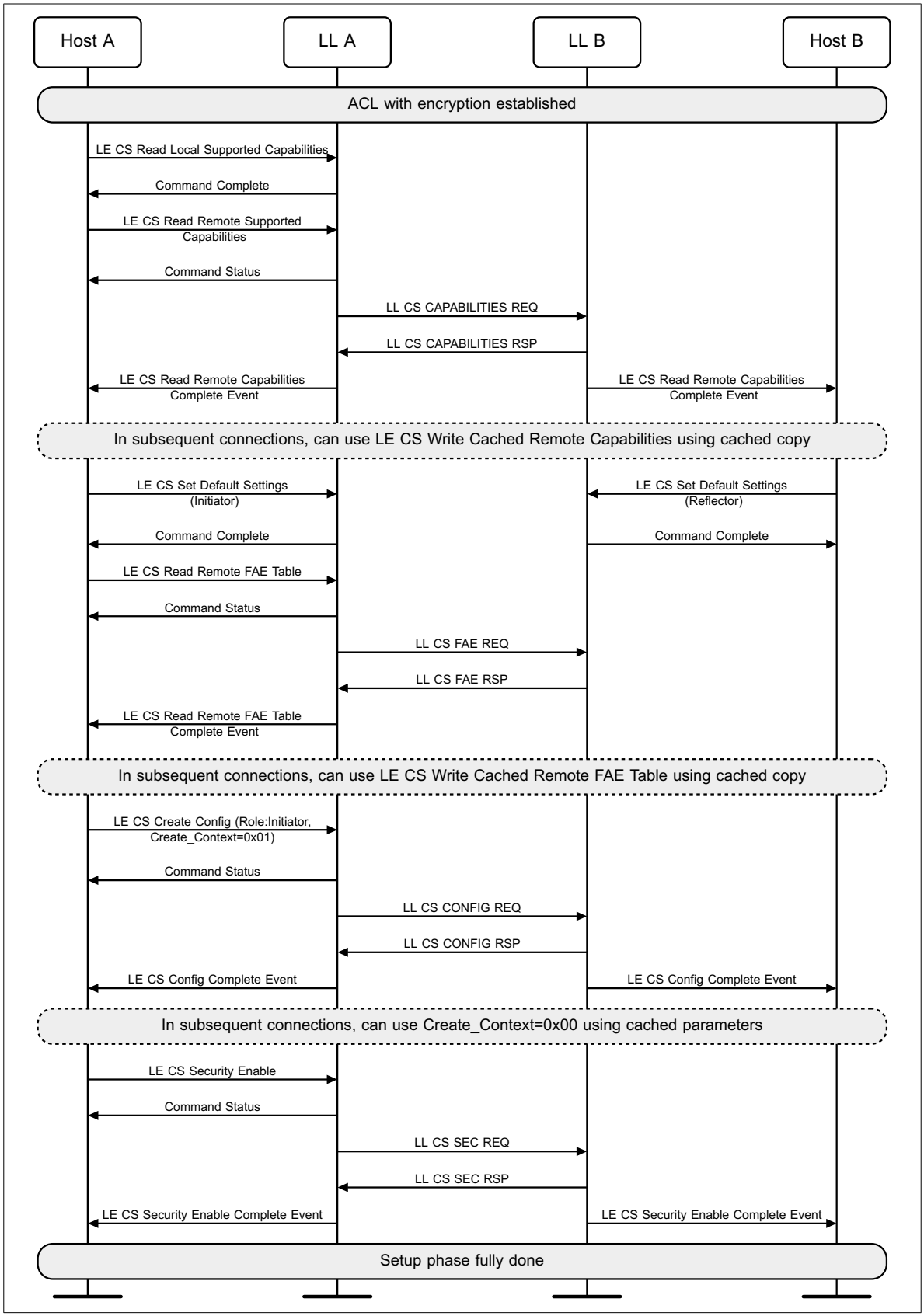



*Figure 6.75: Channel Sounding example setup phase*

## 6.35 Channel Sounding started by Central in initiator role

As mentioned in [Vol 1] Part A, Section 3.3.2.5.3, an LE device may assume either the initiator or reflector role. In the example shown in Figure 6.76, an application running on a Central device in the initiator role starts a CS procedure.

*Message Sequence Charts*

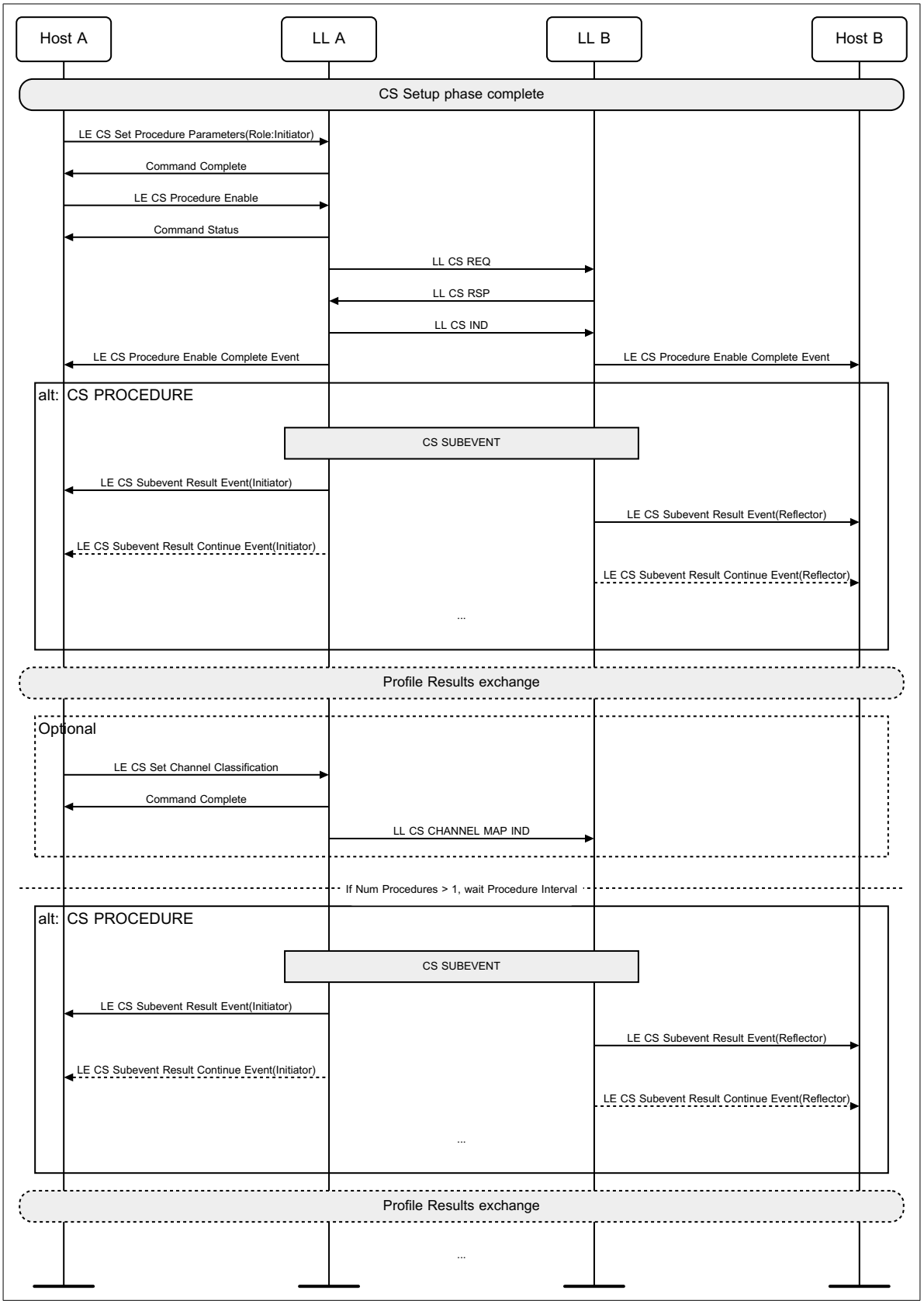



*Figure 6.76: Channel Sounding started by Central in initiator role*

## 6.36 Channel Sounding started by Peripheral in reflector role

As mentioned in [Vol 1] Part A, Section 3.3.2.5.3, an LE device may assume either the initiator or reflector role. In the example shown in Figure 6.77, an application running on a Peripheral device in the reflector role starts a CS procedure.

*Message Sequence Charts*

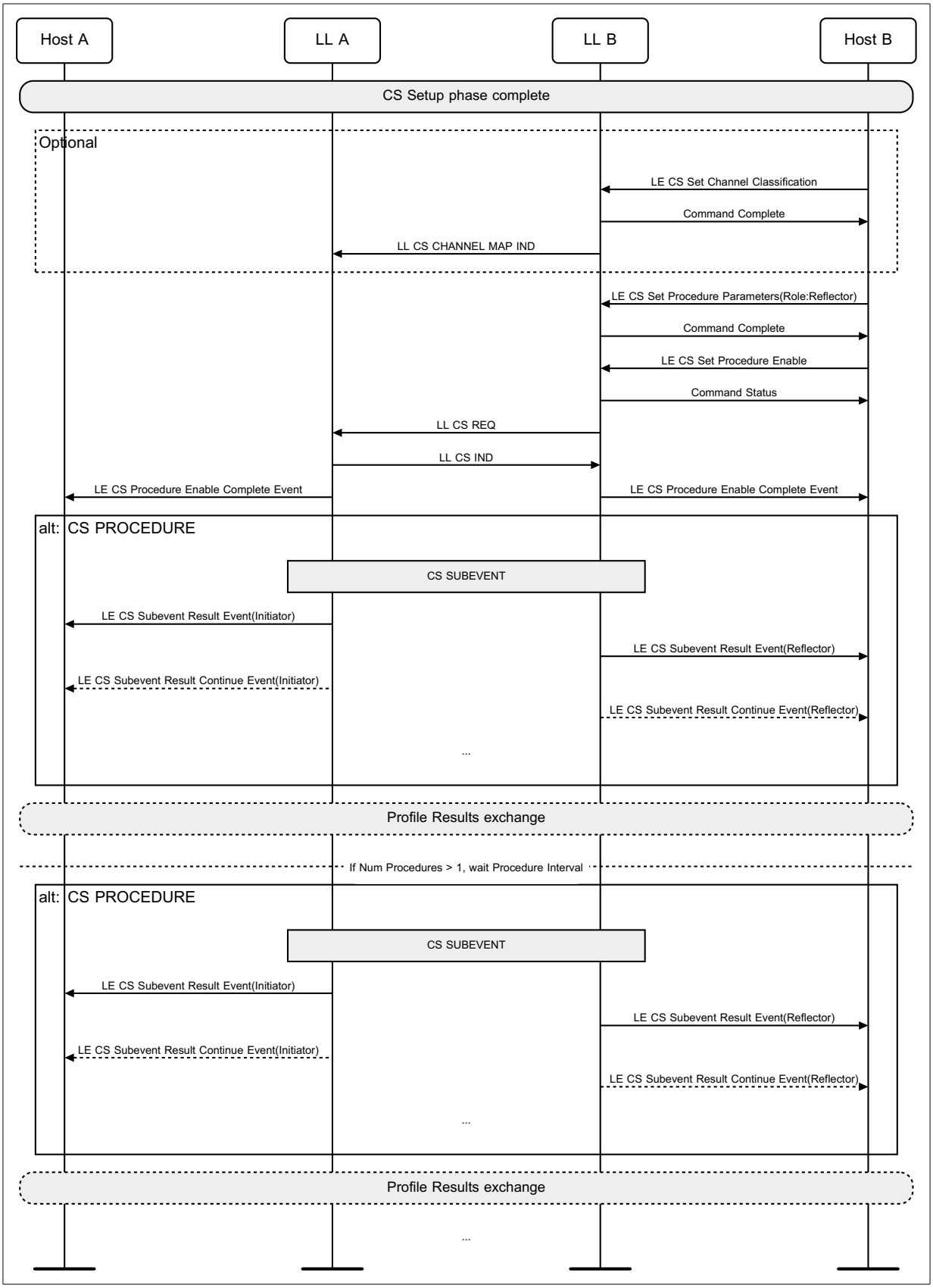



*Figure 6.77: Channel Sounding started by Peripheral in reflector role*

## 6.37 Channel Sounding started by Central, rejected by Peripheral

In the example shown in Figure 6.78, an application running on a Central device in the initiator role starts a CS procedure. The Peripheral rejects the LL_CS_REQ because the requested parameters are not acceptable.

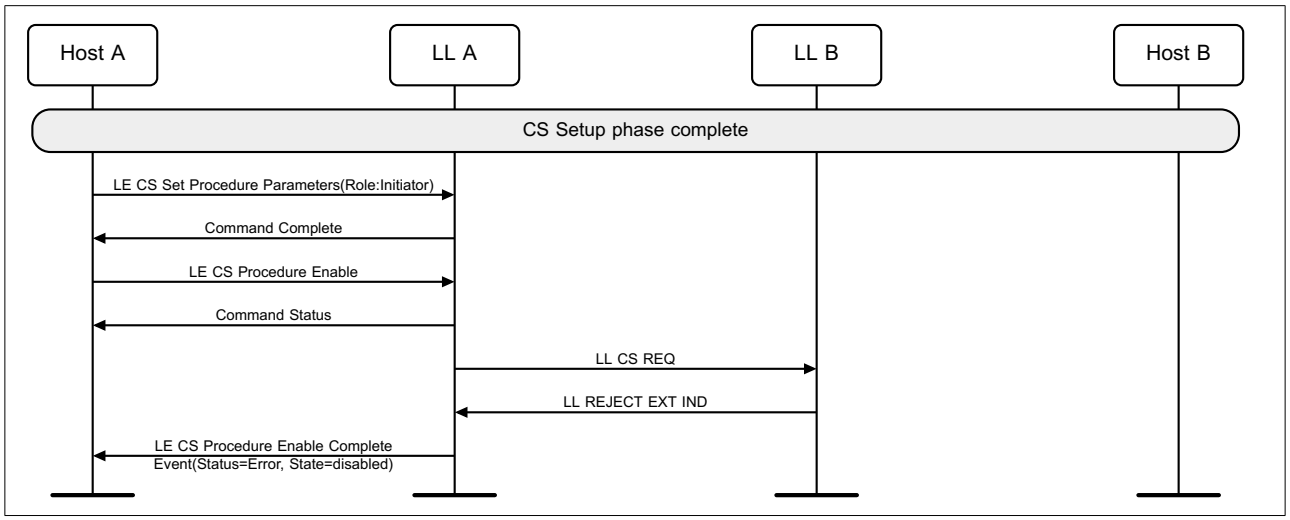



*Figure 6.78: Channel Sounding started by Central, rejected by Peripheral*

## 6.38 Channel Sounding configuration removal during an active CS procedure

In the example shown in Figure 6.79, an application running on a Central device in the initiator role starts a CS procedure. To remove a CS configuration that is being used

in an active CS procedure, the Host first terminates the active CS procedure and then issues a LE CS Remove Config command.

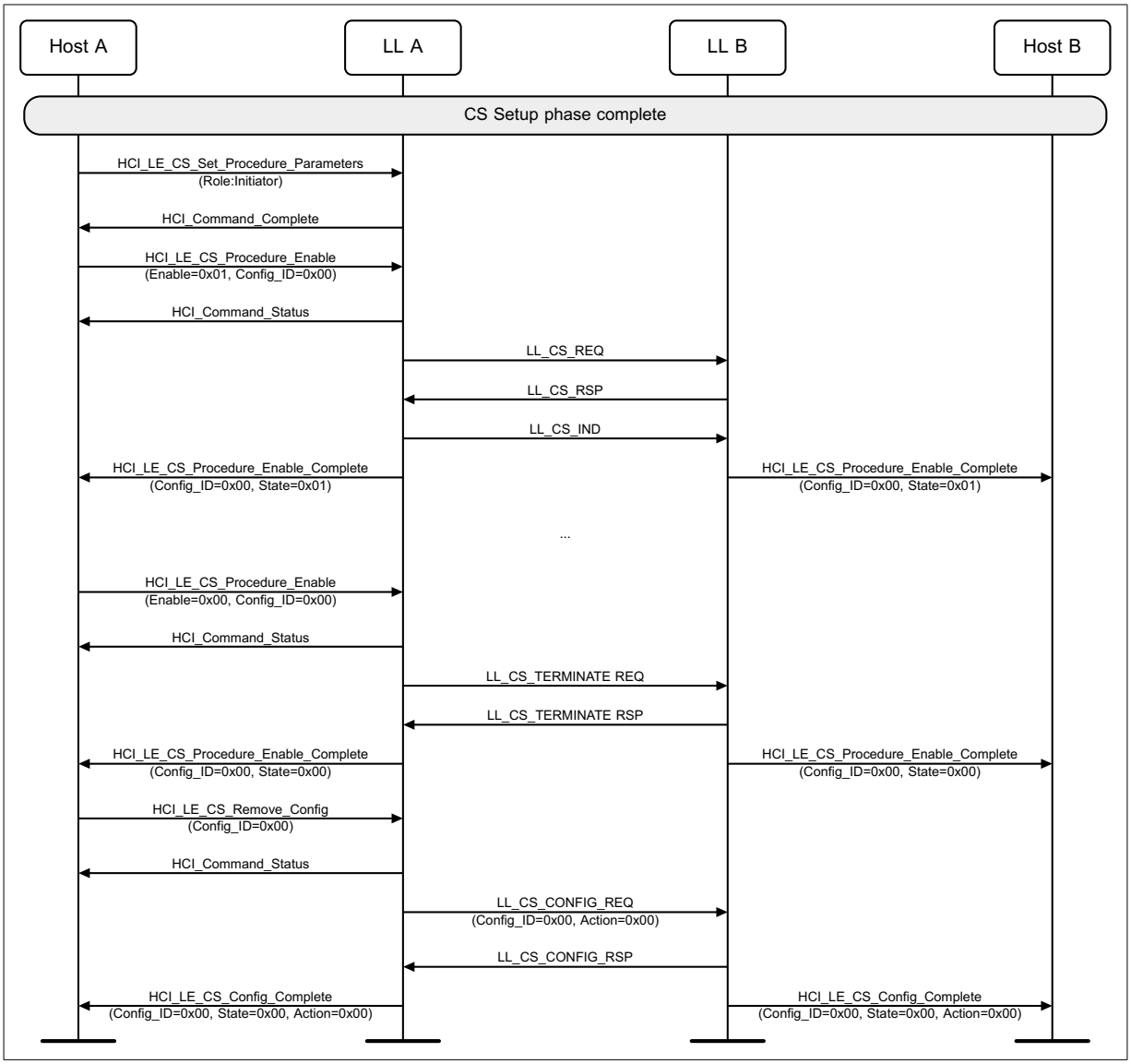



*Figure 6.79: Channel Sounding configuration removal during an active CS procedure*

## 6.39 Frame Space Update

Either the Host or the Controller of device A requests a change to the frame space values of the connection, specifying a change to T_IFS_CIS, T_MSS_CIS, or a PHY that is not the PHY used on the ACL.

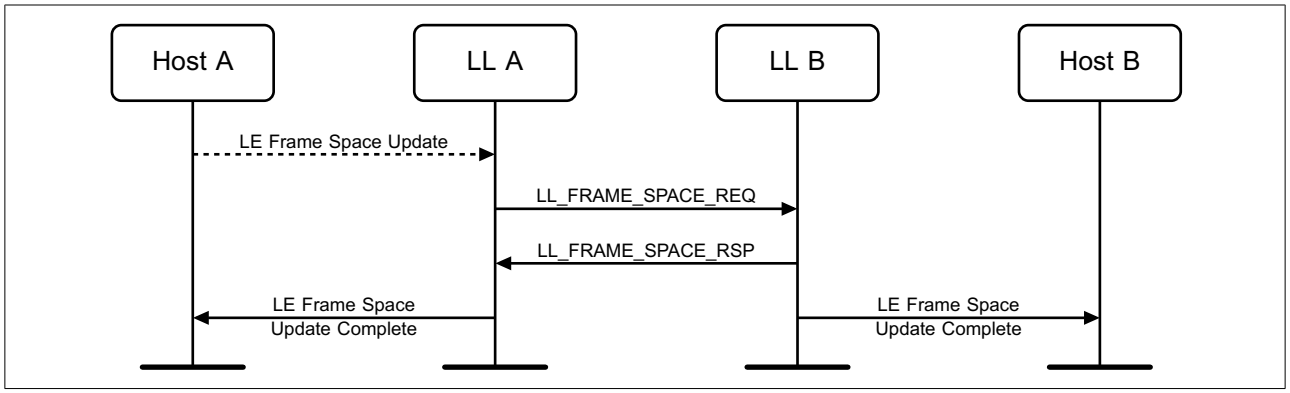



*Figure 6.80: Frame Space Update affecting CIS only*

Either the Host or the Controller of device A requests a change to the frame space value of the connection, specifying a change to T_IFS_CIS and a change to either T_IFS_ACL_CP, T_IFS_ACL_PC, or T_MCES. Device B responds with only a change to a PHY that is not the PHY used on the ACL.

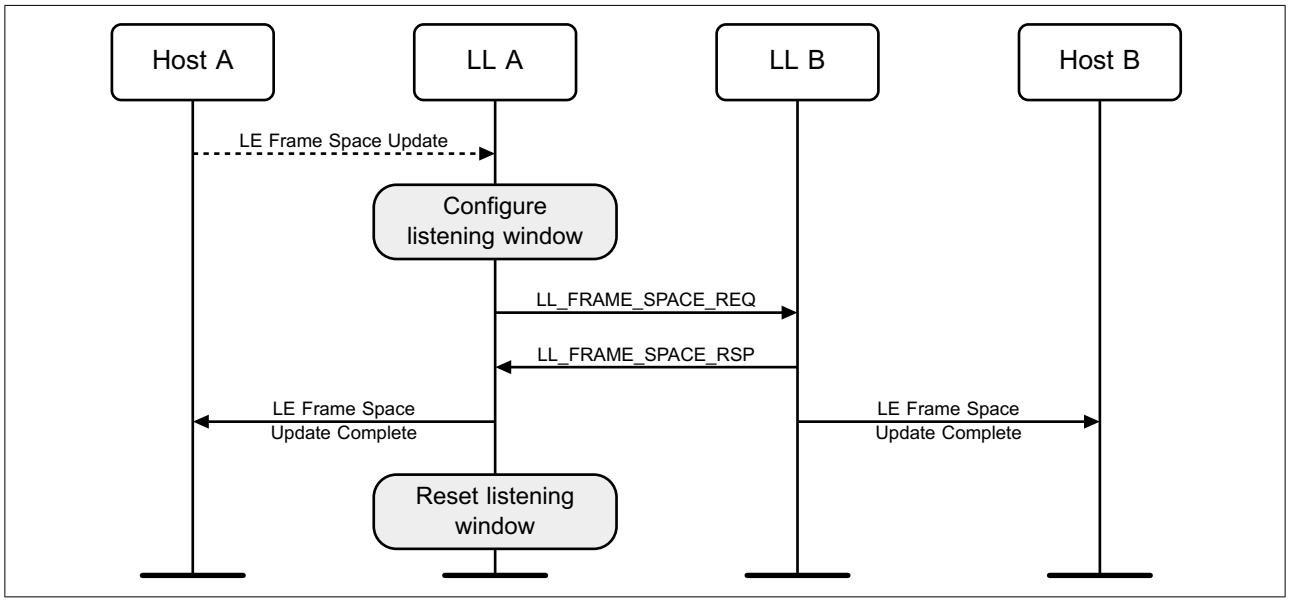



*Figure 6.81: Frame Space Update procedure where the responding device does not update the frame space for the PHYs used for the connection*

Either the Host or the Controller of device A requests a change to the frame space value of the connection. Device B responds that it can modify T_IFS_ACL_CP on the PHY used on the ACL.

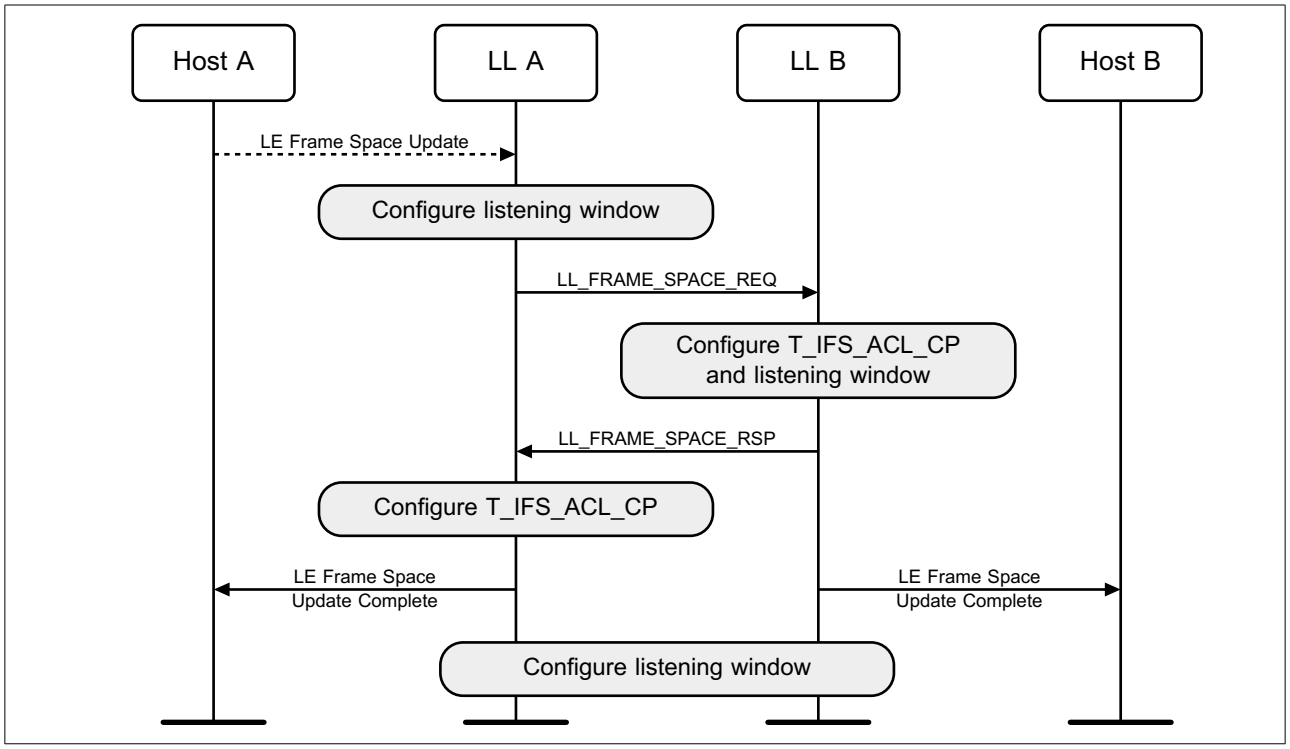



*Figure 6.82: Frame Space Update where the frame space for the connection is affected*

## 6.40 Modifying the Connection Rate

Central A in a connection with Peripheral B modifies the connection rate (see Figure 6.83).

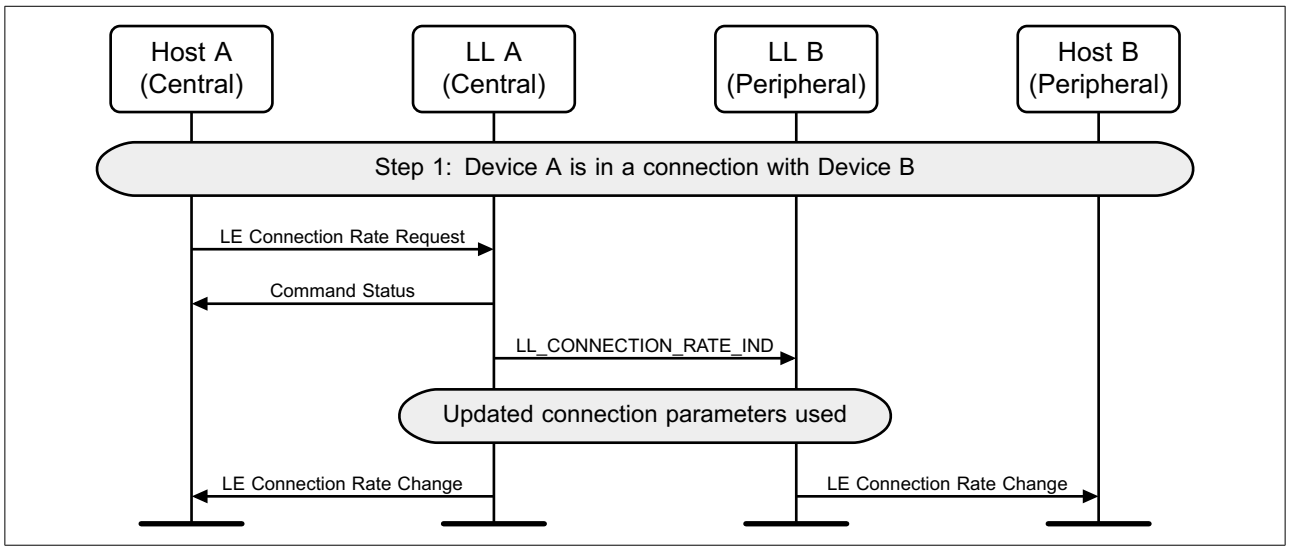



*Figure 6.83: Central A modifies the connection rate*

Central A in a connection with Peripheral B attempts to modify the connection rate. Peripheral B rejects the procedure due to invalid parameters, or the instant in the past (see Figure 6.84).

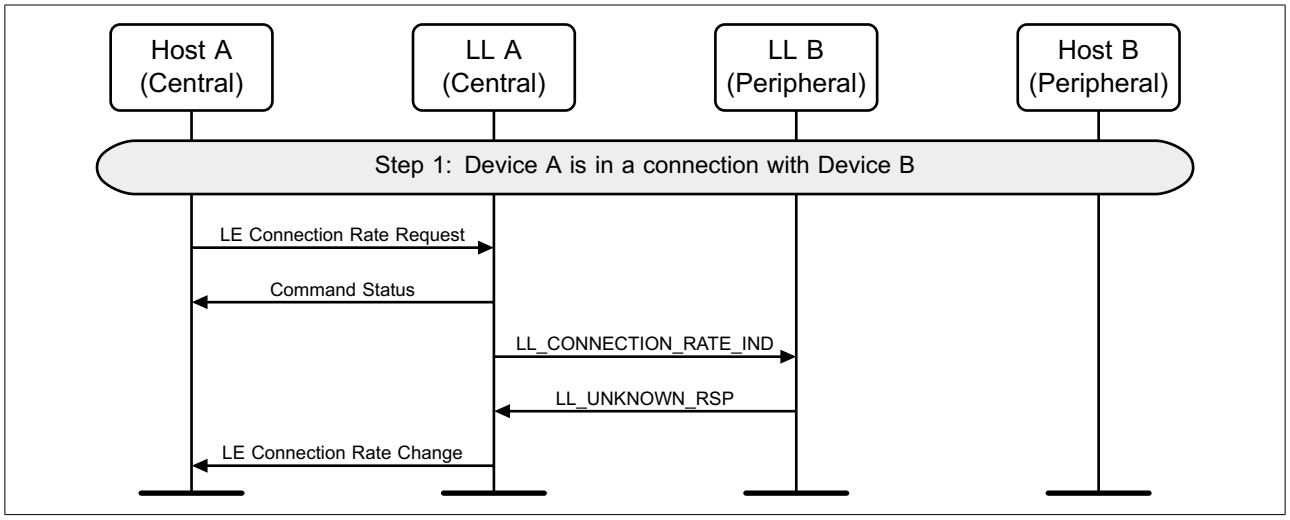



*Figure 6.84: Central A attempts to modify the connection rate, Peripheral B rejects*

Peripheral B requests a new rate for its connection with Central A. Central A accepts the request (see Figure 6.85).

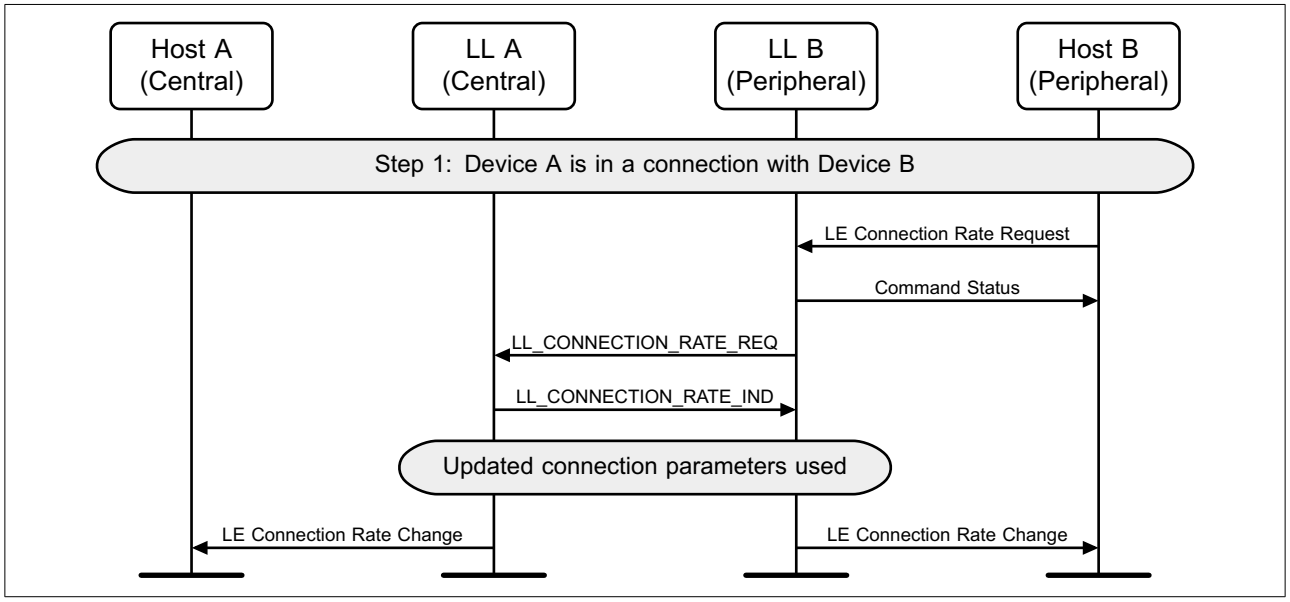



*Figure 6.85: Peripheral B requests a change to the connection rate, Central A accepts*

Peripheral B requests a new rate for its connection with Central A. Central A rejects the change (see Figure 6.86).

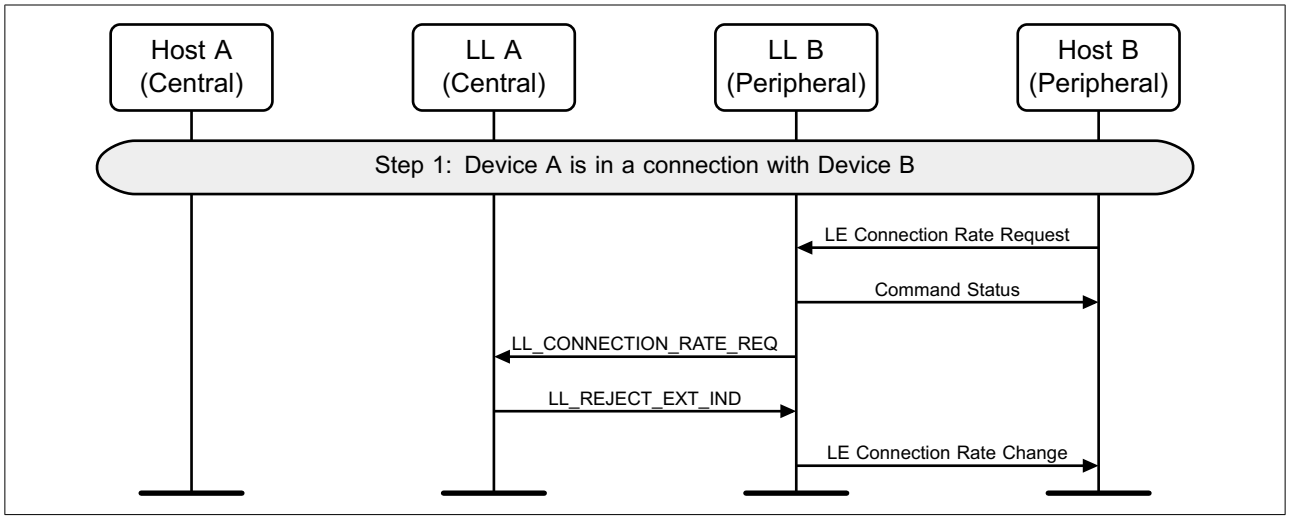



*Figure 6.86: Peripheral B requests a change to the connection rate, which is rejected by Central A*

# 7 PERIODIC ADVERTISING SYNC TRANSFER

In the following examples, device B is carrying out periodic advertising. Either device A or device B is in a connection with device C and transfers periodic advertising synchronization information about the periodic advertising train to device C.

## 7.1 Transfer by scanner, reports initially disabled

The scanning device (A) transfers periodic advertising synchronization information to device C, which starts listening to the periodic advertising train but only sends reports to the Host when explicitly requested.

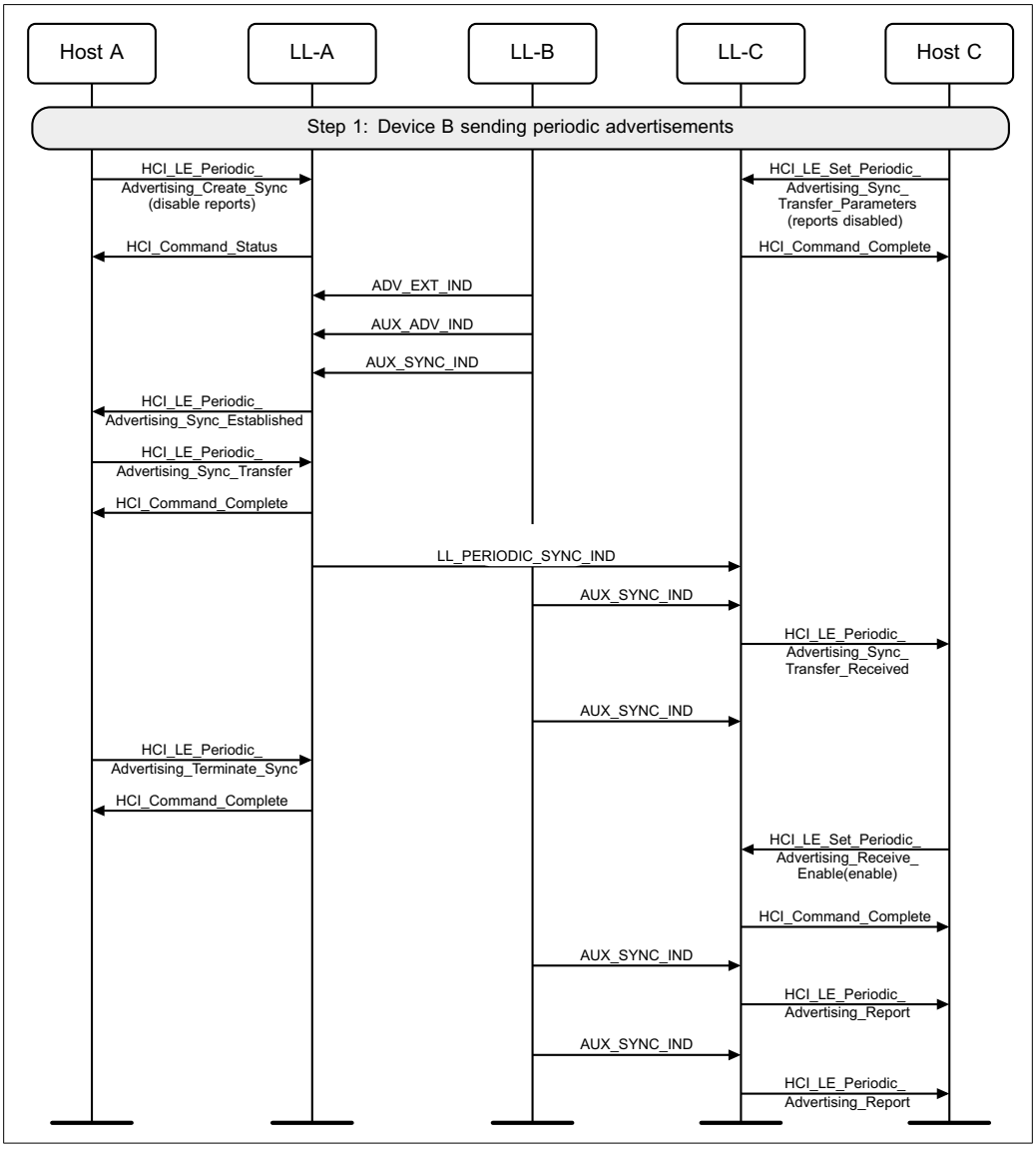



*Figure 7.1: Periodic Advertising Sync Transfer by scanner, reports initially disabled*

## 7.2 Transfer by scanner, reports initially enabled

The scanning device (A) transfers periodic advertising synchronization information to device C, which starts listening to the periodic advertising train and immediately sends reports to the Host.

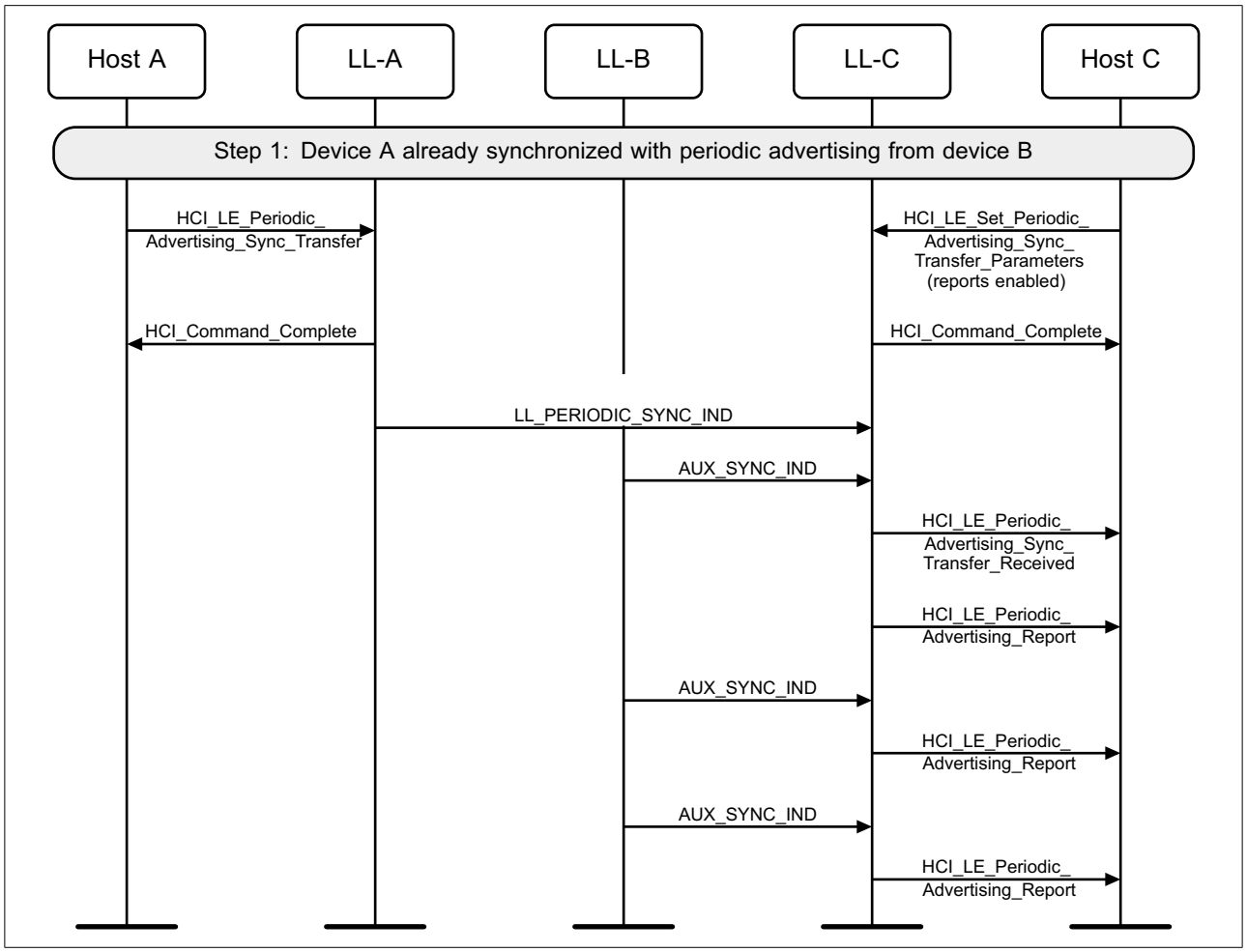



*Figure 7.2: Periodic Advertising Sync Transfer by scanner, reports initially enabled*

## 7.3 Transfer by the advertiser

The advertiser (B) directly transfers periodic advertising synchronization information about its periodic advertising to device C. In this example reports to Host C are disabled.

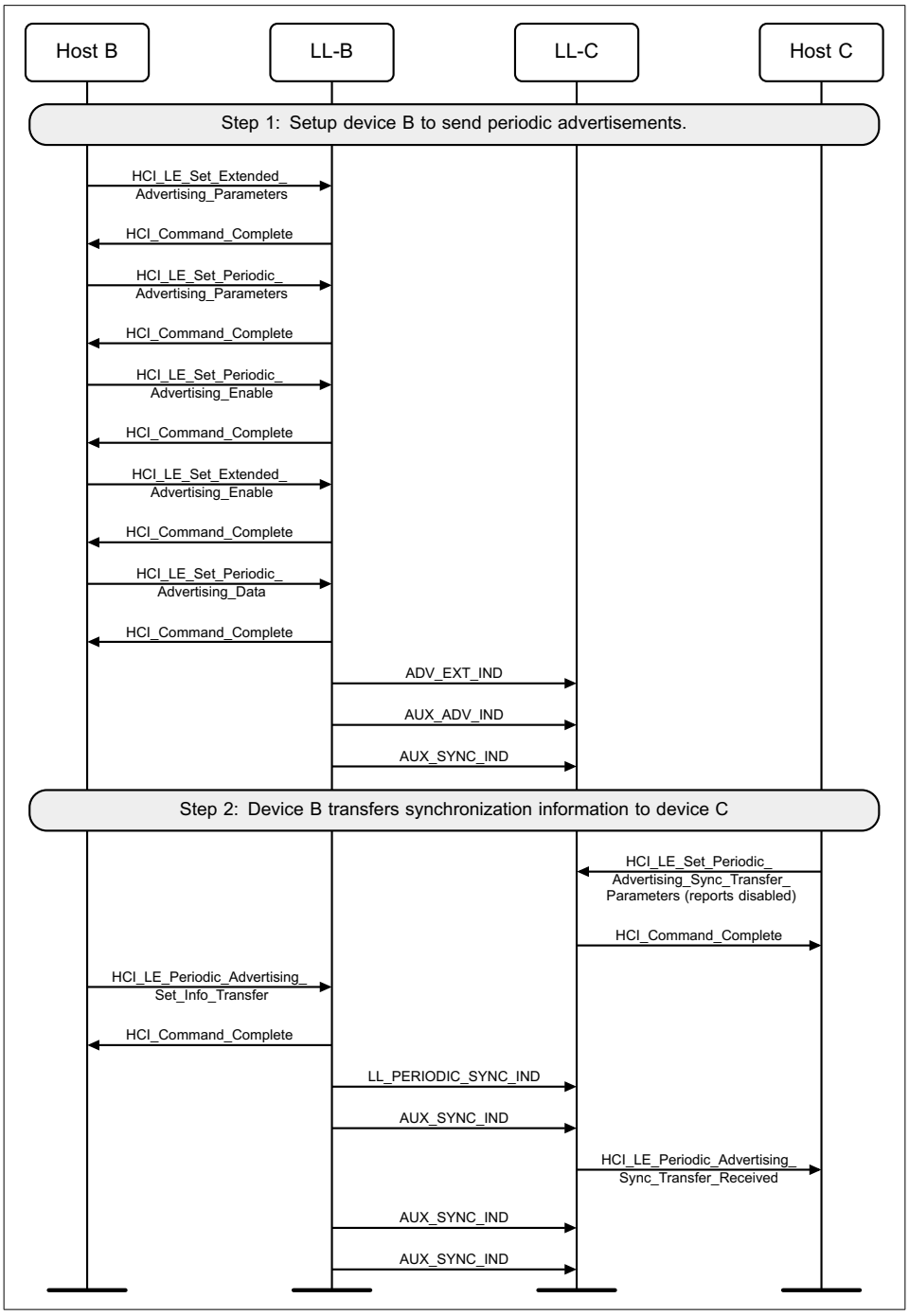



*Figure 7.3: Periodic Advertising Sync Transfer by advertiser, reports initially disabled*

# 8 SYNCHRONIZATION STATE

## 8.1 Synchronizing with a Broadcast Isochronous Group

A device enters the Synchronization state, receives the synchronization information from the periodic advertising train that is associated with the BIG, and synchronizes to a BIS (see Figure 8.1).

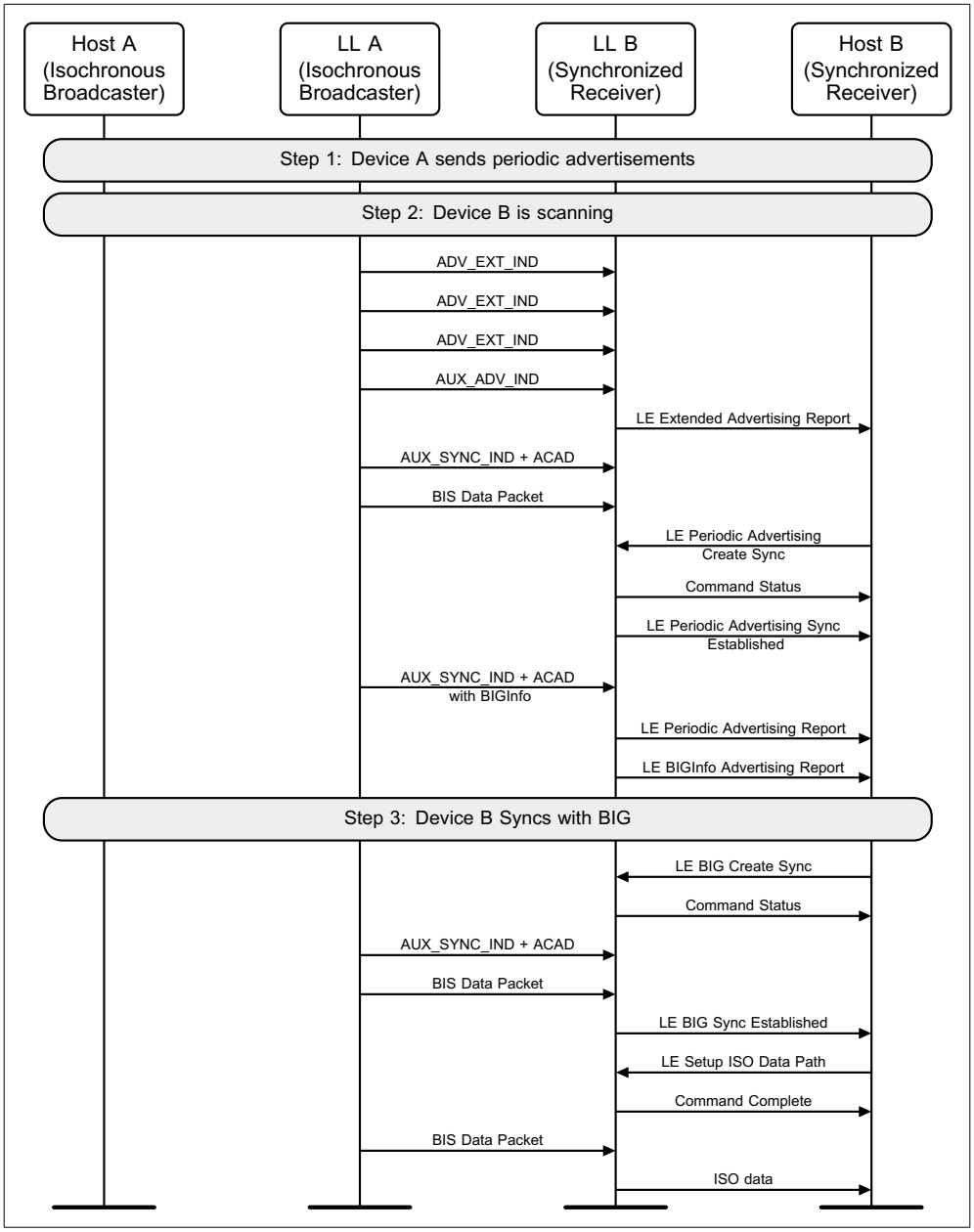



*Figure 8.1: Device B synchronizes with a BIG*

## 8.2 Terminate Synchronization with a BIG

A device terminates synchronization with a BIG (see Figure 8.2).

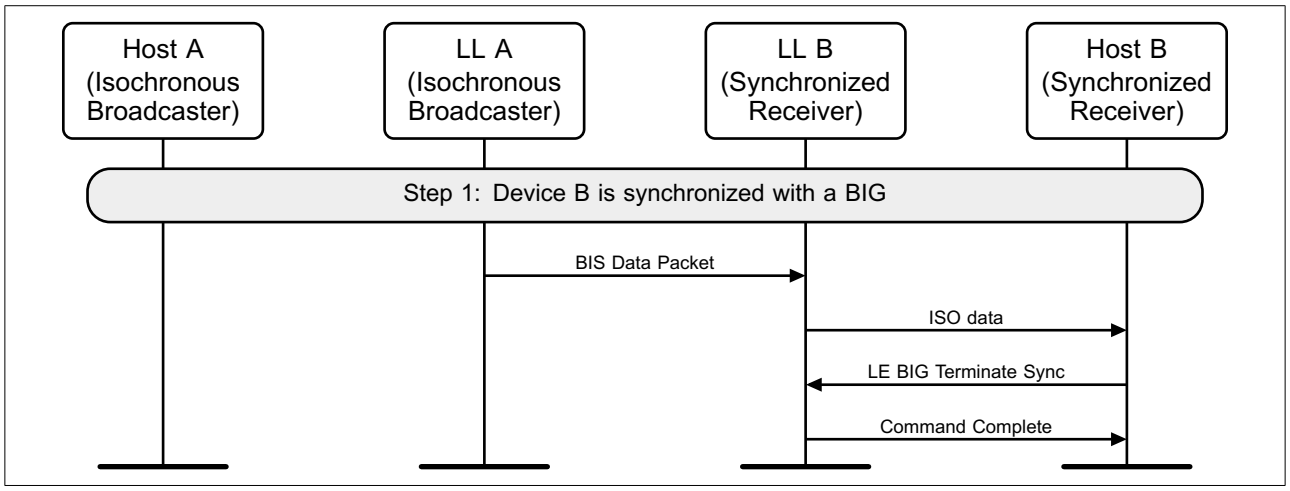



*Figure 8.2: Synchronization with a BIG is terminated*

## 8.3 New Channel Map for Broadcast Isochronous Group

A device receives a channel map update for a BIG (see Figure 8.3).

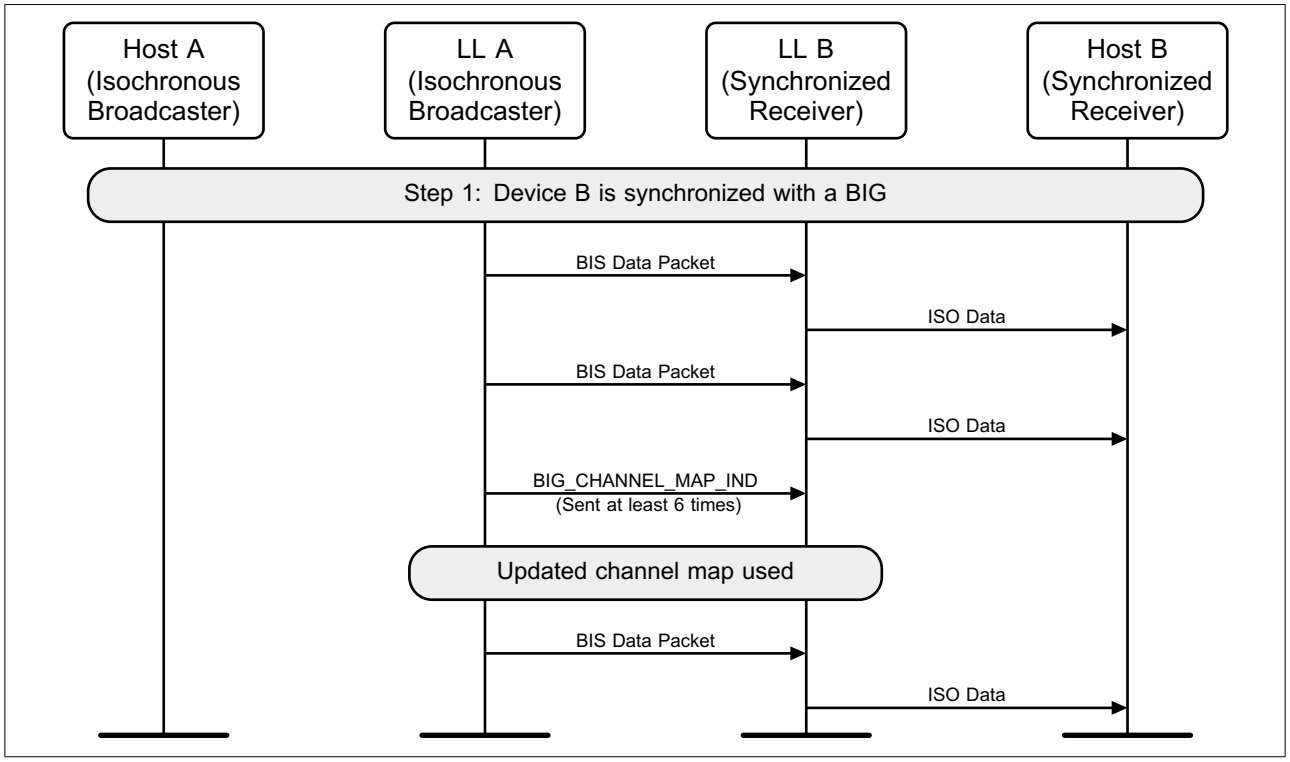



*Figure 8.3: Device A sends a channel map update for a BIG*

## 8.4 Lost Synchronization with a Broadcast Isochronous Group

A device loses synchronization with a BIG (see Figure 8.4).

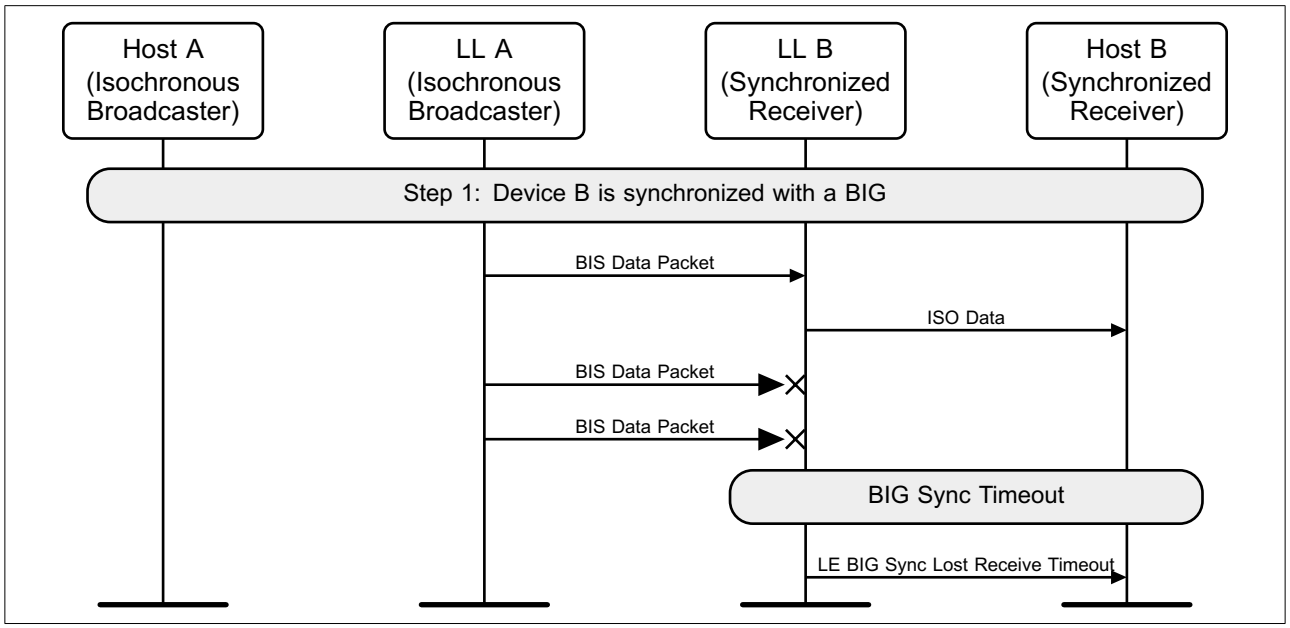



*Figure 8.4: Device B loses synchronization with a BIG*

## 8.5 Data path setup for a BIS

Figure 8.5 shows an example of an isochronous broadcast stream from the Broadcaster A. Host A sets up the ISO data path with codec processing in the Controller to create the PDUs transmitted over the BIS. Device B becomes a Synchronized Receiver for

the BIS and sets up the ISO data path to do codec processing in the Controller on the received data PDUs.

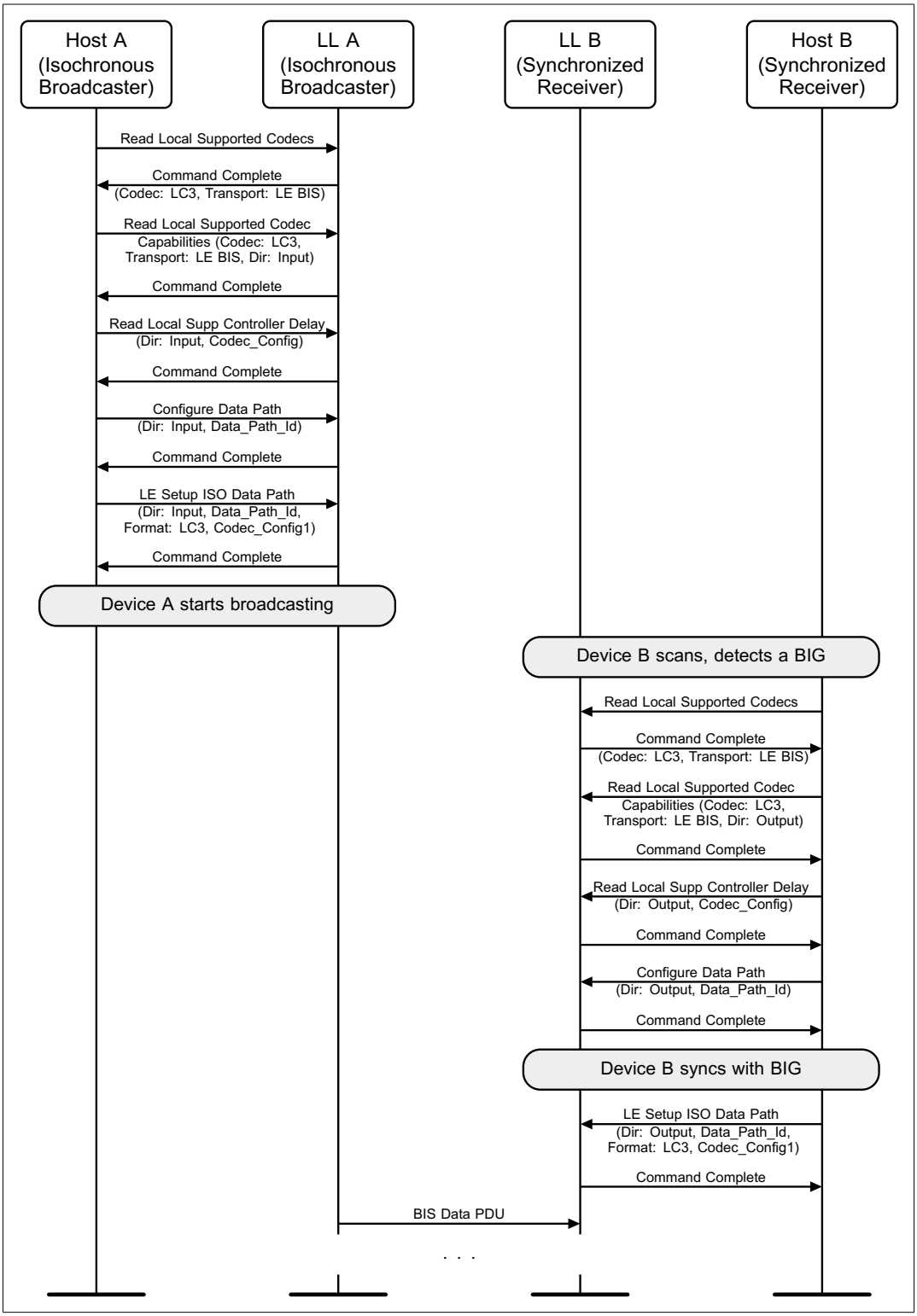



*Figure 8.5: Isochronous broadcasting between devices which both have a codec in the Controller*

Low Energy Controller
Part E

# LOW ENERGY LINK LAYER SECURITY

*This Part describes the Link Layer security for Bluetooth Low Energy.*

# CONTENTS

**1 Encryption and authentication overview ..... 3711**
1.1 Cryptographic Toolbox ..... 3712
1.1.1 Group Session Key Derivation Function h8 ..... 3712
1.1.2 Derivation of Group Session Key ..... 3713

**2 CCM ..... 3714**
2.1 CCM nonce ..... 3714
2.2 Counter mode blocks ..... 3716
2.3 Encryption blocks ..... 3717
2.4 Session Keys ..... 3717

**3 Deterministic Random Bit Generator ..... 3719**
3.1 Cryptographic toolbox ..... 3720
3.1.1 Octet and bit ordering ..... 3721
3.1.2 DRBG chain function f7 ..... 3722
3.1.3 DRBG derivation function f8 ..... 3723
3.1.4 DRBG update function f9 ..... 3725
3.1.5 DRBG instantiation function h9 ..... 3727
3.1.6 Random bit generation function CS_DRBG ..... 3728
3.1.7 DRBG backtracking resistance ..... 3728
3.2 DRBG based on a block cipher ..... 3729
3.2.1 DRBG nonce V ..... 3729
3.2.2 DRBG nonce increment function ..... 3730

**4 References ..... 3733**

# 1 ENCRYPTION AND AUTHENTICATION OVERVIEW

The Link Layer provides encryption and authentication using Counter with Cipher Block Chaining-Message Authentication Code (CCM) Mode, which shall be implemented consistent with the algorithm as defined in IETF RFC 3610 (http://www.ietf.org/rfc/rfc3610.txt) in conjunction with the AES-128 block cipher as defined in NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf). A description of the CCM algorithm can also be found in the NIST Special Publication 800-38C (http://csrc.nist.gov/publications/PubsSPs.html).

This specification uses the same notation and terminology as the IETF RFC except for the block cipher key K that in this specification is called the session key (see Section 2.4) and for the Message Authentication Code (MAC) that in this specification is called the Message Integrity Check (MIC) to avoid confusion with the term Media Access Controller.

CCM has two size parameters, M and L. The Link Layer defines these to be:

- M = 4; indicating that the MIC (authentication field) is 4 octets
- L = 2; indicating that the Length field is 2 octets

CCM requires a new session key whenever encryption is started. CCM also requires a unique nonce value for each PDU that is protected by a given session key. The CCM nonce shall be 13 octets.

An ACL connection or a BIS (if present), may be either encrypted and authenticated or unencrypted and unauthenticated. A CIS shall be encrypted and authenticated if, and only if, the associated ACL connection is.

When encryption is enabled, all Data Physical Channel and Isochronous Physical Channel PDUs with a non-zero length Payload shall be encrypted and authenticated. Authentication is performed by appending a MIC field to the Payload. The MIC shall be calculated over the PDU's Payload and the first octet of the header with certain bits (specified in Table 2.3) masked to zero. Encryption shall be applied to the PDU's Payload and MIC.

Each new encrypted PDU with a non-zero length Payload shall be decrypted and authenticated before being sent to the Host or processed by the Link Layer. Authentication is unrelated to the Link Layer acknowledgment scheme; authentication does not have to be performed before the packet is acknowledged by the Link Layer.

In the event of an authentication failure being detected in a Data Physical Channel PDU or CIS Data PDU, the connection or stream carrying the affected PDU shall be

considered lost (see [Vol 6] Part B, Section 4.5.12). The notification to the Host shall specify the error code *Connection Terminated due to MIC Failure* (0x3D). The peer Link Layer will detect this loss of connection through the supervision timeout procedure.

In the event of a MIC failure being detected in a Broadcast Isochronous PDU, synchronization shall be considered lost. The Link Layer shall not receive any further packets on the BIS and the Host shall be notified of the loss of synchronization with the error code *Connection Terminated due to MIC Failure* (0x3D).

## 1.1 Cryptographic Toolbox

The following cryptographic functions are defined:

- h8 is used to generate group session keys for BIGs.

The building block for the cryptographic functions h8 is the security function AES-CMAC.

When a multi-octet integer parameter is used as input to AES-CMAC inside the h8 function, the most significant octet of the integer shall be the first octet of the stream and the least significant octet of the integer shall be the last octet of the stream.

The functions h6 and h7 used in this Part are defined in [Vol 3] Part H, Section 2.2.

### 1.1.1 Group Session Key Derivation Function h8

The function h8 is used to generate the Group Session Key (GSK) for encrypting or decrypting payloads of an encrypted BIS. The definition of the h8 function makes use of the AES-CMAC function. The inputs to the function h8 are:

K is 128 bits
S is 128 bits
keyID is 32 bits

For the first AES-CMAC function, K is used as the data m and S is used as the key. The output of the first AES-CMAC function IK (intermediate key which is 128 bits) is used as the key for the second AES-CMAC function and keyID is used as the data m:

$IK = AES\text{-}CMAC_S(K)$
$h8(K, S, keyID) = AES\text{-}CMAC_{IK}(keyID)$

### 1.1.2 Derivation of Group Session Key

The Link Layer shall derive the Group Long Term Key (GLTK) and Group Session Key (GSK) as follows:

IGLTK = h7(“BIG1”, Broadcast_Code)
GLTK = h6(IGLTK, “BIG2”)
GSK = h8 (GLTK, GSKD, “BIG3”)

The string "BIG1" is mapped into a SALT using ASCII as 0x00000000_00000000_00000000_42494731.

The string "BIG2" is mapped into a keyID using ASCII as 0x42494732.

The string "BIG3" is mapped into a keyID using ASCII as 0x42494733.

# 2 CCM

This section provides the details for using the CCM algorithm. As specified, the CCM algorithm requires the payload and some additional parameters to be formatted into the CCM nonce, counter-mode blocks and encryption blocks. The CCM nonce provides uniqueness to each packet. The counter-mode blocks are used to calculate the MIC. The encryption blocks provide the keystream that is used to encrypt the payload and the MIC of the Data Physical Channel PDU, CIS Data PDU, BIS Data PDU, and BIG Control PDU.

Sample data of the blocks (see Section 2.2 and Section 2.3) can be found in [Vol 6] Part C, Section 1 and Section 1.2.

## 2.1 CCM nonce

The CCM nonce is constructed from a 39-bit *packetCounter*, 1-bit *directionBit* and an 8-octet IV (initialization vector). The format of the 13-octet nonce shall be as shown in Table 2.1.

| Octet | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Nonce0 | 1 | variable | *packetCounter*[7:0] |
| 1 | Nonce1 | 1 | variable | *packetCounter*[15:8] |
| 2 | Nonce2 | 1 | variable | *packetCounter*[23:16] |
| 3 | Nonce3 | 1 | variable | *packetCounter*[31:24] |
| 4 | Nonce4 | 1 | variable | Bit 6 – Bit 0: *packetCounter*[38:32]<br>Bit 7: *directionBit* |
| 5 | Nonce5 | 1 | variable | IV[7:0] |
| 6 | Nonce6 | 1 | variable | IV[15:8] |
| 7 | Nonce7 | 1 | variable | IV[23:16] |
| 8 | Nonce8 | 1 | variable | IV[31:24] |
| 9 | Nonce9 | 1 | variable | IV[39:32] |
| 10 | Nonce10 | 1 | variable | IV[47:40] |
| 11 | Nonce11 | 1 | variable | IV[55:48] |
| 12 | Nonce12 | 1 | variable | IV[63:56] |

*Table 2.1: CCM nonce format*

The Link Layer shall maintain one *packetCounter* per Role for each ACL and CIS connection and one for each BIS that the Link Layer is transmitting or is synchronized to.

For each ACL connection, the *packetCounter* shall be set to zero for the first encrypted Data Physical Channel PDU sent during the encryption start procedure. The *packetCounter* shall then be incremented by one for each new Data Physical Channel PDU that is encrypted. The *packetCounter* shall not be incremented for retransmissions.

For each CIS, *packetCounter* shall equal *cisPayloadCounter* defined in [Vol 6] Part B, Section 4.5.13.1.

For each BIS, *packetCounter* shall equal *bisPayloadCounter* defined in [Vol 6] Part B, Section 4.4.6.3

The value of *packetCounter* used for a BIG Control PDU shall be equal to the value of *packetCounter* used in the first subevent of the BIG event that the BIG Control PDU is part of.

The *directionBit* shall be set to 1 for Data Physical Channel PDUs and CIS Data PDUs sent by the Central and set to 0 for Data Physical Channel PDUs and CIS Data PDUs sent by the Peripheral. The *directionBit* shall be set to 1 for Broadcast Isochronous PDUs sent by the Isochronous Broadcaster.

The IV is common for both Roles of an ACL and CIS. Whenever encryption is started or restarted, a new 8-octet IV shall be used for each pair of communicating devices. The IV for an ACL is determined as specified in [Vol 6] Part B, Section 5.1.3.1. The IV for a CIS or BIS is calculated from an $IV_{base}$ and the Access Address of the CIS or BIS respectively. For a CIS, the $IV_{base}$ shall be set to the value of IV of the associated ACL. For a BIS, the $IV_{base}$ shall be set to the value of GIV contained in the BIGInfo. IV[31:0] shall equal $IV_{base}$[31:0] XORed with the Access Address of the CIS or BIS while IV [63:32] shall equal $IV_{base}$[63:32], as shown in Figure 2.1. The IV for a BIG control logical link shall be determined in the same way as for a BIS.

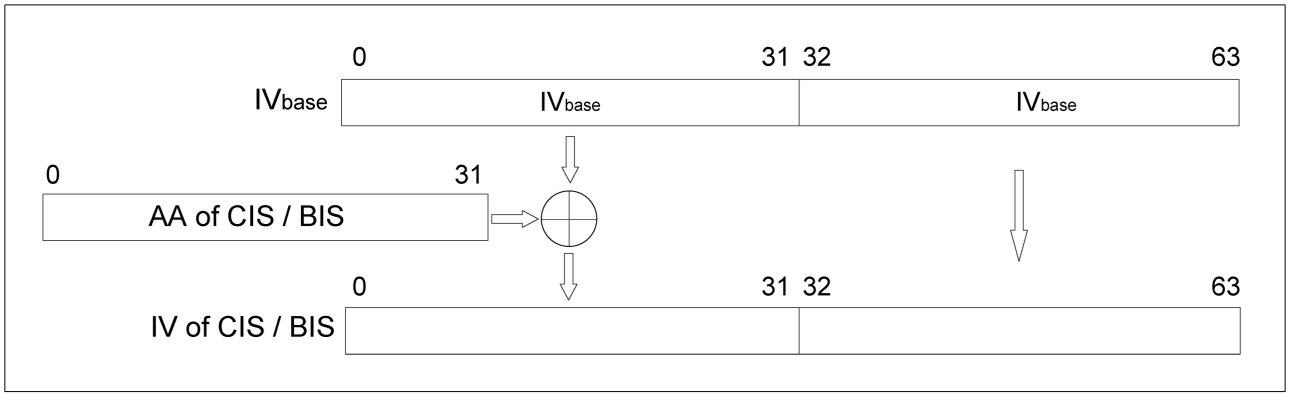



*Figure 2.1: Generation of IV for a CIS or BIS*

Note: If a CIS is terminated or considered lost and then a new CIS with the same CIS_ID is created in the same CIG, the new CIS must still have a new access address (see [Vol 6] Part B, Section 2.1.2).

## 2.2 Counter mode blocks

For calculating the MIC, the multiple counter mode blocks are generated according to the CCM specification. These are referred to as blocks $B_0$ – $B_n$. Table 2.2 defines the format of block $B_0$. Table 2.3 defines the format of block $B_1$ that is devoted to the authentication of the additional authenticated data. Additional B blocks are generated as needed for authentication of the payload.

| Offset (octets) | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Flags | 1 | 0x49 | As per the CCM specification |
| 1 | Nonce | 13 | variable | The nonce as described in Table 2.1.<br>Nonce0 shall have offset 1. Nonce12 shall have offset 13. |
| 14 | Length[MSO] | 1 | 0x00 | The most significant octet of the length of the payload |
| 15 | Length[LSO] | 1 | variable | The least significant octet of the length of the payload |

*Table 2.2: Block $B_0$ format*

| Offset | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | AAD_Length[MSO] | 1 | 0x00 | The most significant octet of the length of the additional authenticated data |
| 1 | AAD_Length[LSO] | 1 | 0x01 | The least significant octet of the length of the additional authenticated data |

| Offset | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 2 | AAD | 1 | variable | The PDU header's first octet with the following bits masked to 0.<br>Data Physical channel PDU: NESN, SN, MD.<br>Connected Isochronous PDU: NESN, SN, NPI, CIE.<br>Broadcast Isochronous PDU: CSSN, CSTF. |
| 3 | Padding | 13 | 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 | These octets are only used to pad the block. They are not part of the packet and never transmitted. |

*Table 2.3: Block $B_1$ format*

## 2.3 Encryption blocks

The CCM algorithm uses the $A_i$ blocks to generate keystream that is used to encrypt the MIC and the PDU payload. Block $A_0$ is always used to encrypt and decrypt the MIC. Block $A_1$ is always used to encrypt and decrypt the first 16 octets of the Payload. Subsequent blocks are always used to encrypt and decrypt the rest of the Payload as needed.

| Offset (octets) | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Flags | 1 | 0x01 | As per the CCM specification |
| 1 | Nonce | 13 | variable | The nonce as described above<br>Nonce0 shall have offset 1. Nonce12 shall have offset 13. |
| 14 | i[MSO] | 1 | variable | The most significant octet of the counter i |
| 15 | i[LSO] | 1 | variable | The least significant octet of the counter i |

*Table 2.4: Block $A_i$ format*

## 2.4 Session Keys

The session key for an ACL connection shall be generated as specified in [Vol 6] Part B, Section 5.1.3.1.

The session key for a CIS shall be the same as that for the associated ACL.

The session key for a BIS shall be the Group Session Key derived in Section 1.1.2.

# 3 DETERMINISTIC RANDOM BIT GENERATOR

The Link Layer provides the ability to generate random bit sequences using a Deterministic Random Bit Generator (DRBG), which is implemented consistent with the recommendations defined in NIST Special Publication 800-90Ar1 [1] with a security strength of 128 bits, in conjunction with the AES-128 block cipher. Table 3.1 shows the operational definitions for the DRBG specification.

| **Block Cipher** | **AES-128** |
|---|---|
| Supported security strength | 128 bits |
| Temporal key length | 128 bits |
| Nonce vector V length | 128 bits |
| Seed length | 256 bits |
| Entropy input length | 128 bits |
| Required minimum entropy for instantiation | 128 bits |
| Personalization string length | 128 bits |
| Maximum number of requests between reseeds | $2^{48}$ |

*Table 3.1: DRBG definitions*

The DRBG requires a temporal key $K_{DRBG}$ and a nonce vector $V_{DRBG}$. These two values shall both be instantiated locally within each peer Controller before the first invocation of the DRBG within each LE connection on which CS is being used. The inputs to this instantiation procedure are the initialization vector, instantiation nonce, and personalization vector, which are the result of the CS Security Start procedure described in [Vol 6] Part B, Section 5.1.23. The nonce vector $V_{DRBG}$ shall be incremented at each invocation of the DRBG, and its value shall not repeat for the life of the DRBG temporal key. This increment function is described in Section 3.2.2.

The values of $K_{DRBG}$ and $V_{DRBG}$ shall not be exported outside the local Controller.

Figure 3.1 provides a functional overview of the DRBG function. Entropy material (a) in the form of an initialization vector, instantiation nonce, and personalization vector used to seed the DRBG, is generated from the CS Security Start procedure described in [Vol 6] Part B, Section 5.1.23. The instantiation function h9 (b) described in Section 3.1.5 is invoked to generate the security material for the DRBG. A CS step counter, transaction ID, and transaction counter are used to increment (c) the DRBG security nonce, as described in Section 3.2.2, which in turn is used as an input to the DRBG (d) to generate the required random bits (e) as described in Section 3.1.6. Finally, backtracking resistance (f) is invoked every time a new CS procedure is started, as described in Section 3.1.7.

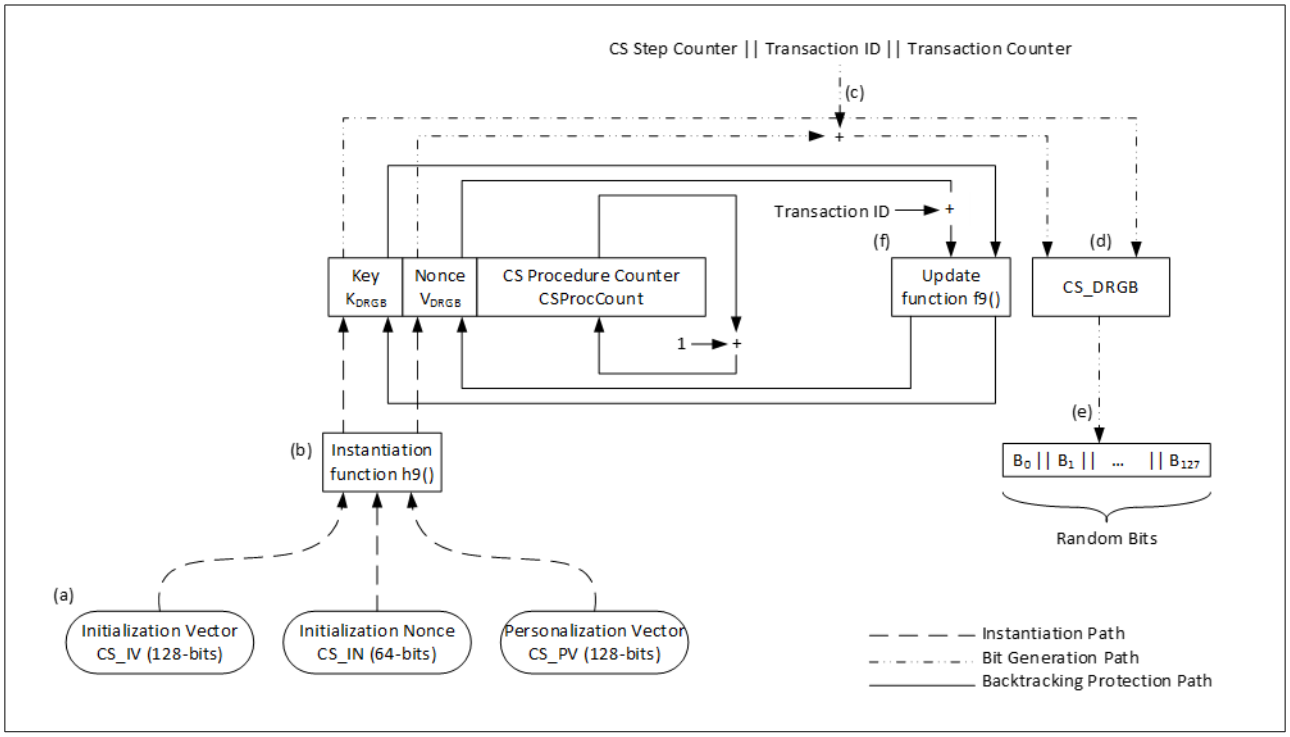



*Figure 3.1: DRBG functional overview*

## 3.1 Cryptographic toolbox

Section 3.1 defines several cryptographic functions for the DRBG. The instantiation function h9 and random bit generation function CS_DRBG are the primary cryptographic procedures, which in turn invoke subordinate functions. Figure 3.2 shows the call tree for the cryptographic toolbox described in this section.

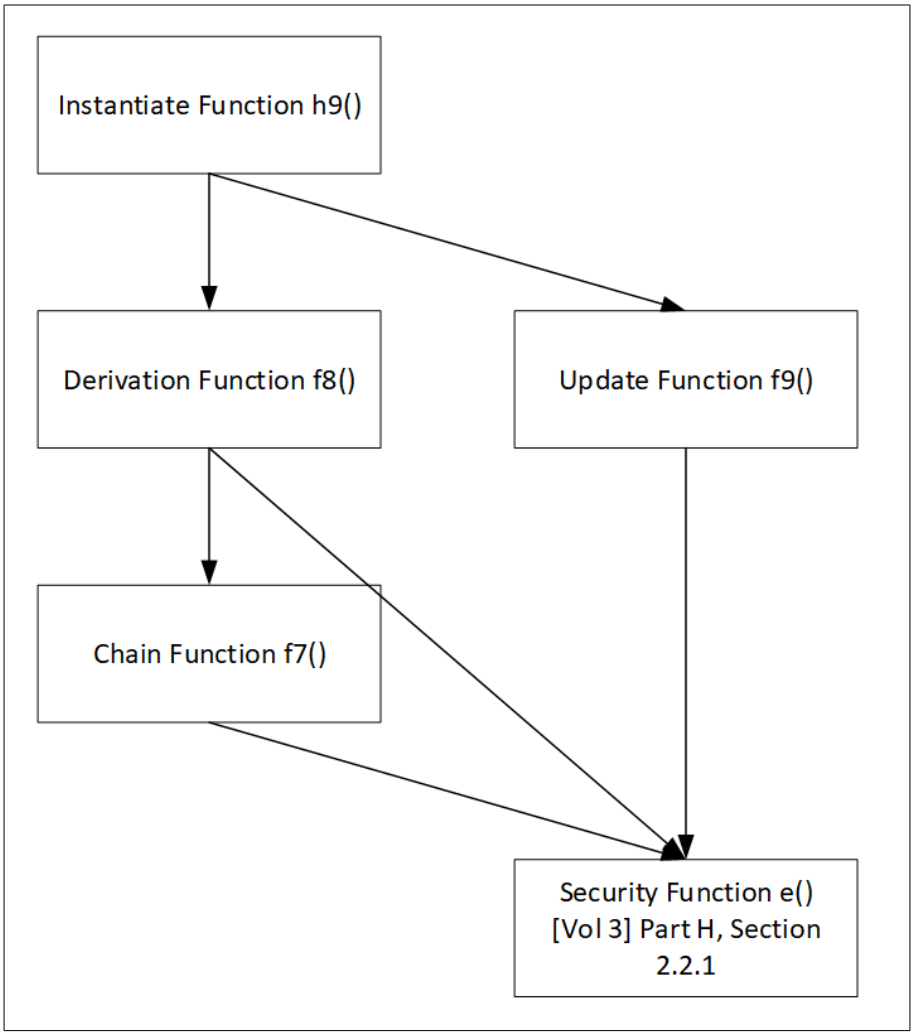



*Figure 3.2: CS cryptographic call tree*

Within these cryptographic procedures, the following operations are used.

| Operation | Meaning |
|---|---|
| len( X ) | Computes the 8-bit byte length of X in octets |
| leftmost( X ) | Returns the leftmost 128 bits of X |
| rightmost( X ) | Returns the rightmost 128 bits of X |

*Table 3.2: Operators used by the cryptographic toolbox*

### 3.1.1 Octet and bit ordering

With the exception of the random bit generation function CS_DRBG (see Section 3.1.6), all toolbox functions operate on one or more input arguments represented by a series of octets, and all toolbox functions output results that are a series of octets. With the exception of $V_{DRBG}$, the leftmost to rightmost representation is indexed starting from

octet 0, which is then followed by octet 1, and so on. For $V_{DRBG}$, the leftmost to rightmost representation is indexed starting from octet 15 and ending with octet 0.

In several toolbox functions, the hexadecimal notation of the form 0xXXXXXX is used to represent a series of octet values necessary for the processing of that function. In these cases, the leftmost to rightmost representation is used. In the example below, 0x01 represents the leftmost octet and 0x06 represents the rightmost octet.

0x010203040506

In several toolbox functions, octet and bit strings may be converted to integers to perform a mathematical operation such as increment or modulus function. Let $\{b_0, b_1, \ldots b_n\}$ represent that string in its leftmost to rightmost format. The following summation is then used to convert that bit string into an integer where *i* represents the position of each bit from the original bit string and is indexed from [0...n].

$$\sum \left(2^{n-i} \times b_i\right) \quad \text{(EQ 1)}$$

The reverse operation is used to convert an integer into a string of bits, using the same leftmost to rightmost format.

### 3.1.2 DRBG chain function f7

The DRBG chain function f7 takes a 128-bit key and an input string whose length is a multiple of 128 bits and generates an output that is 128 bits long using a cipher block chaining technique. The input string is segmented into 128-bit blocks that are processed in sequence. The security function e (see [Vol 3] Part H, Section 2.2.1) is invoked on each block whose output is XORed with the next block. This process is repeated until all input blocks are processed.

The following value is returned from f7:

hout is 128 bits

The following temporary values are used:

block[ i ] is an array of 128-bit string quantities

The input/output format of h7 is as follows:

hout = f7( k, input_bit_string )

f7 processing is as follows:

hout = 0x00000000000000000000000000000000

Starting with the leftmost bits of input_bit_string, split into 128-bit blocks from block[ 1 ] to block[ n ]

for each block[ i ]

hout = hout xor block[ i ]

hout = e( k, hout );

return hout

The block diagram representing function f7() is shown in Figure 3.3.

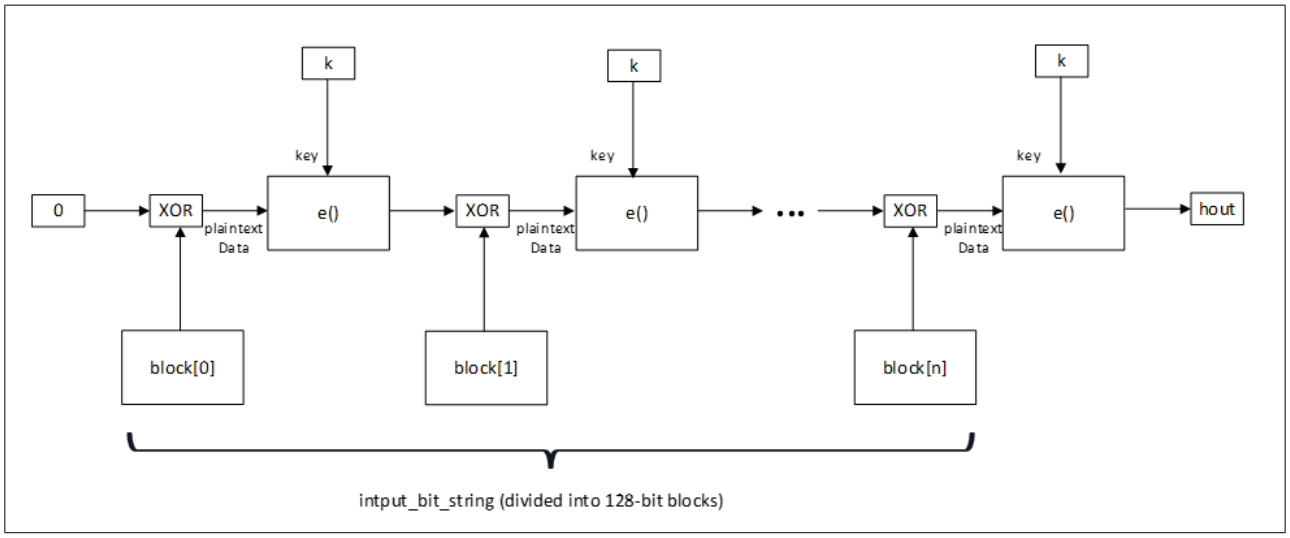



*Figure 3.3: f7() block diagram*

### 3.1.3 DRBG derivation function f8

The DRBG derivation function f8 is used to generate DRBG seed material that is 256 bits long. It takes an input bit string that is 320 bits long. Both f7 (see Section 3.1.2) and security function e (see [Vol 3] Part H, Section 2.2.1) are invoked by f8.

The following values are returned from f8:

SM is 256 bits

The following temporary values are used:

S is 512 bits

K is 128 bits

K2 is 128 bits

V is 128 bits

X is 128 bits

The input/output format of f8 is as follows:

SM = f8( input_bit_string )

The derivation function uses the input_bit_string to form a temporary bit string S constructed as shown below.

S = 0x0000002800000020 || input_bit_string || 0x80 || 0x00000000000000000000000000000000

K = 0x000102030405060708090A0B0C0D0E0F

V = 0x00000000000000000000000000000000

K2 = f7( K, V || S )

V = 0x00000001**1**00000000000000000000000

X = f7( K, V || S )

SM = e( K2, X )

SM = SM || e( K2, SM )

return SM

The block diagram representing function f8() is shown in Figure 3.4.

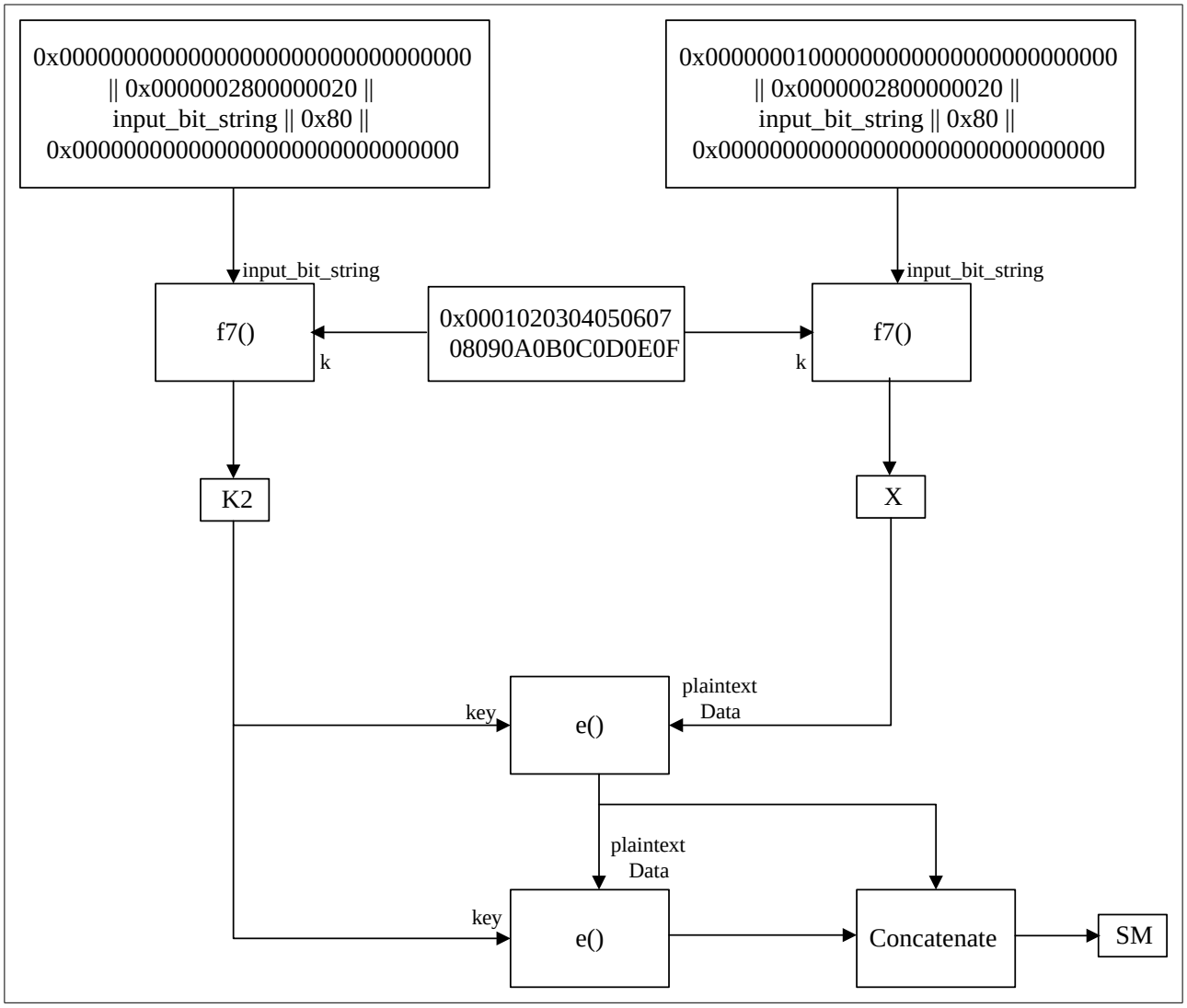



*Figure 3.4: f8() block diagram*

### 3.1.4 DRBG update function f9

The DRBG update function f9 is used to update and refresh a DRBG 128-bit temporal key K and a 128-bit nonce vector V using a 256-bit seed material (SM) that may carry fresh entropy. The SM value may also be 0 if f9 is called for backtracking purposes, as described in Section 3.1.7. The security function e (see [Vol 3] Part H, Section 2.2.1) is invoked by f9.

The following values are returned from f9:

$K_{out}$ is 128 bits

$V_{out}$ is 128 bits

The following temporary values are used:

X is 256 bits

The input/output format of f9 is as follows:

$K_{out}$, $V_{out}$ = f9( SM, K, V )

f9 processing is as follows:

V = V[127:8] || (( V[7:0] + 1 ) mod $2^8$)

X = e( K, V )

V = V[127:8] || (( V[7:0] + 1 ) mod $2^8$)

X = X || e( K, V )

X = X $\oplus$ SM

$K_{out}$ = leftmost( X )

$V_{out}$ = rightmost( X )

return $K_{out}$, $V_{out}$

The block diagram representing function f9() is shown in Figure 3.5.

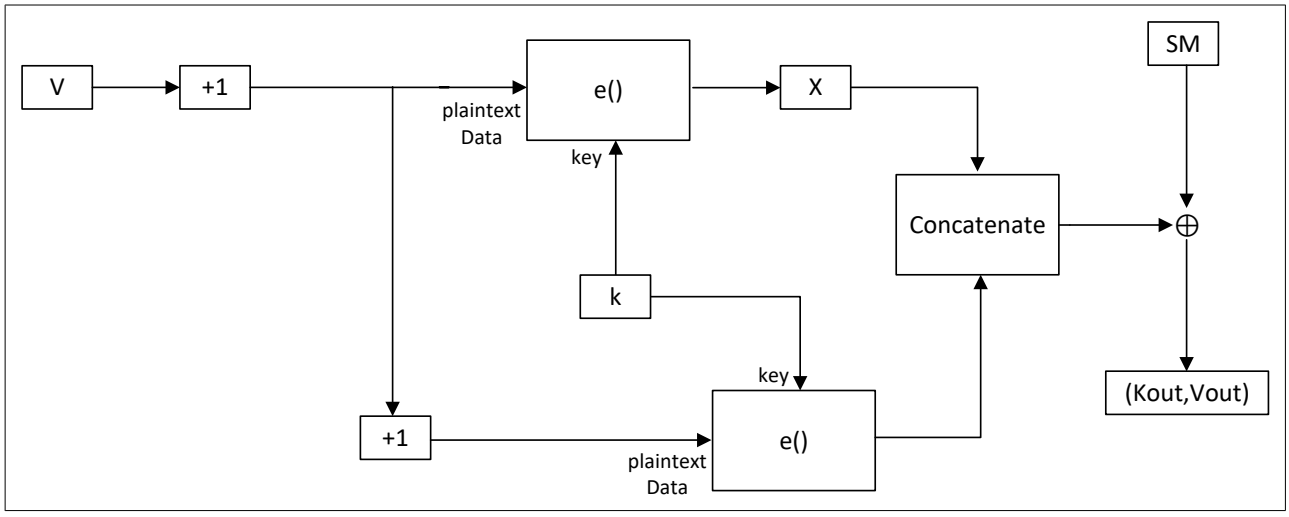



*Figure 3.5: f9() block diagram*

### 3.1.5 DRBG instantiation function h9

The DRBG instantiation function h9 is used to instantiate the DRBG temporal key $K_{DRBG}$ and a nonce vector $V_{DRBG}$. The inputs to h9 are the 128-bit CS_IV, 64-bit CS_IN, and 128-bit CS_PV values that are the result of the CS Security Start procedure described in [Vol 6] Part B, Section 5.1.23. SM below represents the derived seed material for the DRBG instantiation.

The following values are returned from h9:

$K_{DRBG}$ is 128 bits

$V_{DRBG}$ is 128 bits

The following temporary values are used:

SM is 256 bits

K is 128 bits

V is 128 bits

The input/output format of h9 is as follows:

$K_{DRBG}$, $V_{DRBG}$ = h9( CS_IV, CS_IN, CS_PV )

In the instantiation function, both a seed derivation function f8 (see Section 3.1.3) and a key and nonce vector update function f9 (see Section 3.1.4) are invoked.

SM = f8( CS_IV || CS_IN || CS_PV )

K = 0x00000000000000000000000000000000

V = 0x00000000000000000000000000000000

$K_{DRBG}$, $V_{DRBG}$ = f9( SM, K, V )

return $K_{DRBG}$, $V_{DRBG}$

The block diagram representing function h9() is shown in Figure 3.6.

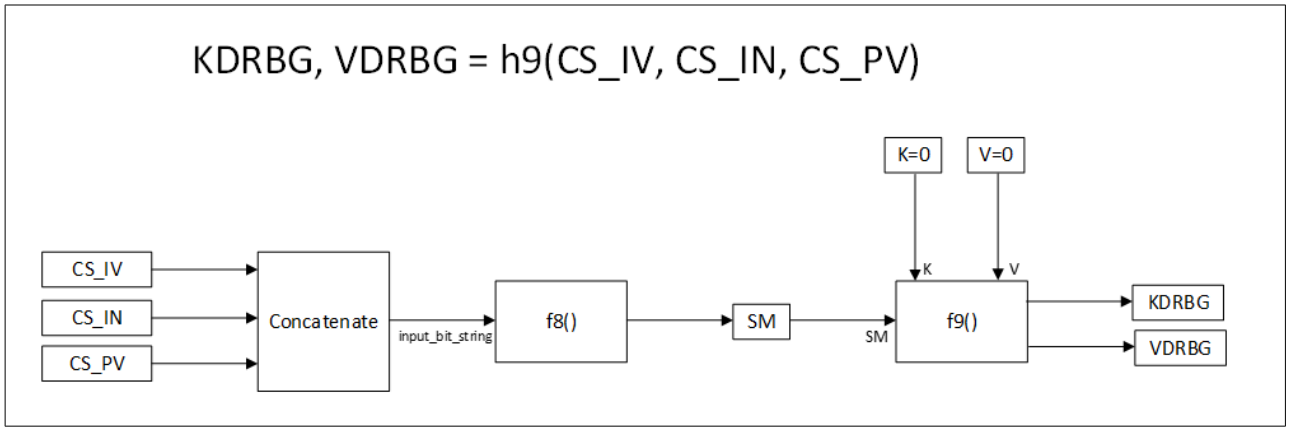



*Figure 3.6: h9() block diagram*

### 3.1.6 Random bit generation function CS_DRBG

Random bit generation function CS_DRBG makes use of the security function e (see [Vol 3] Part H, Section 2.2.1) to generate a fresh 128-bit random bit sequence. The instantiated key and nonce vector V values are described in Section 3.

randomBits[0:127] = e( $K_{DRBG}$, $V_{DRBG}$ + counters )

CS_DRBG( number of required bits ) ⊆ randomBits[0:127]

Refer to Section 3.2.1 for a description of the counters that contribute the nonce V value.

The maximum number of bits that may be generated in any single invocation is 128 bits. The number of required bits may vary from operation to operation. If less than 128 bits are required, then the return value of CS_DRBG shall be a subset of the bits returned from security function e The subset of bits shall be extracted starting from the most significant bit in leftmost to rightmost order. For example, on a new invocation of e, if an operation requires 8-bit values, then randomBits[0:7] would be the first set used, randomBits[8:15] would be the second set used, and so on. If at any point the number of required bits exceeds the number of fresh bits remaining to draw upon, then the remaining bits shall be discarded, the nonce vector $V_{DRBG}$ shall be updated (see Section 3.2.2), and the security function e shall be invoked again.

Depending on the operation, the random bits may also be used as unsigned integer values. If unsigned integer values are used, the most significant bit returned shall represent the most significant bit of the unsigned integer that is formed from that set of bits.

### 3.1.7 DRBG backtracking resistance

Backtracking resistance as described in [1] is included in DRBG. Backtracking resistance shall be invoked every time the CSProcCount is incremented (see [Vol

6] Part B, Section 5.1.26) before the first invocation of the CS_DRBG for that CS procedure (see Section 3.1.6).

The procedure for backtracking resistance makes use of the DRBG update function f9 (see Section 3.1.4) to update the $K_{DRBG}$ and $V_{DRBG}$ values for any subsequent CS procedure. The inputs to f9 are the running set of values for $K_{DRBG}$ and $V_{DRBG}$. From this value, the CS Transaction_Identifier field shall be set as described in [Vol 6] Part H, Section 4.8. The CS Step_Counter and CS Transaction_Counter fields shall be reset to zero relative to this $V_{DRBG}$ value. The procedure for applying these modifications of the $V_{DRBG}$ value is described in Section 3.2.2.

The following values are returned from f9 when used for backtracking resistance:

$K_{DRBG}$ is 128 bits

$V_{DRBG}$ is 128 bits

f9 is invoked as follows using the values for $K_{DRBG}$ and $V_{DRBG}$ as described above as inputs:

$K_{DRBG}$, $V_{DRBG}$ = f9( 0, $K_{DRBG}$, $V_{DRBG}$ )

return $K_{DRBG}$, $V_{DRBG}$

## 3.2 DRBG based on a block cipher

The Deterministic Random Bit Generator uses a block cipher as described in [1]. The block cipher employed is the security function e described in [Vol 3] Part H, Section 2.2.1. The inputs to e for the DRBG are key $K_{DRBG}$ and nonce vector $V_{DRBG}$, both of which shall be instantiated using security function h9 as described in Section 3.1.5, and afterwards updated using security function f9 as described in Section 3.1.4, before any use of the DRBG.

### 3.2.1 DRBG nonce V

The instantiation content of the nonce V vector, as described in Section 3.1.5 represents the starting value from which the DRBG increment function operates. From that starting value, several nonce counter fields are defined, as shown in Table 3.3. Each counter field octet is structured in leftmost to rightmost bit format, from bit 7 to bit 0.

| Octet | Field | Size (octets) | Value | Description |
|---|---|---|---|---|
| 0 | Counter0 | 1 | variable | 8-bit CS Transaction_Counter, with bit 7 being the most significant bit |
| 1 | Counter1 | 1 | variable | 8-bit CS Transaction_Identifier, with bit 7 being the most significant bit |
| 2 | Counter2 | 1 | variable | Octet 0 (LSO) of the CS Step_Counter, with bit 7 being the most significant bit |
| 3 | Counter3 | 1 | variable | Octet 1 (MSO) of the CS Step_Counter, with bit 7 being the most significant bit |
| 4 | Counter4 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 5 | Counter5 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 6 | Counter6 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 7 | Counter7 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 8 | Counter8 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 9 | Counter9 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 10 | Counter10 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 11 | Counter11 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 12 | Counter12 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 13 | Counter13 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 14 | Counter14 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |
| 15 | Counter15 | 1 | fixed | Bits remain set as instantiated or updated for the life of the DRBG key |

*Table 3.3: DRBG nonce/counter format*

### 3.2.2 DRBG nonce increment function

The Link Layer shall maintain one DRBG nonce/counter (V) value per ACL connection as described in Section 3.2.1.

The management of the nonce V CS Step_Counter field, CS Transaction_Identifier field, and CS Transaction_Counter field are described in this section. All other fields in the nonce V counter shall be set relative to the $V_{DRBG}$ content resulting from either the procedure described in Section 3.1.5 or the Backtracking Resistance procedure described in Section 3.1.7.

The nonce V CS Step_Counter field shall be initialized from the $V_{DRBG}$ value resulting from either the instantiation procedure described in Section 3.1.5 or the Backtracking Resistance procedure described in Section 3.1.7. This value shall be used for invocations of the DRBG that are necessary to the start of the first step of a CS procedure. Thereafter, the nonce V CS Step_Counter initialization value shall be added to the step number, *CSStepCount* at the time that the DRBG is invoked again, as described in [Vol 6] Part H, Section 4.3, modulo $2^{16}$.

$$\text{CS Step_Counter} = (V_{DRBG}[31:16] + \textit{CSStepCount}) \textit{ mod } 2^{16}$$

The CS Step_Counter value shall not exceed a value of $2^{16} - 1$ when used within the CS Step_Counterfield of the nonce V value in each CS procedure instance. The DRBG is not invoked at every CS step within a CS procedure.

The nonce V CS Transaction_Identifier field shall be set to a specific transaction identifier, *CSTransactionID* as defined in [Vol 6] Part H, Section 4.8, added to the $V_{DRBG}$ value resulting from either the instantiation procedure (see Section 3.1.5) or the Backtracking Resistance procedure (see Section 3.1.7), modulo $2^8$.

$$\text{CS Transaction_Identifier} = (V_{DRBG}[15:8] + \textit{CSTransactionID}) \textit{ mod } 2^{8}$$

The nonce V CS Transaction_Counter field is incremented in conjunction with each CS transaction identifier type defined in [Vol 6] Part H, Section 4.8. That is, for each CS transaction identifier type, it might be necessary to invoke the DRBG one or more times. The CS Transaction_Counter field shall be incremented each time the DRBG is invoked for that specific CS transaction identifier type at a specific CS step. A separate CS Transaction_Counter field value shall be maintained for each possible CS transaction identifier type because the invocation of the DRBG might be interleaved with different CS transaction identifier types at any one CS step. In the next two paragraphs, the possible CS Transaction_Counter field values are referred to collectively as the CS Transaction_Counter field.

The CS Transaction_Counter field shall be initialized to the $V_{DRBG}$ value resulting from either the instantiation procedure (see Section 3.1.5) or the Backtracking Resistance procedure (see Section 3.1.7), each time the nonce V CS Step_Counter field is set to a new value. The CS Transaction_Counter initialization value shall be added to the transaction counter, *CSTransactionCounter* as defined in [Vol 6] Part H, Section 4.8

before each invocation of the DRBG, for the specific CS Transaction_Identifier field setting.

$$\text{CS Transaction_Counter} = (V_{DRBG}[7:0] + \textit{CSTransactionCounter}) \textit{ mod } 2^8$$

The CS Transaction_Counter field shall not be incremented more than $2^8 - 1$ times for any combined single setting for the CS Transaction_Identifier and the CS Step_Counter fields.

# 4 REFERENCES

[1] NIST Special Publication 800-90A Rev. 1, "Recommendation for Random Number Generation Using Deterministic Random Bit Generators", https://csrc.nist.gov/publications/detail/sp/800-90a/rev-1/final

Low Energy Controller
Part F

# RFPHY TEST MODES

*This Part describes the Test modes available for RFPHY testing of Bluetooth Low Energy devices.*

# CONTENTS

**1 Introduction ..... 3737**

**2 Low Energy test scenarios ..... 3739**
2.1 Test sequences ..... 3739
2.2 Message sequence charts ..... 3740
2.3 Channel Sounding test commands ..... 3741
2.4 Channel Sounding message sequence charts ..... 3742

**3 UART Test Interface ..... 3748**
3.1 UART Interface characteristics ..... 3748
3.2 UART functional description ..... 3748
3.3 Commands and events ..... 3749
3.3.1 Command and event behavior ..... 3749
3.3.2 Commands ..... 3749
3.4 Events ..... 3753
3.4.1 LE_Test_Status event ..... 3753
3.4.2 LE_Packet_Report event ..... 3755
3.5 Timing - command and event ..... 3756

**4 LE Test packet definition ..... 3757**
4.1 LE Test packets format ..... 3757
4.1.1 Whitening ..... 3758
4.1.2 Preamble and synchronization word ..... 3758
4.1.3 CRC ..... 3758
4.1.4 LE Test packet PDU ..... 3758
4.1.5 LE Test packet payload description ..... 3760
4.1.6 LE Test packet interval ..... 3761
4.1.7 Constant Tone Extension ..... 3761
4.1.8 LE Channel Sounding Test packet trailer ..... 3762
4.1.9 LE Channel Sounding Test packet payload ..... 3762
4.1.10 LE Channel Sounding Test exchanges ..... 3762

**5 Unified Test Protocol ..... 3763**
5.1 Introduction ..... 3763
5.2 Message format ..... 3765
5.3 Configuration messages ..... 3766
5.3.1 UTP_Set_RF_Channel ..... 3767
5.3.2 UTP_Set_Packet_Payload ..... 3767
5.3.3 UTP_Set_Payload_Length ..... 3768
5.3.4 UTP_Set_PHY ..... 3768

5.3.5 UTP_Set_Modulation_Index ..... 3769
5.3.6 UTP_Set_CTE_Length ..... 3769
5.3.7 UTP_Set_CTE_Type ..... 3770
5.3.8 UTP_Set_CTE_Slot_Durations ..... 3770
5.3.9 UTP_Set_CTE_Antenna_IDs ..... 3770
5.3.10 UTP_Set_Packet_Count ..... 3771
5.3.11 UTP_Set_Tx_Power_Level ..... 3771
5.3.12 UTP_Set_OTA_Exclusion_Period ..... 3771
5.3.13 UTP_Set_Vendor_Specific_Data ..... 3772
5.4 Control messages sent by the Upper Tester ..... 3772
5.4.1 UTP_Query_Supported_Features ..... 3773
5.4.2 UTP_Reset ..... 3773
5.4.3 UTP_Start_Test ..... 3773
5.4.4 UTP_Stop_Test ..... 3774
5.4.5 UTP_Vendor_Specific_Control ..... 3774
5.5 Control messages sent by the IUT ..... 3775
5.5.1 UTP_Accept ..... 3775
5.5.2 UTP_Reject ..... 3775
5.5.3 UTP_Reset_Accept ..... 3776
5.5.4 UTP_Test_Ended ..... 3776
5.6 Report messages ..... 3776
5.6.1 UTP_Report_Supported_Features ..... 3777
5.6.2 UTP_Report_IQ_Samples ..... 3777
5.6.3 UTP_Report_Receiver_Quality_Counters ..... 3779
5.6.4 UTP_Report_Vendor_Specific_Data ..... 3780

**6 UTP transport ..... 3782**
6.1 Introduction ..... 3782
6.2 2-wire UART ..... 3782
6.3 HCI ..... 3782
6.4 OTA ..... 3782
6.4.1 Introduction ..... 3782
6.4.2 Connection interval requirements ..... 3783
6.4.3 Transmitter test ..... 3784
6.4.4 Receiver test ..... 3785

# 1 INTRODUCTION

Direct Test Mode and Unified Test Protocol are protocols that are used by an Upper Tester to control the Implementation Under Test (IUT) and to provide a report back to the Upper Tester.

DTM shall operate over a physical transport interface between the Upper Tester and the IUT. The transport used may be either HCI (see Section 2) or 2-wire UART (see Section 3).

UTP may operate over a physical transport interface (UTP HCI Mode) or, if supported, an OTA transport interface (UTP OTA Mode) between the Upper Tester and the IUT. The transport used may be any of the following:

- 2-wire UART (see Section 6.2)
- HCI (see Section 6.3)
- OTA (see Section 6.4)

Figure 1.1 illustrates the alternatives for RFPHY Test modes setup using a physical transport interface.

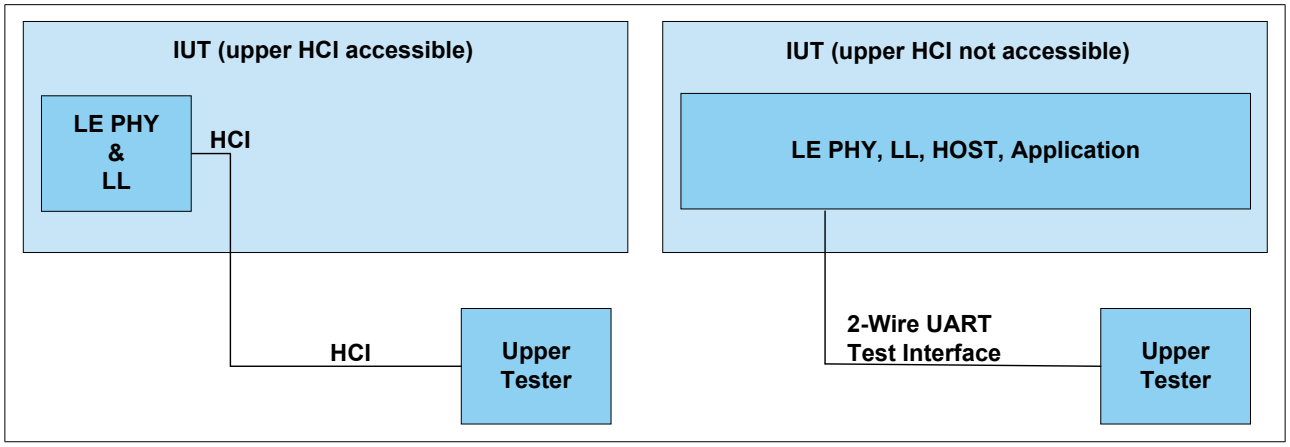



*Figure 1.1: Setup alternatives for RFPHY Test modes*

Figure 1.2 illustrates the RFPHY Test modes setup principle using a 2-wire UART interface.

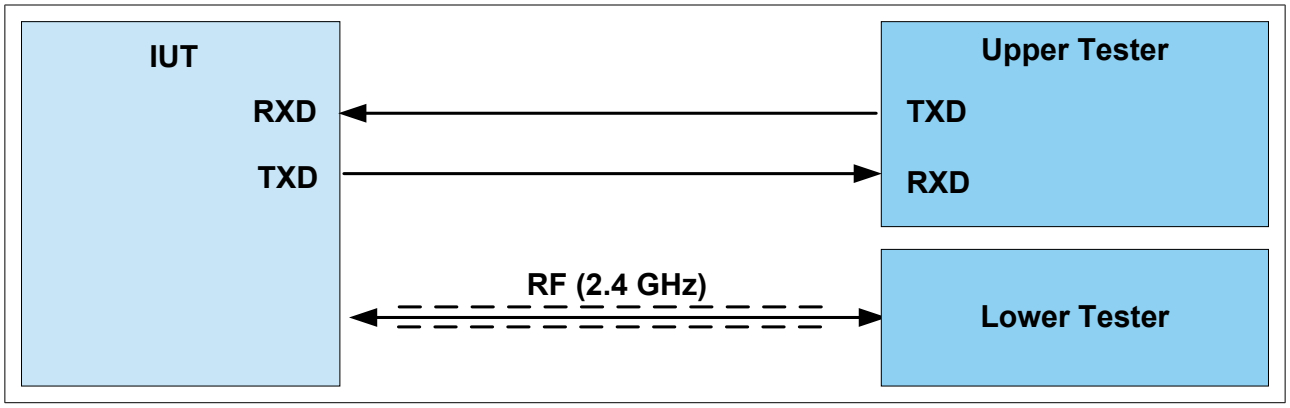



*Figure 1.2: Setup for RFPHY Test modes (2-wire UART control)*

Figure 1.3 illustrates the RFPHY Test mode setup for the OTA transport interface. This setup removes the requirement for the physical test interfaces shown in Figure 1.1 and Figure 1.2.

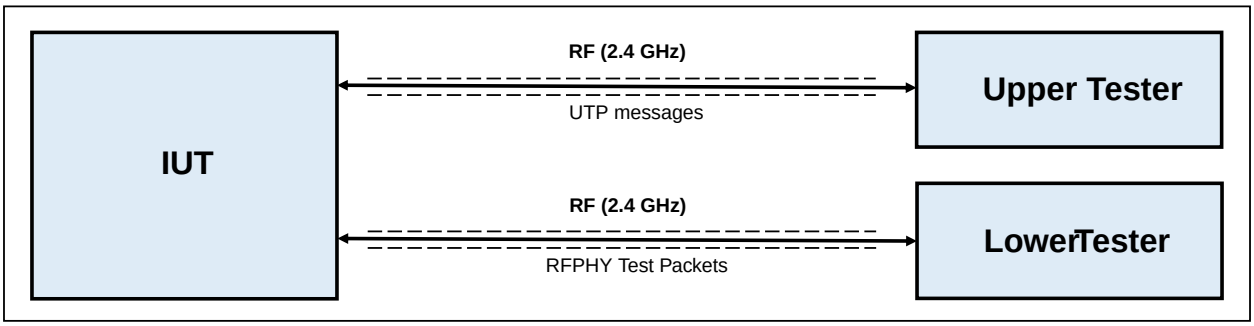



*Figure 1.3: Setup for RFPHY Test modes (OTA control)*

For LE devices supporting the Channel Sounding feature, DTM shall only be available when an accessible Host Controller Interface is provided, as shown in Figure 1.1 (left). The applicable CS commands and event are defined in Section 2.3.

# 2 LOW ENERGY TEST SCENARIOS

## 2.1 Test sequences

These sequences are used as routines and used to control an LE IUT with an accessible HCI or a 2-wire UART interface for RF testing.

The following mapping shall be performed from the RF testing commands to HCI commands and events or 2-wire UART commands and events, depending on which method is used:

| RF Test command / event | HCI command / event | 2-wire UART command / event |
|---|---|---|
| LE_Transmitter_Test command | HCI_LE_Transmitter_Test command | LE_Transmitter_Test command |
| LE_Receiver_Test command | HCI_LE_Receiver_Test command | LE_Receiver_Test command |
| LE_Test_End command | HCI_LE_Test_End command | LE_Test_End command |
| LE_Status event | HCI_Command_Complete event | LE_Test_Status event |
| LE_Packet_Report event | HCI_Command_Complete event | LE_Packet_Report event |

*Table 2.1: Mapping table of HCI / 2-wire UART commands and events*

The HCI commands and events used in Direct Test Mode are defined in [Vol 4] Part E, Section 7.8 and [Vol 4] Part E, Section 7.7.65 respectively.

## 2.2 Message sequence charts

### Transmitter Test

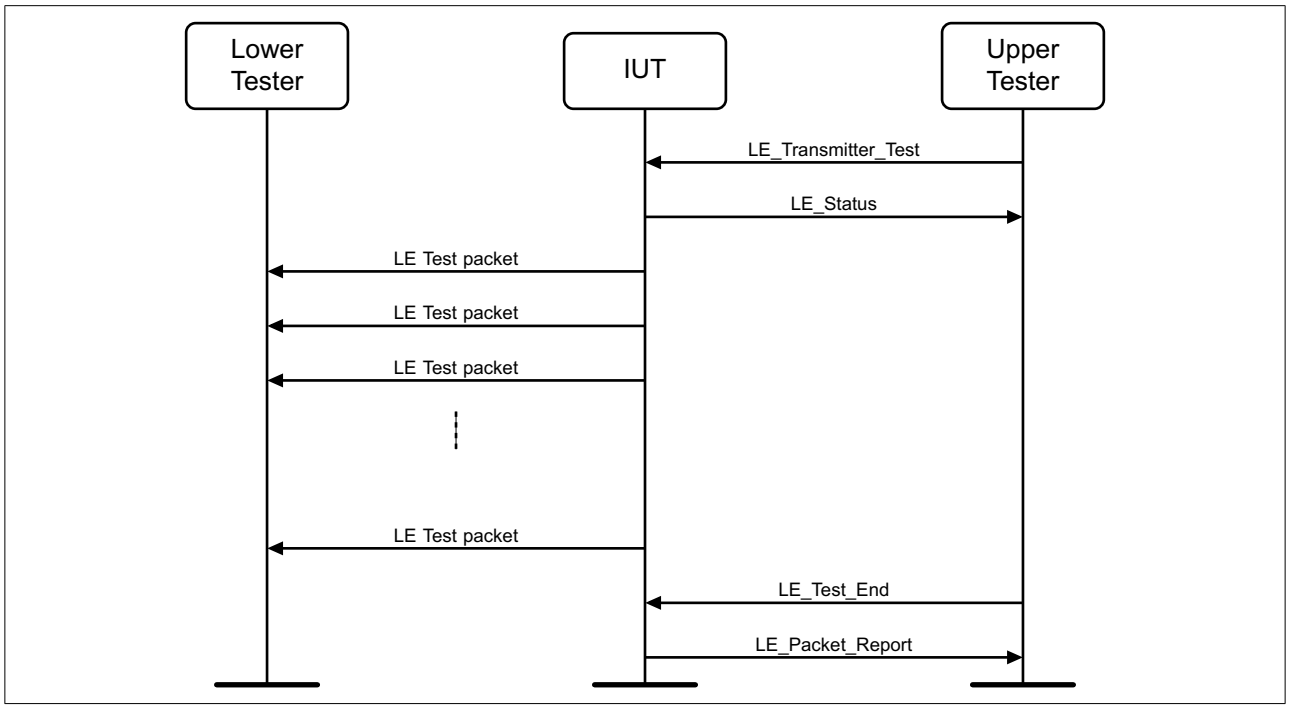



*Figure 2.1: Transmitter Test MSC*

### Receiver Test

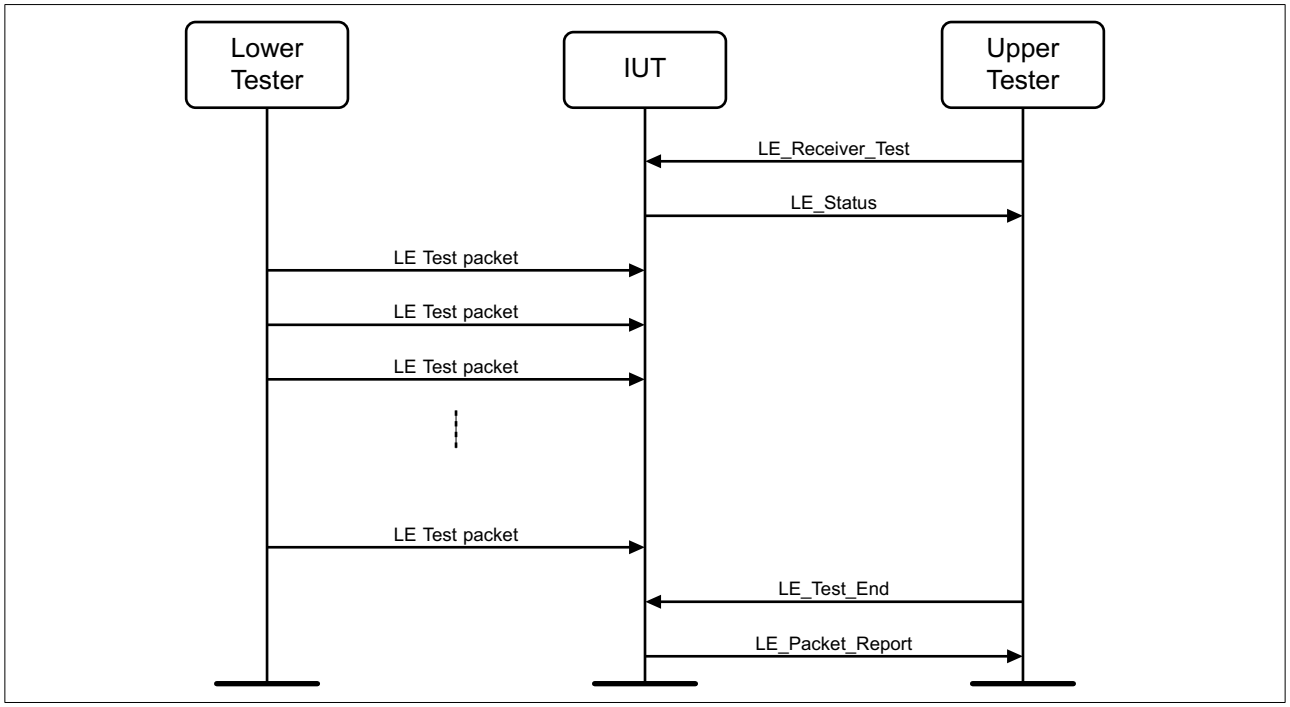



*Figure 2.2: Receiver Test MSC*

## 2.3 Channel Sounding test commands

The relevant HCI commands and events for performing RFPHY Channel Sounding testing are defined in Table 2.2.

| HCI command / event | Description |
|---|---|
| HCI_LE_CS_Test command | This command is used to begin a CS test where the IUT is placed in the role of either the initiator or reflector. A single CS procedure which consists of one or more CS subevents is scheduled. |
| HCI_LE_CS_Subevent_-Result event | This event is generated when the local Controller has results to report for a CS subevent within the CS procedure. |
| HCI_LE_CS_Subevent_-Result_Continue event | This event is generated after the local Controller has comple-ted a new CS subevent measurement and has already sent an HCI_LE_CS_Subevent_Result event for the specified subevent. |
| HCI_LE_CS_Test_End com-mand | This command is used to terminate any CS test that is in progress. |
| HCI_LE_CS_Test_End_-Complete event | This event is generated when the local Controller has stopped an ongoing CS test as a result of the HCI_LE_CS_Test_End command. |

*Table 2.2: Channel Sounding HCI commands and events*

The HCI commands and events used in Direct Test Mode are defined in [Vol 4] Part E, Section 7.8.

## 2.4 Channel Sounding message sequence charts

Test scenarios for Channel Sounding require the Lower Tester and IUT to have interchangeable roles, one side being in the initiator role and the other in the reflector role. Figure 2.3 shows an MSC with the IUT in the initiator role, with Figure 2.4 the IUT in the reflector role.

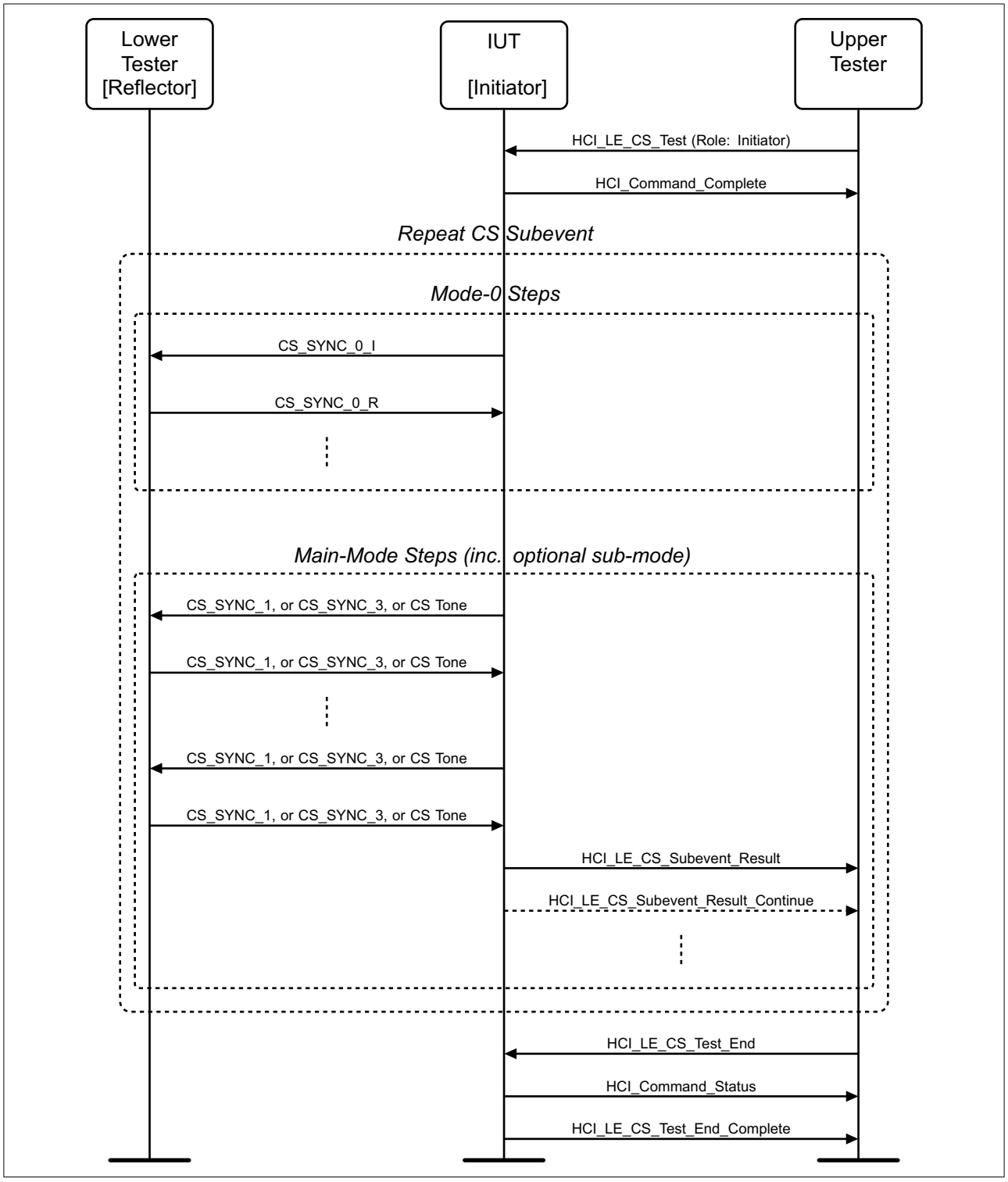



*Figure 2.3: Channel Sounding Test MSC: IUT in Initiator Role*

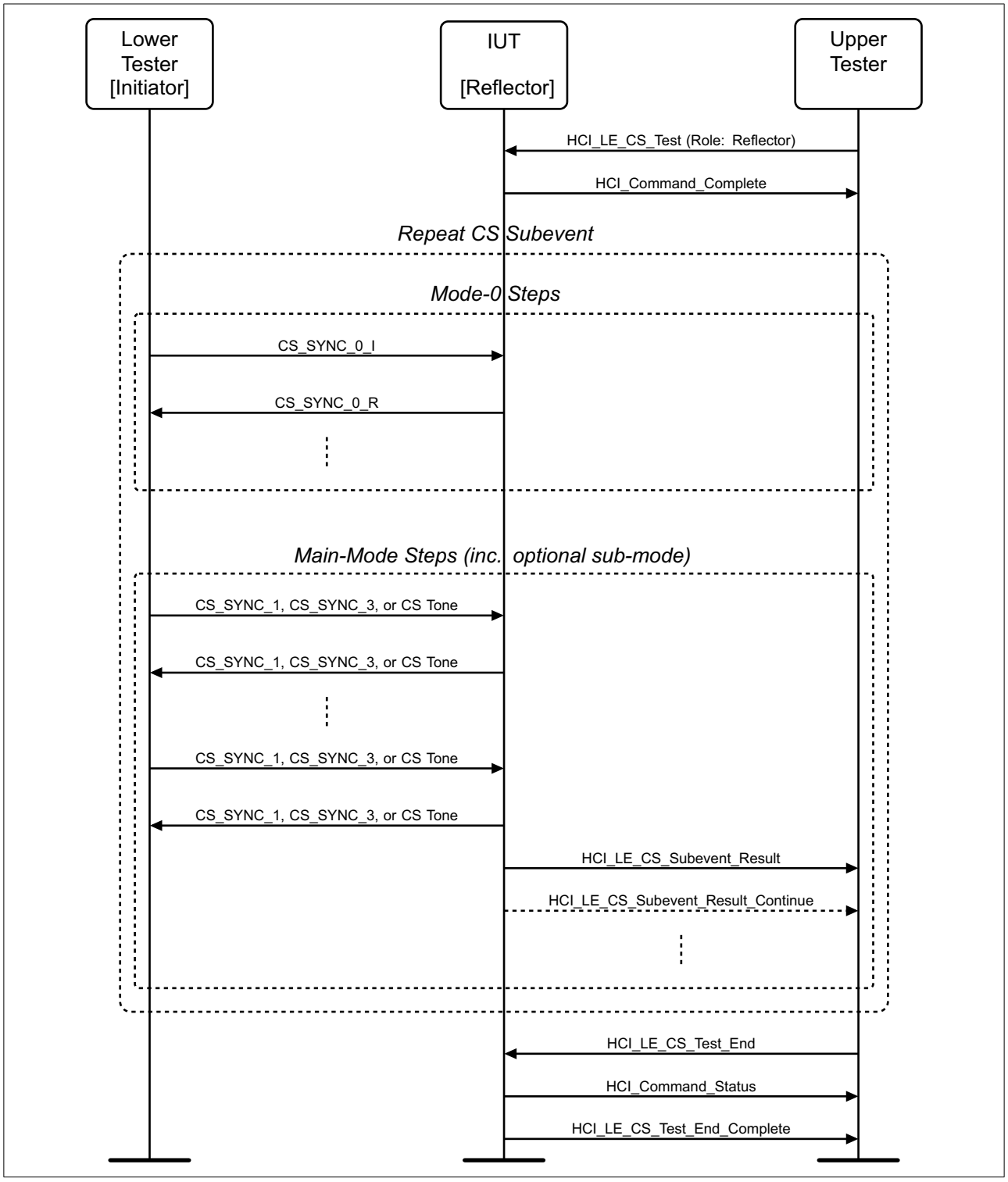



*Figure 2.4: Channel Sounding Test MSC: IUT in Reflector Role*

The test procedure below applies to all test scenarios, the IUT can assume either of the Initiator or Reflector roles as described in Figure 2.3, and Figure 2.4.

1. The Upper Tester sends an HCI_LE_CS_Test command to the IUT with Role configured appropriately to either 0x00 (initiator), or 0x01 (reflector) and receives a successful HCI_Command_Complete event response. All Channel Sounding parameters referred to in this test procedure are derived from the HCI_LE_CS_Test command. The first channel in the Channel parameter list is used as the starting channel for the test. At the beginning of the test, either the IUT or Lower Tester in the reflector role shall listen on this channel until it receives the first mode-0 transmission from the initiator.
2. The initiator sends the mode-0 CS_SYNC_0_I transmission to the reflector.
3. The reflector responds with the mode-0 CS_SYNC_0_R transmission to the initiator.
4. Repeat steps 2 and 3 the number of times specified by the Mode_0_Steps parameter.
5. The initiator sends the Main-Mode step transmission to the reflector.
6. The reflector responds with a Main-Mode transmission to the initiator.
7. Optional Sub_Mode steps specified by Sub_Mode_Type may be exchanged.
8. Repeat steps 5 to 7 with the parameters configured by the HCI_LE_CS_Test command from step 1 for this Channel Sounding test procedure observing the Subevent_Len, Subevent_Interval, the computed channel map or Channel array override, and so on.
9. The IUT sends an HCI_LE_CS_Subevent_Result event to the Upper Tester for each CS subevent based on the procedure parameters from step 1. The IUT may also send one or more HCI_LE_CS_Subevent_Result_Continue events for each CS subevent to the Upper Tester.
10. The Upper Tester sends an HCI_LE_CS_Test_End command to the IUT. The IUT shall respond to the Upper Tester with an HCI_Command_Status event. When the IUT has successfully sent all pending CS subevent results it shall generate an HCI_LE_CS_Test_End_Complete event.

The test scenario for the Stable Phase measurement differs from every other Channel Sounding RFPHY test. The Stable Phase test is enabled via the Override_Config Bit 10 in the HCI_LE_CS_Test command, defined in [Vol 4] Part E, Section 7.8.143. The Lower Tester shall assume the role of the reflector and IUT the role of initiator.

The Stable Phase measurement period is equal to the maximum CS step mode-2 duration as defined in [Vol 6] Part A, Section 3.4.

The Stable Phase measurement is performed on CS_Tone transmissions. As CS_Tone transmissions are continuous unmodulated carrier waves the Lower Tester cannot accurately synchronize. Instead, the Lower Tester shall rely on the preceding mode-0 step transmission (CS_SYNC_0_I) sent from the IUT to provide a satisfactory measurement anchor point. The Lower Tester is not required to respond to the IUT mode-0 transmission in this test, but the IUT shall still transmit a CS_Tone as if the Lower Tester had properly transmitted its portion of the mode-0 exchange. The IUT is not required to provide a measurement report on the mode-0 packet (CS_SYNC_0_R) if sent back from the Lower Tester. The measurement period T_PM_MEAS shall begin following this mode-0 step as well as the T_FCS frequency hop period defined in [Vol 6] Part H, Section 4.3. This measurement shall not include the T_PM exclusion period defined in [Vol 6] Part H, Section 4.6.

Figure 2.5 shows the Stable Phase measurement MSC. Each Stable Phase measurement cycle uses a single CS procedure containing a single CS subevent. Each CS subevent contains a single mode-0 step, followed by a single CS_Tone. As such, there is a single measurement period T_PM_MEAS available for obtaining phase measurement samples per CS procedure. The number of measurement cycles may be adjusted dependent upon the number of phase measurement samples required.

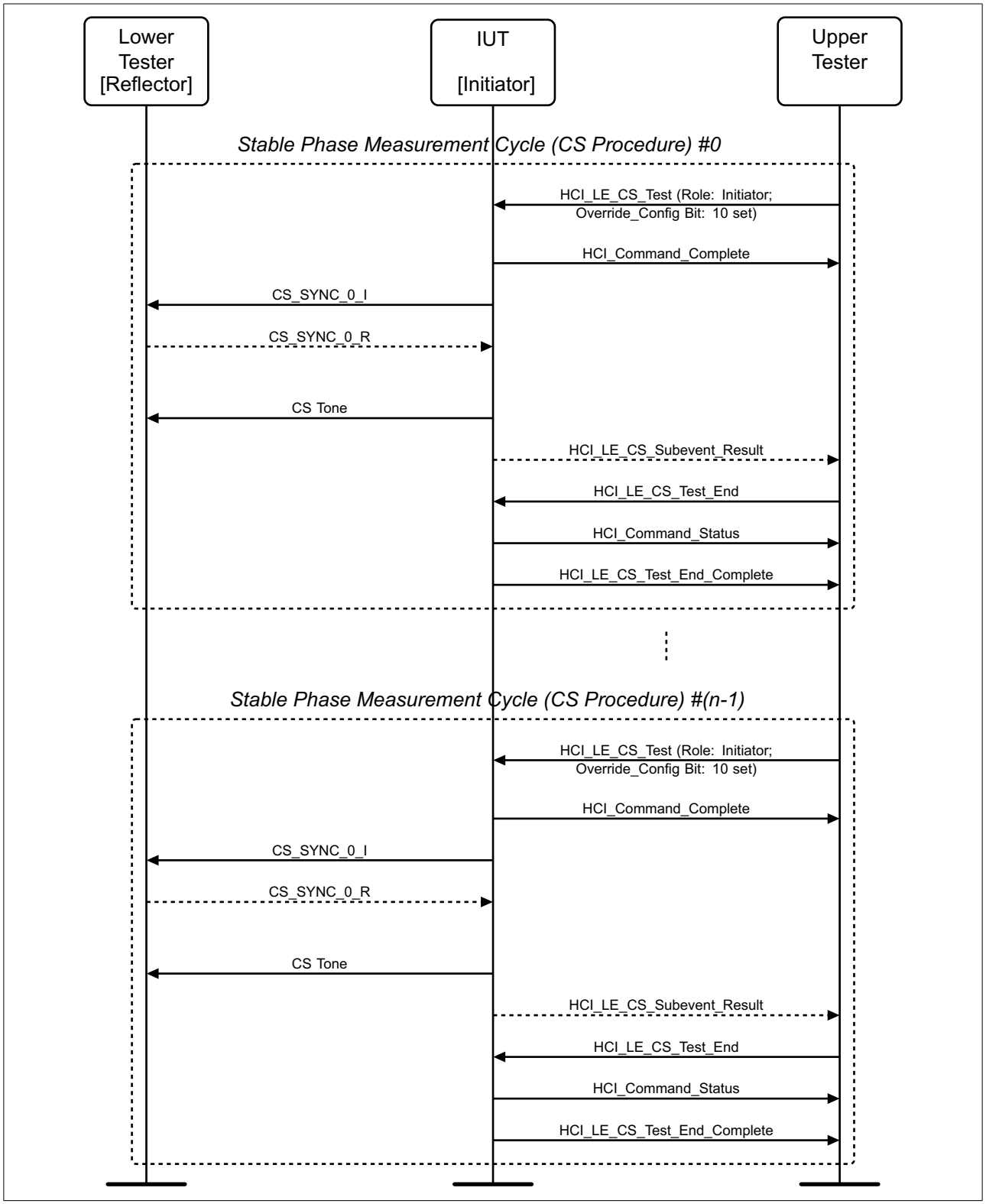



*Figure 2.5: Stable Phase Channel Sounding Test MSC*

# 3 UART TEST INTERFACE

## 3.1 UART Interface characteristics

The UART interface characteristics shall be set to use the following parameters:

- Baud rate: One of the following shall be supported by the IUT:

  1200, 2400, 9600, 14400, 19200, 38400, 57600, 115200, 230400, 460800, 500000, 576000, 921600, 1000000, 1152000, 2000000, 3000000, 3500000, 4000000
- Number of data bits: 8
- No parity
- 1 stop bit
- No flow control (RTS or CTS)

## 3.2 UART functional description

The Upper Tester shall always initiate a test scenario using the UART interface. The IUT shall respond to the commands from the Upper Tester.

The Upper Tester sends test commands to the IUT. The IUT shall respond with a test status event or packet report event.

The Upper Tester shall not transmit further commands before it receives a response from the IUT. If the Upper Tester does not receive a response from the IUT within the time $t_{TIMEOUT}$, the Upper Tester shall transmit a reset command (i.e., a test setup command with the control argument set to 0x00) to the IUT and display an appropriate error message. For the reset command, $t_{RESPONSE}$ and $t_{TIMEOUT}$ do not apply.

On reception of a reset command, the IUT shall reset all parameters to their default state.

**Definitions**

- All commands and events consist of 16 bits (2 bytes).
- The most significant bit is bit number 15.
- The least significant bit is bit number 0.
- The most significant byte is from bit 15 to 8.
- The least significant byte is from bit 7 to 0.
- Commands and events are sent most significant byte (MSB) first, followed by the least significant byte (LSB).

## 3.3 Commands and events

### 3.3.1 Command and event behavior

Table 3.1 outlines the set of commands which can be received by the IUT and the corresponding response events that can be transmitted by the IUT.

| Command (IUT RXD) | Event (IUT TXD) |
|---|---|
| LE_Test_Setup | LE_Test_Status SUCCESS<br>LE_Test_Status FAIL |
| LE_Receiver_Test | LE_Test_Status SUCCESS<br>LE_Test_Status FAIL |
| LE_Transmitter_Test | LE_Test_Status SUCCESS<br>LE_Test_Status FAIL |
| LE_Test_End | LE_Packet_Report<br>LE_Test_Status FAIL |

*Table 3.1: 2-Wire command and event behavior*

### 3.3.2 Commands

Command packet formats are shown in Figure 3.1 and Figure 3.2.

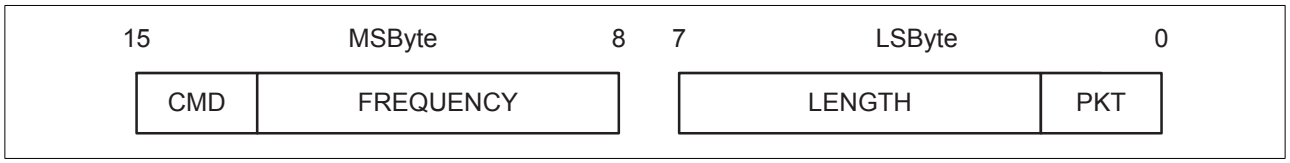



*Figure 3.1: Command message format for LE_Transmitter_Test and LE_Receiver_Test commands*

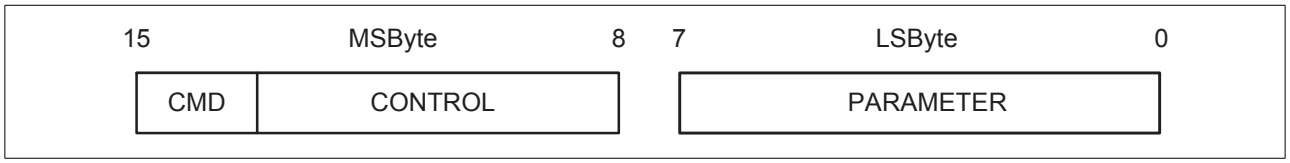



*Figure 3.2: Command message format for LE_Test_Setup and LE_Test_End commands*

Note: Some cases of the Test Setup and Test End commands have four parameter values, differing only in the bottom 2 bits, specified for the same action. In these cases the specific value chosen does not affect the behavior of the command.

*CMD (command):* *Size: 2 bits*

| Value $b_1b_0$ | Parameter Description |
|---|---|
| 00 | LE_Test_Setup command |
| 01 | LE_Receiver_Test command |
| 10 | LE_Transmitter_Test command |
| 11 | LE_Test_End command |

*Test Setup command:* *Size: 14 bits*

| Control (6 bits) | Parameter (8 bits) | Description |
|---|---|---|
| 0x00 | 0x00 to 0x03 | RESET; the upper 2 bits of the data length for any LE_Transmitter_Test or LE_Receiver_Test commands following are set to 00, the PHY is set to LE 1M, the receiver assumes the transmitter has a standard modulation index, and no Constant Tone Extension is present. |
| | Any other value | Reserved for future use |
| 0x01 | 0x00 to 0x0F | Set the upper 2 bits of the data length for any LE_Transmitter_Test or LE_Receiver_Test commands following to bits 2 and 3 of the parameter (to enable a length greater than 0x3F to be used) |
| | Any other value | Reserved for future use |
| 0x02 | 0x04 to 0x07 | PHY set to LE 1M |
| | 0x08 to 0x0B | PHY set to LE 2M |
| | 0x0C to 0x0F | PHY set to LE Coded; transmitter is to use S=8 data coding |
| | 0x10 to 0x13 | PHY set to LE Coded; transmitter is to use S=2 data coding |
| | Any other value | Reserved for future use |
| 0x03 | 0x00 to 0x03 | Receiver assumes transmitter has a standard modulation index |
| | 0x04 to 0x07 | Receiver assumes transmitter has a stable modulation index |
| | Any other value | Reserved for future use |
| 0x04 | 0x00 to 0x03 | Read the test case supported features.<br>The LE_Test_Status event will return the state of the test case supported features as detailed in the LE_Test_Status event (Section 3.4.1). |
| | Any other value | Reserved for future use |

| Control (6 bits) | Parameter (8 bits) | Description |
|---|---|---|
| 0x05 | 0x00 to 0x03 | Read supportedMaxTxOctets (see [Vol 6] Part B, Section 4.5.10) |
| | 0x04 to 0x07 | Read supportedMaxTxTime (see [Vol 6] Part B, Section 4.5.10) |
| | 0x08 to 0x0B | Read supportedMaxRxOctets (see [Vol 6] Part B, Section 4.5.10) |
| | 0x0C to 0x0F | Read supportedMaxRxTime (see [Vol 6] Part B, Section 4.5.10) |
| | 0x10 | Read maximum length of Constant Tone Extension supported |
| | Any other value | Reserved for future use |
| 0x06 | 0x00 | No Constant Tone Extension |
| | Any other value | CTEInfo (see [Vol 6] Part B, Section 2.5.2) |
| 0x07 | 0x01 | Sample Constant Tone Extension with 1 µs slots |
| | 0x02 | Sample Constant Tone Extension with 2 µs slots |
| | Any other value | Reserved for future use |
| 0x08 | Bits 0 to 6:<br>0x01 to 0x4B | Number of antennae in the antenna array |
| | Any other value | Reserved for future use |
| | Bit 7:<br>0 | Antenna switching pattern A: 1, 2, 3, ..., n, 1, 2, 3, ..., n, ... (where n is the number of antennae in the antenna array) |
| | 1 | Antenna switching pattern B: 1, 2, 3, ..., n, n-1, n-2, ..., 1, ... (where n is the number of antennae in the antenna array) |
| 0x09 | -127 to +20 | Set transmitter to the specified or the nearest transmit power level<br>Units: dBm |
| | 0x7E | Set transmitter to minimum transmit power level |
| | 0x7F | Set transmitter to maximum transmit power level |
| | All other values | Reserved for future use |

If an AoD Constant Tone Extension is selected when transmitting, Control 0x08 shall be used before starting the test. If an AoA Constant Tone Extension is selected when receiving, Controls 0x07 and 0x08 shall be used before starting the test.

Control 0x07 does not affect receiving AoD Constant Tone Extensions or any transmissions. Control 0x08 does not affect transmitting AoA Constant Tone Extensions or receiving AoD Constant Tone Extensions.

In the receiver test, the CTEInfo field specified using Control 0x06 indicates the expected type and length of the Constant Tone Extension. If either the length or type of the Constant Tone Extension in a received LE Test packet does not match the expected value, then the IUT shall discard that packet.

*LE_Test_End command:* *Size: 14 bits*

| Control (6 bits) | Parameter (8 bits) | Description |
|---|---|---|
| 0x00 | 0x00 to 0x03 | LE_Test_End command |
| 0x00 | Any other value | Reserved for future use |
| 0x01 to 0x3F | Any value | Reserved for future use |

### LE_Transmitter_Test and LE_Receiver_Test commands:

*Frequency:* *Size: 6 bits*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x27 | The frequency to be used; a value of N represents a frequency of (2N+2402) MHz (the available range is therefore even values from 2402 MHz to 2480 MHz) |
| 0x28 to 0x3F | Reserved for future use |

*Length:* *Size: 6 bits*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x3F | The lower 6 bits of the packet length in bytes of payload data in each packet (the top two bits are set by the LE_Test_Setup command) |

*PKT (Packet Type):* *Size: 2 bits*

| Value $b_1b_0$ | Parameter Description |
|---|---|
| 00 | PRBS9 Packet Payload |
| 01 | 11110000 Packet Payload |
| 10 | 10101010 Packet Payload |
| 11 | On the LE Uncoded PHYs: Vendor Specific<br>On the LE Coded PHY: 11111111 |

## 3.4 Events

There are two types of events sent by the IUT:

1. LE_Test_Status event
2. LE_Packet_Report event

The event packet format is shown in Figure 3.3. This packet format is used for both LE_Test_Status events and LE_Packet_Report events.

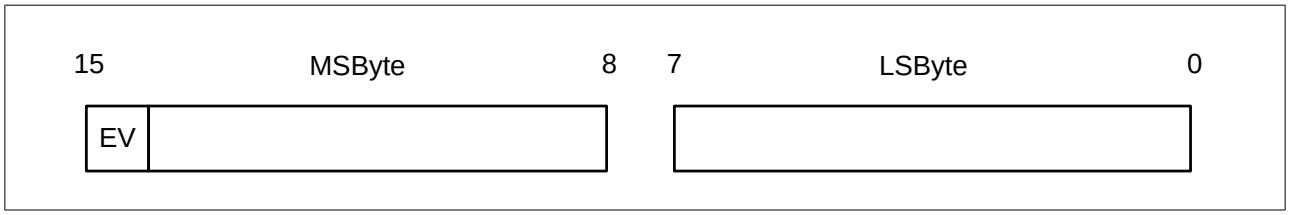



*Figure 3.3: Event packet format*

*EV (event):* *Size: 1 bit*

| Value | Parameter Description |
|---|---|
| 0 | LE_Test_Status event |
| 1 | LE_Packet_Report event |

### 3.4.1 LE_Test_Status event

The LE_Test_Status event packet format is as shown in Figure 3.4.

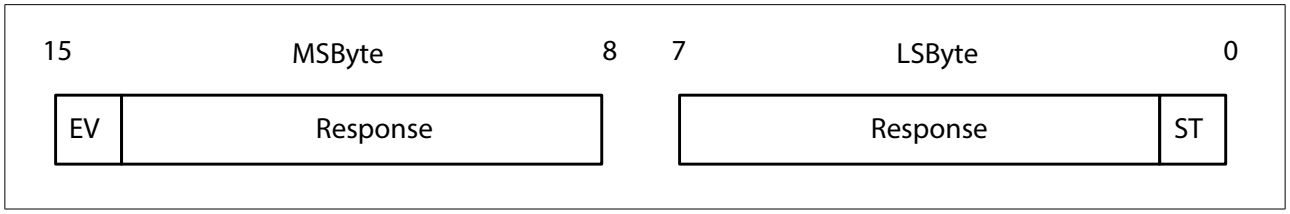



*Figure 3.4: LE_Test_Status event*

*ST (status):* *Size: 1 bit*

| Value | Parameter Description |
|---|---|
| 0 | Success |
| 1 | Error |

*Response:* [1] *Size: 14 bits*

[1]If the event has a status of "Error" or was generated in response to a command other than LE_Test_Setup, then this field is Reserved for future use.

| **LE_Test_-Setup command control parameter** | **Value bits 1 to 14**[2] | **Description** |
|---|---|---|
| 0x04 | Bit 1 | LE Data Packet Length Extension feature supported |
| | Bit 2 | LE 2M PHY supported |
| | Bit 3 | Transmitter has a Stable Modulation Index |
| | Bit 4 | LE Coded PHY supported |
| | Bit 5 | Constant Tone Extension supported |
| | Bit 6 | Antenna switching supported |
| | Bit 7 | 1 µs switching supported for AoD transmission |
| | Bit 8 | 1 µs sampling supported for AoD reception |
| | Bit 9 | 1 µs switching and sampling supported for AoA reception |
| | Bits 10 to 14 | Reserved for future use |
| 0x05 | Bits 1 to 14 | One of the following values (depending on the parameter in the original query):<br>• Maximum transmit or receive time, in microseconds, that the local Controller supports for transmission of a single Link Layer Data Physical Channel PDU, divided by 2.<br>• Maximum number of payload octets that the local Controller supports for transmission of a single Link Layer Data Physical Channel PDU.<br>• Maximum length of the Constant Tone Extension that the local Controller supports for transmission in a Link Layer packet, in 8 µs units.<br><br>Range:<br>• 0x00A4 to 0x2148 for times or<br>• 0x001B to 0x00FF for number of octets.<br>• 0x02 to 0x14 for the maximum Constant Tone Extension length<br><br>All values outside the range are reserved for future use. |

| LE_Test_-Setup command control parameter | Value bits 1 to 14[2] | Description |
|---|---|---|
| 0x09 | Bits 1 to 8 | Actual transmit power level set by the transmitter<br>Range: -127 to +20<br>Units: dBm |
| | Bit 9 | Set to 1 if the transmitter is at minimum transmit power level |
| | Bit 10 | Set to 1 if the transmitter is at maximum transmit power level |
| | Bits 11 to 14 | Reserved for future use |
| All other values | | Reserved for future use |

[2]This field is described as having bits 1 to 14 rather than 0 to 13 to avoid confusion.

### 3.4.2 LE_Packet_Report event

The LE_Packet_Report event packet format is shown in Figure 3.5. The *Packet Count* parameter indicates the number of received LE Test packets. The *Packet Count* in the LE_Packet_Report event ending a transmitter test shall be 0.

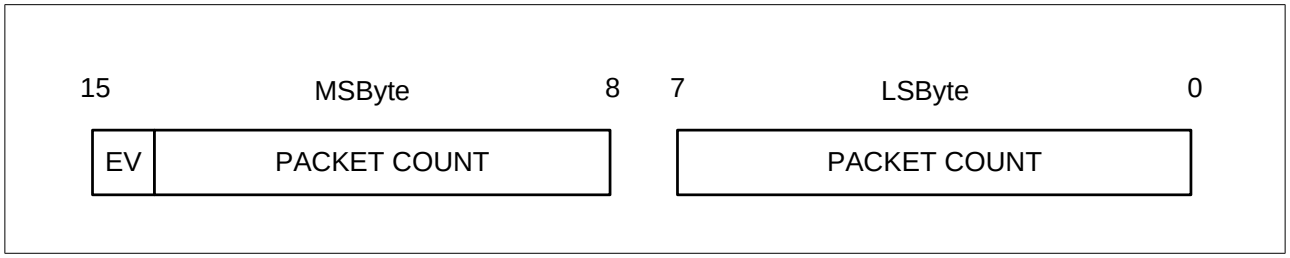



*Figure 3.5: LE_Packet_Report event*

*PACKET COUNT:* *Size: 15 bits*

| Value | Parameter Description |
|---|---|
| N | N is the number of packets received<br>Range = 0 to 32767. |

Note: The IUT is not responsible for any overflow conditions of the packet count. That responsibility belongs with the RFPHY test system or other auxiliary equipment.

## 3.5 Timing - command and event

The timing requirements are as shown in Table 3.2.

| Symbol | Parameter | Min. | Max. | Unit |
|---|---|---|---|---|
| $b_{ERR}$ | Baud rate accuracy | | ±5 | % |
| $t_{MIN}$ | The time between the first and second byte of the command or event (end of stop bit to start of start bit) | 0 | 5 | ms |
| $t_{RESPONSE}$ | The time from a IUT receiving a command (end of stop bit) until the IUT responds (start of start bit) | 0 | 50 | ms |
| $t_{TURNAROUND}$ | The time from when the Upper Tester receives a response (end of stop bit) until the Upper Tester sends another command (start of start bit) | 5 | - | ms |
| $t_{TIMEOUT}$ | The time from when the Upper Tester sends a command (end of stop bit) until the Upper Tester times out (not having received end of the stop bit in the response) | 51 | 100 | ms |

*Table 3.2: Parameter requirements table for 2-wire UART interface*

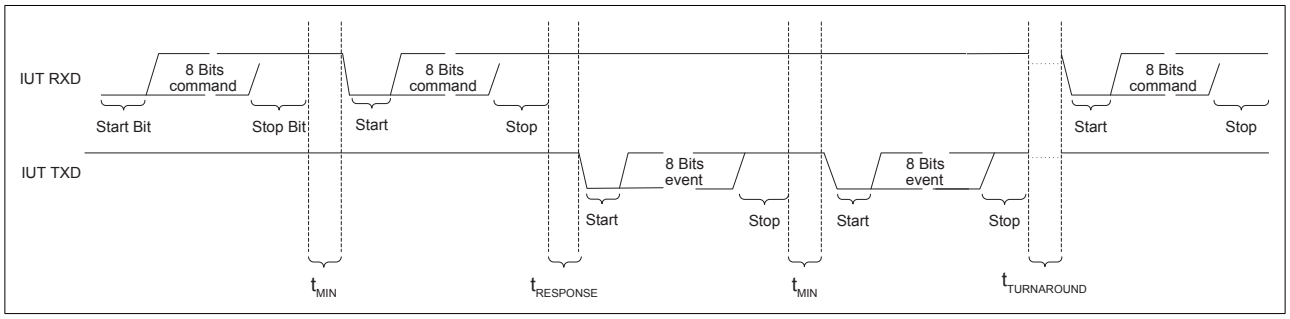



*Figure 3.6: Command and event timing on 2-wire UART interface*

The commands and events shall be transmitted with two 8-bit bytes with a maximum time between the 2 transmissions. A timeout is required for no response or an invalid response from the IUT.

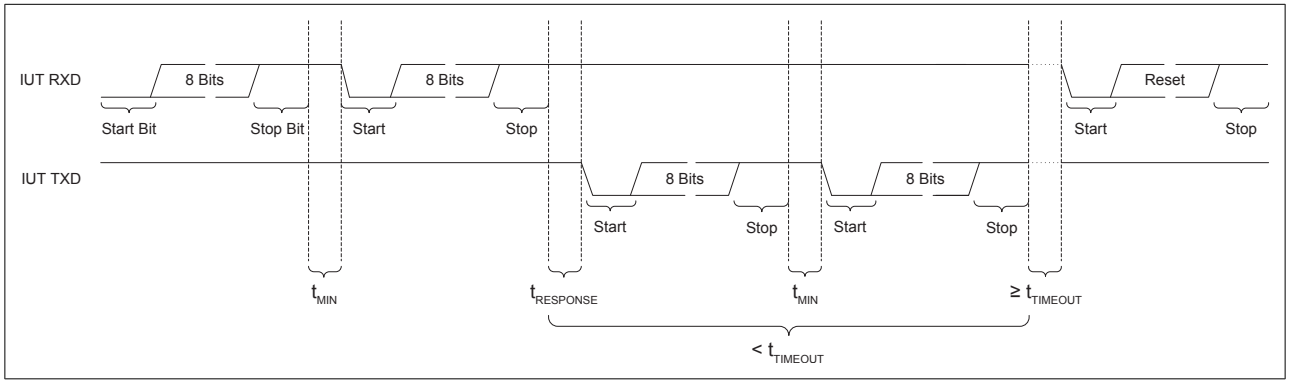



*Figure 3.7: Command and event timing on 2-wire UART interface showing timeout*

# 4 LE TEST PACKET DEFINITION

## 4.1 LE Test packets format

The LE Test packet format for the LE Uncoded PHYs shall be as shown in Figure 4.1. The LE Test packet format for the LE Coded PHY shall be as shown in Figure 4.2. LE test packets are required for LE RFPHY conformance testing using Direct Test Mode. Except as modified by this section, the LE Test packet formats shall be identical to the formats specified in [Vol 6] Part B, Section 2.1 and [Vol 6] Part B, Section 2.2.

Depending on the test, the packet payload content may vary.

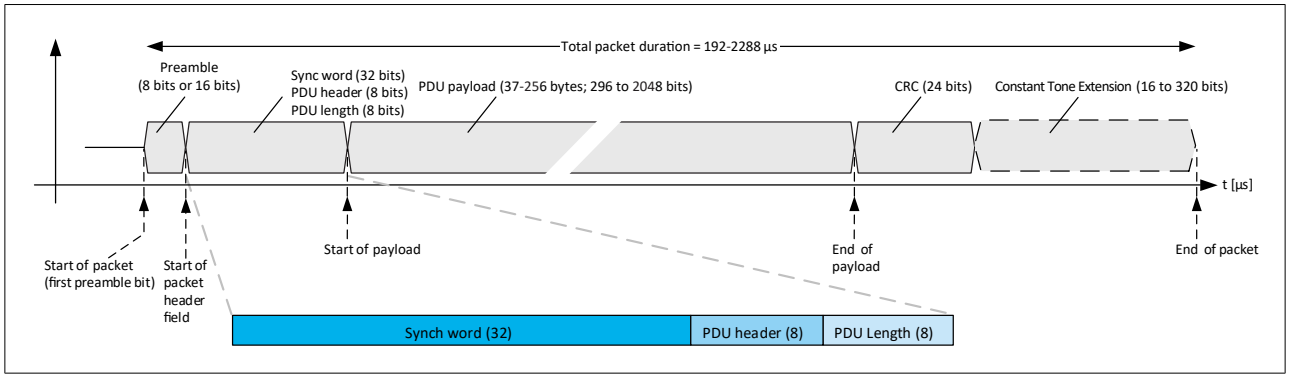



*Figure 4.1: LE Test packet format for the LE Uncoded PHYs*

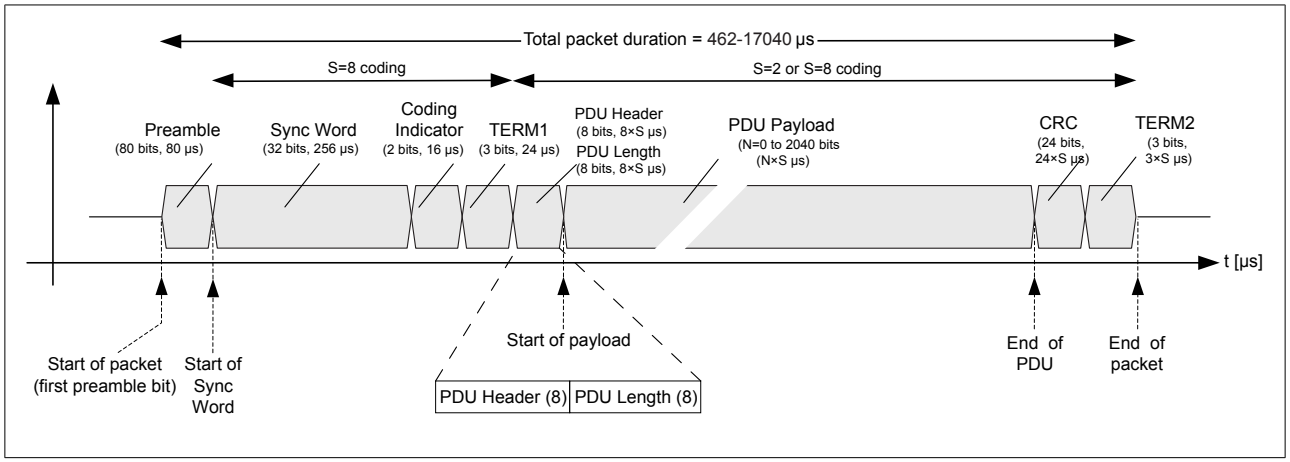



*Figure 4.2: LE Test packet format for the LE Coded PHY*

The LE Channel Sounding test packet (CS_SYNC) format shall be as shown in Figure 4.3. Except as modified by this section, the LE CS Test packet format shall be identical to the format specified in [Vol 6] Part H, Section 2.

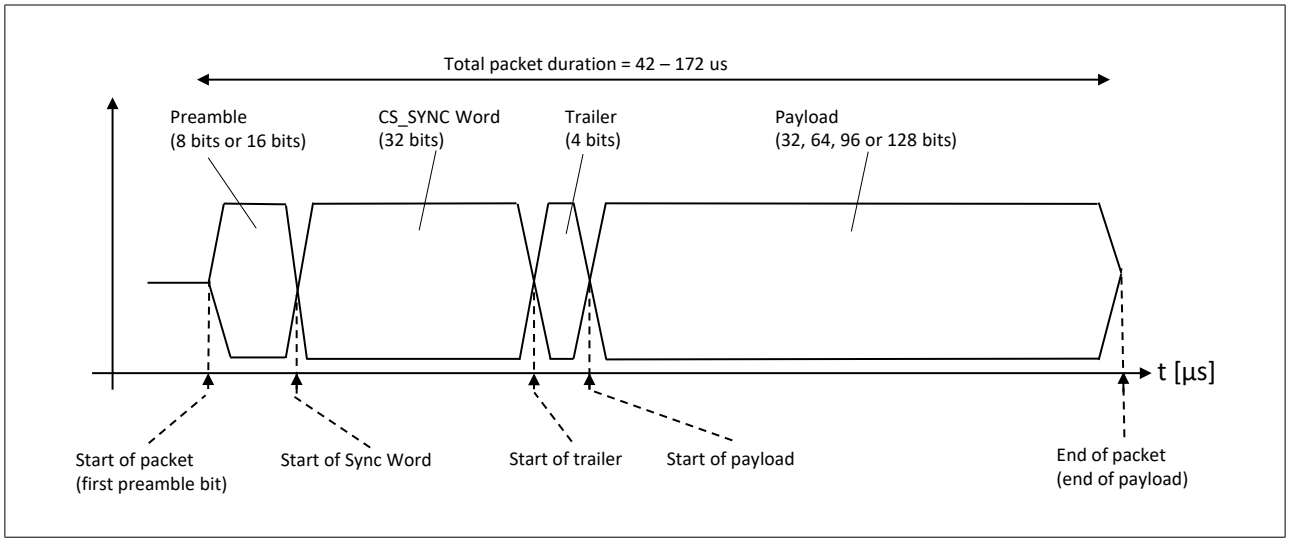



*Figure 4.3: LE CS Test packet format*

### 4.1.1 Whitening

LE test packets and LE CS test packets shall not use whitening.

### 4.1.2 Preamble and synchronization word

LE test packets shall have ‘10010100100000100110111010001110’ (in transmission order) as the synchronization word. The preamble for all LE test packets is thus ‘10101010’ (in transmission order) when the implementation under test is configured for the LE 1M PHY, ‘1010101010101010’ (in transmission order) if the implementation under test is configured for the LE 2M PHY, and the preamble described in [Vol 6] Part B, Section 2.2.1 if the implementation under test is configured for the LE Coded PHY.

LE CS test packets use Access Addresses (CS_SYNC Words) as defined by the HCI_LE_CS_Test command, [Vol 4] Part E, Section 7.8.

### 4.1.3 CRC

The CRC shift register shall be preset with 0x555555 for every LE test packet. The CRC portion is not applicable for LE CS test packets.

### 4.1.4 LE Test packet PDU

The LE test packet PDU consists of an 8-bit header, an 8-bit length field, an optional 8-bit CTEInfo field, and a variable size payload. Its structure is as shown in Figure 4.4 and Figure 4.5.

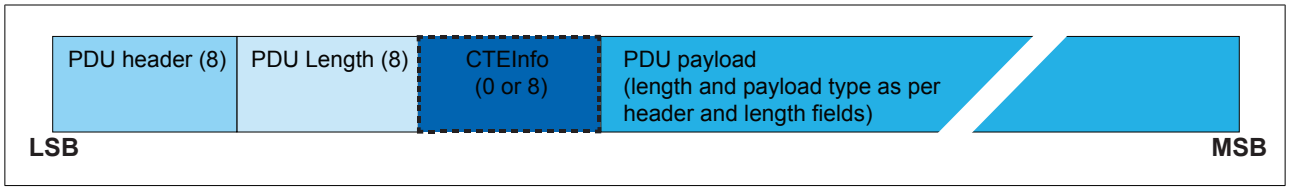



*Figure 4.4: LE Test packet PDU structure*

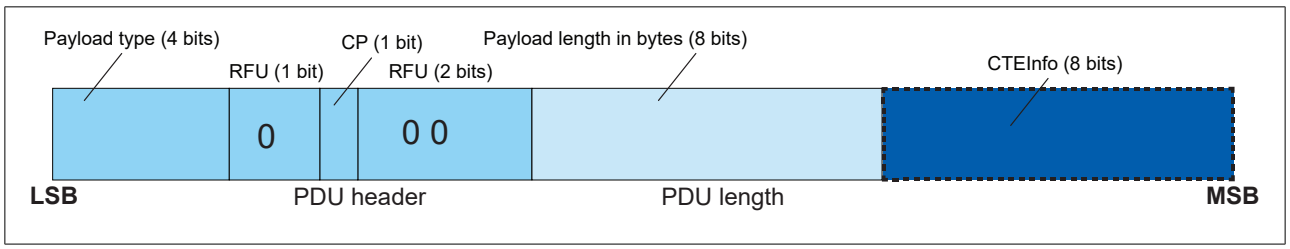



*Figure 4.5: LE Test packet header and length field structure*

The first four bits of the PDU header field indicate the payload content type as defined in Table 4.1. The CTEInfo Present (CP) field of the PDU header indicates whether the CTEInfo field is present and therefore whether the test packet has a Constant Tone Extension. If the CP field is 0, then no CTEInfo field is present and there is no Constant Tone Extension in the test packet. If the CP field is 1, then the CTEInfo field is present and the test packet includes a Constant Tone Extension. The CTEInfo field is defined in [Vol 6] Part B, Section 2.5.2. The length field expresses the Payload length in bytes.

Note: On the LE Coded PHY, this section defines the PDU contents before coding.

| **Payload type $b_3b_2b_1b_0$** | **Payload description** |
|---|---|
| 0b0000 | PRBS9 sequence ‘11111111100000111101...’ (in transmission order) as described in Section 4.1.5 |
| 0b0001 | Repeated ‘11110000’ (in transmission order) sequence as described in Section 4.1.5 |
| 0b0010 | Repeated ‘10101010’ (in transmission order) sequence as described in Section 4.1.5 |
| 0b0011 | PRBS15 sequence as described in Section 4.1.5 |
| 0b0100 | Repeated ‘11111111’ (in transmission order) sequence |
| 0b0101 | Repeated ‘00000000’ (in transmission order) sequence |
| 0b0110 | Repeated ‘00001111’ (in transmission order) sequence |
| 0b0111 | Repeated ‘01010101’ (in transmission order) sequence |

*Table 4.1: LE Test packet PDU header’s Type field encoding*

*Example*: For LE test packets with 0x0F payload contents (‘11110000’ in transmission order) and with an LE test packet payload length of 37 bytes (296 bits), the LE test packet header and length type field will be ‘1000000010100100’ in transmission order.

### 4.1.5 LE Test packet payload description

The LE test packet payload content alternatives required for the Bluetooth Low Energy RFPHY conformance tests are:

**PRBS9:**

A 9-bit pseudorandom binary sequence used for wanted signal payload content. The PRBS9 sequence repeats itself after the ($2^9$ -1 = 511) bit. The PRBS9 sequence may be generated in a nine stage shift register whose $5^{th}$ and $9^{th}$ stage outputs are XORed (see Figure 4.6) and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONEs (i.e. the shift register is initialized with nine ONEs).

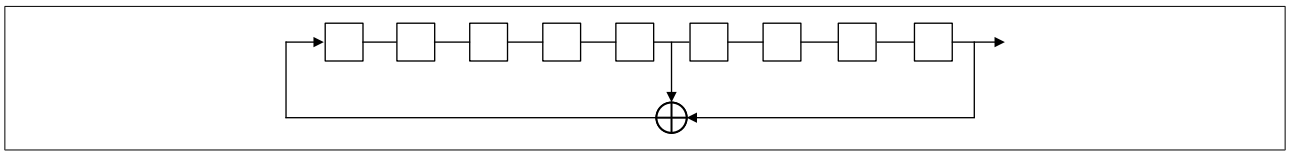


*Figure 4.6: Linear feedback shift register for generation of the PRBS9 sequence*

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated).

**PRBS15:**

A 15-bit pseudorandom binary sequence that is used for the interfering signal and can optionally be used for wanted signal payload content. The PRBS15 sequence repeats itself after the ($2^{15}$ -1 = 32767) bit. The PRBS15 sequence may be generated in a fifteen stage shift register whose 14th and 15th stage outputs are XORed (see Figure 4.7) and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 15 consecutive ONEs (i.e., the shift register is initialized with fifteen ONEs).

This PRBS15 definition is consistent with ITU T-REC-01 150-199605-I. SERIES O: SPECIFICATIONS OF MEASURING EQUIPMENT - Equipment for the measurement of digital and analogue/digital parameters.

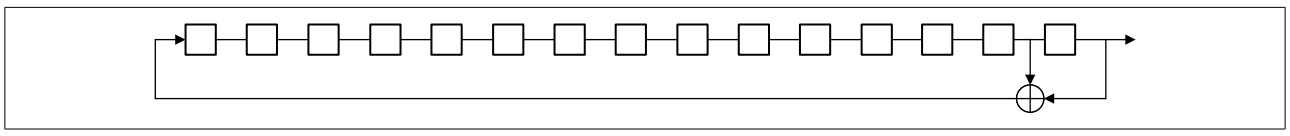


*Figure 4.7: Linear feedback shift register for generation of the PRBS15 sequence*

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated).

**10101010:**

Repeated sequence of alternating 1's and 0's, starting at the first payload bit and ending at the start of the first bit in the CRC. This pattern is used to verify the frequency deviation and the Gaussian filtering properties of the transmitter modulator.

**11110000:**

Repeated sequence of alternating 0's and 1's in groups of four (i.e. 1111000011110000...), starting at the first payload bit and ending at the start of the first bit in the CRC. This pattern is used to verify the frequency deviation and the Gaussian filtering properties of the transmitter modulator.

### 4.1.6 LE Test packet interval

While in LE direct TX mode, LE test packets shall be transmitted from the IUT with a packet interval I(L) as defined below; see the top half of Figure 4.8 for reference.

While in LE direct RX mode, the nominal packet interval of the LE test packets transmitted from the Lower Tester is I(L), but the packet interval may be extended to a maximum of T(L) upon change of the DTX parameter settings and during verification of the IUT PER reporting functionality. See the bottom half of Figure 4.8 for reference.

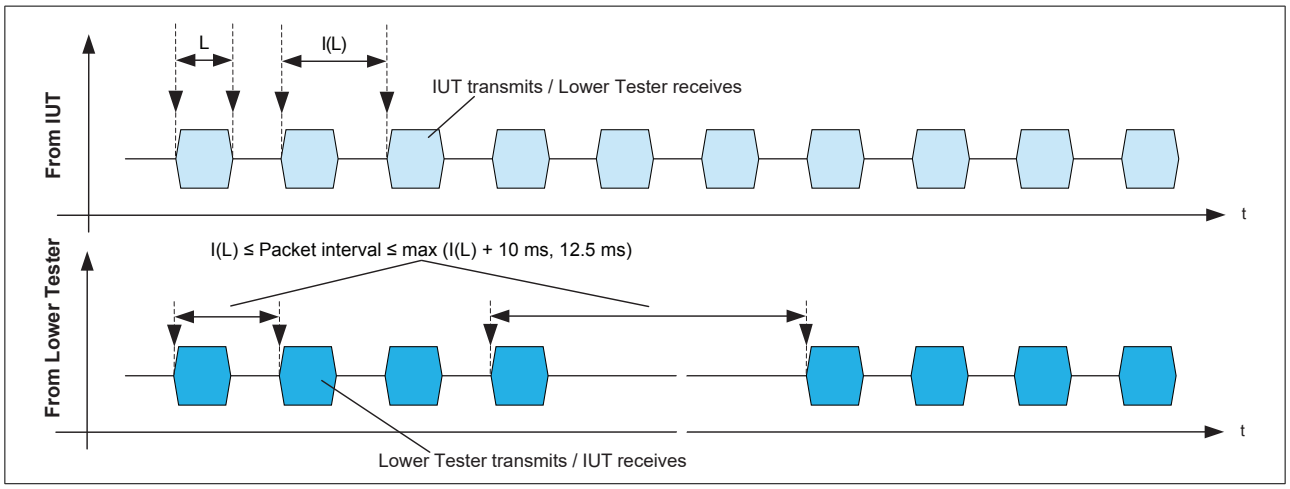



*Figure 4.8: LE Test packet interval in LE Direct Test mode*

For an LE Test packet length of L µs, I(L) = $\lceil (L + 249) \div 625 \rceil$ × 625 µs and T(L) = max (I(L) + 10 ms, 12.5 ms).

### 4.1.7 Constant Tone Extension

The Constant Tone Extension is an optional field that consists of a constantly modulated series of unwhitened 1s. It is 16 to 160 bits when operating at 1 Msym/s modulation or 32 to 320 bits when operating at 2 Msym/s modulation. The Constant Tone Extension

is not included in CRC or MIC calculations. The Constant Tone Extension shall only be present on the LE Uncoded PHYs.

If Direct Test Mode is being used over HCI and a Constant Tone Extension is present in a received packet, the Controller may generate events containing IQ samples of the Constant Tone Extension (see [Vol 4] Part E, Section 7.7.65.21).

### 4.1.8 LE Channel Sounding Test packet trailer

The CS trailer is a sequence of 4 bits, alternating between 0 and 1 bits. The trailer is 1010 (in transmission order) when the most significant bit of the CS Access Address is a 0, and 0101 when the most significant bit of the CS Access Address is a 1.

### 4.1.9 LE Channel Sounding Test packet payload

The CS Test packet payload is defined by the HCI_LE_CS_Test command, [Vol 4] Part E, Section 7.8. The payload length shall be either 32, 64, 96, or 128 bits, defined in [Vol 6] Part H, Section 2.5. There are no associated PDU header or PDU length fields for LE CS Test packet.

### 4.1.10 LE Channel Sounding Test exchanges

In some instances Channel Sounding tests involve exchanges between initiator and reflector devices. In these cases both devices shall operate using active clock accuracy as described in [Vol 6] Part B, Section 4.2.1.

# 5 UNIFIED TEST PROTOCOL

## 5.1 Introduction

The Unified Test Protocol (UTP) is a set of configuration, control, and reporting messages used to perform the RFPHY transmitter and receiver tests for Bluetooth LE devices. The UTP may be used on any of the RFPHY Test mode transports (see Section 6).

UTP messages are sent between the Upper Tester and the IUT across a test control interface. There are three types of messages:

- UTP configuration messages are used to set test parameters at the Upper Tester and the IUT. See Table 5.1 for a list of available configuration messages.
- UTP control messages are sent by the Upper Tester to control the test execution and are used by the IUT to accept or reject requests from the Upper Tester as required. See Table 5.3 and Table 5.4 for a list of available control messages.
- UTP report messages are sent by the IUT to report test results to the Upper Tester. See Table 5.5 for a list of available report messages.

The Lower Tester, Upper Tester, and IUT shall behave as defined by each of the associated roles specified by the test procedure.

Figure 5.1 provides an overview of a typical RFPHY UTP HCI mode scenario showing how transmitter or receiver measurements are performed using UTP.

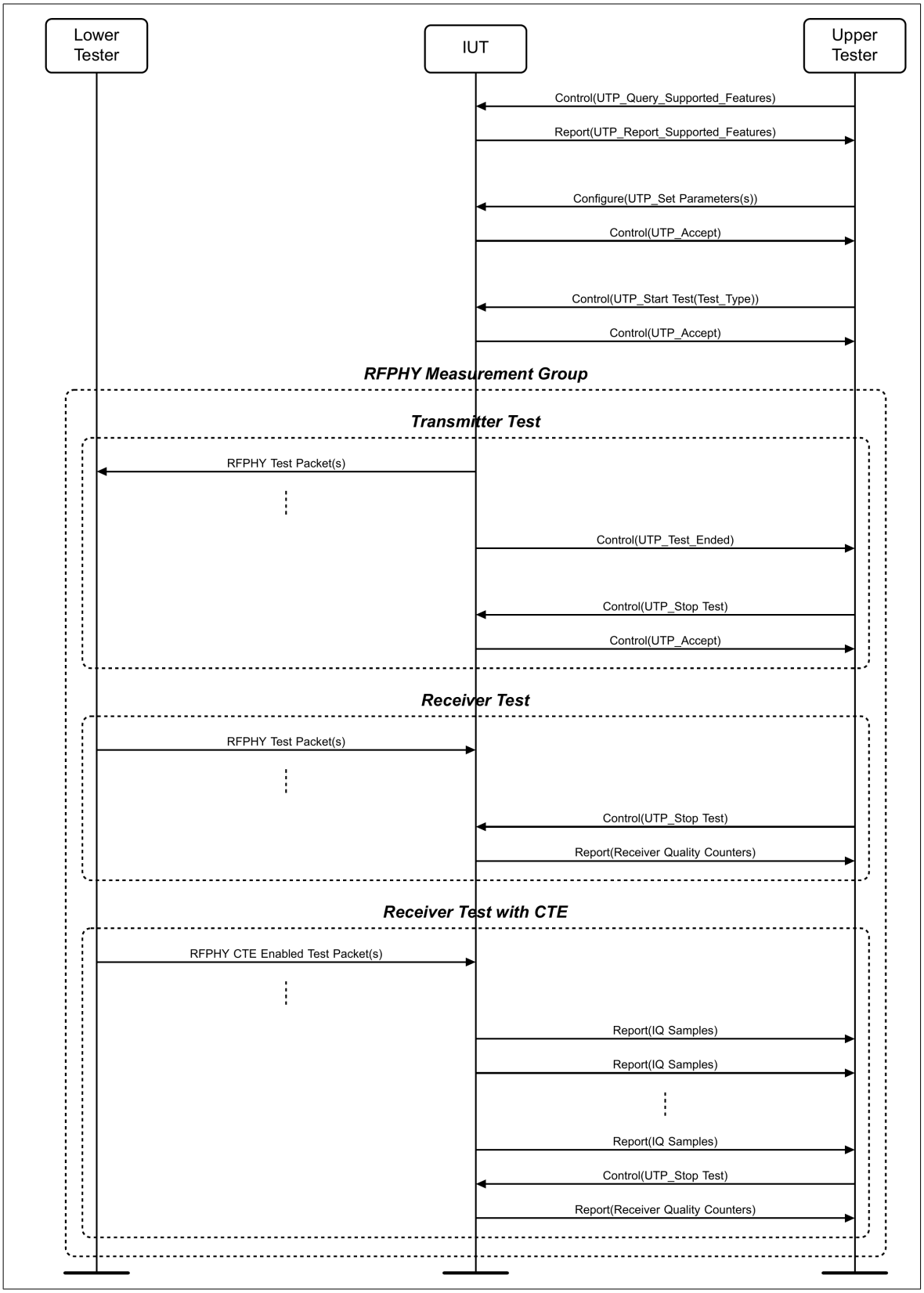



*Figure 5.1: Unified Test Protocol (UTP) overview (example using HCI mode)*

A typical UTP scenario involves the following steps:

1. The Upper Tester queries the UTP supported features of the IUT. The IUT reports the queried UTP supported features back to the Upper Tester.
2. The Upper Tester resets the UTP mode parameters configured at the IUT (optional).
3. The Upper Tester configures the IUT's UTP mode parameters as required for the transmitter or receiver test(s) to be performed. The IUT accepts or rejects each message accordingly.
4. The Upper Tester initiates the transmitter or receiver test(s) by sending a start test control message to the IUT. The IUT accepts the control message.
5. The transmitter or receiver test(s) are performed using the configuration(s) made in step 3.
6. The Upper Tester ends the test by sending a stop test control message to the IUT. For a transmitter test, the IUT accepts the control message. For a receiver test, the IUT sends a UTP Receiver Quality Counters report message to the Upper Tester.

Each device shall process UTP messages, and reply if required, in the order received.

All configuration and control messages sent by the Upper Tester to the IUT shall either be accepted (see Section 5.5.1) or rejected (see Section 5.5.2).

The implementation is transport dependent; see Section 6.

## 5.2 Message format

Each UTP message shall be composed of a Type-Length-Value (TLV) triplet as shown in Figure 5.2.

The Type field indicates the message type (opcode). The Length field specifies the length of the Value field in octets. When the Length field is set to zero, the Value field is not included. An octet with value 0x00 where the Type field would be indicates padding. The message recipient shall ignore this octet and continue parsing any following octet.

| Triplet | | |
|---|---|---|
| Type | Length | Value |
| (1 octet) | (2 octets) | (Length octets) |

*Figure 5.2: TLV format for UTP messages*

Message parameters values other than those specified shall be reserved for future use.

## 5.3 Configuration messages

UTP configuration messages shall be sent by the Upper Tester to set the test parameters used by the IUT. Table 5.1 lists the configuration messages.

| Opcode | Triplet Type | Controller Support |
|---|---|---|
| 0x01 | UTP_Set_RF_Channel | M |
| 0x02 | UTP_Set_Packet_Payload | M |
| 0x03 | UTP_Set_Payload_Length | M |
| 0x04 | UTP_Set_PHY | C.1 |
| 0x05 | UTP_Set_Modulation_Index | C.2 |
| 0x06 | UTP_Set_CTE_Length | C.3 |
| 0x07 | UTP_Set_CTE_Type | C.3 |
| 0x08 | UTP_Set_CTE_Slot_Durations | C.3 |
| 0x0A | UTP_Set_CTE_Antenna_IDs | C.3 |
| 0x0B | UTP_Set_Packet_Count | M |
| 0x0C | UTP_Set_Tx_Power_Level | C.4 |
| 0x0D | UTP_Set_OTA_Exclusion_Period | C.5 |
| 0xFF | UTP_Set_Vendor_Specific_Data | O |

C.1: Mandatory if LE Feature (LE 2M PHY) or LE Feature (LE Coded PHY) is supported, otherwise optional.

C.2: Mandatory if LE Feature (Stable Modulation Index) is supported, otherwise optional.

C.3: Mandatory if LE Feature (Connectionless CTE Transmitter) or LE Feature (Connectionless CTE Receiver) is supported, otherwise optional.

C.4: Mandatory if LE Feature (LE Power Control) is supported, otherwise optional.

C.5: Mandatory if LE Feature (UTP OTA mode) is supported, otherwise optional.

*Table 5.1: Unified Test Protocol configuration messages*

UTP configuration messages apply to both the device's transmitter and the receiver, unless stated otherwise. The Upper Tester sends configuration messages to the IUT that are based on the test scenario(s) to be performed.

The Upper Tester and the IUT shall set the parameters to the default values in Table 5.2 when the UTP transport is initialized or when the UTP_Reset message (see Section 5.4.2) is sent or received.

| Parameter | Value |
|---|---|
| RF_Channel | 0x00 |
| Packet_Payload | 0x02 |
| Payload_Length | 0x0025 |
| PHY | 0x01 |
| Modulation_Index | 0x00 |
| CTE_Length | 0x00 |
| CTE_Type | *any valid value* |
| CTE_Slot_Durations | *any valid value* |
| CTE_Antenna_IDs | *any valid value* |
| Packet_Count | 0x0A |
| Tx_Power_Level | 0x7F |
| OTA_Exclusion_Period | 0x03 |

*Table 5.2: Default parameter values for UTP configuration messages*

### 5.3.1 UTP_Set_RF_Channel

**Description:**

The UTP_Set_RF_Channel message is used to set the RF channel.

**Message Parameters:**

*RF_Channel:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Range: N = 0x00 to 0x4E<br>Frequency used to transmit and receive RFPHY test packets<br>Frequency: (N + 2402) MHz<br>Available frequency range: 2402 MHz to 2480 MHz |

### 5.3.2 UTP_Set_Packet_Payload

**Description:**

The UTP_Set_Packet_Payload message is used to set the packet payload.

**Message Parameters:**

*Packet_Payload:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | PRBS9 sequence '11111111100000111101…' (in transmission order) as described in [Vol 6] Part F, Section 4.1.5 |
| 0x01 | Repeated '11110000' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x02 | Repeated '10101010' (in transmission order) sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x03 | PRBS15 sequence as described in [Vol 6] Part F, Section 4.1.5 |
| 0x04 | Repeated '11111111' (in transmission order) sequence |
| 0x05 | Repeated '00000000' (in transmission order) sequence |
| 0x06 | Repeated '00001111' (in transmission order) sequence |
| 0x07 | Repeated '01010101' (in transmission order) sequence |

### 5.3.3 UTP_Set_Payload_Length

**Description:**

The UTP_Set_Payload_Length message is used to set the payload length.

**Message Parameters:**

*Payload_Length:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length, in octets, of the payload data in each RFPHY test packet<br>Range: 0x0000 to 0x00FF |

### 5.3.4 UTP_Set_PHY

**Description:**

The UTP_Set_PHY message is used to set the PHY for test packets. If OTA mode is being used, the PHY used by the ACL shall not be affected by this message.

**Message Parameters:**

*PHY:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01 | LE 1M PHY |
| 0x02 | LE 2M PHY |
| 0x03 | LE Coded PHY with S=8 data coding |
| 0x04 | LE Coded PHY with S=2 data coding |

### 5.3.5 UTP_Set_Modulation_Index

**Description:**

The UTP_Set_Modulation_Index message is used to set the assumed modulation index at the receiver.

**Message Parameters:**

*Modulation_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Receiver assumes transmitter has a standard modulation index |
| 0x01 | Receiver assumes transmitter has a stable modulation index |

### 5.3.6 UTP_Set_CTE_Length

**Description:**

The UTP_Set_CTE_Length message is used to set the CTE length.

**Message Parameters:**

*CTE_Length:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Do not transmit a Constant Tone Extension, or do not expect a Constant Tone Extension in a receiver test |
| 0x02 to 0x14 | Length of the Constant Tone Extension in 8 µs units |

### 5.3.7 UTP_Set_CTE_Type

**Description:**

The UTP_Set_CTE_Type message is used to set the CTE type of the transmitter or the CTE type expected by the receiver.

**Message Parameters:**

*CTE_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AoA Constant Tone Extension |
| 0x01 | AoD Constant Tone Extension with 1 µs slots |
| 0x02 | AoD Constant Tone Extension with 2 µs slots |

### 5.3.8 UTP_Set_CTE_Slot_Durations

**Description:**

The UTP_Set_CTE_Slot_Durations message is used to set the CTE switching and sampling slot size of the receiver.

**Message Parameters:**

*Slot_Durations:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Switching and sampling slots are 1 µs each |
| 0x01 | Switching and sampling slots are 2 µs each |

### 5.3.9 UTP_Set_CTE_Antenna_IDs

**Description:**

The UTP_Set_CTE_Antenna_IDs message is used to set the Antenna IDs for transmitter and receiver tests in the IUT and the Upper Tester.

**Message Parameters:**

*Antenna_IDs[i]:* *Size: Length x 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | List of Antenna IDs in the pattern |

### 5.3.10 UTP_Set_Packet_Count

**Description:**

The UTP_Set_Packet_Count message is used to set the number of packets sent in transmitter tests by the IUT or sent in receiver tests by the Lower Tester.

**Message Parameters:**

*Packet_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 | Packets transmitted continuously until explicitly stopped by sending the UTP_Stop_Test message (see Section 5.4.4) |
| All other values | Number of test packets to be transmitted during the test |

### 5.3.11 UTP_Set_Tx_Power_Level

**Description:**

The UTP_Set_Tx_Power_Level message is used to set the IUT's transmit power.

**Message Parameters:**

*Tx_Power_Level:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 (min), to 0x14 (max) | Power level to use (signed integer). If this level is not supported, then use the nearest supported level<br>Range: -127 to +20<br>Units: dBm |
| 0x7E | Use the minimum transmit power level the IUT supports |
| 0x7F | Use the maximum transmit power level the IUT supports |

### 5.3.12 UTP_Set_OTA_Exclusion_Period

**Description:**

The UTP_Set_OTA_Exclusion_Period message is used to configure the time from each ACL anchor point until test packets may be sent via the OTA transport.

**Message Parameters:**

*OTA_Exclusion_Period:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x03 to 0xFF | Time after the start of each connection event before the first test packet may be transmitted<br>Unit: 625 µs<br>Range: 0x03 to 0xFF |

### 5.3.13 UTP_Set_Vendor_Specific_Data

**Description:**

The UTP_Set_Vendor_Specific_Data message is used to configure parameters at the IUT that are vendor-specific (e.g., calibration data). The first two data octets shall contain a company identifier from Assigned Numbers. The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.

**Message Parameters:**

*Vendor_Specific_Data:* *Size: Length octets*

| Value | Parameter Description |
|---|---|
| Octets 0 and 1 | Company ID, see Assigned Numbers for Company Identifier |
| Octets 2 to Length-1 | Vendor-specific data |

## 5.4 Control messages sent by the Upper Tester

UTP control messages are sent by the Upper Tester to control the test execution. Table 5.3 lists the control messages.

| Opcode | Triplet Type | Controller Support |
|---|---|---|
| 0x0E | UTP_Query_Supported_Features | M |
| 0x0F | UTP_Reset | M |
| 0x10 | UTP_Start_Test | M |
| 0x11 | UTP_Stop_Test | M |
| 0xFE | UTP_Vendor_Specific_Control | O |

*Table 5.3: Unified Test Protocol control messages sent by the Upper Tester*

### 5.4.1 UTP_Query_Supported_Features

**Description:**

The UTP_Query_Supported_Features message is used to obtain the features supported by the IUT (see Section 5.6.1).

### 5.4.2 UTP_Reset

**Description:**

The UTP_Reset message is used to reset the default UTP mode values in the IUT. See Table 5.2 for the default parameter values.

*Reset_Seq_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Sequential reset number used to identify each individual reset message<br>For every consecutive UTP_Reset sent, Reset_Seq_Number shall be incremented by 1 (modulo 256). |

### 5.4.3 UTP_Start_Test

**Description:**

The UTP_Start_Test message is used by the Upper Tester to start a transmitter or receiver test.

If the set of test parameters specified by the default parameters and any previous configuration messages is not valid for testing (e.g., two parameters have inconsistent values), then the Upper Tester shall reject the UTP_Start_Test message using a UTP_Reject message (see Section 5.5.2) that contains the opcode and error code of the configuration message containing the parameter that caused the error; more than one UTP_Reject message may be sent in the case of inconsistent parameters.

*Test_Type:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Transmitter test.<br>No transmission of RFPHY test packets shall occur until the IUT has sent a UTP_Accept message (see Section 5.5.1). |
| 0x01 | Receiver test.<br>No transmission of RFPHY test packets shall occur until the Upper Tester has received a UTP_Accept message (see Section 5.5.1). |

*Reset_Counters:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Do not reset counters |
| 0x01 | Reset counters |

### 5.4.4 UTP_Stop_Test

**Description:**

The UTP_Stop_Test message is used to stop a test.

- Transmitter test: The *Send_Report* parameter shall be set to 0x00 (do not send the report). The IUT shall respond with an acknowledgment when it completes transmitting the RFPHY test packets. The IUT may send a UTP_Test_Ended command in advance of responding to the UTP_Stop_Test command.
- Receiver test: If the *Send_Report* parameter is set to 0x00 (do not send the report), then the IUT shall respond with an acknowledgment. If the *Send_Report* parameter is set to 0x01 (send the report), then the IUT shall respond with the appropriate measurement-dependent report (see Section 5.6.3).

*Send_Report:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Do not send the report |
| 0x01 | Send the report (applicable to receiver test only) |

### 5.4.5 UTP_Vendor_Specific_Control

**Description:**

The UTP_Vendor_Specific_Control message is used by the Upper Tester to initiate a vendor-specific operation or process at the IUT. The first two data octets shall contain a company identifier from Assigned Numbers. The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.

**Message Parameters:**

*Vendor_Specific_Data:* *Size: Length octets*

| Value | Parameter Description |
|---|---|
| Octets 0 and 1 | Company ID, see Assigned Numbers for Company Identifier |
| Octets 2 to Length-1 | Vendor-specific data |

## 5.5 Control messages sent by the IUT

UTP control messages are sent by the IUT to respond to requests from the Upper Tester. Table 5.4 lists the control messages.

| Opcode | Triplet Type | Controller Support |
|---|---|---|
| 0x12 | UTP_Accept | M |
| 0x13 | UTP_Reject | M |
| 0x14 | UTP_Reset_Accept | M |
| 0x15 | UTP_Test_Ended | M |

*Table 5.4: Unified Test Protocol control messages sent by the IUT*

### 5.5.1 UTP_Accept

**Description:**

The UTP_Accept message is used to indicate that the IUT has correctly processed a message. If a message is rejected by the IUT, then a UTP_Reject message shall be sent instead.

*Opcode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Opcode of the Upper Tester's message |

### 5.5.2 UTP_Reject

**Description:**

The UTP_Reject message is used when the IUT rejects a message. Any message that the IUT does not support, including a parameter value that is reserved for future use, shall be rejected and the Controller shall return the error code *Unsupported Feature or Parameter Value* (0x11).

*Opcode:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Opcode of the Upper Tester's message |

*Error_Code:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason that the received message was rejected. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

### 5.5.3 UTP_Reset_Accept

**Description:**

The UTP_Reset_Accept message is used to individually accept reset messages (see Section 5.4.2).

*Reset_Seq_Number:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Sequential reset number used to identify each individual reset message<br>Reset_Seq_Number shall match the value in the UTP_Reset message being accepted |

### 5.5.4 UTP_Test_Ended

**Description:**

The UTP_Test_Ended message shall be used for Transmitter tests only. This message is used to notify the Upper Tester that the last RFPHY test packet has been transmitted. The message may be sent even though the IUT has not received a UTP_Stop Test command. The Upper Tester shall not acknowledge receipt of the message.

*Packet_Count:* *Size: 4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Number of test packets transmitted during the test |

## 5.6 Report messages

UTP report messages are sent by the IUT to report results back to the Upper Tester. Table 5.5 lists these report messages.

| Opcode | Triplet Type | Controller Support |
|---|---|---|
| 0x16 | UTP_Report_Supported_Features | M |
| 0x17 | UTP_Report_IQ_Samples | C.1 |
| 0x18 | UTP_Report_Receiver_Quality_Counters | M |

| Opcode | Triplet Type | Controller Support |
|---|---|---|
| 0xFD | UTP_Report_Vendor_Specific_Data | O |
| C.1: Mandatory if LE Feature (Connectionless CTE Transmitter) or LE Feature (Connectionless CTE Receiver) is supported, otherwise optional. | | |

*Table 5.5: UTP report messages*

### 5.6.1 UTP_Report_Supported_Features

**Description:**

The UTP_Report_Supported_Features message returns the supported features of the IUT.

**Message Parameters:**

*Supported_Features:* *Size: Length octets*

| Octet | Bit Number | Parameter Description |
|---|---|---|
| 0 | 0 | Extended Packet Error Rate (PER) measurements (measurements in addition to the number of packets received with valid CRC)<br>• Number of packets received with invalid CRC<br>• Number of packets received with payload error<br>• Number of packets received with payload length error |
| | 1 | Bit Error Rate (BER) measurement |

Until the Upper Tester receives a UTP_Report_Supported_Features message from the IUT, the Upper Tester shall assume that the IUT does not support any of these features.

If the message is not long enough to include bits for all the features listed, then the Upper Tester shall assume that the IUT does not support any features beyond those in the last octet in the message.

### 5.6.2 UTP_Report_IQ_Samples

**Description:**

The UTP_Report_IQ_Samples message returns the IQ data from the Constant Tone Extension of a packet received by the IUT to the Upper Tester. IQ samples shall only be reported for packets that contain a CTE.

### Message Parameters:

*Status:* *Size: 1octet*

| Value | Parameter Description |
|---|---|
| 0x00 | IQ Samples collected |
| 0x01 to 0xFF | Error in collecting IQ samples. See [Vol 1] Part F, Controller Error Codes for a list of error codes and descriptions |

*Channel_Index:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 to 0x27 | The physical channel index on which the packet was received, see [Vol 6] Part B, Section 1.4.1 |

*RSSI:* *Size: 2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | RSSI of the packet<br>Range: -1270 to +200<br>Units: 0.1 dBm |

*RSSI_Antenna_ID:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Antenna ID of the antenna on which the RSSI was measured |

*Packet_Status:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | CRC was correct |
| 0x01 | CRC was incorrect and the Length and CTETime fields of the packet were used to determine sampling points |
| 0x02 | CRC was incorrect but the Controller has determined the position and length of the Constant Tone Extension in some other way |
| 0xFF | Insufficient resources to sample |

*Sample_Count:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No samples provided (only permitted if Packet_Status is 0xFF) |
| 0x09 to 0x52 | Total number of sample pairs (there shall be the same number of I samples and Q samples)<br>Note: This number is dependent on the switch and sample slot durations used |

*I_Sample[i]:* *Size: Sample_Count x 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | I sample for the reported PDU (signed integer)<br>The list is in the order of the sampling points within the packet |

*Q_Sample[i]:* *Size: Sample_Count x 1 octet*

| Value | Parameter Description |
|---|---|
| 0x80 | No valid sample available |
| All other values | Q sample for the reported PDU (signed integer)<br>The list is in the order of the sampling points within the packet |

I & Q samples shall be interleaved in the order I[0], Q[0], I[1], Q[1], ..., I[N-1], Q[N-1].

### 5.6.3 UTP_Report_Receiver_Quality_Counters

**Description:**

The UTP_Report_Receiver_Quality_Counters message shall return the supported receiver quality parameters (see Section 5.6.1) from the IUT.

Receiver_Quality_Counters_Config is a bit-field that shall indicate which data is being returned in the message.

Receiver_Quality_Counters_Data shall contain the data items specified by Receiver_Quality_Counters_Config, in ascending order of bit number (i.e., the lowest numbered bit set in Receiver_Quality_Counters_Config corresponds to the first item in Receiver_Quality_Counters_Data).

The Upper Tester shall process the reported receiver quality parameters based on the IUT's reported supported feature set. Invalid counter values reported for unsupported features shall be ignored by the Upper Tester.

**Message Parameters:**

*Receiver_Quality_Counters_Config:* *Size: 1 octet*

| Bit Number | Parameter Description |
|---|---|
| 0 | Number of correctly received packets (with correct CRC).<br>Counter incremented when the CRC matches the packet contents.<br>Size: 4 octets |
| 1 | Number of received packets with invalid CRC.<br>Counter incremented when the CRC does not match the packet contents.<br>Size: 4 octets |
| 2 | Number of received packets with incorrect packet payload.<br>Counter incremented when the packet payload content does not match the Packet_-Payload parameter provided by the Upper Tester (see Section 5.3.2).<br>Size: 4 octets |
| 3 | Number of received packets with incorrect payload length.<br>Counter incremented when the payload length does not match the Payload_Length parameter provided by the Upper Tester (see Section Section 5.3.3).<br>Size: 4 octets |
| 4 | Total number of bit errors in received packets with invalid CRC.<br>Counter is not incremented if either of the Packet_Payload (see Section 5.3.2) or Payload_Length (see Section 5.3.3) parameters do not match those provided by the Upper Tester.<br>If both these parameters match those provided, then the counter increments for every payload bit received in error.<br>Size: 4 octets |

*Receiver_Quality_Counters_Data:* *Size: Length-1 octets*

| Value | Parameter Description |
|---|---|
| Variable | Parameter values for those parameters specified by Receiver_Quality_Counters_Config |

### 5.6.4 UTP_Report_Vendor_Specific_Data

**Description:**

The UTP_Report_Vendor_Specific_Data message returns parameters at the IUT that are vendor-specific (e.g., calibration data). The first two data octets shall contain a company identifier from Assigned Numbers. The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.

### Message Parameters:

*Vendor_Specific_Data:* *Size: Length octets*

| Value | Parameter Description |
|---|---|
| Octets 0 and 1 | Company ID, see Assigned Numbers for Company Identifier |
| Octets 2 to Length-1 | Vendor-specific data |

# 6 UTP TRANSPORT

## 6.1 Introduction

All UTP messages may be concatenated rather than sent individually over any supported transport. UTP messages may be fragmented (e.g., when too large to fit in a single message) and then recombined at the target. The sender decides when to use concatenation or fragmentation. Both can be used at the same time and over any type of transport. A UTP message can be fragmented into any number of parts with a minimum of one octet of data. There is no timeout between the transmission of two parts.

This section describes how the UTP is carried over the following transports:

- 2-wire UART (see Section 6.2)
- HCI (see Section 6.3)
- OTA (see Section 6.4)

## 6.2 2-wire UART

The 2-wire protocol is transported over a serial interface between the Upper Tester and the IUT (see Figure 1.2). See Section 3 for details of the UART test interface used to carry the UTP.

## 6.3 HCI

HCI is transported over a physical interface between the Upper Tester and the IUT's accessible upper HCI interface (see Figure 1.1).

UTP is carried over this HCI transport by the HCI_LE_UTP_Send command (see [Vol 4] Part E, Section 7.8.153) and the HCI_LE_UTP_Receive event (see [Vol 4] Part E, Section 7.7.65.49).

## 6.4 OTA

### 6.4.1 Introduction

UTP OTA mode is an alternative method for performing RFPHY testing without the need for a physical transport interface between the Upper Tester and the IUT (see Figure 1.3).

See Section 5.1 for a description of the UTP mode procedure. The test system shall, in addition to the UTP procedure, perform the following steps when using the OTA transport:

1. UTP OTA mode is enabled at the IUT.
2. The Lower Tester and IUT shall establish an encrypted ACL connection between them. Either the Lower Tester or the IUT can assume the Central role in the connection.
3. The ACL connection shall be terminated when the test session is completed.
4. The IUT shall disable UTP OTA mode when the test session has completed.

The connection shall be maintained between the Lower Tester and the IUT during the entirety of the test session. If the connection is terminated or lost, then the test session shall immediately end.

Throughout the test session, *connSubrateFactor* and *connPeripheralLatency* (see [Vol 6] Part B, Section 4.5.1) shall be set to 1 and 0, respectively.

The RFPHY test packets shall be sent on a static configured frequency (see Section 5.3.1) between connection events (see Section 6.4.2). The number of test packets sent during a connection interval is dependent on the configured connection interval, the test packet PHY, and the test packet length.

### 6.4.2 Connection interval requirements

The number of RFPHY test packets that can be sent in a connection interval depends on the PHY, payload length, DTX, interval between test packets, exclusion period, and ACL connection interval. In addition, the last test packet within a connection interval shall end at least max (625 µs, T_IFS + Window Widening) before the next anchor point.

The connection interval period shall be agreed between the Upper Tester and IUT based on the above parameters and shall be long enough to allow at least one test packet in each connection interval.

The UTP OTA mode timings are shown in Figure 6.1.

The OTA exclusion period is described in Section 5.3.12.

L and I(L) are defined in Section 4.1.6.

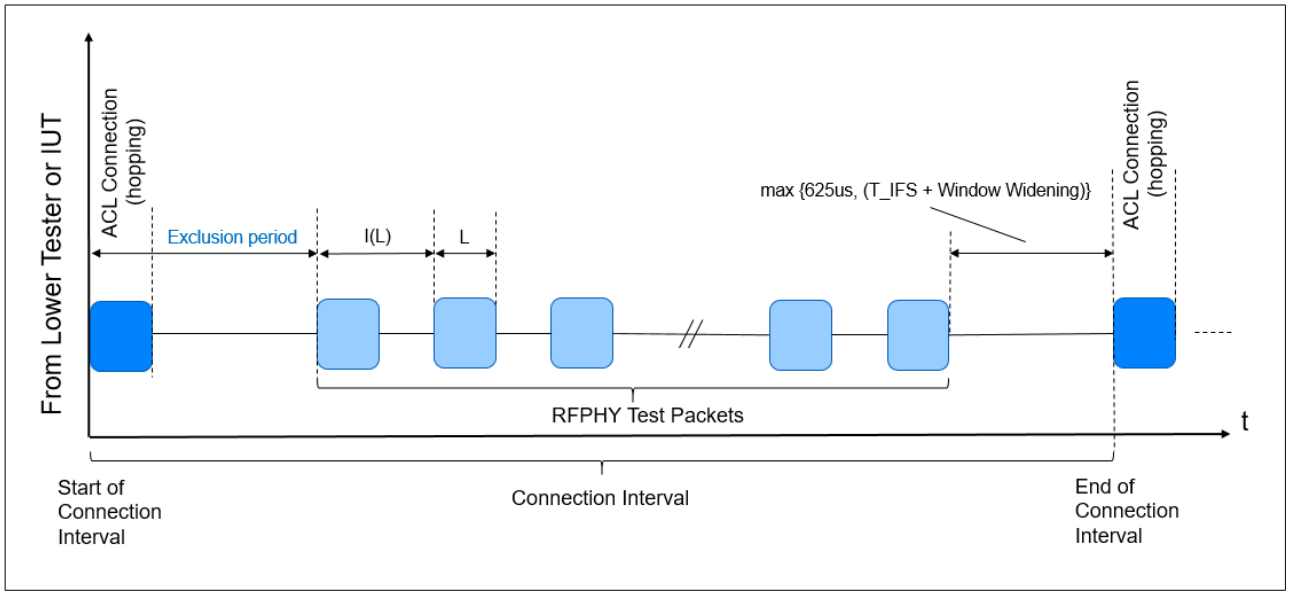



*Figure 6.1: UTP OTA mode connection interval requirements*

### 6.4.3 Transmitter test

The Upper Tester may initiate a transmitter test using the UTP_Start_Test message (see Section 5.4.3). The IUT shall transmit RFPHY test packets to the Lower Tester on the configured frequency for the number of connection interval(s) required to satisfy the configured packet count. The IUT shall stop sending RFPHY test packets in the connection interval in which the configured packet count is reached.

Examples of OTA transmitter tests are shown in Figure 6.2, Figure 6.3, and Figure 6.4.

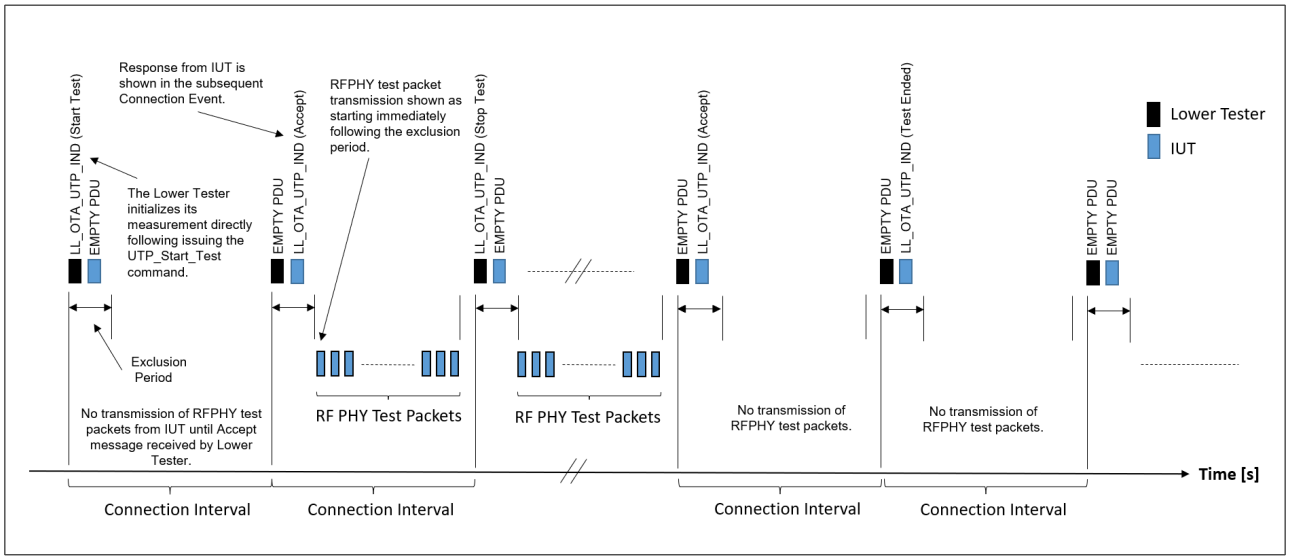



*Figure 6.2: OTA transmitter test example: Lower Tester ends the sequence early, packet count not reached*

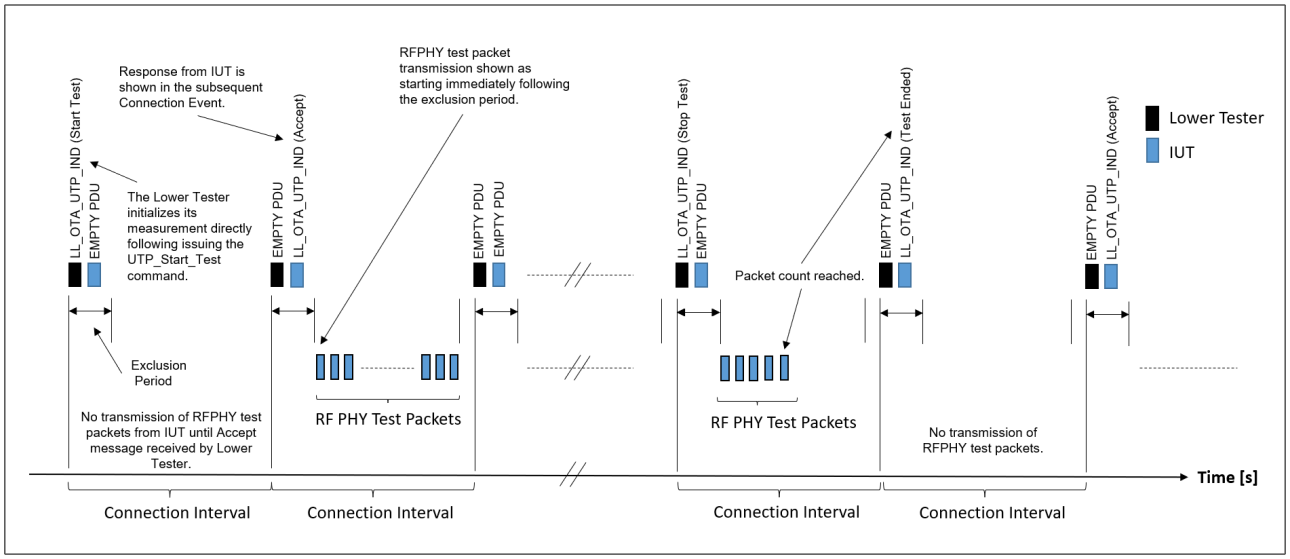



*Figure 6.3: OTA transmitter test example: Lower Tester ends the sequence early, packet count reached*

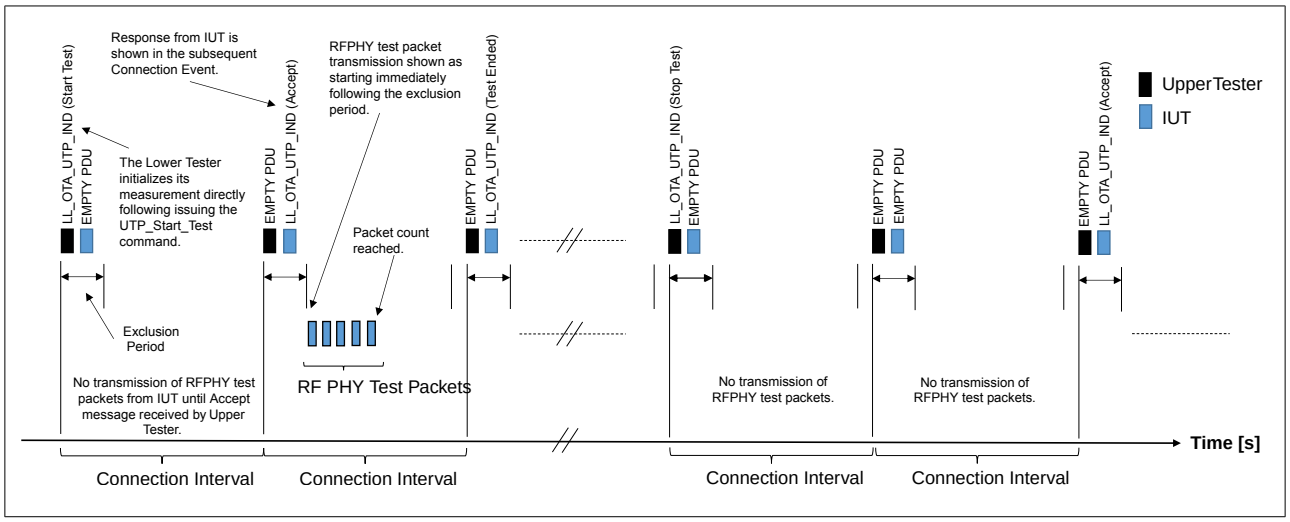



*Figure 6.4: OTA transmitter test example: IUT ends the sequence, packet count reached*

### 6.4.4 Receiver test

The Upper Tester may initiate a receiver test using the UTP_Start_Test message (see Section 5.4.3). In this test, the Lower Tester shall transmit RFPHY test packets on the configured frequency for the number of connection interval(s) required to satisfy the configured packet count. The Lower Tester shall begin the transmission of RFPHY test packets immediately after the end of an exclusion period and shall not omit any transmissions.

An example of an OTA receiver test is shown in Figure 6.5.

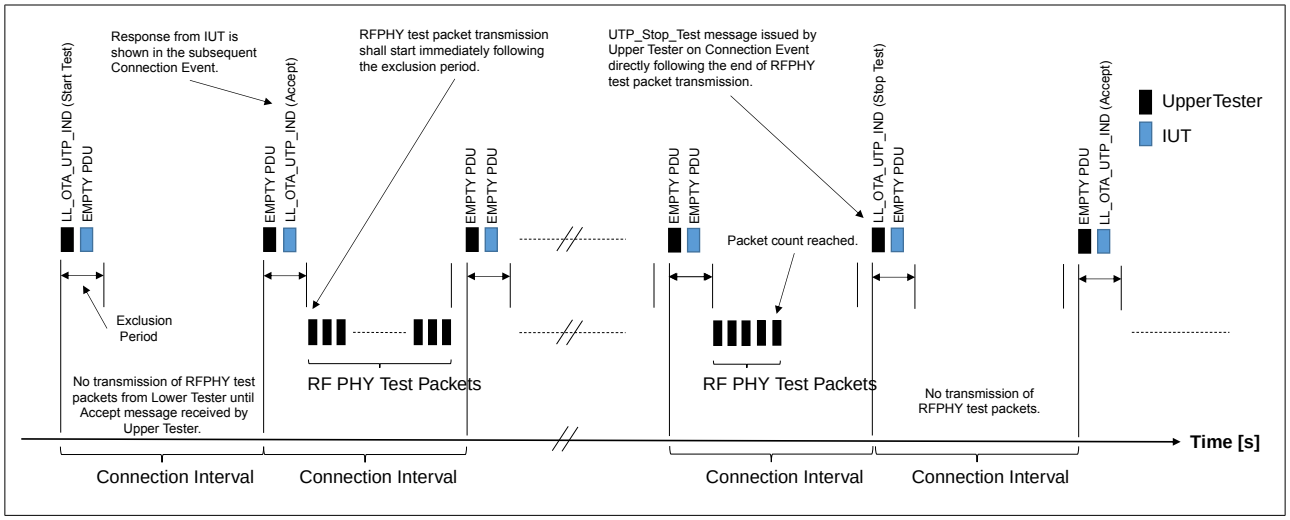



*Figure 6.5: OTA receiver test example*

Low Energy Controller
Part G

# ISOCHRONOUS ADAPTATION LAYER

*This Part describes the Isochronous Adaptation Layer (ISOAL) which supports segmentation and reassembly, and fragmentation and recombination of packets to and from a higher layer.*

# CONTENTS

**1 Introduction ..... 3789**
1.1 Terminology ..... 3789

**2 ISOAL features ..... 3790**
2.1 Unframed PDU ..... 3792
2.2 Framed PDU ..... 3793

**3 Time_Stamp and Time_Offset ..... 3797**
3.1 Time_Offset in framed PDUs ..... 3797
3.2 SDU synchronization reference ..... 3799
3.2.1 SDU synchronization reference using framed PDUs . 3799
3.2.2 SDU synchronization reference using unframed PDUs ..... 3801
3.3 Time Stamp for SDU ..... 3804

**4 SDU Recombination and Reassembly ..... 3806**

# 1 INTRODUCTION

The Isochronous Adaptation Layer (ISOAL) provides segmentation, fragmentation, reassembly and recombination services for conversion of SDUs from the upper layer to PDUs of the Link Layer and vice versa. The ISOAL accepts or generates SDUs, each with a length up to the maximum length (Max_SDU), at a rate that is supported by the Controller. SDUs are transferred to and from the upper layer using either HCI ISO Data packets or over an implementation-specific transport.

Note: SDUs in the ISOAL has no relation to SDUs in the L2CAP layer.

## 1.1 Terminology

The following terms are used in the ISOAL:

| Term | Description |
|---|---|
| Upper layer | The upper layer generates and or receives isochronous data packets known as SDUs. The interface between the upper layer and ISOAL may be the HCI or a proprietary interface. |
| SDU (Service Data Unit) | An SDU is a packet containing isochronous data that is generated or received by the upper layer. |
| PDU (Protocol Data Unit) | A PDU is a data packet that is transmitted or received in a Connected or Broadcast Isochronous Stream. |
| Fragment | An SDU or part of an SDU, resulting from the fragmentation operation. |
| Segment | An SDU or part of an SDU, with a header, resulting from the segmentation operation. |
| Isochronous interval | The time between two consecutive BIS or CIS events (designated ISO_Interval in the Link Layer). |
| Unframed PDU | A PDU that contains a fragment. |
| Framed PDU | A PDU that contains one or more segments. |
| SDU interval | The nominal time between two consecutive SDUs that are sent or received by the upper layer. |
| Max_SDU | Maximum size of an SDU. |
| Synchronization Reference | A time reference of an SDU that allows synchronization of isochronous data in multiple devices. |
| Time_Stamp | Time stamp (32-bit value in microseconds) of a packet. |
| Time_Offset | Time offset (24-bit value in microseconds) of a packet. |
| ISO Data PDU | Either a CIS Data PDU or a BIS Data PDU. |

*Table 1.1: Terminology*

# 2 ISOAL FEATURES

Figure 2.1 shows the architectural diagram of the ISOAL. The multiplexer in Figure 2.1 shows the route of SDUs to either unframed PDU or framed PDU path.

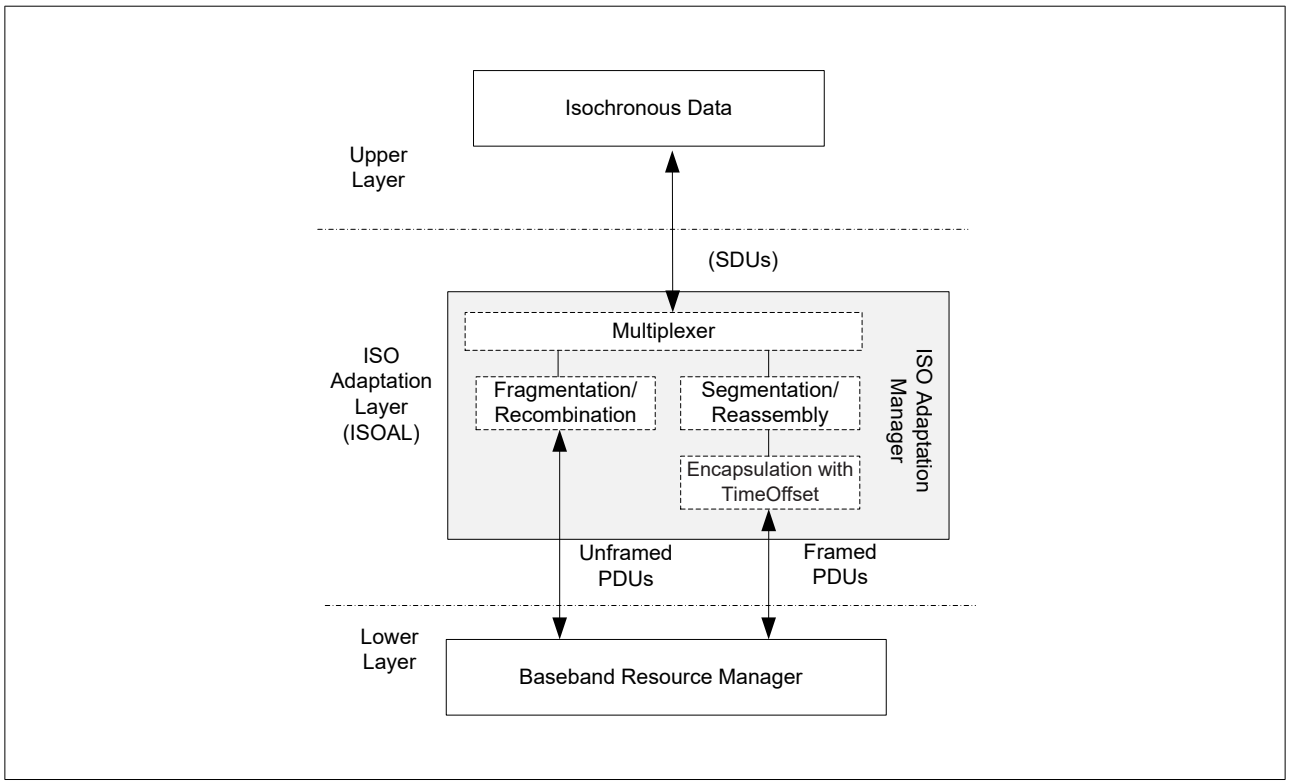



*Figure 2.1: ISOAL architectural blocks*

- Fragmentation and recombination

  The fragmentation process splits an SDU into one or more fragments which are carried by one or more unframed PDUs (see Section 4) in a Connected or Broadcast Isochronous Stream. A fragment shall contain only isochronous data. An SDU with a length less than or equal to the Max_PDU shall be sent in a single unframed PDU. An SDU with a length greater than the Max_PDU shall be sent in multiple fragments in multiple unframed PDUs. The fragmentation process shall use the minimum number of unframed PDUs to transmit the SDU. The recombination process generates an SDU from one or more fragments received in unframed PDUs.

- Segmentation and reassembly

  The segmentation process splits an SDU into one or more segments which are carried by one or more framed PDUs (see Figure 2.1) in a Connected or Broadcast Isochronous Stream. A segment shall contain a Segmentation Header (see Section 3) and may contain Time_Offset and isochronous data. The reassembly process

combines data from one or more segments received in framed PDUs and generates one or more SDUs.

- Encapsulation and Time_Offset

  In the encapsulation process Segmentation Headers and a Time_Offset parameter are added before the first segment of each new SDU allowing timing reconstruction at the peer device (see Section 3.)

Unframed PDUs shall only be used when the ISO_Interval is equal to or an integer multiple of the SDU_Interval and a constant time offset alignment is maintained between the SDU generation and the timing in the isochronous transport. This requires the upper layer to synchronize generation of its data to the effective transport timing. When the Host requests the use of framed PDUs, the Controller shall use framed PDUs.

Framed PDUs are not restricted by the limitations of unframed PDUs and can support any valid combinations of SDU_Interval and ISO_Interval.

SDU_Interval shall be a value, in microseconds, between 0x0000FF and 0x0FFFFF.

When the Host requests a framing packet type and mode, the Controller may use another permitted framing packet type and mode if permitted in Table 2.1.

Note: If the original type and mode cannot be used, then the Controller is not required to use another type and mode.

| **Requested framing packet type and mode** | **Permitted framing packet type and mode** | | |
|---|---|---|---|
| | Unframed | Framed, Segmentable mode | Framed, Unsegmented mode |
| Unframed | Yes | Yes | Yes |
| Framed, Segmentable mode | No | Yes | Yes |
| Framed, Unsegmented mode | No | Yes | Yes |

*Table 2.1: Permitted framing modes in isochronous PDUs*

SDUs sent to the upper layer by the ISOAL shall be given a sequence number which is initialized to 0 when the CIS or BIS is created. SDUs received by the ISOAL from the upper layer shall be given a sequence number which is initialized to 0 when the CIS or BIS is created. The upper layer may synchronize its sequence number with the sequence number in the ISOAL once the Datapath is configured and the link is established.

The sequence number shall be incremented by one for each SDU_Interval, whether or not an SDU was received from the upper layer or included in reports sent to the upper layer.

The sequence number shall be 16 bits and shall be included in the Packet_Sequence_Number parameter in HCI ISO Data Packets.

## 2.1 Unframed PDU

An unframed PDU is an ISO Data PDU; it shall contain payload from the SDU without additional headers in the payload. An unframed PDU shall only contain payload from a single SDU.

There are two types of unframed PDUs. An unframed PDU containing an end fragment or a complete SDU, and an unframed PDU containing a start or a continuation fragment. Unframed PDUs are identified by the LLID field as described below:

LLID in the header of the ISO Data PDU shall be set to 0b00 in the following conditions:

- When the payload of the ISO Data PDU contains the end fragment of an SDU.
- When the payload of the ISO Data PDU contains a complete SDU.

LLID in the header of the ISO Data PDU shall be set to 0b01 in the following conditions:

- When the payload of the ISO Data PDU contains a start or a continuation fragment of an SDU.
- When the ISO Data PDU is used as padding. This is required when the fragments do not add up to the configured number of PDUs specified by the BN parameter per BIS or CIS event.

In the receiving device the payload from a PDU with LLID = 0b00 shall append the payload from PDUs with LLID = 0b01 to derive the length of the payload of the SDU. The SDU can contain some invalid or missing data due to missing or invalid PDUs. All SDUs shall be sent to the upper layer including the indication of validity of data. A report shall be sent to the upper layer if the SDU is completely missing.

Each fragment shall be sent in a new unframed PDU. Multiple fragments shall not be sent in a single PDU.

BN, Max_PDU and ISO_Interval parameters of the Connected or Broadcast Isochronous Stream shall be set such that the bandwidth of the data transmitted by the Link Layer shall be greater than or equal to the bandwidth of data from the upper layer.

The following two additional parameters are defined for unframed PDUs. USPI is the number of SDUs scheduled per ISO_Interval and equals ISO_Interval ÷ SDU_Interval. UPPS is the maximum number of fragments that an SDU is divided into and therefore the number of PDUs allocated to transmit each SDU. Both these parameters are integers.

UPPS shall be at least $\lceil \text{Max_SDU} \div \text{Max_PDU} \rceil$. BN shall equal UPPS × USPI.

Each SDU shall generate UPPS fragments. All these fragments shall be sent to the Link Layer before any fragments of the next SDU. If an SDU generates less than UPPS fragments, empty payloads shall be used to make up the number. Empty payloads shall be PDUs with LLID 0b01 with zero length payload, e.g., when UPPS=4 and only 3 PDUs are generated from the SDU, an additional padding PDU needs to be added.

When an SDU contains zero length data, the corresponding PDU(s) shall be of zero length and the LLID field shall be set to 0b00.

## 2.2 Framed PDU

A framed PDU is an ISO Data PDU where the payload field is made up of segments or is empty. Each segment is encapsulated by a Segmentation Header and, for each first segment of a new SDU, a Time_Offset field is included to allow for reconstruction of the original SDU timing.

Framed PDUs shall be used when the requirements for using unframed PDUs are not met. Framed PDUs support the aggregation of data from multiple SDUs into a single PDU. The maximum allowed drift (MaxDrift) on the average timing of SDU delivery shall not exceed 100 ppm from the configured value of SDU_Interval.

There are two modes for use with framed[1] PDUs: Segmentable mode and Unsegmented mode. Segmentable mode shall be supported; Unsegmented mode may be supported.

In the Segmentable mode, the Controller may segment an SDU over multiple PDUs and a PDU may contain segments from more than one SDU. In the Unsegmented mode, each SDU shall be contained in a single segment in a single PDU; a PDU may contain more than one SDU.

Both modes use the same Segmentation Header format.

In Segmentable mode, the BN, Max_PDU and ISO_Interval parameters of the Connected or Broadcast Isochronous Stream shall be set such that the bandwidth of the data transmitted by the Link Layer shall be greater than or equal to the bandwidth of data from the upper layer plus the amount of space needed for the headers (including an allowance for MaxDrift). This requirement is met if:

$$\text{BN} \times (\text{Max_PDU} - 2) \geq \lceil F \rceil \times 5 + \lceil F \times \text{Max_SDU} \rceil$$

where F = (1 + MaxDrift) × ISO_Interval ÷ SDU_Interval

[1] Before the introduction of Unsegmented mode, “Framed” only referred to Segmentable mode.

The requirement can be met without the inequality being satisfied.

In Unsegmented Mode, the Max_PDU parameter shall be set to $\lceil (\text{ISO_Interval} \div \text{SDU_Interval} \times (1 + \text{MaxDrift}) \div \text{BN}) \rceil \times (5 + \text{Max_SDU})$. For example, 40-octet SDUs generated at 10 ms intervals (with the worst-case drift of 100 ppm) will require a Max_PDU of 45 octets when sent at 7.5 ms ISO_Interval (since there is at most one SDU per PDU, BN will equal 1).

Figure 2.2 shows how multiple segments from multiple SDUs may fit in multiple ISO Data PDUs. Each row in the figure shows the payload field of one PDU. The number and length of segments in the payload shall be limited such that the length of the ISO Data PDU does not exceed the Max_PDU set for the Connected or Broadcast Isochronous Stream.

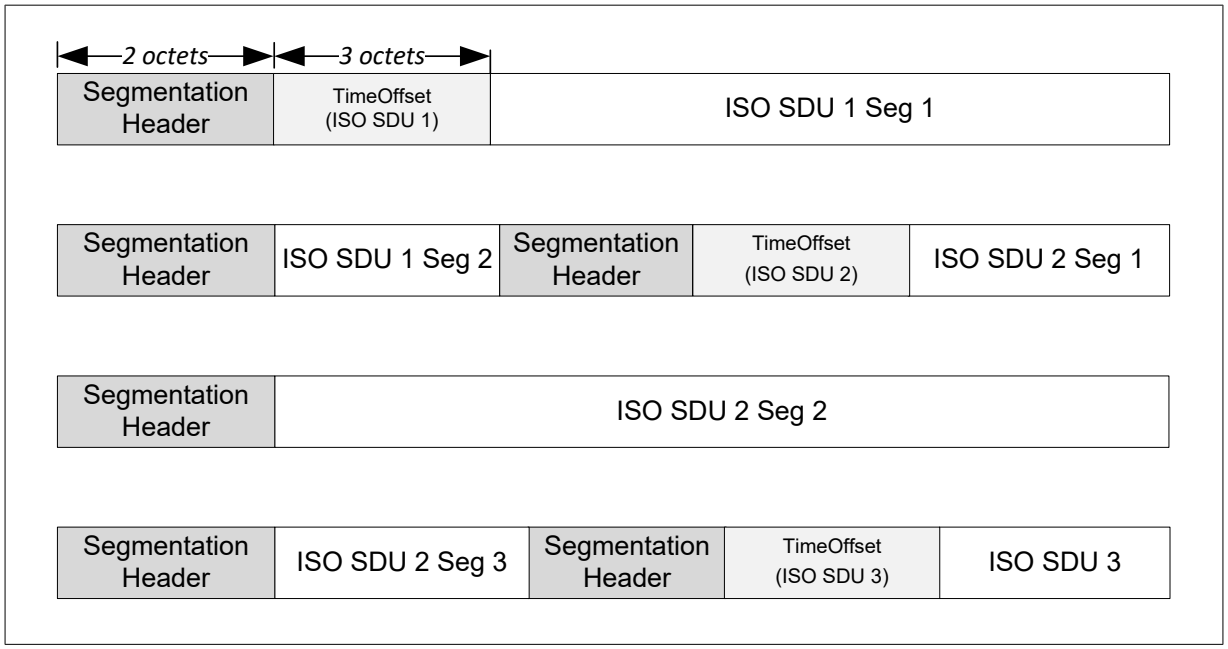



*Figure 2.2: Examples of ISO data PDU payload fields with one or more segments*

The format of a Segmentation Header is shown in Figure 2.3.

| Segmentation Header | | | |
|---|---|---|---|
| **LSB** | | | **MSB** |
| SC<br>(1 bit) | CMPLT<br>(1 bit) | RFU<br>(6 bits) | Length<br>(8 bits) |

*Figure 2.3: Segmentation header format*

The fields in the Header is shown in Table 2.2.

| Field Name | Description |
|---|---|
| SC | The Start or Continuation (SC) field indicates that the data following the Segmentation Header is the start of a new SDU or the continuation of a previous SDU. |
| CMPLT | The Completion (CMPLT) field indicates that the segment following the Segmentation Header in the PDU is the end segment of an SDU. |
| Length | The length field indicates the size, in octets, of the segment that follows the Segmentation Header in this PDU and, when present, includes the Time_Offset parameter. |

*Table 2.2: Segmentation Header fields*

Each framed PDU starts with a Segmentation Header that contains a SC, CMPLT and a Length field. Depending on the SC field, an additional Time_Offset parameter is included between the Segmentation Header and the start of SDU data to indicate the relative timing difference between the SDU and CIG reference point or BIG anchor point timing.

A framed PDU with non-zero length shall contain one or more segments. A framed PDU with zero length can be used as a padding PDU; such a PDU does not have a Segmentation Header. Padding is required when the data does not add up to the configured number of PDUs that are specified by the BN parameter per BIS or CIS event.

The Segmentation Header includes the SC and the CMPLT fields with the following requirements:

- When SC is set to 0, it indicates the start of a new SDU and that the data that follows the header is part of a new SDU. If SC is set to 1, it indicates a continuation of an SDU that was partially transmitted in a previous isochronous PDU.
- When SC is set to 0, the Segmentation Header shall be followed by a Time_Offset field, otherwise the Time_Offset is omitted and the header is directly followed by a segment of an SDU.
- When CMPLT is set to 0, not all data of the SDU has been included in the current PDU. One or more additional framed PDUs are required to complete the SDU transfer. When CMPLT is set to 1, all remaining data of the SDU is included in the segment and the SDU may be transferred to the higher layer.
- CMPLT is independent of the value of SC.

The Segmentation Header shall contain a Length field. The Length field indicates the number of octets of the SDU data segment following the Segmentation Header including the length of Time_Offset when present (see Section 3).

Data from a single SDU shall not be split over multiple segments in a single PDU. Additional Segmentation Headers and data from other SDUs may be added depending

on available remaining octets in that PDU. When an SDU has zero length, the SDU shall be included with a Segmentation Header with SC set to 0, CMPLT set to 1, and the length field set to the length of the Time_Offset. Such an empty SDU shall contain a correct Time_Offset, to allow the higher layer of the device that receives the empty SDU to maintain synchronization.

Because an isochronous transport is not a reliable transport, Segmentation Headers shall integrally be contained in a single PDU. When insufficient octets remain in a framed isochronous PDU to contain the Segmentation Header and the Time_Offset field, no new Segmentation Header can be added, as shown in Table 2.3. When only 5 octets remain, a new Segmentation Header, including the start of the next SDU, may be included but is not recommended.

| SC Start/ continuation | COMPLT Completion | Time_Offset Added | Description |
|---|---|---|---|
| 0 | 0 | Yes | The start of a new SDU, where not all SDU data is included in the current PDU, and additional PDUs are required to complete the SDU. |
| 0 | 1 | Yes | The start of a new SDU that contains the full SDU data in the current PDU. |
| 1 | 0 | No | The continuation of a previous SDU. The SDU payload is appended to the previous data and additional PDUs are required to complete the SDU. |
| 1 | 1 | No | The continuation of a previous SDU. Frame data is appended to previously received SDU data and completes in the current PDU. |

*Table 2.3: Description of Segmentation Header types in framed isochronous PDUs*

A start of a new SDU with SC = 0 shall only be used when the previous frame has completed. A continuation of an SDU with SC = 1 shall only be used if the previous transmitted Segmentation Header had the completion flag set to 0.

When one or more ISO Data PDUs are not received, the receiving device may discard all SDUs affected by the missing PDUs. Any partially received SDU may also be discarded. A report shall be sent to the upper layer for each discarded SDU.

# 3 TIME_STAMP AND TIME_OFFSET

Framed PDUs include a Time_Offset field to allow the reconstruction of the original SDU timing. When using HCI ISO Data packets, a Time_Stamp field may be included. This section defines the usage of these parameters. Every Time_Stamp and Time_Offset shall be derived from a free running reference clock associated with the Controller (the "Controller's clock") that is not affected by adjustments to synchronize with other devices, such as a Peripheral synchronizing to a received packet from the Central (see [Vol 6] Part B, Section 4.5.7) or an adjustment done as part of the Piconet Clock Adjustment feature (see [Vol 2] Part B, Section 8.6.10).

## 3.1 Time_Offset in framed PDUs

At the transmitter, the Time_Offset is measured from the reference time of the SDU at the source to the CIG Reference point or BIG anchor point of the corresponding CIG or BIG event of the isochronous payload containing the first Segmentation Header of that SDU. This computation ignores any potential retransmissions or missed subevents, resulting in the time of transmission under perfect link conditions. The Time_Offset shall be a positive value.

The reference time of the SDU is determined based on the local timing in the Controller based on either a time stamp of that SDU (e.g., provided via the Time_Stamp parameter in the HCI ISO Data packet) or another mechanism used by the implementation tracking the SDU timing to the transport timing.

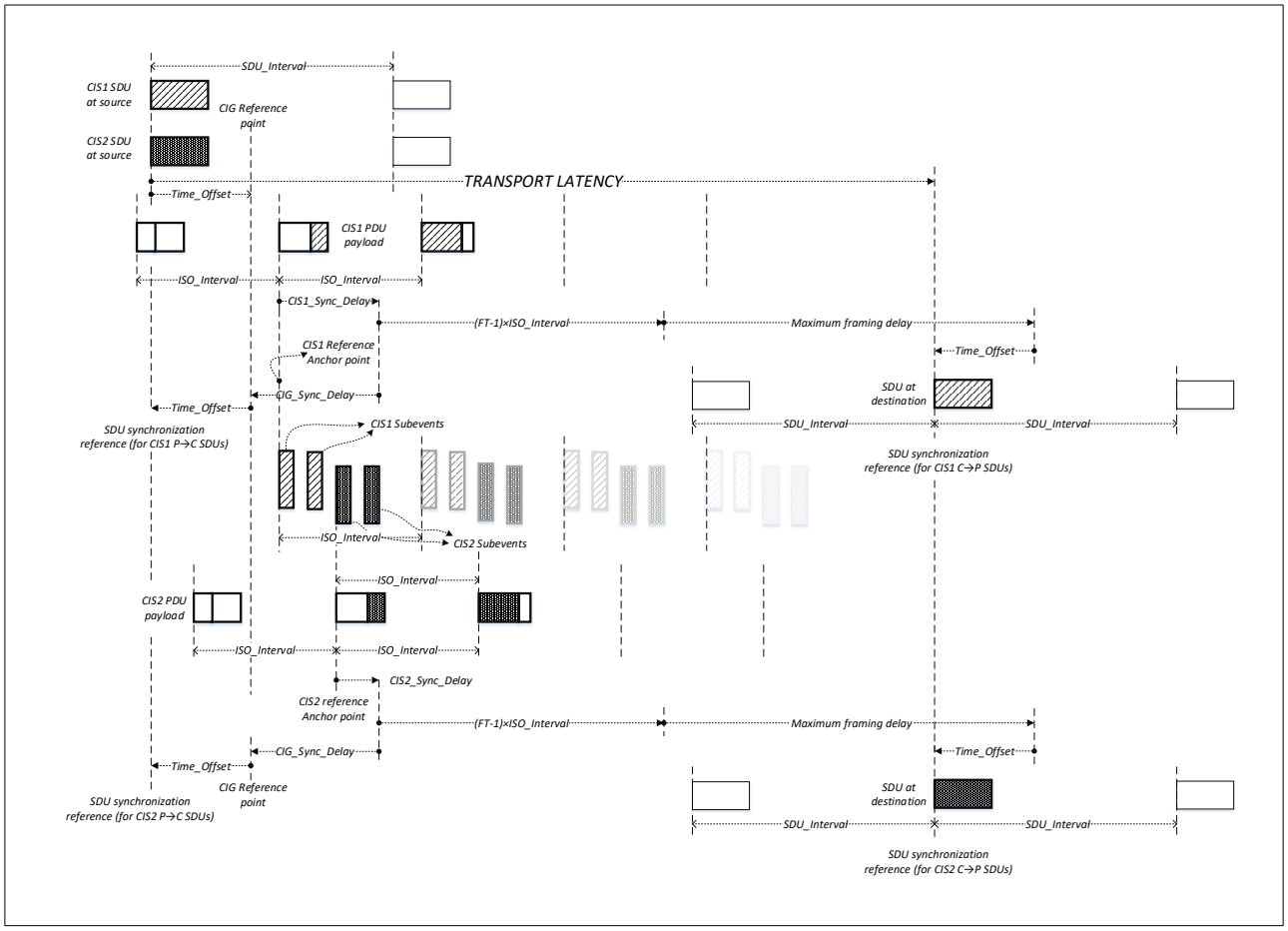



*Figure 3.1: SDU synchronization reference using the Time_Offset parameter*

For details on the computation of the SDU synchronization reference and transport latency for all options of framed and unframed PDU in CIS and BIS, see Section 3.2.

At the receiver, the CIS reference anchor point is computed ignoring any retransmissions or missed subevents, resulting in the time of transmission under perfect link conditions. Similar calculations can be performed for BIG.

Any potential delays in effective transmission shall be omitted in the calculation of Time_Offset.

When BN is greater than 1, all PDUs belonging to the same CIS or BIG event use the same reference anchor point.

For example, in a CIS with BN=2 and FT=10, payload number 720 can have its first transmission potentially in CIS event 360 and its last potential transmission in CIS event 369. Even when the transfer effectively takes place in CIS event 365, the CIS reference anchor point for this payload is the anchor point of CIS event 360.

The Controller transmitting an SDU may use any of the following methods to determine the value of the SDU reference time, indicated in Figure 3.1, to define the Time_Offset for that SDU:

- A captured timestamp of the SDU
- A timestamp provided by the higher layer
- A computed timestamp based on a sequence counter provided by the higher layer
- Any other method of determining Time_Offset

## 3.2 SDU synchronization reference

This section describes the method of managing the SDU synchronization reference in framed and unframed PDUs.

### 3.2.1 SDU synchronization reference using framed PDUs

Using framed PDUs introduces additional delay in transferring SDUs. This additional delay shall be included when computing the SDU synchronization reference.

When using framed PDUs, the additional delay caused by the segmentation in a system where SDU creation is not synchronized to the transport timing equals one SDU interval plus one Isochronous interval.

The transport latency is the duration between the CIG reference point (or BIG anchor point) of the first possible CIG or BIG event containing the payload and the CIG or BIG synchronization point of the last possible CIG or BIG event containing the payload, plus the maximum framing delay[1] (shown in Figure 3.1 for CIG). Transport latency is measured from the reference time of the SDU to its SDU_Synchronization_Reference and does not include any implementation specific processing times or internal transport delays (e.g., delays over the HCI transport or PDU processing times).

The maximum framing delay is one isochronous interval plus one SDU interval in Segmentable mode and is one isochronous interval in Unsegmented mode. In the following formulae, Framing_Delay_C, Framing_Delay_P, and Framing_Delay_B equal SDU_Interval_C_To_P, SDU_Interval_P_To_C, and SDU_Interval respectively in Segmentable mode and all equal zero in Unsegmented mode.

The transport latency for a CIG is the actual latency of transmitting payloads of all CISes in the CIG, and is calculated as:

Transport_Latency_C_To_P = CIG_Sync_Delay + (FT_C_To_P) × ISO_Interval + Framing_Delay_C

[1]This parameter was named “segmentation delay” in earlier specification versions.

Transport_Latency_P_To_C = CIG_Sync_Delay + (FT_P_To_C) × ISO_Interval + Framing_Delay_P

The transport latency for a BIG is the actual latency of transmitting payloads of all BISes in the BIG, and is calculated as:

Transport_Latency_BIG = BIG_Sync_Delay + PTO × (NSE ÷ BN – IRC) × ISO_Interval + ISO_Interval + Framing_Delay_B

The calculated transport latencies shall be less than or equal to those set by the Host.

Each SDU is given a synchronization reference based on the Reference Anchor point.

The CIS reference anchor point is computed ignoring any retransmissions or missed subevents and shall be set to the anchor point of the CIS event in which the first PDU containing the SDU could have been transferred.

The BIG reference anchor point is the anchor point of the BIG event that the PDU is associated with (see [Vol 6] Part B, Section 4.4.6.6).

For the Central to Peripheral direction or from the Isochronous Broadcaster to Synchronized Receiver direction, the SDU synchronization point represents the absolute time where the data from a CIG or BIG should be available, and can be used to synchronize between multiple devices.

For SDUs sent from the Central to the Peripheral in a CIS using framed PDUs the SDU_Synchronization_Reference shall be calculated as follows:

SDU_Synchronization_Reference = CIS Reference Anchor point + CIS_Sync_Delay + Framing_Delay_C + FT_C_To_P × ISO_Interval – Time_Offset

For the Peripheral to Central direction, the SDU synchronization point represents the absolute reference time of the origin of an SDU and can be used to synchronize data coming from multiple sources. This point is at the start of the CIG reference point. It could be earlier for framed PDUs.

For SDUs sent from the Peripheral to the Central in CIS using framed PDUs, the SDU_Synchronization_Reference shall be calculated as follows:

SDU_Synchronization_Reference = CIS reference anchor point + CIS_Sync_Delay – CIG_Sync_Delay – Time_Offset

For SDUs received in a BIS using framed PDUs, the SDU_Synchronization_Reference shall be calculated as follows:

SDU_Synchronization_Reference = BIG reference anchor point + BIG_Sync_Delay + ISO_Interval + Framing_Delay_B – Time_Offset

For example, when Segmentable mode is in use, the SDU interval is 10.6 ms, and the isochronous interval is 10 ms, an additional 20.6 ms is added due to segmentation, while if Unsegmented mode is in use, only 10 ms is added.

This example clarifies the usage of Time_Offset in the Central to Peripheral direction of CIS: at the source an SDU has a reference time X at the Controller's clock. The start of the CIS event of the first potential transmission of the PDU that can contain the first part of that SDU is 6.123 ms later in time. The Time_Offset for the initial segment for that SDU will contain the value 6123 decimal or 0x0017EB. During the exchange over the isochronous transport, the PDU was not transmitted at the first possible opportunity, rather the PDU was received by the peer in the $p^{th}$ subevent n events later due to actual conditions of physical transport and potential Controller behavior such as scheduling conflicts at a subevent with starting time Y. Time Y is captured at the Controller's clock of the receiving device.

The receiving device computes the CIS reference anchor point time Y1 as follows:

$$Y1 = Y – (p–1) \times Sub_Interval – n \times ISO_Interval$$

The SDU synchronization reference for the SDU known as Y2 in µs is computed as:

$$Y2 = Y1 + CIS_Sync_Delay + SDU_Interval_C_To_P + (FT_C_To_P \times ISO_Interval) – 6123$$

The SDU may be exchanged over multiple PDUs, but the synchronization reference point shall only be computed based on the PDU timing of the PDU containing the first Segmentation Header of the applicable SDU.

### 3.2.2 SDU synchronization reference using unframed PDUs

Using unframed PDUs does not introduce additional delay when the SDU interval equals the isochronous interval. When the Isochronous interval is larger than the SDU interval, multiple SDUs are received per BIS or CIS event which increases the delay. This additional delay shall be included when computing the SDU_Synchronization_Reference.

The transport latency is the duration between the CIG reference point (or BIG anchor point) of the first possible CIG or BIG event containing the payload and the CIG or BIG synchronization point of the last possible CIG or BIG event containing the payload, plus the additional delay when grouping multiple SDUs in a single ISO_Interval (as shown in Figure 3.2 for CIG). Transport latency is measured from the reference time of the SDU to its SDU_Synchronization_Reference and does not include any implementation specific processing times or internal transport delays (e.g., delays over the HCI transport or PDU processing times).

The Transport_Latency for a CIG is the actual latency of transmitting payloads of all CISes in the CIG, and is calculated as:

> Transport_Latency_C_To_P = CIG_Sync_Delay + (FT_C_To_P – 1) × ISO_Interval + (USPI_C_To_P – 1) × SDU_Interval_C_To_P
>
> Transport_Latency_P_To_C = CIG_Sync_Delay + (FT_P_To_C – 1) × ISO_Interval + (USPI_P_To_C – 1) × SDU_Interval_P_To_C

where USPI_C_To_P and USPI_P_To_C are the values of USPI for the two directions.

These calculations can be simplified as follows:

> Transport_Latency_C_To_P = CIG_Sync_Delay + FT_C_To_P × ISO_Interval – SDU_Interval_C_To_P
>
> Transport_Latency_P_To_C = CIG_Sync_Delay + FT_P_To_C × ISO_Interval – SDU_Interval_P_To_C

The Transport_Latency for a BIG is the actual latency of transmitting payloads of all BISes in the BIG, and is calculated as:

> Transport_Latency = BIG_Sync_Delay + PTO × (NSE ÷ BN – IRC) × ISO_Interval + (USPI – 1) × SDU_Interval

This calculation can be simplified as follows:

> Transport_Latency = BIG_Sync_Delay + (PTO × (NSE ÷ BN – IRC) + 1) × ISO_Interval – SDU_Interval

For example, when the SDU interval is 10 ms and the Isochronous interval is 20 ms an additional 10 ms is added due to combination of 2 SDUs in one BIS or CIS event.

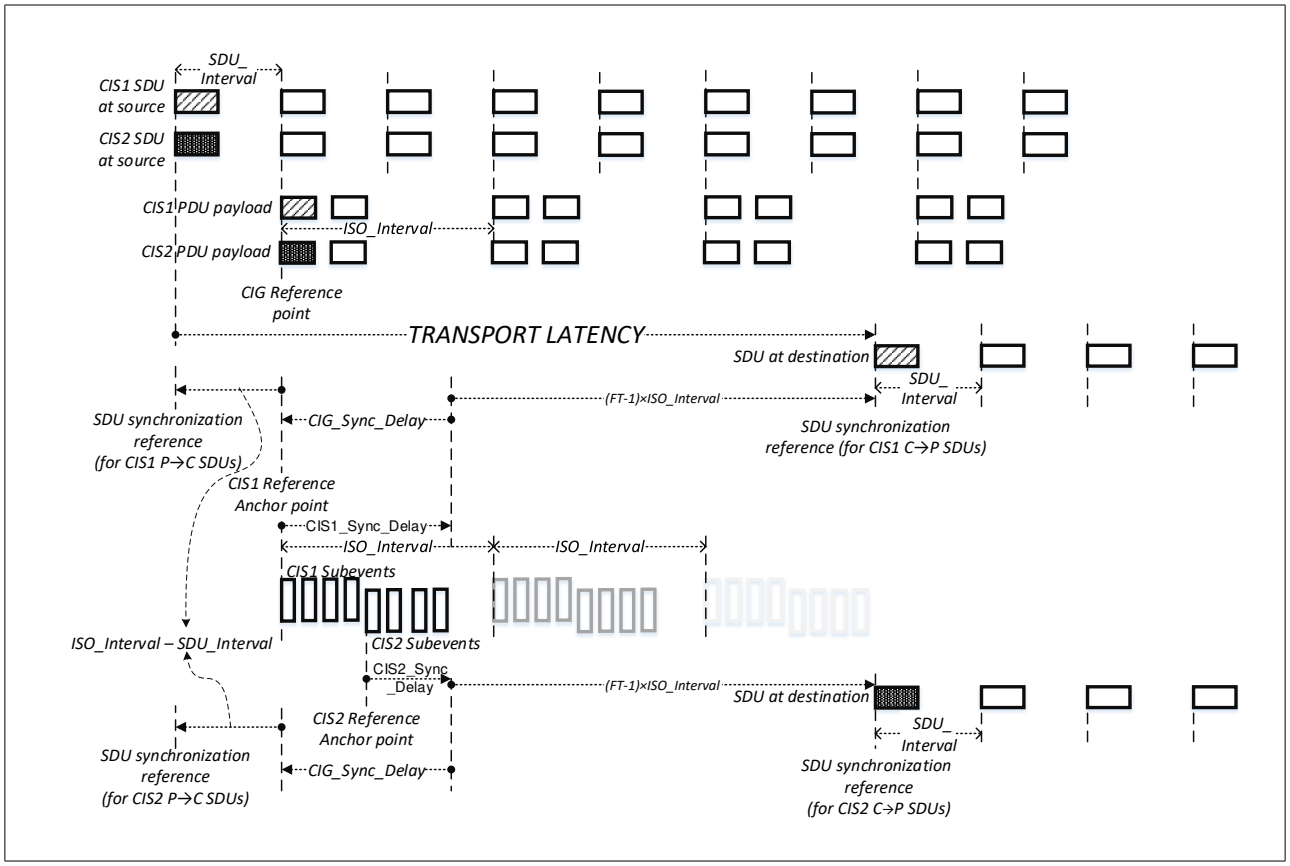



*Figure 3.2: SDU synchronization reference using the unframed PDUs*

Each SDU is given a synchronization reference point based on the reference anchor point.

The CIS reference anchor point is computed ignoring any retransmissions or missed subevents and shall be set to the start of the CIS event in which the first PDU containing the SDU could have been transferred.

The BIG reference anchor point is the anchor point of the BIG event that the PDU is associated with (see [Vol 6] Part B, Section 4.4.6.6).

For the Central to Peripheral direction or from the Isochronous Broadcaster to Synchronized Receiver direction, the SDU synchronization point represents the absolute time where the data from a CIG or BIG should be available, and can be used to synchronize between multiple devices.

For SDUs sent from the Central to the Peripheral in CIS using unframed PDUs the SDU_Synchronization_Reference for the first SDU received in a burst of PDUs shall be calculated as follows:

SDU_Synchronization_Reference = CIS reference anchor point + CIS_Sync_Delay + (FT_C_To_P – 1) × ISO_Interval

For the Peripheral to Central direction, the SDU synchronization point represents the absolute reference time of the origin of an SDU and can be used to synchronize data coming from multiple sources. This point is at the start of the CIG reference point.

For SDUs sent from the Peripheral to the Central in CIS using unframed PDUs, the SDU_Synchronization_Reference for the first SDU received in a burst of PDUs shall be calculated as follows:

> SDU_Synchronization_Reference = CIS reference anchor point + CIS_Sync_Delay – CIG_Sync_Delay + SDU_Interval – ISO_Interval

For SDUs received in BIS using unframed PDUs, the SDU_Synchronization_Reference shall be calculated as follows:

> SDU_Synchronization_Reference = BIG reference anchor point + BIG_Sync_Delay

All PDUs belonging to a burst as defined by the configuration of BN have the same reference anchor point. When multiple SDUs have the same reference anchor point, the first SDU uses the reference anchor point timing. Each subsequent SDU increases the SDU_Synchronization_Reference timing by one SDU interval.

## 3.3 Time Stamp for SDU

When the Time_Stamp field is included in an HCI ISO Data packet from the Controller to the Host, the value of Time_Stamp is set to the synchronization reference for the SDU, as defined in Section 3.2, and is based on the CIS or BIG reference anchor point of the Controller's clock. The Controller should include a Time_Stamp.

The Host may determine the CIG reference point or BIG anchor point for the last transmitted SDU from the TX_Time_Stamp and Time_Offset return parameters of the HCI_LE_Read_ISO_TX_Sync command, as shown in Figure 3.1. It may then use this information when providing the Time_Stamp in future HCI ISO Data packets. For each HCI ISO Data packet where the Host can provide a valid Time_Stamp value, it should include a Time_Stamp in those HCI ISO Data packets that it sends.

When an HCI ISO Data packet sent by the Host does not contain a Time_Stamp or the Time_Stamp value is not based on the Controller's clock, the Controller should determine the CIS or BIS event to be used to transmit the SDU contained in that packet based on the time of arrival of that packet.

In each direction, in a given packet, the Time_Stamp provided (if any) shall be one that applies to the SDU corresponding to the Packet_Sequence_Number.

When using the Time_Stamp or another suitable method for synchronization, both the higher layer and Controller should implement a mechanism to compensate potential

clock drift and jitter. These mechanisms should include long term drift compensation and tolerances on the synchronization method used.

The Controller or the higher layer sending or receiving an SDU may use any of the following methods to transfer the synchronization reference of the SDU to the higher layer:

- Timestamps at the Controller’s clock
- Any other signaling method suitable to the higher layer

# 4 SDU RECOMBINATION AND REASSEMBLY

The ISOAL shall reconstruct SDUs from the received framed PDUs and pass all correct SDUs to the upper layer.

As an isochronous transport is not a reliable transport, errors can occur in the transfer of PDUs. Upon packet loss or erroneous reception, multiple SDUs can be affected. The number of affected SDUs will be (new synchronization reference – last valid SDU synchronization reference) ÷ SDU_Interval, rounded to the nearest integer, unless clock drift on the Controller's clock in this period is greater than SDU_Interval.

Each SDU shall be reported to the upper layer. SDUs that are completely missing shall be reported as "lost data". The following SDUs shall either be discarded and reported as “lost data” or be reported as “data with possible errors”:

- SDUs with violations in the Segmentation Headers of framed PDUs.
- SDUs with a length exceeding Max_SDU. In this case, the length of the SDU reported to the upper layer shall not exceed the Max_SDU length. The SDU shall be truncated to Max_SDU octets.
- SDUs with missing isochronous data due to loss of PDUs or only containing empty data PDUs with LLID=0b01.
- SDUs with unreliable isochronous data due to CRC errors in PDUs.
- Unframed SDUs without exactly one fragment with LLID=0b00.
- Unframed SDUs where the fragment with LLID=0b00 is followed by a fragment with LLID=0b01 and containing at least one octet of data.
- PDUs on unframed links with LLID=0b10.
- PDUs on framed links with LLID=0b00 or LLID=0b01.

Low Energy Controller
Part H

# CHANNEL SOUNDING

# CONTENTS

**1 Channel Sounding physical channels ..... 3811**

**2 Packet formats for Channel Sounding ..... 3812**
2.1 Preamble ..... 3812
2.2 Channel Sounding Access Address ..... 3813
2.2.1 Channel Sounding Access Address selection rules .. 3814
2.2.2 Channel Sounding Access Address checking ..... 3814
2.3 Trailer ..... 3815
2.4 Sounding sequence ..... 3815
2.5 Random sequence ..... 3816
2.6 Channel Sounding extended packet formats ..... 3817

**3 Channel Sounding bit stream processing ..... 3819**
3.1 Measuring RTT ..... 3819
3.1.1 Reference receiver for round-trip time measurements ..... 3820
3.1.2 ToD and ToA reporting accuracy ..... 3821
3.2 Timing estimate based on an Access Address ..... 3822
3.2.1 Timestamps using a native clock ..... 3822
3.2.2 Timing estimate based on a pseudo-noise bit sequence ..... 3823
3.3 Fractional timing estimate based on a sounding sequence ..... 3823
3.3.1 Phase-based PCT estimate based on a sounding sequence ..... 3824
3.3.1.1 Reference receiver for phase-based ranging from a sounding sequence ..... 3827
3.3.1.2 Accuracy requirements ..... 3828
3.4 Fractional timing estimate based on a random sequence ..... 3829
3.5 Attack detection requirements ..... 3829
3.5.1 Normalized attack detector metric ..... 3829
3.5.2 Reference signal modulated with h=0.5 GFSK ..... 3831
3.5.3 Early commit attacks with phase based detection ..... 3832
3.5.3.1 Sounding sequence attack signal definition ..... 3832
3.5.3.2 Random sequence attack signal definition 3835
3.5.3.3 Raw attack detector metric based on a sounding sequence or a random sequence ..... 3836

3.5.3.4 Phase-based attack detection requirements for RTT with sounding sequence and random sequence packets . 3839
3.5.4 Amplitude-based attacks with DFT detection ..... 3840
3.5.4.1 Amplitude-based attack signal definition .... 3840
3.5.4.2 Raw attack detector metric based on a DFT ..... 3843
3.5.4.3 Amplitude-based attack detection requirements for RTT with sounding sequence and random sequence ..... 3844

**4 Channel Sounding interface protocol ..... 3853**
4.1 Channel selection Algorithm #3 ..... 3853
4.1.1 Conventions ..... 3853
4.1.2 Channel index shuffling function cr1 ..... 3853
4.1.3 Channel selection Algorithm #3a for mode-0 steps ... 3854
4.1.4 Channel index selection for non-mode-0 steps ..... 3855
4.1.4.1 Channel Selection Algorithm #3b ..... 3855
4.1.4.2 Channel Selection Algorithm #3c ..... 3855
4.2 Channel Sounding channel indices ..... 3866
4.3 Channel Sounding steps ..... 3867
4.3.1 Channel Sounding step mode-0 ..... 3868
4.3.2 Channel Sounding step mode-1 ..... 3870
4.3.3 Channel Sounding step mode-2 ..... 3871
4.3.4 Channel Sounding step mode-3 ..... 3873
4.4 Channel Sounding subevent and mode sequencing ..... 3876
4.4.1 Tone extension slots ..... 3876
4.4.2 CS subevent structure ..... 3878
4.4.3 Sub_Mode insertion ..... 3879
4.4.4 Main_mode repetition ..... 3879
4.4.5 Channel Sounding procedure and procedure repeat desynchronization ..... 3880
4.5 Timing of steps ..... 3881
4.6 Phase measurements during T_PM ..... 3882
4.7 Phase measurements with antenna switching ..... 3884
4.7.1 Antenna switching in the 1:1 configuration ..... 3884
4.7.2 Antenna switching in the N_AP:1 configuration ..... 3884
4.7.3 Antenna switching in the 1:N_AP configuration ..... 3886
4.7.4 Antenna switching in the 2:2 (N_AP = 4) configuration ..... 3887
4.7.5 Antenna path permutations ..... 3889
4.8 Channel Sounding random bit generation ..... 3892

4.8.1 Channel Sounding random number generation function hr1 ..... 3893

# 1 CHANNEL SOUNDING PHYSICAL CHANNELS

Channel Sounding (CS) defines 79 RF channels in the 2.4 GHz ISM band and defines a new channel index for each of these channels. Table 1.1 shows the relationship between CS channel indices and RF center frequency. Table 1.1 also shows some of the CS channel indices are not allowed for CS communication.

| CS Channel Index | RF Center Frequency | Allowed |
|---|---|---|
| 0 | 2402 MHz | No |
| 1 | 2403 MHz | No |
| 2 | 2404 MHz | Yes |
| … | … | … |
| 22 | 2424 MHz | Yes |
| 23 | 2425 MHz | No |
| 24 | 2426 MHz | No |
| 25 | 2427 MHz | No |
| 26 | 2428 MHz | Yes |
| … | … | … |
| 76 | 2478 MHz | Yes |
| 77 | 2479 MHz | No |
| 78 | 2480 MHz | No |

*Table 1.1: Mapping of CS channel indices to RF physical channels*

# 2 PACKET FORMATS FOR CHANNEL SOUNDING

CS uses a specific modulated bit sequence known as CS_SYNC. The CS_SYNC packet format is similar to a packet format for LE Uncoded PHY, except that the CS_SYNC packet has no PDU, CRC, or CTE fields. The format of the CS_SYNC packet is shown in Figure 2.1.

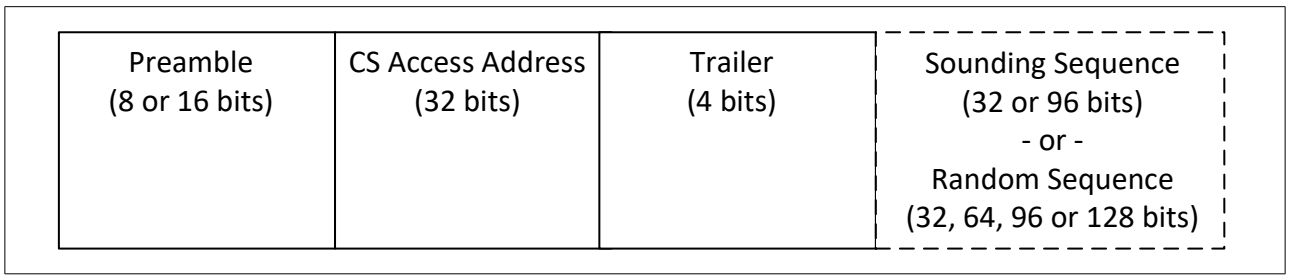



*Figure 2.1: CS_SYNC packet format*

The preamble is 8 bits when transmitting or receiving on the LE 1M PHY and 16 bits when transmitting or receiving on the LE 2M and the LE 2M 2BT PHYs. The CS Access Address is 32 bits. The trailer is 4 bits. The Sounding Sequence and Random Sequence fields are optional. If present, the Sounding Sequence field shall be of length 32 or 96 bits. If the Random Sequence field is present, it shall be of length 32, 64, 96, or 128 bits.

The variable length of the optional Sounding Sequence and Random Sequence fields allows an implementation to optimize the total packet length versus the correlation accuracy of the field. CS packets with no Sounding Sequence or Random Sequence fields take 44 µs to transmit when sent using the LE 1M PHY and 26 µs to transmit when using the LE 2M and the LE 2M 2BT PHYs. CS packets that include a Sounding Sequence or Random Sequence field take longer to transmit based on the length of the field and the PHY selection.

## 2.1 Preamble

The same rules defined for the preamble of the LE Uncoded PHY packet format as described in [Vol 6] Part B, Section 2.1.1 shall apply to the preamble of the CS packet format. The preamble for CS_SYNC is shown in Figure 2.2.

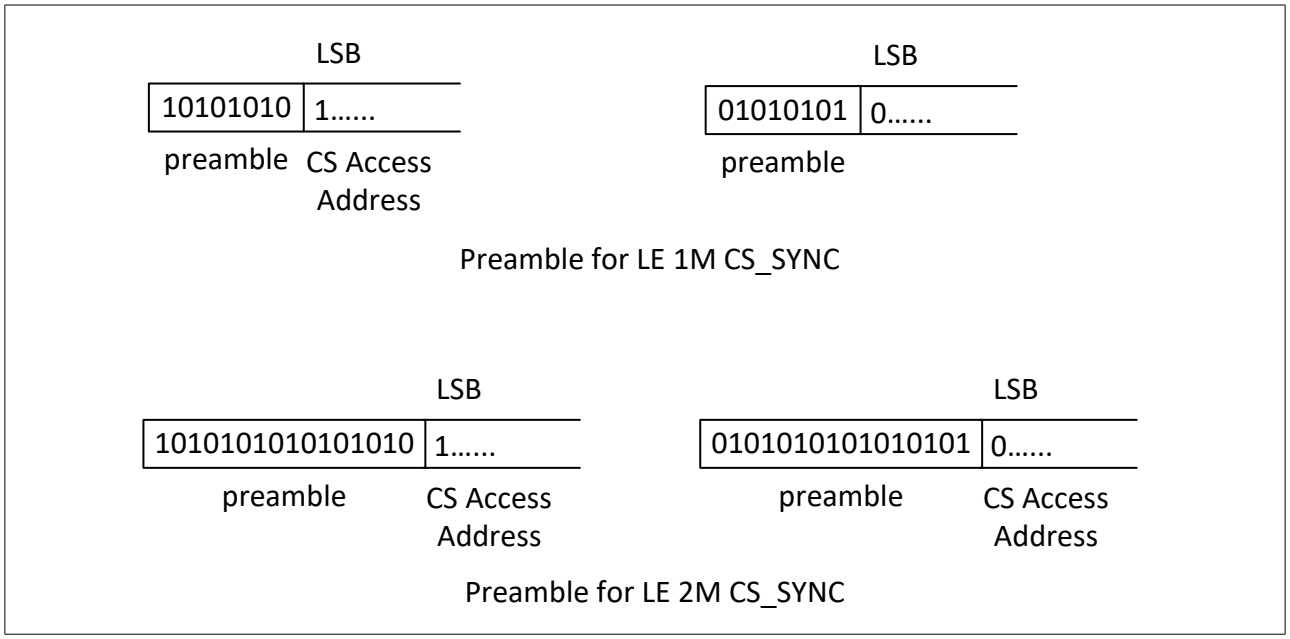



*Figure 2.2: CS_SYNC preamble*

## 2.2 Channel Sounding Access Address

Each CS Access Address is a sequence of bits that shall be cryptographically generated using the CS DRBG described in [Vol 6] Part E, Section 3.1.6. Devices that communicate with CS shall generate a common set of CS Access Addresses using the CS Access Address selection rules described below. The CS Access Addresses are used for synchronization, security, and round-trip time purposes.

Each CS step that transmits a CS_SYNC packet requires two CS Access Addresses that are derived from four DRBG output vectors of 32 bits each. The four vectors form four bit sequences $s_0$, $s_1$, $s_2$, and $s_3$ in the order the vectors were generated. Each bit sequence is constructed directly from the ordered bit output from the DRBG, with the first bit from the DRBG forming bit 31 (the most significant bit) of the sequence, the second bit forming bit 30, and so on, with the final bit forming bit 0 (the least significant bit) of the sequence. This ordered bit assignment is shown in Figure 2.3.

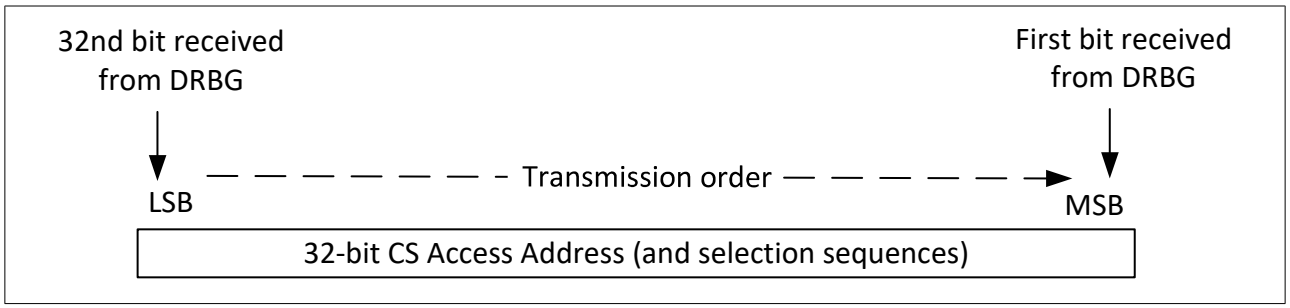



*Figure 2.3: CS Access Address and selection sequence bit ordering*

Sequences $s_0$ and $s_1$ are fed into the CS Access Address selection rules to produce a 32-bit sequence that becomes the CS Access Address of the first part of the CS step. Similarly, sequences $s_2$ and $s_3$ are fed into the CS Access Address selection rules to produce a 32-bit sequence that becomes the CS Access Address of the second part of the CS step.

The CS Access Address is transmitted least significant bit first (i.e., from bit 0 to bit 31).

### 2.2.1 Channel Sounding Access Address selection rules

The CS Access Address selection rules determine which of the two 32-bit candidates is selected to become the CS Access Address. These two 32-bit values shall be generated using the CS DRBG described in Section 4.8 at the step that requires the RTT exchange. The two random 32 bits for the CS Access Address candidates used in the CS_SYNC from the initiator to reflector shall be generated first. These two 32-bit values are referred to as $s_0$ and $s_1$ per the notation in Section 2.2. The two random 32 bits for the CS Access Address candidates used in the CS_SYNC from the reflector to the initiator shall be generated second. These two 32-bit values are referred to as $s_2$ and $s_3$ per the notation in Section 2.2. Filtering rules assign one autocorrelation score to each pair of 32-bit candidates. The sequence with the lower autocorrelation score is selected as the CS Access Address. If both sequences achieve the same autocorrelation score, then the second of the sequence pair (i.e., $s_1$ and $s_3$) shall be selected.

The CS autocorrelation score is a function that applies to a 32-bit sequence that is composed of 32 individual bits $s_i$ (i in the 1 to 32 range) and is defined as:

$$score = abs(2 \times C_1 - 31) + abs(2 \times C_2 - 30) + abs(2 \times C_3 - 29)$$

Where each of the $C_k$ elements is defined as:

$$C_k = \sum_{i=1}^{32-k} (s_i \oplus s_{i+k})$$

### 2.2.2 Channel Sounding Access Address checking

A device shall return a CS Access Address quality indication value, as shown in Table 2.1, for a CS_SYNC packet received or intended to be received in a CS step. Although the generation of an Access Address quality indication value is implementation specific, the meaning noted in Table 2.1 should be followed. Only packets with an Access Address quality indication value of 0 shall be used for testing RTT accuracy measurements.

| Value | Meaning |
|---|---|
| 0 | CS Access Address check is successful and all bits match the expected sequence. For non-Mode-0 packets, the device has obtained a ToA-ToD (or ToD-ToA) measurement from the packet. |
| 1 | CS Access Address check contains 1 or more bit errors. For non-Mode-0 packets, the device has obtained a ToA-ToD (or ToD-ToA) measurement from the packet. |
| 2 | CS Access Address not found |
| 3 | RFU |

*Table 2.1: CS Access Address integrity check result values*

## 2.3 Trailer

The CS trailer is a sequence of 4 bits, alternating between 0 and 1 bits. As shown in Figure 2.4, the trailer is 1010 (in transmission order) when the most significant bit of the CS Access Address is a 0, and 0101 when the most significant bit of the CS Access Address is a 1.

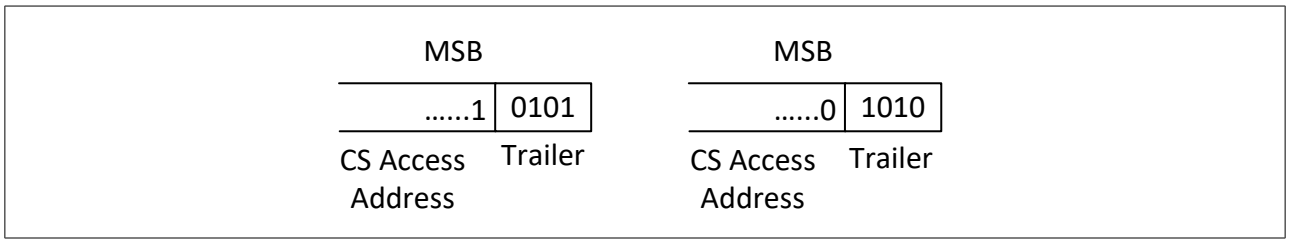



*Figure 2.4: CS_SYNC trailer*

## 2.4 Sounding sequence

The optional sounding sequence is a sequence of bits alternating between 0 and 1, starting with a 0 as the first bit (LSB) in transmission order, which is then partially overwritten by one or two marker signals for resilience against spoofing attacks. Both the marker signal position and bit pattern are selected separately for each CS_SYNC transmitted.

The marker signal is 4 bits long. There are two possible signals that are selected randomly, seeded by the CS DRBG as described in Section 4.8. CS DRBG shall be invoked to generate a single random bit (see Section 4.8). If this bit is a 0, then the marker signal consists of the bits ‘1100’ in transmission order. If the bit is a 1, then the marker signal consists of the bits ‘0011’ in transmission order. This marker signal selection process shall be repeated if a sounding sequence of 96 bits is used and only if the second marker is not omitted.

Each marker signal shall overwrite four consecutive bits of the sounding sequence starting at a position determined by invoking the random number generation function

hr1 (see Section 4.8.1). For the 32-bit sounding sequence, a single marker shall be used starting at bit number hr1(29) of the sequence (where the first bit is bit number 0). For the 96-bit sounding sequence, one or two markers shall be used. The first marker shall start at bit number hr1(64) and the second marker shall start at bit number hr1(75) + 67; if the starting bit position for the second marker exceeds 92, then the second marker shall be omitted.

See Section 4.8 for CS DRBG invocation ordering rules.

Figure 2.5 shows the construction of the sounding sequence with a single marker inserted. The sounding sequence is transmitted with the least significant bit first, and the most significant bit transmitted last.

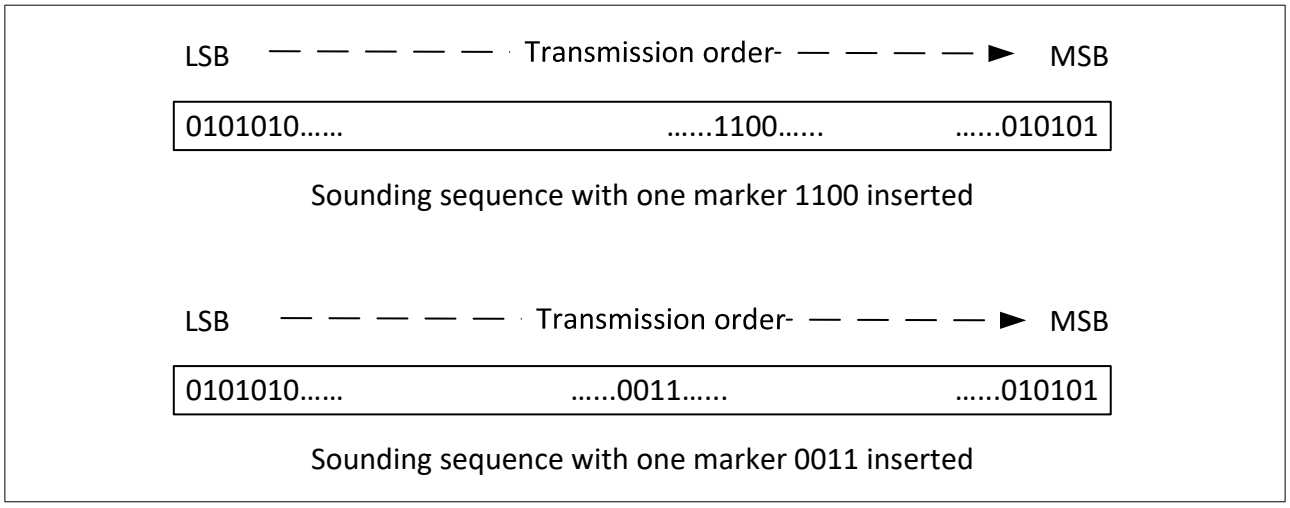



*Figure 2.5: CS_SYNC sounding sequence*

## 2.5 Random sequence

The optional random sequence payload is a sequence of randomized bits. This random sequence is generated separately for each CS_SYNC transmitted. This bit sequence shall be generated using the output of the CS DRBG described in Section 4.8. The length of this bit sequence is identified by the RTT_Types selected during the CS configuration procedure described in [Vol 6] Part B, Section 5.1.25.

See Section 4.8 for CS DRBG invocation ordering rules.

The random sequence is constructed directly from the ordered bit output from the DRBG, with the first bit from the DRBG occupying the most significant bit of this field, which is the bit position equal to the length of the random sequence minus 1. The final bit from the DRBG occupies the least significant bit at position 0, as shown in Figure 2.6.

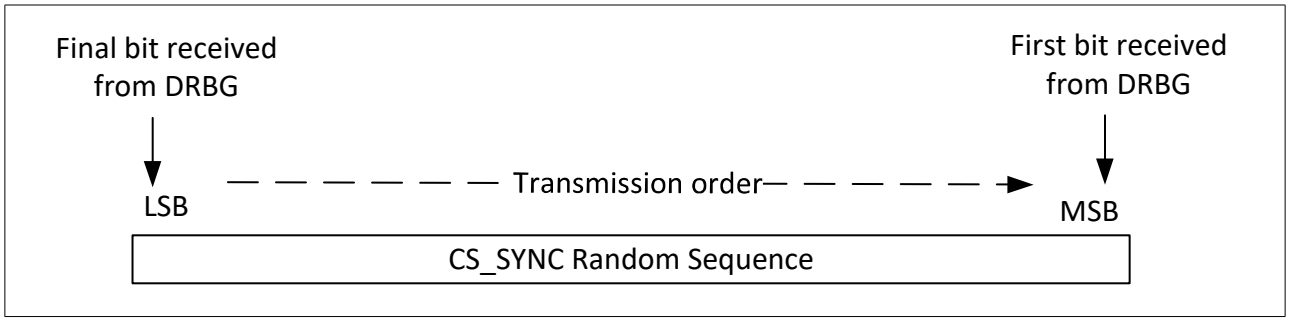



*Figure 2.6: CS_SYNC random sequence*

The random sequence is transmitted with the least significant bit first and the most significant bit transmitted last.

## 2.6 Channel Sounding extended packet formats

CS defines two additional packet formats.

A CS_SYNC followed by a CS tone is shown in Figure 2.7. The CS_SYNC has two mode-specific variations: a CS_SYNC_0_R, as described in Section 4.3.1, and a CS_SYNC_3_I, as described in Section 4.3.4.

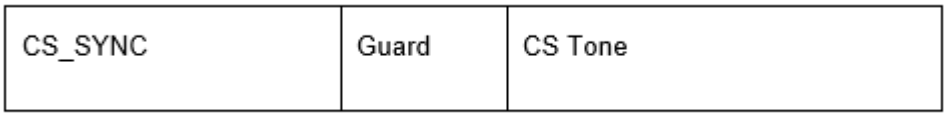



*Figure 2.7: CS_SYNC followed by a CS tone*

A CS tone followed by a CS_SYNC is shown in Figure 2.8. In this case, the CS_SYNC occurrence is the CS_SYNC_3_R as described in Section 4.3.4.

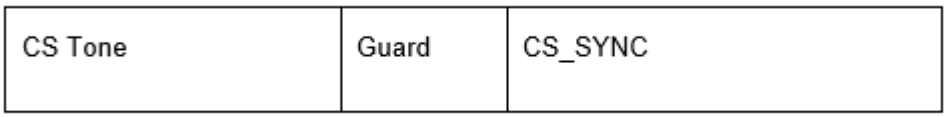



*Figure 2.8: CS tone followed by a CS_SYNC*

In both cases, the guard time (T_GD) is 10 µs in duration, independent of the LE PHY being used. The transmitted signal during the guard time should be ignored by the receiver.

In both cases, the CS tone and CS_SYNC are transmitted on the same RF frequency and are subject to the rules of frequency and phase stability established in [Vol 6] Part A, Section 3.5. The duration of the two additional packet formats is variable and depends on various configuration factors, as described in Section 4.3.1 and Section 4.3.4.

# 3 CHANNEL SOUNDING BIT STREAM PROCESSING

## 3.1 Measuring RTT

Devices may use CS_SYNC exchanges to measure the propagation channel's round-trip time (RTT).

Device A is the device that begins the RTT procedure with the transmission of the first packet in the RTT exchange. Device B is the device that receives the transmission from device A and then sends a response transmission back to device A.

Let $ToD_A$ represent the time of departure, measured at the antenna port, of the first packet transmitted in this exchange from device A. Also, let $ToA_A$ represent the time of arrival of the received packet, measured at the antenna port.

Similarly, let $ToA_B$ represent the time of arrival of the received packet from device A. Also, let $ToD_B$ represent the time of departure.

Consolidated, let $ToX_Y$ represent the time of departure from device Y if X is replaced with D, or the time of arrival of arrival if X is replaced with A measured from device Y at its antenna port.

The value of $ToX_Y$ is determined by optimizing the solution to the equation

$$ToX_Y\left[k\right] = \underset{\tau}{\operatorname{argmax}}\left|\int_{\tau+t_A}^{\tau+t_B} \hat{x}_Y(k,t)\hat{x}_Y^*(k,t-T_{sym})s^*\left(k,t-\tau\right)s\left(k,t-\tau-T_{sym}\right)\ dt\right| \qquad \text{(EQ 1)}$$

where $\hat{x}_Y(k,t)$ is the signal at the antenna port of Device Y during step k, after applying the reference down-conversion for a CS packet as defined in Section 3.1.1, and $s(k,t)$ is the reference transmitted signal of the CS packet for step k as defined in Section 3.5.2. The value $T_{sym}$ represents the symbol period. The value $t_A$ represents the start of the CS access address of the reference transmitted signal, where the preamble starts at $t = 0$. The value $t_B$ represents the end of the CS access address of the reference transmitted signal in the case of no random payload; alternatively, it represents the end of the random payload.

The RTT measurement obtained by Device A is obtained by using the difference of the values of $ToA_A - ToD_A$ and $ToD_B - ToA_B$ as expressed in the equation:

$$RTT = 2ToF = \left[\frac{1}{1+FFO_{RX^*}10^{-6}}(ToA_A - ToD_A)\right] - (ToD_B - ToA_B)$$

$FFO_{RX^*}$ is defined in [Vol 6] Part A, Section 3.5.1.

A single step for this exchange is shown in Figure 3.1.

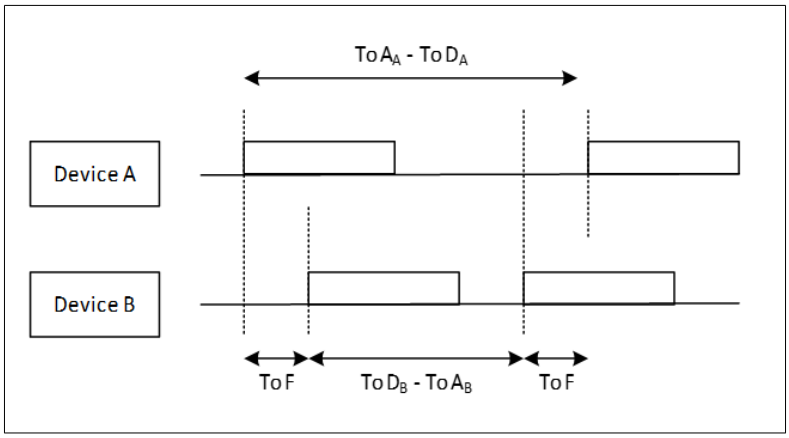



*Figure 3.1: Round-trip time*

Devices should measure the ToD and ToA values as accurately and precisely as possible. Devices shall compensate for any known internal delays from the antenna to the actual measurement point for these values. Device A shall compensate for the effects of clock drift based on the FFO estimate of device B, relative to its own clock as described in [Vol 6] Part A, Section 3.5.1. Device A shall implement frequency-based timing compensation based on the equation above and using the value of $FFO_{RX^*}$. Device B shall not attempt any related compensation for timing drift. Devices may choose to delay the start of their transmissions by up to one symbol period of the expected transmit point to increase the overall security of the RTT measurements.

Three variations of RTT timing estimates are defined in this section for the purpose of coordination between devices. A method for simultaneous extraction of a phase-based estimate along with an RTT timing estimate is also defined. Both device A and device B shall select the same method when performing both RTT timing and CS_SYNC phase-based estimates.

### 3.1.1 Reference receiver for round-trip time measurements

The reference down-conversion for the CS packet for step $k = M + 1, \ldots, M + K$, is defined at the antenna of the device as

$$\hat{x}\left(k,t\right) = LPF\left[x(t)e^{-j2\pi f_E[k]t}\right]$$

where $x(t)$ is the signal at the antenna of device as defined in [Vol 6] Part A, Appendix B and LPF is a low-pass filter that removes the high frequency components. $f_E[k]$ represents the expected center frequency of the CS_SYNC packet as defined in [Vol 6] Part A, Section 3.5.2. $\hat{x}(t)$ is defined for all time t within step k.

### 3.1.2 ToD and ToA reporting accuracy

Let *2ToF* represent the expected known two-way round-trip time of flight between device A and device B. In this section, the initiator is represented by device A and the reflector is represented by device B.

Denote $(ToD - ToA)_B[k]$ as the measurement value reported by Device B for step k.

Define the value of $(ToD - ToA)'_B[k]$

$$(ToD - ToA)'_B[k] = \frac{((ToD - ToA)_B[k] + T_SY_CENTER_DELTA[k])}{\left(1 + FFO_E \cdot 10^{-6}\right)}$$

where *T_SY_CENTER_DELTA*[*k*] is the value of T_SY_CENTER_DELTA for the mode used at step k as defined in Section 4.3.2 and Section 4.3.4. $FFO_E$ is the fractional frequency offset of the CS subevent as defined in [Vol 6] Part A, Section 3.5.2.

The error in Device B's reported time measurement for step k, is defined as

$$\Delta T_{RESP}[k] = (ToD_B[k] - ToA_B[k]) - (ToD - ToA)'_B[k]$$

where $ToD_B[k]$, $ToA_B[k]$ are the values measured as described in Section 3.1.

Similarly, denote $(ToA - ToD)_A[k]$ as the measurement value reported by Device A for step k that must include the correction for clock drift as described in Section 3.1.

Define the value of $(ToA - ToD)'_A[k]$ as

$$(ToA - ToD)'_A[k] = (ToA - ToD)_A[k] + T_SY_CENTER_DELTA[k]$$

Let *2ToF* represent the expected known two-way round-trip time of flight between device A and device B.

Device A's RTT measurement error for step k, is defined as

$$\Delta T_{RTT}[k] = \left(ToA_A[k] - ToD_A[k]\right) \cdot \left(1 + FFO_E \cdot 10^{-6}\right) - (ToA - ToD)'_A[k] - 2ToF\,.$$

where $ToD_A[k]$, $ToA_A[k]$ are the values measured by the Lower Tester as described in Section 3.1.

Let MeasurementValid[k] be equal to 1 if the CS Access Address quality indication value for the CS_SYNC received at step k is reported as 0, as described in Section 2.2.2. MeasuremetValid[k] is equal to 0 if the CS Access Address quality indication value is reported otherwise.

Within a single procedure i, with N steps of single packet exchanges, Device B's procedure wise response time error is defined as

$$\Delta T_{RESP,PROC} = \frac{\sum_{k=1}^{N} \Delta T_{RESP}[k] \cdot MeasurementValid[k]}{\sum_{k=1}^{N} MeasurementValid[k]}$$

Similarly, Device A's procedure wise RTT measurement error is given by

$$\Delta T_{RTT,PROC} = \frac{\sum_{k=1}^{N} \Delta T_{RTT}[k] \cdot MeasurementValid[k]}{\sum_{k=1}^{N} MeasurementValid[k]}$$

For Device B, the absolute bias $B$ is the absolute value of the mean of $\Delta T_{RESP,PROC}$ for Device B across procedures. The standard deviation of $\Delta T_{RESP,PROC}$ for Device B across procedures is denoted as $\sigma$. For Device A, the absolute bias $B$ is the absolute value of the mean of $\Delta T_{RTT,PROC}$ for Device A across procedures. The standard deviation of $\Delta T_{RTT,PROC}$ for Device A across procedures is denoted as $\sigma$.

Each device shall be characterized and the values of $B$ and $\sigma$ shall meet one of two possible criteria at a receiver input level of -70 dBm:

$$2\sigma + B < 10 \text{ ns} \quad \text{(EQ 2)}$$

for a minimum number of mode-1 and mode-3 CS_SYNC packet exchanges *N* per procedure, where *N* ≤ 255, or alternatively

$$2\sigma + B < 150 \text{ ns} \quad \text{(EQ 3)}$$

for a minimum number of mode-1 and mode-3 CS_SYNC packet exchanges *N* per procedure, where *N* ≤ 255.

## 3.2 Timing estimate based on an Access Address

This section describes two methods for deriving ToA timing estimates based on the CS Access Address.

### 3.2.1 Timestamps using a native clock

Timestamping of the ToA may be done using the receiver's native clock. The resolution provided by this method for a single sample is therefore proportional to the accuracy and clock rate of that receiver.

To improve the overall resolution of the measurement, measurements over several packets may be taken and averaged. Because the sampling clocks of a pair of devices are unrelated in phase, the distribution derived from several measurements is used to normalize the ToA value to a more precise average.

To improve the distribution of results, a transmitter may choose to transmit a CS_SYNC with an added fractional timing delay of up to one symbol period, which itself is pseudo-random with respect to the expected transmit point. The selection of the pseudo-random offset is left to the implementation.

### 3.2.2 Timing estimate based on a pseudo-noise bit sequence

RTT timing measurements may use a pseudo-noise sequence in a CS Access Address or payload, of sufficient length to obtain an indication of the timing error of the received signal with regard to the receiver's local sampling clock. This timing error is the difference between the optimum sampling point and the actual sampling point, also known as the fractional timing component.

Several methods exist for extraction of this fractional timing component. One such method searches for the symbol correlation peak across all the symbols in the pseudo-noise sequence. Fractional timing extraction is not dependent on pseudo-random clock phase distribution between packet exchanges.

## 3.3 Fractional timing estimate based on a sounding sequence

Timing estimates may be derived from a modulated repeating [0 1] bit sequence. The modulation is either at the LE 1M PHY, the LE 2M PHY, or the LE 2M 2BT PHY. The periodicity of this sequence creates two distinct tones in the output spectrum, of which the periodic components are associated with the basic repetition of the two symbols. At the 1 Msym/s rate, this periodicity is at 2 µs. At the 2 Msym/s rate, it is at 1 µs.

Let

*f* represent the frequency of the complex sinusoid, and

*α* represent a common complex gain on these two signals.

The two complex sinusoid baseband signals are then denoted as

$$S_{+f}(t) = ae^{+j2\pi ft}$$

$$S_{-f}(t) = ae^{-j2\pi ft}$$

Now let

$T_s$ represent the sampling period at the receiver,

*n* represent the number of samples of the respective complex sinusoids from a time *t*, and

*M* represent the integration period in units of full cycles of the complex sinusoids.

Then *L*, which represents the number of samples taken within that integration period, is denoted as

$$L = \frac{M}{fT_s} - 1$$

Denote the signal containing the modulated repeating [0 1] bit sequence as

$$S(n,t) = s(t + nT_s)$$

Then the phase of the two complex sinusoids is measured through the following two correlations:

$$\varphi\left(t, +f\right) = \angle\left(\sum_{n=0}^{L} S\left(n,t\right) \times e^{-j2\pi fnT_s}\right) \quad \text{(EQ 4)}$$

$$\varphi\left(t, -f\right) = \angle\left(\sum_{n=0}^{L} S\left(n,t\right) \times e^{+j2\pi fnT_s}\right)$$

Because of the time quantization error and random sampling clock offset at the receiver, the estimated ToA timing measured at the receiver's sampling clock rate is not the same as the actual ToA. Denote the coarse estimate of the start of the sounding sequence measured at the receiver's antenna port as $t_{\text{sync}}$. Using the two equations above, the fractional delay is calculated as:

$$\Delta t = \frac{\varphi\left(t_{sync}, +f\right) - \varphi\left(t_{sync}, -f\right)}{4\pi f}$$

The final estimate of the actual ToA is $t_{\text{sync}} + \Delta t$, which is used for the ToA value in Section 3.1.

### 3.3.1 Phase-based PCT estimate based on a sounding sequence

The properties of the modulated repeating [0 1] bit sequence also allow for the extraction of distance estimates using phase-based calculations. Similar to RTT exchanges, the phase measurements consist of two exchanges and measurements, first in the direction from a device A to a receiving device B, then in the reverse direction. Exchanges over at least two channels are required to resolve the 2π ambiguity in the distance estimate.

Let

$\varphi_0^a$ represent the RF initial phase of device *A* transmitting the signal,

$\varphi_0^b$ represent the RF receive phase of the signal as seen at the receiving device *B*,

$\omega_{lo}$ represent the carrier frequency of that signal, and

$\omega_m$ represent the frequency of the modulated tones.

The complex representation of the two dominant spectral components of the sounding sequence (ignoring the large tone on the center frequency) at the transmitting device *A* is

$$S_{rf_a}\left(t\right) = e^{j(\omega_{lo}t + \varphi_0^a)} \cdot \left(e^{+j\omega_m t} + e^{-j\omega_m t}\right)$$

Let

*D* represent the propagation distance, and

*C* represent the speed of light.

At the receiving device *B*, the representation of the receive signal (ignoring frequency dispersion inside the channel) is

$$S_{rf_b}\left(t\right) = e^{j\left(\omega_{lo}t + \varphi_0^a - \frac{\omega_{lo}D}{C}\right)} \cdot \left(e^{+j\omega_m t} + e^{-j\omega_m t}\right)$$

After processing at the receiver, the equation above can be represented as

$$S_{rf_b}\left(t\right) = e^{j\left(\varphi_0^a - \varphi_0^b - \frac{\omega_{lo}D}{C}\right)} \cdot \left(e^{+j\omega_m t} + e^{-j\omega_m t}\right)$$

Applying the same correlations represented in Section 3.3, EQ 4 yields the following phase results for each of the two complex sinusoids:

$$SS_PCT_1 = \varphi\left(t, +f\right) = \left(\sum_{n=0}^{L} S_{rf_b}\left(n,t\right) \times e^{-j2\pi f n T_s}\right)$$

$$SS_PCT_2 = \varphi\left(t, -f\right) = \left(\sum_{n=0}^{L} S_{rf_b}\left(n,t\right) \times e^{+j2\pi f n T_s}\right)$$

At device *B*, to correct for phase change of the local oscillator while transmitting, the phase correction should be performed by complex multiplying both correlations by exp(j×phase_correction), where the phase correction is expressed in radians. The reference point for this definition is the local device's antenna. The reference point for this definition is the local device's antenna. SS_PCT results shall be consistent with the

format for PCT results as described in Section 4.6. If a subevent contains PCT results, the same reference power level (RPL) should be used for the SS_PCT. Otherwise, the RPL shall be encoded as an 8-bit signed number. The RPL expressed in dBm shall correspond to the power of the sounding sequence sidelobe that produces a SS_PCT whose amplitude is 2048, obtained by integrating the received signal over the payload of the CS_SYNC packet at the expected sidelobe frequency. In all cases, a single RPL is used for all the SS_PCT values in a CS subevent.

The conversion between SS_PCT IQ values as unitless values to power in dBm can be expressed as denoted by EQ 6 in Section 4.6. In all cases the magnitude should be adjusted to give the power in the sounding sequence tone plus 6 dB.

These two equations can be further simplified, where $t_i$ is the same arbitrary time at which the correlations start, to

$$\varphi\left(t, +f\right) = e^{j\left(\varphi_0^a - \varphi_0^b - \frac{\omega_{lo}D}{C}\right)} \cdot \left(e^{+j\omega_m t_i}\right)$$

$$\varphi\left(t, -f\right) = e^{j\left(\varphi_0^a - \varphi_0^b - \frac{\omega_{lo}D}{C}\right)} \cdot \left(e^{-j\omega_m t_i}\right)$$

Then the phase-based rotation result from device A to device B is

$$\varphi_{ab} = \left(\angle\varphi(t, +f) + \angle\varphi(t, -f)\right) \div 2 = \left(\varphi_0^a - \varphi_0^b - \frac{\omega_{lo}D}{C}\right)$$

Device *B* then reciprocates with a transmission of its own, and a similar result is obtained with device *B* as the transmitter and device *A* as the receiver. If both devices retain (or correct for) phase continuity in their local oscillators, then the phase-based rotation result from device *B* to device *A* is

$$\varphi_{ba} = \left(\angle\varphi(t, +f) + \angle\varphi(t, -f)\right) \div 2 = \left(\varphi_0^b - \varphi_0^a - \frac{\omega_{lo}D}{C}\right)$$

When combined, these two results provide an estimated distance *D* between the two devices.

$$\Phi_{2w}\left(\omega_{lo}, D\right) = -\left(\varphi_{ab} + \varphi_{ba}\right) = \frac{2\omega_{lo}D}{C}$$

This estimate is compromised by the 2π ambiguity. To obtain an unambiguous range, the results from measurements over several frequencies, as defined in [Vol 2] Part A, Section 2, may be combined:

$$\Delta_\Phi = \Phi_{2w}\left(\omega_{lo}{}^1, D\right) - \Phi_{2w}\left(\omega_{lo}{}^0, D\right) = \frac{2\Delta\omega_{lo}D}{C}$$

$$D = \frac{C}{2\Delta\omega_{lo}}\Delta_\Phi \ mod \frac{C}{2\Delta\omega_{lo}}$$

#### 3.3.1.1 Reference receiver for phase-based ranging from a sounding sequence

Let the CS step at which a sounding sequence is exchanged occur at step index k as defined for CS tones in [Vol 6] Part A, Section 6.1. The reference down-conversion for the sounding sequence is then defined at the antenna of the device as

$$\hat{x}(t) = LPF\left[x(t)e^{-j2\pi f_E[k]t}\right]$$

where $x(t)$ is the signal at the antenna of a device used as defined in [Vol 6] Part A, Appendix B, and LPF is a low-pass filter that removes the high frequency components. $f_E[k]$ represents the center frequency of the CS_SYNC packet carrying the sounding sequence, as defined in [Vol 6] Part A, Section 3.5.2. $\hat{x}(t)$ is defined for all time $t$ within step $k$.

Define the start time of the transmission of the sounding sequence from the vector signal generator as $t_{SS}^{TX}[k]$.

Denote the time of arrival of the CS Access Address of the transmission from the implementation under test as $t_{SS}^{RX}[k]$.

In the reference receiver, the following are also defined:

$T_s$ represents the sampling period at the receiver.

$L_{ss}$ represents the length of the sounding sequence in terms of the number of symbols (see Section 2.4).

*symbol_time* is 1 µs for LE 1M and 0.5 µs for LE 2M and LE 2M 2BT.

$f = \frac{1}{2 \times \text{symbol_time}}$ represents the frequency of the complex exponential function used for correlations.

Then $L$, which represents one less than the number of samples taken within the integration period, is denoted as

$$L = \frac{L_{SS} \times \text{symbol_time}}{T_S} - 1$$

#### 3.3.1.2 Accuracy requirements

The observed average transmitted phase $\varphi_{tx}(k, +f)$ and $\varphi_{tx}(k, -f)$ for step *k* is measured through the following two correlations:

$$\varphi_{tx}(k, +f) = \angle\left(\sum_{n=0}^{L} \hat{x}\left(t_{SS}^{TX}[k] + nT_s\right) \times e^{-j2\pi fnT_s}\right)$$

$$\varphi_{tx}(k, -f) = \angle\left(\sum_{n=0}^{L} \hat{x}\left(t_{SS}^{TX}[k] + nT_s\right) \times e^{+j2\pi fnT_s}\right)$$

These results are then combined to give the observed average transmitted phase for step *k*, $\Phi_{TX}[k]$:

$$\Phi_{TX}[k] = (\varphi_{tx}(k, +f) + \varphi_{tx}(k, -f))$$

The observed average received phase $\varphi_{rx}(k, +f)$ and $\varphi_{rx}(k, -f)$ for step *k* is measured through the following two correlations:

$$\varphi_{rx}(k, +f) = \angle\left(\sum_{n=0}^{L} \hat{x}\left(t_{SS}^{RX}[k] + nT_s\right) \times e^{-j2\pi fnT_s}\right)$$

$$\varphi_{rx}(k, -f) = \angle\left(\sum_{n=0}^{L} \hat{x}\left(t_{SS}^{RX}[k] + nT_s\right) \times e^{+j2\pi fnT_s}\right)$$

These results are then combined for step *k*, according to the following equation:

$$\Phi_{RX}[k] = (\varphi_{rx}(k, +f) + \varphi_{rx}(k, -f))$$

The internal phase offset for step *k* is defined as

$$\theta_C[k] = principal(\Phi_{TX}[k] - \Phi_{RX}[k] + 2 \times \angle(SS_PCT_1[k] + SS_PCT_2[k]))$$

where *principal*( ) is defined in [Vol 6] Part A, Section 5.2.1, and $SS_PCT_1[k]$ and $SS_PCT_2[k]$ are the correlations defined in Section 3.3.1 for step *k*, provided by the IUT.

Denote $\theta_{C,UW}[k]$ as the phase-unwrapped version of $\theta_C[k]$.

Denote $\alpha f_{E,ord}[k] + \beta$ as the solution to the linear regression of the set of points defined by $(f_{E,ord}[k], \theta_{C,UW}[k])$.

For any subevent where

- An RTT Sounding Sequence of at least 32 bits is exchanged at least once for all CS channels,
- the transmit signal received signal satisfies the conditions for frequency offset described in [Vol 6] Part A, Section 3.5.2, and
- the receiver input level is -55dBm,

the solution to the linear regression shall satisfy

$$|\alpha| < 2\pi \times 10.2 \text{ ns},$$

for 95% of subevents.

## 3.4 Fractional timing estimate based on a random sequence

Fractional RTT timing measurements based on a random sequence may use the same properties for the extraction of fractional timing components based on a pseudo-noise bit sequence as those described in Section 3.2.2. The symbol correlation peak derived from the individual symbols of an added random sequence bit field may be used to further refine the fractional timing extracted from the CS Access Address.

## 3.5 Attack detection requirements

This section describes a metric for reflecting the level of symbol attack detection, describes a reference signal construction, and provides reference attack signal definition. Reference attack signal definitions are used as a basis for detecting the presence of attack signals.

### 3.5.1 Normalized attack detector metric

The normalized attack detector metric (NADM) provides a measure of how much a received GFSK modulated packet signal differs from the expected packet signal as described in Section 3.5.2. A NADM value range indicates a progressively increased chance that an attacker is present. Refer to Table 3.1 for a description of these ranges.

Figure 3.2 shows the components of the overall attack detector system. The attack detector system uses a Controller-based NADM algorithm to determine a NADM value from each received packet in each CS device on both sides of the link. The NADM values are aggregated while an attack detector algorithm determines the threat level using the NADM values collected from one or more packet exchanges within a CS procedure. A user application can use the attack detector algorithm to determine its future actions.

The specifics of the user application and attack detector algorithm are beyond the scope of this specification.

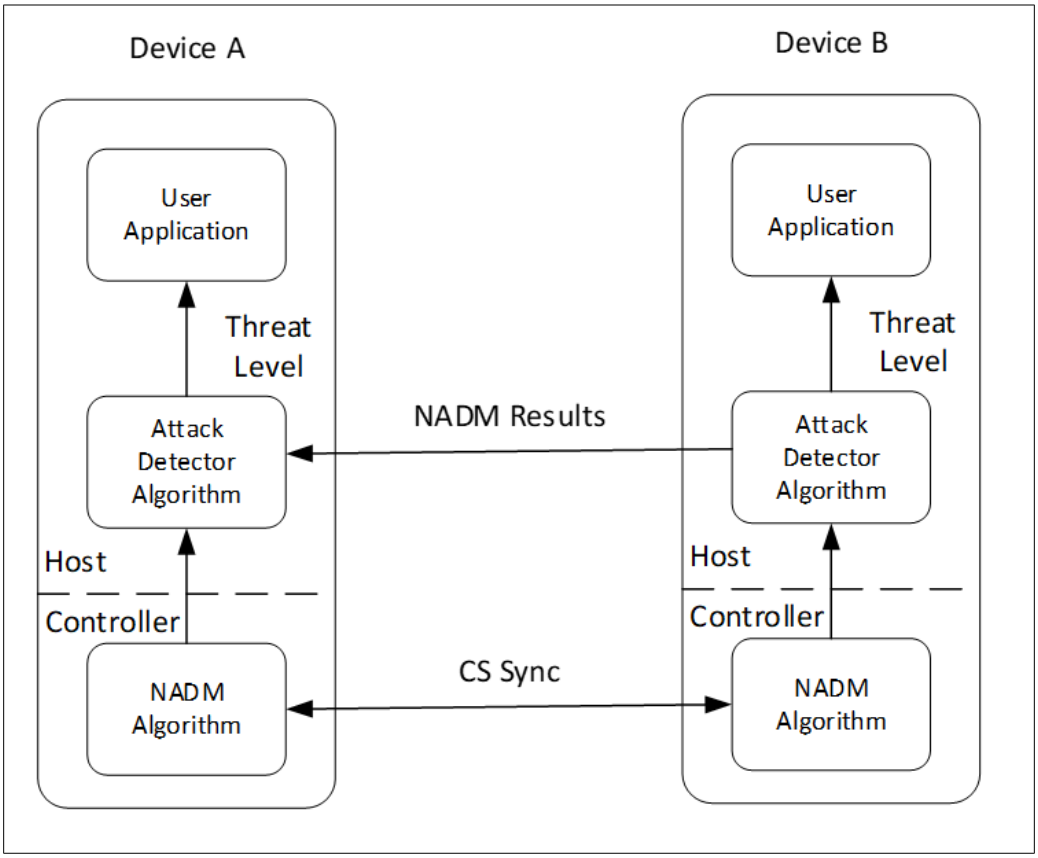



*Figure 3.2: Components of a NADM attack detector system*

Figure 3.3 shows an example NADM system. In this example, a packet is received and its signal content is extracted. This signal content contains information associated with the encoded data symbols from GFSK modulation. Because the data symbols for the received packet are known in advance, they can be modulated to construct a reference signal. A reference signal is described in Section 3.5.2. The received and reference signals should be similar. Differences in the content of the two signals may be used to produce a raw attack detector metric. This raw metric may be combined with other information, such as the number of bit errors in the received signal, to produce the NADM.

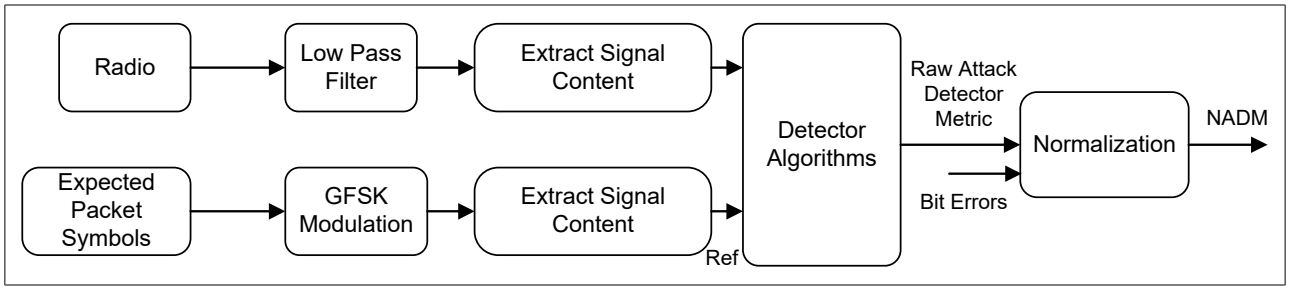



*Figure 3.3: NADM processing*

A raw detector metric for both a phase-based attack and an amplitude-based attack for CS_SYNC packets with sounding sequence or random sequence is described in Section 3.5.3 and Section 3.5.4.

The list of NADM values is shown in Table 3.1.

| NADM | Attack Type |
|---|---|
| 0x00 | Attack extremely unlikely |
| 0x01 | Attack very unlikely |
| 0x02 | Attack unlikely |
| 0x03 | Attack is possible |
| 0x04 | Attack is likely |
| 0x05 | Attack very likely |
| 0x06 | Attack extremely likely |
| 0xFF | Unknown |
| All other values | Reserved for future use |

*Table 3.1: NADM values*

Values starting from “Attack unlikely” through to “Attack extremely unlikely” indicate with increasing confidence the absence of an attacker. Values starting from “Attack is possible” through to “Attack extremely likely” indicate with increasing confidence the presence of an attacker. The implementation shall determine whether an attack is present or not present, and the selection of the level of confidence of that event. If a device is unable to determine a NADM value or does not support NADM processing, then it shall report a NADM value of *Unknown* (0xFF).

### 3.5.2 Reference signal modulated with h=0.5 GFSK

Let B = [$b_0$.. $b_{N-1}$] be a binary sequence of N elements in the [0,1] space.

Let A = [$a_0$.. $a_{N-1}$] be the symbol sequence of N elements corresponding to the binary sequence, following the mapping defined in [Vol 6] Part A, Section 3.1. Also, $a_i = 2b_i - 1$.

Let p(t) be a Gaussian-shaped pulse function, for a normalized symbol period of 1, as defined by:

$$p(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-2\left(\frac{t}{\sigma}\right)^2}$$

Where for BT = 0.5

$$\sigma = \frac{\sqrt{\ln 2}}{\pi}$$

and for BT = 2.0

$$\sigma = \frac{\sqrt{\ln 2}}{4\pi}$$

And g(t) is the convolution of p(t) with a rectangle pulse of normalized duration 1, rect(t) = 1 when 0 < t < 1 and 0 otherwise.

$$g(t) = p(t) * rect(t)$$

Where the * operator represents the time convolution of the two signals.

Based on these definitions, the normalized reference phase is given by:

$$\varphi_{r,1}\left(t\right) = \frac{\pi}{2} \sum_{i=0}^{N-1} a_i \int_{-\infty}^{t} g\left(\tau - i\right) d\tau$$

For a reference signal with symbol period $T_{Sym}$ then the reference signal phase is defined as

$$\varphi_r(t) = \varphi_{r,1}\left(\frac{t}{T_{sym}}\right)$$

and the reference signal is then

$$x_r(t) = \exp(j\varphi_r(t))$$

### 3.5.3 Early commit attacks with phase based detection

#### 3.5.3.1 Sounding sequence attack signal definition

The format of the sounding sequence with marker insertion is described in Section 2.4. An early commit, late detect (ECLD) attack is a relevant MITM attack strategy due to the regular, repeating pattern of the sounding sequence. During an ECLD attack, the attacker receives the first device's transmitted signal, and may regenerate the sounding sequence with a timing advance while searching for the randomized marker. The MITM attacker detects the marker while observing an unexpected bit transition (i.e., 0b1 to a 0b1, or 0b0 to a 0b0, instead of 0b1 to a 0b0, or 0b0 to a 0b1). The MITM may then

change the phase trajectory to correct the rest of the symbol so the second device does not receive an incorrect symbol and the attack is not detected.

Table 3.2 shows the positions of the symbol(s) where an attacker-induced glitch might be detected given the symbol position of the marker start.

| Symbol position of marker start | Symbol(s) with Glitch | |
|---|---|---|
| | Marker Sequence 0b1100 | Marker Sequence 0b0011 |
| k odd, original bit is 1 | k+1 and possibly k+2 | k and possibly k+1 |
| k even, original bit is 0 | k and possibly k+1 | k+1 and possibly k+2 |

*Table 3.2: Sounding sequence detectable symbol glitch positions*

The following definitions are used to generate the attack signal:

- Initialize the reference attack packet $P_A$ as the CS_SYNC packet without any markers in its sounding sequence. Define the oversampled discrete-time phase of the attack signal, $\Phi_A[\mathrm{n}]$, as the phase of the GFSK modulated version of $P_A$. This is defined in terms of the reference signal defined in Section 3.5.2. where binary input sequence is $P_A$. Then $\Phi_A\left[n\right] = \varphi_r\left(\frac{nT_{\mathrm{sym}}}{R}\right)$, where $T_{\mathrm{sym}}$ is the symbol period, and $R$ is the oversampling rate within one symbol period.
- The MITM attack signal is defined based on a detection delay of M samples of the original transmitted signal. The attacker determines a bit flip at the end of the detection delay and then applies a correction to follow its estimate of the original transmitted signal. The duration of this correction is denoted as the transition period. Table 3.3 shows this detection latency as well as other parameters (as explained below) that define this reference attack signal set.

| Parameter | Value | Definition |
|---|---|---|
| $R$ | ≥ 16 | Symbol oversampling rate |
| $M$ | 0.25$R$ | Detection latency period |
| $W$ | 0.25$R$ | Transition period |

*Table 3.3: Sounding Signal attacker signal configuration*

- The correction is a function of a Hanning window, that here is parameterized by its half window size width $W$:

$$Han\left(l\right) = \frac{1}{2} \times \left(1 - \cos\left(\frac{2\pi \times l}{2W+1}\right)\right), l = \left[0, 2W\right].$$

The reference attack signal is further defined as follows:

1. Denote position $p$, as the first bit in the packet that differs with the initial reference attack packet. If this bit is a 1, then the marker is 0b0011, otherwise it is 0b1100.

For some bit positions there will be only one possible start position of marker. For example, if there is a bit flip in position 0 or 30 in the case of a single marker. If there is more than one possible start of the marker, then the attacker then makes a guess whether bit *p* is the first or second bit of the marker. This guess is independently distributed with equal probability. Denote the attacker's first guess of the marker start position as $k_1$. If there is a valid second guess of the marker start position, denote this as $k_2$.

2. Define $P_1$ as the reference attack packet with the marker starting in position $k_1$, and $\Phi_1(n)$ as the signal obtained by GFSK modulating $P_1$. Similarly, define $P_2$ as the reference attack packet with the marker starting in position $k_2$, and $\Phi_2(n)$ as the signal obtained by GFSK modulating $P_2$.
3. Modify the phase trajectory of the reference attack signal $\Phi_A[n]$ at samples from symbol starting at $n = R \times p + 1$.
4. Segment 1: during the first M samples (detection latency period), the current reference attack signal is sent:

$$\Phi_A[n] \leftarrow \Phi_A[n], n \in [R \times p + 1, R \times p + M]$$

Segment 2: during the next W samples (transition period), a (half) Hanning windowed phase transition is applied from $\Phi_A[R \times p + M + 1]$ to the first guess $\Phi_1[R \times p + M + W]$:

$$\Phi_A[n] \leftarrow \Phi_A[R \times p + M] + \Delta\Phi \times \mathrm{Han}(n - R \times p + M),$$
$$n \in [R \times p + M + 1, R \times p + M + W]$$

$$\Delta\Phi = \Phi_1[R \times p + W + M] - \Phi_A[R \times p + M]$$

Segment 3: in this segment, the phase trajectory follows the first guess:

$$\Phi_A[n] \leftarrow \Phi_1[n], \quad n \geq R \times p + M + W + 1$$

The reference attack packet becomes $P_1$ i.e. $P_A \leftarrow P_1$.

5. In the case that the attacker's first guess is wrong, then the attacker observes a bit flip at $p' = p + 1$. The attacker then modifies the phase trajectory of the attack signal $\Phi_A[n]$ to produce a second glitch. Define the phase trajectory as follows.

Segment 1: during the first M samples (detection latency period), the current reference attack signal is sent:

$$\Phi_A[n] \leftarrow \Phi_A[n], n \in [R \times p' + 1, R \times p' + M]$$

Segment 2: during the next W samples (transition period), a (half) Hanning windowed phase transition is applied from $\Phi_A[R \times p' + M + 1]$ to the correct phase $\Phi_2[R \times p' + M + W]$:

$$\Phi_A[n] \leftarrow \Phi_A[R \times p' + M] + \Delta\Phi \times \mathrm{Han}(n - R \times p' + M),$$
$$n \in [R \times p' + M + 1, R \times p' + M + W]$$
$$\Delta\Phi = \Phi_2[R \times p' + W + M] - \Phi_A[R \times p' + M]$$

Segment 3: in this segment, the phase trajectory follows the second guess:

$$\Phi_A[n] \leftarrow \Phi_2[n], n \geq R \times p' + M + W + 1$$

The reference attack packet becomes $P_2$ i.e. $P_A \leftarrow P_2$.

6. If the marker is the first marker in the packet, and there exists a second marker in the packet, then return to step 1. Otherwise, the reference attack signal is $\Phi_A[\mathrm{n}]$.

#### 3.5.3.2 Random sequence attack signal definition

The phase of an attacker signal $\Phi_a(t)$ is the sum of the ideal reference signal $\phi_r(t)$ with symbol period $T_{sym}$ as defined in Section 3.5.2 and an adjustment term $\psi(t)$:

$$\Phi_a(t) = \phi_r(t) + \psi(t)$$

The adjustment term approximates the behavior of an MITM attack to make the sequence appear earlier than the original signal. This adjustment is achieved by applying a phase transition using a Gaussian mono-pulse early within the symbol period. The sign of the pulse acting over symbol $n$ is determined by the symbol value $a_n$.

For each encoded symbol $n = 0, \ldots, N-1$, a pulse is added to $\psi(t)$ centered at $t = (n + p + 1)T_{sym}$, where $p$ is a time adjustment that significantly defines the attacker's desired distance offset.

In this context, a Gaussian mono-pulse is defined as minus the derivative of a Gaussian pulse. The width of the pulse is roughly $T_{sym} \div K$, where $K$ is the pulse sharpness factor. The magnitude $M$, time adjustment $p$, and sharpness factor $K$ are used to manipulate the timing of $\Phi_a(t)$ compared to $\phi_r(t)$.

$\psi(t)$ is defined as

$$\psi\left(t\right) = M \sum_{n=0}^{N-1} a_n \left(\frac{t}{T_{\mathrm{sym}}} - p - n - 1\right) e^{-K\left(\frac{t}{T_{\mathrm{sym}}} - p - n - 1\right)^2} \quad \text{(EQ 5)}$$

where $a_n \in \{-1, 1\}$ is the symbol sequence as defined in Section 3.5.2.

Table 3.4 shows the parameters from EQ 5 that define a Reference attack signal set for LE 1M and LE 2M.

| Parameter | Definition | Value LE1M | Value LE 2M | Value LE 2M 2BT |
|---|---|---|---|---|
| *K* | Pulse sharpness factor | 12 | 12 | 12 |
| *M* | Pulse sharpness factor | 3 | 3 | 3 |
| *p* | Pulse time adjustment factor<br>These values of p equate to approximately 3 meters of distance decrease. | -0.56 | -0.61 | -0.62 |
| $T_{sym}$ | Symbol period in microseconds | 1.0 | 0.5 | 0.5 |

*Table 3.4: Attacker signal configuration*

#### 3.5.3.3 Raw attack detector metric based on a sounding sequence or a random sequence

A raw attack detector estimate may be derived from a modulated random sequence if this sequence is known between a transmitting and receiving set of devices. A preconstructed reference model based on this sequence is used as a basis of comparison for detecting symbol manipulation.

The modulation of the signal for which an attack detection estimate is generated is either at the 1 Msym/s rate or at the 2 Msym/s rate.

Let

*f* represent the frequency of the modulated bits.

$T_s$ represent the sampling period at the receiver.

*M* represent the integration period in units of symbol full cycles.

The integration period for an RTT with sounding sequence is the span of symbols with the possibility of detectable glitches when manipulation is present, as defined in Section 2.4. The integration period for RTT with random sequence is the span of the entire random sequence, as defined in Section 2.5.

Let $l$, which represents the number of samples within each full symbol, be denoted as

$$l = \frac{1}{fT_S}$$

And *L*, which represents the number of samples within the full integration period, is denoted as

$$L = Ml$$

Denote the measured signal containing the random sequence as

$$S(n,t) = s(t + nT_s)$$

Similarly, denote the reference signal of that same random sequence as

$$S_r(n,t) = s(t + nT_s)$$

Then the instantaneous phase of the measured signal and the reference signal, respectively, are

$$\varphi(n,t) = \angle S(n,t)$$
$$\varphi_r(n,t) = \angle S_r(n,t)$$

The phase of the measured signal is then compared to the reference signal to calculate the quality factor. Ambiguity exists between the start instants of the two signals, because the data points of the two signals are collected separately and because of quantization error at the receiver.

Two reference methods for deriving a raw attack detector metric are described.

#### 3.5.3.3.1 Raw attack detector metric based on normalized cross correlation

Normalized cross-correlation may be used to compare a measured signal to a reference signal. In the following description, the mean phase of the reference signal is shifted over a full symbol worth of sample offsets. Then each reference mean phase value is compared with the mean phase of the measured signal.

Let *L'*, which represents the maximum number of samples to process within an integration period that is one full sample less than the full integration period, be denoted as

$$L' = L - l$$

Denote the mean of the phase of the measured signal as

$$\varphi_{Mean} = \frac{1}{L'}\left(\sum_{n=1}^{L'} \varphi\left(n,0\right)\right)$$

Similarly, denote the mean of the phase of the reference signal over a full symbol worth of sample offsets as

$$\varphi_{rMean}\left(z \middle| 1 \le z \le l\right) = \frac{1}{L'}\left(\sum_{n=z}^{L'+z} \varphi_r\left(n,0\right)\right)$$

Cross-correlation is then calculated over the relative phase of the two signals, with a lag span of a full symbol cycle to compensate for receiver quantization error.

$$Corr\left(z \middle| 0 \le z \le l\right) = \frac{\frac{1}{L'}\sum_{n=1}^{L'}(\varphi(n,0) - \varphi_{Mean})(\varphi_r(n+z,0) - \varphi_{rMean}(z))}{\sqrt{\sum_{n=1}^{L'}(\varphi(n,0) - \varphi_{Mean})^2}\sqrt{\sum_{n=1}^{L'}(\varphi_r(n+z,0) - \varphi_{rMean})^2}}$$

The autocorrelation result of the two signals is then

$$MaxCorr = \max\left(Corr(z)\right)$$

From this result, a raw attack detector metric can be derived.

#### 3.5.3.3.2 Raw attack detector metric based on phase minimum square error

Phase minimum square error is an alternate method for comparing a measured signal to a reference signal. In contrast to the description in Section 3.5.3.3.1, the mean phase of the collected signal is shifted over a full symbol worth of sample offsets starting ½ symbol before the nominal start point. Then each measured mean phase value is compared with the mean phase of the reference signal.

Let $l'$ represent the number of samples within ½ of a full symbol, and be denoted as

$$l' = \frac{l}{2}$$

Denote the mean of the phase of the measured signal over sample offsets z in the range $-l' + 1$ *to* $l'$, relative to the nominal start and end of the target symbol(s) used in the comparison, as

$$\varphi_{Mean}\left(z \middle| \left(-l' + 1\right) \le z \le l'\right) = \frac{1}{L}\left(\sum_{n=1}^{L}\varphi(n+z,0)\right)$$

Similarly, denote the mean of the phase of the reference signal as

$$\varphi_{rMean} = \frac{1}{L}\left(\sum_{n=1}^{L}\varphi_r(n,0)\right)$$

The phase square error is then calculated as

$$PhaseSE\left(z\right) = \sum_{n=1}^{L}\left((\varphi(n+z,0) - \varphi_{Mean}(z)) - (\varphi_r(n,0) - \varphi_{rMean})\right)^2$$

The minimum phase mean square error is then

$$\text{PhaseMse} = \min(\text{PhaseSE}(z))\,.$$

From this result, a raw attack detector metric can be derived.

#### 3.5.3.4 Phase-based attack detection requirements for RTT with sounding sequence and random sequence packets

This section describes how the RTT with sounding sequence or RTT with random sequence and the normalized attack detector metric (NADM) shall be tested. CS mode-1 and CS mode-3 steps, described in Section 4.3, carry the packet contents.

This test requires a Lower Tester that can transmit either a normal signal or an attacker signal during mode-1 and mode-3 CS steps. The normal signal is the transmitted signal after either the sounding sequence or random sequence portion has been modulated using normal GFSK. The attacker signal is the normal signal, but with its phase adjusted to approximate a signal from an MITM attack. The attacker signal for RTT with sounding sequence is defined in Section 3.5.3.1. The attacker signal for RTT with random sequence is defined in Section 3.5.3.2.

The Lower Tester shall generate a reference signal of -67 dBm power together with a Gaussian Noise Floor of -152 dBm/Hz for the LE1M PHY. The Lower Tester shall generate a reference signal of -67 dBm power together with a Gaussian Noise Floor of -155 dBm/Hz for the LE2M PHY and the LE2M 2BT PHY.

The Lower Tester acts as an initiator to start a CS procedure with the IUT. Each test comprises one or more CS procedures executing the total CS subevent count specified in Table 3.5 and Table 3.6. Each CS subevent comprises a normal mode-0 CS step, followed by a single mode-1 CS step or a mode-3 CS step with specified sounding and random sequence lengths.

For each CS step, the Lower Tester makes a random decision to transmit either a normal signal or an attacker signal. The Lower Tester records these random decisions over the whole CS procedure. These random decision sequences are later compared with the resulting NADM values from the IUT.

Table 3.5 and Table 3.6 define multiple mandatory tests that depend on the supported capabilities of the IUT. The tests are designed to verify the minimum performance for a NADM value. CS_SYNC packets used in mode-1 and mode-3 are both tested. LE 1M, LE 2M, and LE 2M 2BT are also tested. For RTT with sounding sequence, the attacker signal is designed to exhibit abnormal phase glitch behavior around the associated marker bits. For RTT with random sequence, the attacker signal is designed to appear D meters (see Table 3.6) earlier than the normal signal.

| Test | LE PHY | RTT Step Type | Sounding Sequence Length (bits) | Number of Subevents | Expected min SNR in Reflec-tor-Under-Test / dB | Detection Latency | Transition Period |
|---|---|---|---|---|---|---|---|
| 1 | LE 1M | Mode-1 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |
| 2 | LE 1M | Mode-3 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |
| 3 | LE 2M | Mode-1 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |
| 4 | LE 2M | Mode-3 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |
| 5 | LE 2M 2BT | Mode-1 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |
| 6 | LE 2M 2BT | Mode-3 | 32 | 100 | 25 | ¼ symbol | ¼ symbol |

*Table 3.5: Definitions of sounding sequence tests*

| Test | LE PHY | RTT Step Type | Random Sequence Length (bits) | Number of Subevents | Expected min SNR in Reflector-Under-Test / dB | Manipulated Distance D for Attacker (meters) |
|---|---|---|---|---|---|---|
| 1 | LE 1M | Mode-1 | 32 | 100 | 25 | -3 |
| 2 | LE 1M | Mode-3 | 32 | 100 | 25 | -3 |
| 3 | LE 2M | Mode-1 | 32 | 100 | 25 | -3 |
| 4 | LE 2M | Mode-3 | 32 | 100 | 25 | -3 |
| 5 | LE 2M 2BT | Mode-1 | 32 | 100 | 25 | -3 |
| 6 | LE 2M 2BT | Mode-3 | 32 | 100 | 25 | -3 |

*Table 3.6: Definitions of random sequence tests*

After each test is complete, the Lower Tester shall compare the NADM values from each CS_SYNC packet against the random attack decision sequence it recorded earlier. For each test, 90% of the NADM values shall correctly identify the presence or absence of an attacker as defined in Section 3.5.1.

### 3.5.4 Amplitude-based attacks with DFT detection

#### 3.5.4.1 Amplitude-based attack signal definition

The amplitude-based attack signal approximates the behavior of an MITM attack using a periodic amplification profile synchronized with the symbol timing grid to create an amplitude-to-phase distortion in the victim receiver. This makes the received signal

appear with a timing advance when compared to the legitimate signal. During an amplitude-based attack, the attacker estimates the timing grid of the symbols of the victim transmitter prior to the first symbol of the CS_SYNC packet and applies a repeating attack pattern $A_p(t)$ gain profile that has the same period as the packet symbols, $T_{sym}$. The attacker shifts the timing offset of the attack pattern relative to the starting position of each symbol by a fraction of the symbol period. Aligning the timing offset of the attack pattern is necessary for the attacker to create the desired distortion in the victim receiver.

The attack begins at the first symbol of the preamble. Let $n_0$ represent the starting bit position of the preamble. Thus, for the LE 1M PHY, $n_0$=-8, and for the LE 2M and LE 2M 2BT PHY, $n_0$=-16. The indices $n = n_0 \ldots -1$ then represent the CS_SYNC preamble symbol positions as described in Section 2.1. Further, let the indices $n = 0 \ldots N-1$ represent the CS_SYNC symbol positions after the preamble as described in Section 2.2 to Section 2.5. The attack amplifier term $A_a(t)$ is defined as the summation of a repeating pattern $A_p(t)$ over symbols $n=n_0 \ldots N-1$:

$$A_a(t) = \sum_{n=n_0}^{N-1} A_p(t - n \times T_{sym})$$

Figure 3.4 depicts the relationship of the attack amplifier term $A_a(t)$ relative to the attack pattern term $A_p(t)$. $A_a(t)$ is defined over the packet time interval starting at time $n_0 \times T_{sym}$ and ending at time $N \times T_{sym}$. $A_p(t)$ is defined symbol time interval starting at time 0 and ending at time $T_{sym}$. To align $A_p(t)$ with the desired symbol window, $A_p(t)$ is shifted by $-n \times T_{sym}$. When $n$ is positive, $A_p(t)$ is shifted to the right and when $n$ is negative, it is shifted to the left.

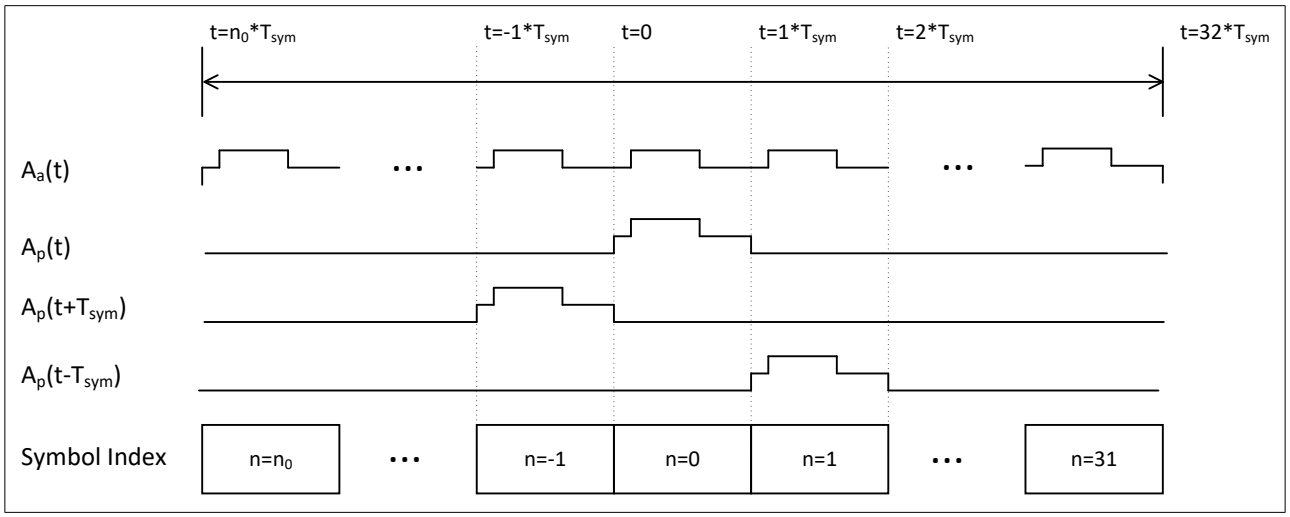



*Figure 3.4: Waveforms depicting the relationship of Aa(t) and Ap(t)*

The amplitude-based attack signal $x_a(t)$ is the time varying periodic change in the magnitude of the signal at the victim's receiving antenna $x(t)$ when multiplied by the real attack amplifier term $A_a(t)$:

$$x_a(t) = x(t) \times A_a(t)$$

The victim receiver down-converts and filters $x_a(t)$. Thus, the timing manipulation caused by $A_a(t)$ in the victim receiver is also dependent on the characteristics of the victim receiver. A carefully selected amplifier term that is properly aligned to the symbol timing grid may cause a distortion in the victim receiver that makes the attacked signal appear with a timing advance relative to the legitimate signal. The attacker characterizes the victim receiver's device and chooses the parameters that define the pattern $A_p(t)$ (defined in Section 3.5.4.1.1) and the phase offset $p_o$ (described in Section 3.5.4.3.2) effective to manipulate the distance by the desired amount.

#### 3.5.4.1.1 Square wave attack pattern

A square wave attack pattern simulates an attacker with a fixed gain amplifier with gain $A_g$ that is pulsed on for a duty cycle $DC$, which is a fraction of a symbol period and is offset by a fraction of a symbol period $p_o$. This attack pattern is applied at a frequency that is a multiple of the legitimate signal symbol frequency. While the amplifier is pulsed off, the gain of the system is the unity gain of 1.0 magnitude of the legitimate signal. $A_p(t)$ for the square wave with the frequency equal to the legitimate signal's symbol frequency is given by the following pseudo code:

$$A_p(t) = \begin{cases} 0, & if \quad t < 0 \ \ else \\ A_g, & if \quad t \leq (DC + p_o - 1) \times T_{sym} \ \ else \\ 1, & if \quad t \leq p_o \times T_{sym} \ \ else \\ A_g, & if \quad t \leq min(DC + p_o, 1) \times T_{sym} \ \ else \\ 1, & if \quad t \leq T_{sym} \ \ else \\ 0, & if \quad t > T_{sym} \end{cases}$$

$A_p(t)$ is equal to zero outside of a selected symbol period. Within this symbol period, $A_p(t)$ is equal to either 1 or $A_g$. This represents the passive pass through and application of the amplifier term respectively, within the respective symbol period. For any given symbol, the beginning point in time when the gain $A_g$ is applied is given by $p_o \times T_{sym}$. Similarly, the ending point in time is given by $(DC + p_o) \times T_{sym}$. Because $DC + p_o$ can exceed 1.0, the period where gain $A_g$ is applied can overlap the signal symbol boundaries. Figure 3.5 illustrates the case when $A_g$ is both applied within symbol boundaries and when it is applied across symbol boundaries.

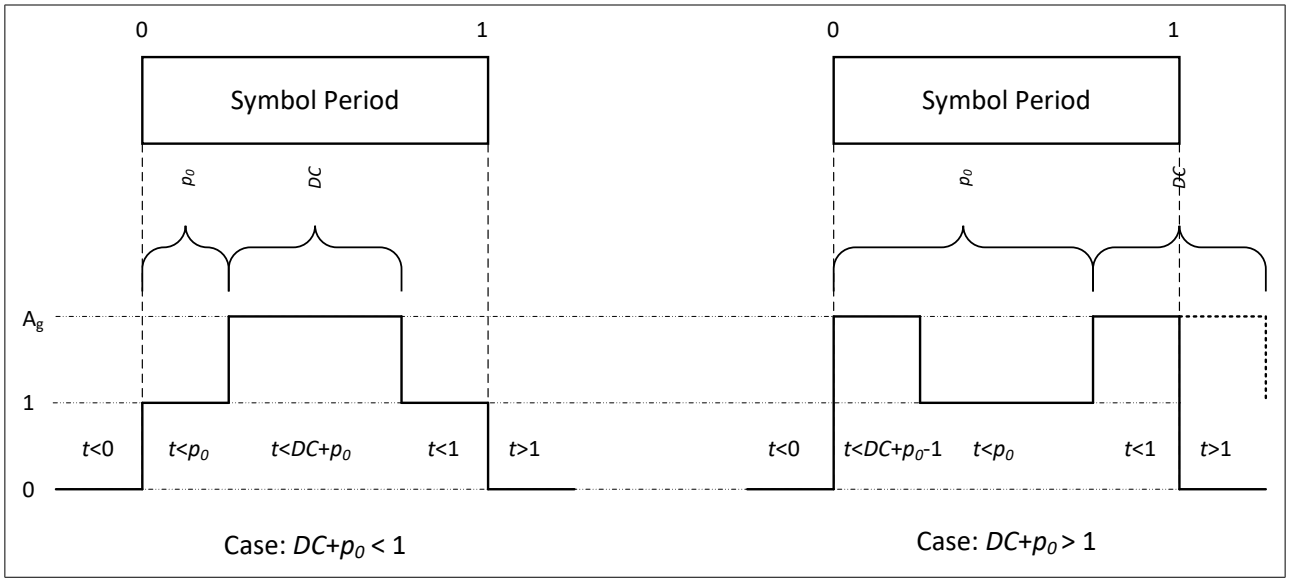



*Figure 3.5: $A_p(t)$ application of $A_g$ relative to signal symbol period timing*

#### 3.5.4.2 Raw attack detector metric based on a DFT

The presence of an amplitude-based symbol attack may be detected with the DFT metric. The periodicity of this attack creates additional distinct tones in the output spectrum, of which the periodic components are associated with the symbol rate. At the 1 Msym/s rate, this periodicity is 1 µs. At the 2 Msym/s rate, it is 0.5 µs.

The amplitude-based attack signal is defined in Section 3.5.4.1.

In the reference receiver, the following are defined:

$T_s$ represents the sampling period at the receiver.

$L_{ss}$ represents the length of the RTT random sequence (see Section 2.5) or sounding sequence (see Section 2.4) in terms of the number of symbols.

$T_{sym}$ is 1 µs for the 1 Msym/s rate and 0.5 µs for 2 Msym/s rate.

$f_1 = \frac{1}{T_{sym}}$ represents the frequency of the complex exponential function at the symbol rate used for DFT correlation and $f_2 = \frac{2}{T_{sym}}$ represents the frequency of the complex exponential function at double the symbol rate.

Then $L$, which represents one less than the number of samples taken within the integration period, is denoted as

$$L = \frac{(L_{ss} \times T_{sym})}{T_s} - 1$$

Denote the discrete time samples of the received signal as

$$X_a[n] = X_a(t_0 + nT_s)$$

where $t_0$ represents the reference time instant where the sampling starts.

Then the DFT metric is measured through the following two correlations where $f_1$ and $f_2$ can be substituted for $f_m$:

$$\varphi(f_m) = \sum_{n=0}^{L} |X_a[n]| \times e^{-j2\pi n f_m T_s}$$

$$\varphi(0) = \sum_{n=0}^{L} |X_a[n]|$$

The DFT metric is then formed by the ratio:

$$DFT\,Metric = 20 \times \log 10 \left( \frac{|(\varphi(f_1))| + |(\varphi(f_2))|}{|(\varphi(0))|} \right)$$

In general, larger values of the DFT metric will indicate a greater chance of an attack being present. There may be a residual DFT metric measured from frequency modulation to amplitude modulation conversion due to the limited bandwidth of the receive filter. The effect will be more pronounced for narrower bandwidth filters.

Transmitters should be designed to limit the amount of correlated amplitude variation to limit residual correlation to the DFT metric.

#### 3.5.4.3 Amplitude-based attack detection requirements for RTT with sounding sequence and random sequence

Devices supporting amplitude-based attack detection requirements for Channel Sounding shall meet the requirements specified in Section 3.5.3.4, in addition to the requirements specified in this section.

Section 3.1.2 defines $\Delta T_{RTT,\,PROC}$ for Device A and $\Delta T_{RESP,\,PROC}$ for Device B as the ToD and ToA error of the average of N CS_SYNC packet exchanges with respect to a tester device. Devices supporting amplitude-based attack detection requirements for Channel Sounding shall meet the 10 ns RTT requirement specified in Section 3.1.2. This limits long-term standard deviation $\sigma$ and the long-term mean of the error $B$. When the residual error term $B$ is zero, the standard deviation is constrained to $\sigma$ < 5 ns.

An effective attack amplification term $A_a(t)$ will, on average, cause a shift of $\Delta T_{RTT,\,PROC}$ and $\Delta T_{RESP,\,PROC}$ within the victim receiver (IUT) by at least -10 ns when compared to a legitimate signal. However, which attack amplification terms correspond to effective

attacks are not known in advance. Within the uncertainty of up to 5 ns of standard deviation per procedure, the IUT is first characterized to find the group of attack amplification terms that represent effective attacks. Thereafter, using this characterization information, the IUT is tested with amplitude-reduced versions of the effective attack to test the IUT's NADM implementation (described in Section Section 3.5.4.3.4).

The amplitude-based attack test requires a tester CS device that can transmit either a reference signal or an attack signal during mode-1 and mode-3 CS steps at -67 dBm power level with a Gaussian noise floor of -152 dBm/Hz for the LE1M PHY and -155 dBm/Hz for the LE2M PHY and LE2M 2BT PHY. The reference signal is as described in Section 3.5.2. The attack signal consists of the reference signal with an amplitude adjustment based on the amplification term $A_a(t)$, as described in Section 3.5.4.1. This amplification term consists of the group of attack patterns $A_p(t)$ described in Section 3.5.4.1.1.

#### 3.5.4.3.1 Determining the effectiveness of an attack point of interest

The tester device shall be capable of measuring $\Delta T_{RESP,PROC}$ for the N CS_SYNC exchanges for the under 10 ns criteria as described in Section 3.1.2 for both normally-modulated and attack-modulated signals. Let $\mu_a$ be the mean and $\sigma_a$ be the standard deviation of M separate $\Delta T_{RESP,PROC}$ measurements of attack-modulated signals. Let $B$ be the mean and $\sigma$ be the standard deviation of M separate $\Delta T_{RESP,PROC}$ measurements of normally-modulated signals.

The tester device shall be capable of testing if the attack amplification term $A_a(t)$ is effective by performing a one-sided Z-test to determine if $\mu_a$ is likely more than 10 ns less than $B$. Let the null and alternative hypothesis be:

$$H_g : \mu_a - B + 10ns \geq 0$$

$$H_a : \mu_a - B + 10ns < 0$$

For each candidate amplification term, let the Z-score be computed as follows:

$$Z = \frac{\mu_a - B + 10\,ns}{\sqrt{\frac{\sigma_a^2}{M} + \frac{\sigma^2}{M}}}$$

Let the critical Z value of -1.65 corresponding to significance level $\alpha$ = 0.05. When Z < -1.65, the tester device shall conclude that the candidate amplification term $A_a(t)$ was effective and the null hypothesis is rejected.

Section 3.5.4.3.3 describes how to employ the $\Delta T_{RESP,PROC}$ measurements to select the attack amplification terms to test. Section 3.5.4.3.4 describes the application of the Z-test on the selected amplification term points of interest.

#### 3.5.4.3.2 Parameterized requirements

The square wave attack is defined by the three parameters listed in Table 3.7:

| Parameter | Definition | Range | Index Range |
|---|---|---|---|
| $p_o$ | Time Offset of the start of the attack pattern relative to the symbol start as a fraction of $T_{sym}$ | 0.03125 to 0.96875 in 0.0625 increments | [1, 16] |
| *DC* | Duty Cycle of the Amplifier On Pulse as a fraction of $T_{sym}$ | 0.03125 to 0.96875 in 0.0625 increments | [1, 16] |
| $A_g$ | Amplifier Gain scaling factor, where 1.0 represents the original signal. | 2.0,1.9,1.8,1.7,1.6,1.5, 1.45, 1.4, 1.35, 1.3, 1.275, 1.25, 1.225, 1.2, 1.175, 1.15, 1.125, 1.1, 1.075, 1.05 | [1, 20] |

*Table 3.7: Square wave parameterization*

The attack amplification term $A_a(t)$ described in Section 3.5.4.1 is produced using a selected value of attack pattern $A_p(t)$. The attack pattern $A_p(t)$ described in Section 3.5.4.1.1 is produced using selected values of $p_o$, *DC,* and $A_g$.

The three settings define a 3-dimensional search space of unique attack amplification terms. A search algorithm is defined to find the settings that are most effective at advancing the estimated timing of the signal. In general, the combination of offset $p_o$ and *DC* define if the attack will advance the timing, and the amplifier $A_g$ will control how much the timing is advanced.

The algorithm is comprised of following phases:

1. Characterization of an IUT by search of a plane in the 3-D parameter space.
   a. For each point of interest, the minimum effective attack amplification term based on $A_g$ is determined.
2. Determine if the IUT effectively discriminates between attack and normally-modulated signals.

The following holds a list of points of interest that are determined during IUT characterization. This list is subsequently used to identify the parameters by which an IUT is tested for detection requirements.

- LocalMinList[][3] – a list with each entry holding three indices representing the selection of parameters from Table 3.7. The parameter names from Table 3.7 are used in the pseudo notation to identify which parameter value is being referenced. This list is set entirely to values of -1 to indicate that it starts as uninitialized. Each entry is subsequently populated as follows:

– LocalMinList[][$p_o$] – index in the range from 1 to the maximum index value of $p_o$ listed in Table 3.7, where a value of 1 represents the lowest value in the valid range of $p_o$, 2 represents the next higher value in the valid range, and so on.

– LocalMinList[][*DC*] – index in the range from 1 to the maximum index value of *DC* listed in Table 3.7, where a value of 1 represents the lowest value in the valid range of *DC*, 2 represents the next higher value in the valid range, and so on.

– LocalMinList[][$A_g$] – index in the range from 1 to the maximum index value of $A_g$ listed in Table 3.7, where a value of 1 represents the highest value in the valid range of $A_g$, 2 represents the next higher value in the valid range, and so on.

#### 3.5.4.3.3 Characterization requirements

A tester CS device first characterizes the IUT operating in the reflector role. A tester CS device shall operate in the initiator role to transmit a series of either normal signal or attacker signal in CS_SYNC packets during mode-1 and mode-3 steps, as described in this section.

The characterization phase fixes the amplifier gain and sweeps a cross-sectional plane by iterating over the range of offsets and duty cycles defined in Table 3.7. This two-dimensional cross-section plane represents a contour of the three-dimensional attack surface parameterized by Table 3.7. This cross-sectional plane represents a contour of the attack effectiveness and contains up to 5 ns of standard deviation as described in Section 3.5.4.3. The plane shall be defined by an index i, representing the duty cycle value ***DC*** and an index j, representing the offset value $p_o$.

To reduce the 5 ns of standard deviation, a spatial filter is applied. The resulting filtered contour is searched to find the local minimums that represent the regions of maximum timing advancement. A local minimum shall be defined as a setting that causes the most negative timing shift within a 5x5 grid centered about the point (i, j).

In this section:

- Let N represent the value used in Section 3.1.2 for satisfying the IUT's ToA and ToD RTT reporting accuracy requirements.
- Let M represent the number of repetitions of the CS procedure described in Section 3.1.2 for accumulation of statistical separation between results for attack-modulated packets versus results for normally-modulated packets. M shall be equal to 50 CS procedures.

In the equations and pseudo code that follows, *idx_16(x)* is defined as follows:

- Input x = [-1.. 18]
- Return [1, 16]

- If x is less than 1, then *idx_16(x)* = x + 16
- Else if x is greater than 16, then *idx_16(x)* = x – 16
- Else *idx_16(x)* = x

The IUT shall be characterized as follows:

1. Set the initial Amplifier Gain $A_g$ to a value of 2.0.
2. For each index i of *DC* and index j of $p_o$, perform a single procedure containing N steps of CS_SYNC packets that are attack-modulated using the fixed $A_g$ and record $\Delta T_{RESP,PROC}$ into the two dimensional array of $\mu_a[i][j]$.
3. Apply a 5x5 spatial filter to $\mu_a[i][j]$ to produce a spatially filtered attack slice $\mu_{as}[i][j]$ by performing the following steps:
   a. Let the spatial filter be

$$g = \begin{vmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 20 & 33 & 20 & 4 \\ 7 & 33 & 55 & 33 & 7 \\ 4 & 20 & 33 & 20 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{vmatrix} \times \frac{1}{331}$$

   with indices a and b ranging from g[-2][-2] at the top left to g[+2][+2] at the bottom right, such that the value 55 is at index (a=0, b=0).

   b. Apply the spatial filter by modulo convolution with the unfiltered characterization contour $\mu_a[i][j]$ such that the convolutional kernel g wraps around the edges of $\mu_a[i][j]$. Apply the spatial filter for all of $\mu_a[i][j]$ per the following equation.

$$\mu_{as}[i][j] = \sum_{a=-2}^{2} \sum_{b=-2}^{2} g[a][b] \times \mu_a[idx_16(i+a)][idx_16(j+b)]$$

4. For each entry of $\mu_{as}[i][j]$, determine if the point (i, j) represents a local minimum. This is done by determining if the index (i,j) is the smallest value within a 5x5 grid centered at point (i,j) where the search grid wraps around the edges of $\mu_a[i][j]$. Then subsequently check if the local minimum value is less than 0 ns. If so, record that as a point of interest for further testing (described in Section 3.5.4.3.4).

   Let LocalMinList[][] represent a list i and j indices of $\mu_a[i][j]$ which meet the local minimum and shift requirements described. In the following procedure, entries for LocalMinList[][$p_o$] and LocalMinList[][*DC*] are determined based on selected i and j indices. The related entry for LocalMinList[][$\boldsymbol{A_g}$] is determined in the subsequent procedure that follows in this section.

In the procedure below, minI, minJ, tmpI, tmpJ, k and m are temporary variables.

For each entry of $\mu_{as}[i][j]$

```
// The value of -1 indicates below that the variables are not set.
Initialize minI = minJ = -1;

// Test μas[i][j] for minimum shift criteria.
// Then the following two “for” loops search through the 5x5 grid.

// if μas[i][j] is greater than or equal to 0 ns then skip processing
If ( μas[i][j] ≥ 0 ) continue to the next entry of μas[i][j]

For k = -2 to 2 do
  For m = -2 to 2 do
    // Normalized k, m indices within μas depicted here.

    tmpI = idx_16( i + k );
    tmpJ = idx_16( j + m );

    // Record the respective μas point coordinates as the lowest minima for this
    // subgrid if it is either the first to be recorded for this μas[i][j] entry or it is
    // the new lowest minima.

    If (( minI == -1 ) || ( μas[tmpI][tmpJ] < μas[minI][minJ]))

      minI = tmpI;
      minJ = tmpJ;
  // end of “For m”
// end of “For k”

// For each subgrid, save off the μas indices of a valid minimum point.
// Append this to the first available entry of LocalMinList.

If ( minI != -1 && μas[i][j] == μas[minI][minJ] )
  LocalMinList[][po] = minI;
  LocalMinList[][DC] = minJ;
```

// End of “For each entry of $\mu_{as}[i][j]$”

5. If LocalMinList is completely empty, then the test concludes with a passed status, otherwise continue to incremental testing of recorded points of interest.

For each point of interest recorded in LocalMinList, a series of reduced attack amplification patterns $A_a(t)$ are created by reducing the gain $A_g$ used to create $A_p(t)$ until

the attack pattern is determined to no longer be effective. Let the index k correspond to the selected gain $A_g$ in Table 3.7 where k=1 corresponds to the maximum $A_g$ value listed in Table 3.7, and the k=maximum index value in Table 3.7 corresponds to the minimum $A_g$ value in Table 3.7. The Z-test described in Section 3.5.4.3.1 is used to determine if the attack amplification term is effective by comparing the average of M procedures of attack-modulated RTT exchanges to the characterization values described in Section 3.1.2.

The tester device shall perform M procedures consisting of N normally-modulated CS_SYNC packets. Compute the mean $B$ and standard deviation $\sigma$ of the M $\Delta T_{RESP, PROC}$ values as defined in Section 3.5.4.3.1. For each point of interest identified in LocalMinList, perform the following steps:

1. Sweep through indices $A_g$ values listed in Table 3.7 starting from k=1, incrementally reducing $A_g$.
2. Create $A_a(t)$ by applying the offset $p_o$ corresponding to the recorded point of interest. Then create $A_p(t)$ using the $DC$ corresponding to the recorded point of interest and the gain $A_g$ corresponding to the iteration index k.
3. Perform M procedures consisting of N attack-modulated CS_SYNC packets. Compute the mean $\mu_a$ and standard deviation $\sigma_a$ of the M $\Delta T_{RESP, PROC}$ values as defined in Section 3.5.4.3.1.
4. Perform the Z-test described in Section 3.5.4.3.1 that compares the mean and standard deviation of the attack amplification term to the mean and standard deviation of normally-modulated signal. When the null hypothesis described in Section 3.5.4.3.1 is not rejected, the tester device shall conclude that the index k is the first index where the attack becomes ineffective, and k-1, if a valid index, is the minimum effective attack index. The index k-1 is recorded in the respective LocalMinList[][$\boldsymbol{A_g}$] entry.

#### 3.5.4.3.4 Detection requirements

This section describes how an IUT NADM implementation is tested to determine if it effectively discriminates between amplitude-based attack-modulated signals and normally-modulated signals.

For each point of interest with a valid amplification term $A_g$ entry aggregated in LocalMinList, the minimum effective attack amplification term is tested to check that the IUT's NADM implementation effectively detects the presence of an attack.

The tester device acts as an initiator to start a CS procedure with the IUT. Each test comprises one or more CS procedures executing the total CS_Sync step exchanges where the IUT returns a NADM value not marked as "Unknown" over a maximum CS subevent count specified in Table 3.8. Each CS subevent comprises a normal mode-0

CS step followed by at least a single mode-1 CS step or a mode-3 CS step with the specified sounding sequence and random sequence lengths.

For each CS_Sync step, the tester device makes a random decision to transmit either a normal signal or an attacker signal. The CS device record these random decisions over the whole set of CS procedures. These random decision sequences are later compared with resulting NADM values from the IUT.

Table 3.8 defines multiple mandatory tests that depend on the supported capabilities of the IUT. The tests verify the minimum performance for a NADM value. CS_SYNC packets are used in mode-1 or mode-3 steps and are both tested. LE 1M, LE 2M, and LE 2M 2BT are also tested.

| Test | LE PHY | CS_Sync Type | Random or Sounding Sequence Length | Number of CS_Sync Steps / Subevents | Expected min SNR in Reflector-Under-Test / dB |
|---|---|---|---|---|---|
| 1 | LE 1M | Mode-1 | 32 | 100 | 25 |
| 2 | LE 1M | Mode-3 | 32 | 100 | 25 |
| 3 | LE 2M | Mode-1 | 32 | 100 | 25 |
| 4 | LE 2M | Mode-3 | 32 | 100 | 25 |
| 5 | LE 2M 2BT | Mode-1 | 32 | 100 | 25 |
| 6 | LE 2M 2BT | Mode-3 | 32 | 100 | 25 |

*Table 3.8: Definitions of random sequence and sounding sequence tests*

For each point of interest listed in LocalMinList with a LocalMinList[][$\boldsymbol{A_g}$] value within the valid range of $A_g$ as reflected in Table 3.7, the tester device shall perform the following steps to select the attack amplification term for NADM testing:

1. Create $A_p(t)$ using the offset $p_o$ corresponding to index LocalMinList[][$p_o$], the *DC* corresponding to index LocalMinList[][*DC*] and the gain $A_g$ corresponding to the recorded value LocalMinList[][$\boldsymbol{A_g}$]. Create $A_a(t)$ by applying this value of $A_p(t)$. Set LocalMinList[][$\boldsymbol{A_g}$] = LocalMinList[][$\boldsymbol{A_g}$] - 1.
2. Send the number of subevents reflected in Table 3.8 to the IUT. Half of the CS_Sync steps shall contain normally-modulated signals and half shall contain attack-modulated signals, selected at random. Record the IUT's NADM value for each CS_Sync step.
3. Compare the NADM values from each CS_SYNC packet for which NADM results are not marked as "Unknown" against the random attack decision selected in step 2. For each test, 90% of the NADM values shall correctly identify the presence or absence of an attacker as defined in Section 3.5.1.

4. The test concludes either when the number of subevents in Table 3.8 have been sent or when the IUT responds to the number of CS_Sync steps in Table 3.8 with NADM results that are not marked as “Unknown”, whichever occurs first.

If the IUT did not respond with the number of CS_Sync steps in Table 3.8 with NADM results that are not marked as “Unknown”, then the test concludes in failure.

If the IUT incorrectly identifies the NADM values relative to the presence or absence of an attack as described step 3 above, then the test concludes in failure.

The tester device shall repeat this procedure starting at step 1 until LocalMinList[][$\boldsymbol{A_g}$] is no longer within the valid range of $A_g$ as reflected in Table 3.7.

When all points of interest have been tested, the test shall conclude in success.

# 4 CHANNEL SOUNDING INTERFACE PROTOCOL

## 4.1 Channel selection Algorithm #3

The algorithms described in this section support Channel Selection for each CS step within a CS procedure. A CS step involves the execution of one of four CS modes, described in Section 4.3. The algorithms described here generate a set of channel indices on which the mode exchanges take place.

Two sets of channel indices are generated. The first sequence defines the hopping pattern used by mode-0 CS steps. The second sequence defines the hop pattern used by non-mode-0 CS steps. Two separate algorithms are defined for the latter sequence.

Each sequence is further filtered by the channel index filter bit map CSFilteredChM, which is described in [Vol 6] Part B, Section 5.1.28. In that bit map, a bit set to 1 indicates that the channel represented by that bit shall be included in the resulting hop set. A bit set to 0 indicates that the channel represented by that bit shall not be included in the resulting hop set. Prohibited channel indices shall be immediately omitted from the resulting hop set (see [Vol 6] Part A, Section 2).

Each resulting channel index array shall be used starting from index 0 in ascending order whenever a CS step (either mode-0 or non-mode-0) is executed. Each entry in the resulting channel index array shall only be used once unless otherwise specified in this specification. After it is exhausted, this array shall be regenerated at the CS step where a new channel index is needed.

### 4.1.1 Conventions

The following procedural conventions are used in this description of Channel Selection Algorithm #3.

| Function | Meaning |
|---|---|
| length_of() | Returns the populated length of an unsigned integer array |

*Table 4.1: Procedural conventions*

### 4.1.2 Channel index shuffling function cr1

When deriving the channel index selection within a CS procedure, the CS DRBG is invoked to shuffle the list of available channels. The channel index shuffling function cr1 is defined for this purpose. The inputs to cr1 are a one-dimensional unsigned integer array, ChannelArray, that lists all available channel indices, and the length of that array, nChannels.

The following are values returned from cr1:

ShuffledChannelArray, a one-dimensional unsigned integer array of length nChannels

The following temporary values are used:

i, j are unsigned integers

The input/output format of cr1 is as follows:

ShuffledChannelArray = cr1( nChannels, ChannelArray )

cr1 processing is as follows:

for i = 0 to nChannels – 1 do

j = random integer such that 0 ≤ j ≤ i

if i ≠ j

ShuffledChannelArray[ i ] = ShuffledChannelArray[ j ]

ShuffledChannelArray[ j ] = ChannelArray[ i ]

The random integer j noted above shall be seeded using the random number generation function hr1 (see Section 4.8.1). The random integer j shall be computed using the increasing index i from above as follows:

hr( i + 1 )

### 4.1.3 Channel selection Algorithm #3a for mode-0 steps

Channel Selection Algorithm #3a is used to generate a randomized channel map with uniform distribution for mode-0 CS steps.

The number of times that the mode-0 channel list is regenerated in a CS procedure depends on several factors, including:

- The size of the list of included filtered channels as indicated by the filter channel bit map CSFilteredChM
- The number of mode-0 steps included in a CS subevent
- The number of CS subevents within a CS procedure
- The run-time construction of CS subevents, whose content may vary from subevent to subevent as described in Section 4.4

The randomized channel map is created by first converting the filtered channel bit map to a one-dimensional unsigned integer array of channel indices that shall include only the indices of the CSFilteredChM marked as included, as described in Section 4.1. This intermediate array is called the filteredChannelArray. The filteredChannelArray is used to create a shuffled channel list in an array of the same size, the Mode0ShuffledChannelArray. The Mode0ShuffledChannelArray is generated using the channel index shuffling function cr1 (see Section 4.1.2) as follows:

> Mode0ShuffledChannelArray = cr1( length_of( filteredChannelArray ), filteredChannelArray )

### 4.1.4 Channel index selection for non-mode-0 steps

Channel Selection Algorithms #3b and #3c support channel selection for each non-mode-0 CS step within a CS procedure. Only one of the two algorithms shall be used within a single CS procedure.

The number of times the non-mode-0 channel list shall be regenerated in a CS procedure depends on several factors, including:

- The size of the list of included filtered channels as indicated by the filter channel bit map CSFilteredChM
- The number of times the filtered channel map is repeated in a procedure for non-mode-0 steps, as defined by CSNumRepetitions (see [Vol 6] Part B, Section 5.1.26)
- The run-time construction of CS subevents whose content may vary from subevent to subevent, as described in Section 4.4

#### 4.1.4.1 Channel Selection Algorithm #3b

Channel Selection Algorithm #3b is used to generate a randomized channel map with uniform distribution for non-mode-0 CS steps.

The filtered channel bit map is first converted to a one-dimensional unsigned integer array of channel indices that shall include only the indices of the CSFilteredChM marked as included, as described in Section 4.1. This intermediate array is called the filteredChannelArray. The resulting shuffled channel list is held in an array of the same size and is known as NonMode0ShuffledChannelArray. To generate this array, the channel index shuffling function cr1 (see Section 4.1.2) shall be invoked as follows:

> NonMode0ShuffledChannelArray = cr1( length_of( filteredChannelArray ), filteredChannelArray )

#### 4.1.4.2 Channel Selection Algorithm #3c

Channel Selection Algorithm #3c integrates rising and falling ramps into the resulting channel map for non-mode-0 CS steps. These ramps are useful for estimating timing

drift and object motion correction. The ramps have the shape of either a hat or an X pattern. In a hat shape, a rising ramp is directly followed by a falling ramp. In an X pattern, rising and falling ramps are interleaved. These shapes are incorporated into a shuffled channel map so that they appear random, preserving equal channel distribution qualities.

A block diagram of the overall algorithm is shown in Figure 4.1. Each process block, as well as inputs and outputs, are described in subsequent sections.

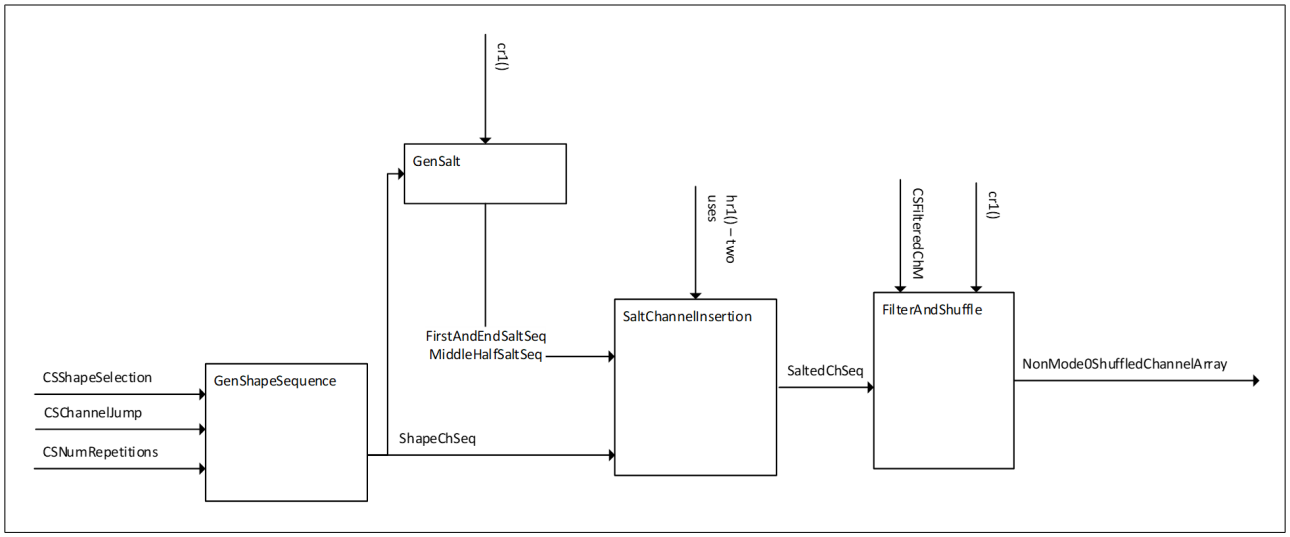



*Figure 4.1: Channel Selection Algorithm #3c functional block diagram*

#### 4.1.4.2.1 Inputs and basic components

Channel Selection Algorithm #3c has the following inputs:

- CSShapeSelection – indicates the selection of either the hat shape or the X pattern.
- CSChannelJump – indicates the number of channels skipped over when rendering the selected shape, based on the CS channel index values described in Section 1.
- CSNumRepetitions – indicates the number of times Channel Selection Algorithm #3c is invoked. This number is the same as the number of times the filtered channel bit map is repeated within that CS procedure.

Table 4.2 shows additional algorithm-related parameters. The value selected for CSChannelJump determines which row in Table 4.2 shall be used in Channel Selection Algorithm #3c.

| CSChannelJump | seq1StartCh | seq2StartCh | maxRepsAllowed | saltRate |
|---|---|---|---|---|
| 2 | 1 | 76 | 1 | 2 |
| 3 | 77 | 0 | 1 | 2 |
| 4 | 78 | 0 | 2 | 2 |
| 5 | 78 | 0 | 2 | 2 |
| 6 | 76 | 1 | 3 | 2 |
| 7 | 74 | 1 | 3 | 2 |
| 8 | 76 | 0 | 3 | 2 |

*Table 4.2: Channel Selection Algorithm #3c parameter blocks based on CSChannelJump parameter selection*

The values of seq1StartCh and seq2StartCh denote the starting channel index used to construct the two ramps of the selected shape indicated by CSShapeSelection. These parameters refer to the CS channel index values described in Section 1.

maxRepsAllowed is the upper limit of the CSNumRepetitions parameter. The value of CSNumRepetitions shall be greater than or equal to 1 and less than or equal to maxRepsAllowed.

saltRate is the rate at which alternate hop channels are inserted into the constructed hopping pattern. These alternate hop channels shall not include the channels used in the shape construction selected by the CSChannelShape parameter. These alternate channels are mixed into the hopping pattern for uniform channel distribution containing the valid set of CS physical channels. This alternate set of channels is referred to as salt channels.

Multiple instances of the non-mode-0 channel map, and therefore multiple iterations of the Channel Selection Algorithm #3c procedure, are required if CSNumRepetitions is greater than 1. The integer value nShapeIteration shall be maintained for each active CS procedure. nShapeIteration shall be initialized to 0 for the first iteration of this algorithm within that CS procedure and shall be incremented by one before the next iteration of the Channel Selection Algorithm #3c procedure.

#### 4.1.4.2.2 Shape generation

Each invocation of Channel Selection Algorithm #3c begins with the identification of the channels used to hold the shape selected by the CSShapeSelection parameter. The function of the process block GenShapeSequence shown in Figure 4.1 is described in this section.

The shape generation function returns the following:

- ShapeChSeq – a one-dimensional unsigned integer array that holds the channel set constructed to form the selected shape

For the purpose of long-term uniform channel distribution, the starting position of the channel sequence that constructs the selected shape is shifted by a value from 0 to CSChannelJump -1 channel indices. The random number generation function hr1 (see Section 4.8.1) is invoked to produce the random startJitter value. If this is the first instance of the non-mode-0 channel map generation procedure with a CS procedure, which shall be the case when nShapeIteration is equal to 0, then the startJitter value is produced as follows:

startJitter = hr1( CSChannelJump )

Otherwise, the value of startJitter shall be held constant for the remaining duration of the CS procedure.

The ShapeChSeq content is then generated as follows:

- The shape ramp offset values, s1Ch and s2Ch, are calculated as follows:

$$offset = (nShapeIteration + startJitter) \bmod CSChannelJump$$
$$s1Ch = seq1StartCh + offset$$
$$s2Ch = seq2StartCh + offset$$

- Each shape ramp is computed starting from the channels identified by s1Ch and s2Ch. Channels are appended through the valid channel map range defined by the CS channel index values described in Section 1. Each computed channel value is appended to the ShapeChSeq array in turn.
- If CSShapeSelection is the X pattern, the channels offset from s1Ch and s2Ch are interleaved when appended to ShapeChSeq starting with s1Ch. If seq1StartCh is less than seq2StartCh, an increment value inc is set to CSChannelJump. If seq1StartCh is greater than or equal to seq2StartCh, inc is set to -1xCSChannelJump. Starting with s1Ch, s1Ch and s2Ch are alternately recomputed as follows and appended to the ShapeChSeq array. For each of the two individual computations, the first few sequential values may fall outside of the channel map range (see Section 1) and if so, those individual values shall not be appended to the ShapeChSeq array. Thereafter, while each newly computed value still remains within the valid channel map range, the values are appended similarly. If one of the two values then falls out of the valid channel map range, then the other value continues to be processed and appended to the ShapeChSeq array until it also falls out of the valid channel map range.

$$s1Ch = s1Ch + inc$$
$$s2Ch = s2Ch - inc$$

- If CSShapeSelection is the hat pattern, the entire rising ramp is appended to ShapeChSeq before the falling ramp by determining which ramp s1Ch and s2Ch represent, using the following procedure. In the procedure, risingCh is the set of channel indices for the rising ramp and fallingCh is the set of channel indices for the falling ramp.
  - If s1Ch is less than s2Ch, then risingCh = s1Ch and fallingCh = s2Ch. If s1Ch is greater than or equal to s2Ch, risingCh = s2Ch and fallingCh = s1Ch.
  - The increment value of CSChannelJump is used.
  - The values based on risingCh are computed first. The first few sequential values may fall outside of the valid channel map range (see Section 1) and if so, shall not be appended to the ShapeChSeq array. Thereafter, the next computed values are appended to the ShapeChSeq array while they remain within the valid channel map range.

$$risingCh = risingCh + CSChannelJump$$

  - Thereafter, the values based on fallingCh are computed. The first few sequential values may fall outside of the valid channel map range (see Section 1) and if so, shall not be appended to the ShapeChSeq array. Thereafter, the next computed values are appended to the ShapeChSeq array while they remain within the valid channel map range.

$$fallingCh = fallingCh - CSChannelJump$$

#### 4.1.4.2.3 Salt generation

Salt channels are used to generate a uniform channel distribution in the valid set of CS physical channels, as defined in Section 1. The function of the process block GenSalt, shown in Figure 4.1, is described in this section.

The salt channel generation function returns the following:

- FirstAndEndSaltChSeq – a one-dimensional unsigned integer array holding an intermediate salt channel list
- MiddleSaltChSeq – another one-dimensional unsigned integer array holding a second intermediate salt channel list

Salt channels are channels that are not part of the shape generation list ShapeChSeq described in Section 4.1.4.2.2 but are still valid CS channel index values as described in Section 1. The list of available salt channels for a specific hop channel generation procedure is derived by direct comparison against the shape generation channel list ShapeChSeq.

The list of candidate salt channels is populated in a one-dimensional unsigned integer array called unusedChSeq, which is first initialized as an empty list. Starting from

channel index 0 in ascending order, each valid CS channel index as described in Section 1 is compared to the ShapeChSeq array. If a channel index is not present, then that channel index is appended to the unusedChSeq list.

The valid CS physical channel range is then divided into four quadrants which approximately surround the channel shape held in the ShapeChSeq. This grouping is shown in Figure 4.2. In the figure, the vertical lines show stepwise sequence boundaries, and the horizontal lines show channel segmentation boundaries for the entire valid CS physical channel range. The boxes denote a delineation of the salt channel set. The boxes labeled First and End contain the same set of channels, corresponding to the first and end quadrants of the shape. Approximately 50% of the valid CS physical channel range is divided between the First and the End, and the remaining physical channel range is contained within the Middle boxes.

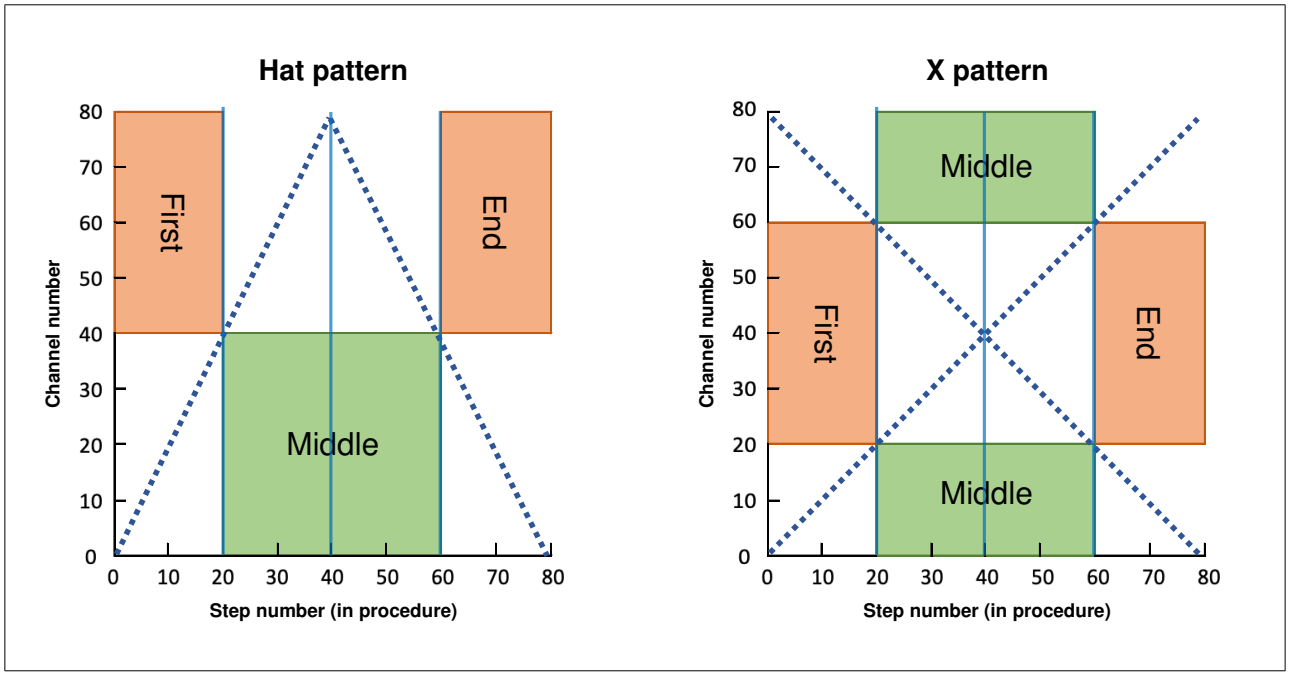



*Figure 4.2: Salt channel groupings for both the hat and the X pattern (represented by the blue dots)*

Four unsigned integer arrays holding intermediate sequences divide the valid CS physical channel range into four approximately equal sized regions:

chQ1All = [0, 19]

chQ2All = [20, 39]

chQ3All = [40, 59]

chQ4All = [60, 78]

From the arrays above, a second set of intermediate sequences are constructed and held in another set of four unsigned integer arrays initialized as empty lists: chQ1Unused, chQ2Unused, chQ3Unused, and chQ4Unused. The content of each of these four channel sets is compared to the previous unusedChSeq in search of common channels. Each channel in chQ1All is compared with the content of unusedChSeq. If a channel in chQ1All appears in unusedChSeq, it is appended to the chQ1Unused array. This process is repeated to construct the content of chQ2Unused, chQ3Unused, and chQ4Unused.

From the content generated for chQ1Unused, chQ2Unused, chQ3Unused, and chQ4Unused, a third set of intermediate sequences is constructed and held in another set of four unsigned integer arrays: firstAndEndUnusedChSeq, firstAndEndAllChSeq, middleUnusedChSeq, and middleAllChSeq. The construction of these sequences depends on the shape selected with CSShapeSelection.

- If CSShapeSelection is the X pattern, then
  - firstAndEndAllChSeq is constructed by concatenating chQ3All to the end of chQ2All.
  - middleAllChSeq is constructed by concatenating chQ4All to the end of chQ1All.
  - firstAndEndUnusedChSeq is constructed by concatenating chQ3Unused to the end of chQ2Unused.
  - middleUnusedChSeq is constructed by concatenating chQ4Unused to the end of chQ1Unused.
- If CSShapeSelection is the hat pattern, then
  - firstAndEndAllChSeq is constructed by concatenating chQ4All to the end of chQ3All.
  - middleAllChSeq is constructed by concatenating chQ2All to the end of chQ1All.
  - firstAndEndUnusedChSeq is constructed by concatenating chQ4Unused to the end of chQ3Unused.
  - middleUnusedChSeq is constructed by concatenating chQ2Unused to the end of chQ1Unused.

The channel index shuffle function cr1 (see Section 4.1.2) shall then be invoked on each of the above four intermediate channel sequences as follows:

firstAndEndAllChSeq = cr1( firstAndEndAllChSeq, length_of( firstAndEndAllChSeq ))

middleAllChSeq = cr1( middleAllChSeq, length_of( middleAllChSeq))

firstAndEndUnusedChSeq = cr1( firstAndEndUsusedChSeq, length_of( firstAndEndUsusedChSeq))

middleUnusedChSeq = cr1( middleUnusedChSeq, length_of( middleUnusedChSeq))

FirstAndEndSaltChSeq is then constructed by concatenating firstAndEndAllChSeq to the end of firstAndEndUnusedChSeq. MiddleSaltChSeq is constructed by concatenating middleAllChSeq to the end of middleUnusedChSeq.

#### 4.1.4.2.4 Shape and salt channel mixing

After salt channels are generated, shape channels are mixed with salt channels. The function process block SaltChannelInsertion shown in Figure 4.1 is described in this section.

The following is returned from the salt channel insertion function:

- SaltedChSeq – a one-dimensional unsigned integer array containing the salted shape channel sequence

SaltedChSeq is first initialized as an empty list.

If this is the first instance of non-mode-0 channel map generation procedure within a CS procedure, which shall be the case when nShapeIteration is equal to 0, then a random set of between 0 and CS3C_N_INITIAL_SALT salt channels shall be appended to SaltedChSeq as follows:

- CS3C_N_INITIAL_SALT shall be equal to 9.
- The random number generation function hr1 (see Section 4.8.1) is invoked to generate the random nInitialSalt value as follows:
  - nInitialSalt = hr1( CS3C_N_INITIAL_SALT + 1 )
- Beginning from element 0, nInitialSalt channels are appended to SaltedChSeq, starting with the FirstAndEndSaltChSeq array and alternating between the FirstAndEndSaltChSeq array and the MiddleSaltChSeq array.

Shape channels and salt channels are then appended in a mixing fashion. Channels from arrays ShapeChSeq (starting from element 0), FirstAndEndSaltChSeq, and

MiddleSaltChSeq are selected and appended to the SaltedChSeq array. Channels from the originating arrays shall be consumed in ascending order and never reused. Previously consumed channels from the FirstAndEndSaltChSeq and MiddleSaltChSeq arrays are also not reused. Channels from these arrays are consumed starting with the next available entry. In the procedure below, i and j are temporary variables.

```
For i = 0 to length_of( ShapeChSeq ) - 1 do

    if ( i mod saltRate ) == 0

        Append a salt step to SaltedChSeq as described below.

    Append the next entry from ShapeChSeq to SaltedChSeq.
```

In the procedure above, when a salt step is appended to the SaltedChSeq, the following procedure shall be used.

```
j = ⌊( ShapeChSeq[i] ÷ 20 )⌋ + 1

if CSShapeSelection is the X pattern

    if ( j == 1 ) or ( j == 4 )

        Append the next entry from FirstAndEndSaltChSeq to SaltedChSeq

    else

        Append the next entry from MiddleSaltChSeq to SaltedChSeq

else if CSShapeSelection is the hat pattern

    if ( j == 1 ) or ( j == 2 )

        Append the next entry from FirstAndEndSaltChSeq to SaltedChSeq

    else

        Append the next entry from MiddleSaltChSeq to SaltedChSeq
```

If this procedure is the last instance of the non-mode-0 channel map generation procedure within the CS procedure, which shall be the case when

nShapeIteration is equal to CSNumRepetitions-1, then a random set of between 0 to CS3C_N_FINAL_SALT salt channels is appended to SaltedChSeq as follows:

- CS3C_N_FINAL_SALT shall be equal to 4.
- The random number generation function hr1 (see Section 4.8.1) is invoked to generate a random nFinalSalt value as follows:
  - nFinalSalt = hr1( CS3C_N_FINAL_SALT + 1 )
- nFinalSalt channels are appended to SaltedChSeq, starting first from the next available FirstAndEndSaltChSeq array and alternating between the next available FirstAndEndSaltChSeq array and the next available MiddleSaltChSeq array.

If more channels from FirstAndEndSaltChSeq than from MiddleSaltChSeq were appended to the SaltedChSeq, then the deficit amount of the next available entries from MiddleSaltChSeq are appended to the end of SaltedChSeq. Alternatively, if more channels from MiddleSaltChSeq than from FirstAndEndSaltChSeq were appended to the SaltedChSeq, then the deficit amount of the next available entries from FirstAndEndSaltChSeq are appended to the end of SaltedChSeq.

Any remaining entries from the FirstAndEndSaltChSeq and MiddleSaltChSeq shall be discarded.

#### 4.1.4.2.5 Filtering and shuffling

After the SaltedChSeq array is generated, it is then filtered through the filter channel bit map CSFilteredChM described in [Vol 6] Part B, Section 5.1.28. The resulting filtered channel set is then shuffled in a block-based fashion. The function process block FilterAndShuffle shown in Figure 4.1 is described in this section.

The following is returned from the filtering and shuffling function. It is also the final output of Channel Selection Algorithm #3c:

- NonMode0ShuffledChannelArray – a one dimensional unsigned integer array containing the non-mode-0 channel index array

If this is the first instance of the non-mode-0 channel map generation procedure with a CS procedure, which shall be the case when nShapeIteration is equal to 0, then NonMode0ShuffledChannelArray is initialized as an empty list. Otherwise, the prior content of NonMode0ShuffledChannelArray is appended to by the procedure described in this section.

filteredSaltedChSeq is an intermediate unsigned integer array that holds the filtered channel content of the SaltedChSeq array. filteredSaltedChSeq is initialized as an empty list.

The content of the SaltedChSeq is filtered using the content of the CSFilteredChM bit map. Each bit position of the CSFilteredChM starting from bit 0, represents a valid CS channel index as described in Section 1. If a bit is set to 0, then that channel index is filtered out. Otherwise, if a bit is set to 1, then that channel index is allowed for use within a CS procedure. Starting from the first element, the content of the SaltedChSeq array is compared with the content of the CSFilteredChM bit map. Channels that are not filtered out are then appended to the filtered sequence filteredSaltedChSeq in order. Channels that are filtered out are discarded.

The content remaining in filteredSaltedChSeq is then shuffled in smaller block quantities. The size of each block is computed as follows.

$$nStepsInBlock = \max\left(10, floor\left(\frac{length_of(\text{ filteredSaltedChSeq })}{4}\right)\right)$$

Therefore, the number of blocks contained in the filteredSaltedChSeq array can be derived as follows.

$$nBlocksToShuffle = \max\left(1, floor\left(\frac{length_of(\text{ filteredSaltedChSeq })}{nStepsInBlock}\right)\right)$$

NonMode0ShuffledChannelArray is then constructed by shuffling filteredSaltedChSeq in blocks of nStepsInBlock channels. This starts from the beginning of the filteredSaltedChSeq and progresses through the entire array content. The channel index shuffling function cr1 (see Section 4.1.2) shall be invoked as follows. Here i is a temporary unsigned integer and tempBlockSeq is a temporary unsigned integer array. filteredSaltedChSeq is referenced starting from the first element at array position 0.

```
For i = 0 to nBlocksToShuffle-1

    delete all content from tempBlockSeq

    if i < nBlocksToShuffle-1 // not the last block in filteredsaltedChSeq

        Copy elements ( i x nStepsInBlock ) to ((( i + 1 ) x nStepsInBlock ) – 1 ) of
        filteredChSeq to tempBlockSeq

    else

        Copy remaining elements filteredChSeq to tempBlockSeq

    tempBlockSeq = cr1( tempBlockSeq, length_of( tempBlockSeq ))

    Append tempBlockSeq to NonMode0ShuffledChannelArray
```

## 4.2 Channel Sounding channel indices

Channel Selection Algorithm #3 shall be used to define the frequency hop pattern for all CS steps within a CS procedure. This hop pattern is seeded by the CSChM and the CSNumRepetitions parameters of the CS configuration selected for that procedure (see [Vol 6] Part B, Section 5.1.26) as well as the running channel map update (see [Vol 6] Part B, Section 5.1.28). The maximum number of non-mode-0 steps included in a CS procedure shall be bounded by the number of used channels in the CSFilteredChM channel map (see [Vol 6] Part B, Section 5.1.28) multiplied by the CSNumRepetitions parameter, plus any salted channel insertions as described in Section 4.1.4.2 and any steps that reuse channels as described in this section and in Section 4.4.

A CS procedure is considered complete and closed when at least one of the following conditions occurs.

- The execution of the next CS step in its entirety would cause the extent of the CS procedure to exceed T_MAX_PROCEDURE_LEN.
- The combined number of mode-0 steps and non-mode-0 steps executed is equal to N_STEPS_MAX as described in [Vol 6] Part B, Section 4.5.18.1.
- The number of CS subevents executed is equal to N_MAX_SUBEVENTS_PER_PROCEDURE.
- If Channel Selection Algorithm #3b is used for non-mode-0 steps, and the channel map generated from CSFilteredChM has been used for CSNumRepetitions cycles for non-mode-0 steps including the use for both Main_Mode and Sub_Mode steps.
- If Channel Selection Algorithm #3c is used for non-mode-0 steps, and CSNumRepetitions invocations of Channel Selection Algorithm #3c have been completed for non-mode-0 steps including the use for both Main_Mode and Sub_Mode steps.

For CS steps besides those in mode-0, a new channel shall be selected from the Channel Selection Algorithm #3 non-mode-0 shuffled channel list each time a Main_Mode step is transmitted. The channel selected shall be re-used (i.e., not refreshed from the shuffled channel list content) when a subsequent mode-1 Sub_Mode step is scheduled. If no subsequent mode-1 Sub_Mode step is scheduled, a new channel shall be selected from the shuffled channel list for the Sub_Mode transmission. The channel selected shall also be re-used when repeating CS steps at the beginning of a CS subevent relative to the steps at the end of the previous CS subevent, as described in Section 4.4.

For mode-0 CS steps, a new channel shall be selected from the Channel Selection Algorithm #3 mode-0 shuffled channel list each time a mode-0 step is scheduled.

## 4.3 Channel Sounding steps

CS defines a set of interlocking transfers between two devices. The device initiating the CS procedures is known as the initiator, and the device responding to those procedures is known as the reflector. Within a CS procedure, *CSStepCount* shall be set to the CS step number, which starts at zero and is incremented by one at each subsequent step within that procedure, even if that step is not executed. CS steps are composed of a combination of modulated packets and modulated CS tones.

There are four different modes for the CS steps. Each of these modes has different usage goals: measuring frequency offset between devices (mode-0), measuring round-trip times (mode-1), measuring phase rotations due to distance (mode-2), and measuring both round-trip times and phase rotations (mode-3).

CS steps require a precise timing synchronization between the devices involved in the transactions (i.e., the initiator and the reflector). This timing synchronization is described in Section 4.5.

CS steps are separated by time periods that allow for performing a frequency hop. The structure is shown in Figure 4.3.

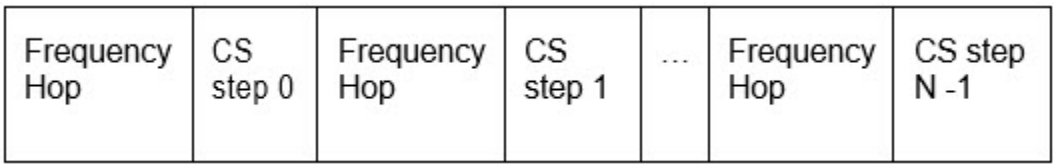



*Figure 4.3: A frequency hop period separates each CS step from the next*

The time allowed for the frequency hop is known as T_FCS. The permitted values for T_FCS are shown in Table 4.3.

| T_FCS Index | T_FCS | Mandatory/Optional/Conditional |
|---|---|---|
| 0 | 15 µs | O |
| 1 | 20 µs | O |
| 2 | 30 µs | O |
| 3 | 40 µs | C.1 |
| 4 | 50 µs | O |
| 5 | 60 µs | O |
| 6 | 80 µs | C.1 |
| 7 | 100 µs | O |
| 8 | 120 µs | O |

| T_FCS Index | T_FCS | Mandatory/Optional/Conditional |
|---|---|---|
| 9 | 150 µs | M |
| C.1: Mandatory if any shorter T_FCS value is supported, otherwise optional. | | |

*Table 4.3: Permitted T_FCS values*

Within a CS procedure, the same T_FCS value shall be used for each of the frequency hops. Devices may use the T_FCS period to perform internal calibrations, in addition to performing the frequency change. Devices may also use this time period to allow settling of the next transmitted RF signal so that it has reached a stable state when the subsequent CS step begins.

During T_FCS, the output power of the reflector in the RF channel that is being switched to shall be at least 40 dB less than the output power to be used for transmissions by the reflector during the next CS step, as measured at the reflector's antenna connector.

### 4.3.1 Channel Sounding step mode-0

CS step mode-0 is used to measure the frequency offset between the initiator and the reflector at a given frequency. Devices supporting the CS feature shall implement CS step mode-0. Table 4.4 shows the structure for CS step mode-0.

The CS initiator performs a measurement of the frequency offset of the transmitted RF signal from the CS reflector during CS step mode-0. The CS initiator shall use this information to compensate for the frequency and timing error contributions in all further transmissions within subsequent CS steps in a CS subevent. For each CS subevent, the CS initiator shall report to the Host the frequency value used for this compensation. Requirements for this compensation are described in [Vol 6] Part A, Section 3.5.

| Role/Duration | T_SY | T_RD | T_IP1 | T_SY + T_GD + T_FM | T_RD |
|---|---|---|---|---|---|
| Initiator | CS_SYNC (CS_SYNC_0_I) | ramp down | | | |
| Reflector | | | | CS_SYNC followed by a CS tone (CS_SYNC_0_R) | ramp down |

*Table 4.4: CS step mode-0*

The initiator shall first transmit a CS_SYNC (CS_SYNC_0_I). The duration of the CS_SYNC (T_SY) corresponds to the definitions described in Section 2 and also depends on the CS_SYNC_PHY being used. The CS_SYNCs for mode-0 shall not carry a sounding sequence or random sequence as part of the CS_SYNC.

After the transmission from the initiator is completed, a defined ramp-down window of 5 µs (T_RD) is allowed for the initiator to remove the transmitted energy from the RF

channel. After T_RD, the output power from the initiator in the RF channel shall be at least 40 dB less than the output power used during the transmission of the CS_SYNC, as measured at the initiator's antenna connector.

T_IP1 represents the idle time between the end of the transmission from the initiator and the start of the transmission from the reflector. Devices may use this time for internal calibrations if needed. The reflector may use this time to ramp up its transmitted signal, if necessary, in advance of the transmission of the CS_SYNC followed by a CS tone (CS_SYNC_0_R), which is described in Section 2.6. The permitted values for T_IP1 are shown in Table 4.5.

| T_IP1 Index | T_IP1 | Mandatory/Optional/Conditional |
|---|---|---|
| 0 | 10 µs | O |
| 1 | 20 µs | O |
| 2 | 30 µs | O |
| 3 | 40 µs | C.1 |
| 4 | 50 µs | O |
| 5 | 60 µs | O |
| 6 | 80 µs | C.1 |
| 7 | 145 µs | M |
| C.1: Mandatory if any shorter T_IP1 value is supported, otherwise optional. | | |

*Table 4.5: Permitted values for T_IP1*

After T_IP1, the reflector shall transmit its CS_SYNC_0_R starting with the transmission of a CS_SYNC. The duration of the CS_SYNC (T_SY) depends on the CS_SYNC_PHY being used.

A guard time of T_GD length, as described in Section 2.6, then follows.

A frequency measurement period of T_FM follows next. The duration of T_FM for CS step mode-0 shall be 80 µs.

After the transmission from the reflector is complete, a ramp-down window of T_RD shall be present using the same requirements as defined for the initiator.

The reflector may transmit its CS_SYNC_0_R even if it does not receive a CS_SYNC from the initiator or if the CS_SYNC was received with bit errors. The time duration of a CS step mode-0 can be expressed as:

$$2 \times T_{SY} + 2 \times T_{RD} + T_{GD} + T_{FM} + T_{IP1}$$

The set of values of the time of departure of the mode-0 packets sent by an initiator in a subevent is denoted here as $ToD_I[m]$, where $m$ = 1, .., $M$ and $M$ is the number of mode-0

steps in a subevent, and therefore $ToD_I[1] = 0$. Within any subevent, the range of the set of values described by the set

$$\{ToD_I[m] - (m-1) \times (2 \times T_{SY} + 2 \times T_{RD} + T_{GD} + T_{FM} + T_{IP1} + T_{FCS}) | m = 1, \ldots, M\}$$

shall be less than or equal to 0.25 µs.

### 4.3.2 Channel Sounding step mode-1

CS step mode-1 is used to measure the round-trip time between the initiator and the reflector. Devices supporting the CS feature shall implement CS step mode-1. Table 4.6 shows the structure defined for CS step mode-1.

| Role/Duration | T_SY | T_RD | T_IP1 | T_SY | T_RD |
|---|---|---|---|---|---|
| Initiator | CS_SYNC (CS_SYNC_1) | ramp down | | | |
| Reflector | | | | CS_SYNC (CS_SYNC_1) | ramp down |

*Table 4.6: CS step mode-1*

The initiator starts by transmitting a CS_SYNC (CS_SYNC_1). The duration of the CS packet (T_SY) corresponds to the definitions described in Section 2 and also depends on the CS_SYNC_PHY being used. The CS_SYNCs for mode-1 may include a sounding sequence or random sequence as part of the CS_SYNC.

The transmission SNR output level may be adjusted during the CS_SYNC transmission. The permitted SNR output levels are described in [Vol 6] Part A, Section 3.1.3.

After the transmission from the initiator is completed, a defined ramp-down window of 5 µs (T_RD) is allowed for the initiator to remove the transmitted energy from the RF channel. After T_RD, the output power in the RF channel shall be at least 40 dB less than the output power used during the transmission of the CS_SYNC, measured at the initiator's antenna connector.

T_IP1 represents the idle time between the transmission from the initiator and the transmission from the reflector. Devices may use this time for internal calibrations if needed. The reflector may use this time to ramp up its transmitted signal, if necessary, in advance of the transmission of the CS_SYNC. The duration of T_IP1 is defined in Section 4.3.1.

After T_IP1, the reflector transmits its CS_SYNC_1. Within this CS_SYNC, the presence or absence of a sounding sequence or random sequence, and the length of the sequence if present, shall match that of the intended initiator's transmission.

The transmission SNR output level may be adjusted during the CS_SYNC transmission. The permitted SNR output levels are described in [Vol 6] Part A, Section 3.1.3.

After the transmission from the reflector is complete, a ramp-down window of T_RD shall be present using the same requirements as defined for the initiator.

The reflector may transmit its CS_SYNC_1 even if it does not receive a CS_SYNC from the initiator or if the CS_SYNC was received with bit errors. The time duration of a CS step mode-1 can be expressed as:

$$2 \times T_{SY} + 2 \times T_{RD} + T_{IP1}$$

Section 3.1 describes how round-trip time is computed using CS_SYNC exchanges. When reporting round-trip time to the Host, each device omits nominal known time offsets from the reported time. This time is equivalent to the time period between the reception and transmission of the center of the CS_SYNC fields, *T_SY_CENTER_DELTA*. For CS step mode-1, *T_SY_CENTER_DELTA* is equal to the following:

$$T_SY_CENTER_DELTA = T_{SY} + T_{RD} + T_{IP1}$$

### 4.3.3 Channel Sounding step mode-2

CS step mode-2 is used to measure the phase rotations of the RF signal between the initiator and the reflector. Devices supporting the CS feature shall implement CS step mode-2. Table 4.7 shows the structure defined for CS step mode-2.

| Role/Duration | (T_SW+T_PM)× (N_AP+1) | T_RD | T_IP2 | (T_SW+T_PM)× (N_AP+1) | T_RD |
|---|---|---|---|---|---|
| Initiator | CS tone | ramp down | | | |
| Reflector | | | | CS tone | ramp down |

*Table 4.7: CS step mode-2*

The initiator transmits a CS tone. The duration of the CS tone shall be (T_SW+T_PM)× (N_AP+1), where T_PM is the Phase Measurement period, T_SW is the antenna switch duration, and N_AP is the number of antenna paths. The permitted values for T_PM are shown in Table 4.8.

| T_PM Index | T_PM | Mandatory/Optional/Conditional |
|---|---|---|
| 0 | 10 µs | O |
| 1 | 20 µs | C.1 |

| T_PM Index | T_PM | Mandatory/Optional/Conditional |
|---|---|---|
| 2 | 40 µs | M |
| C.1: Mandatory if any shorter T_PM value is supported, otherwise optional. | | |

*Table 4.8: Permitted values for T_PM*

ASK modulation is employed during the CS tone extension slot (see Section 4.4). Here each N_AP set of T_PM length transmissions shall be followed by a single CS tone extension slot of the same T_PM length. This CS tone extension slot can carry a transmission. If a transmission is present, it shall be identical to that of the last T_PM slot and shall use the same antenna element used in that prior slot. If a transmission is not present in the CS tone extension slot, then that T_PM period shall still be present but shall not carry a transmission. The presence of a physical transmission in the extension transmission slot shall be seeded by the CS DRBG described in Section 4.8. The process of determining if a transmission is present is described in Section 4.4.

After the transmission from the initiator is completed, a defined ramp-down window of 5 µs (T_RD) is allowed for the initiator to remove the transmitted energy from the RF channel. After T_RD, the output power in the RF channel shall be at least 40 dB less than the output power used during the transmission of the CS tone, measured at the initiator's antenna connector. This ramp-down window shall also apply directly after the initiator's transmission is completed in the case where the CS tone extension slot does not carry an actual transmission. In this case, the period of T_RD shall still also be present after the CS tone extension slot and shall also not carry any added transmission from the initiator.

T_IP2 represents the idle time between the transmissions, including the CS tone extension slot from the initiator and the transmission from the reflector. Devices may use this time for internal calibrations, if needed. The reflector may use this time to ramp up its transmitted signal, if necessary, in advance of the transmission of the CS tone. Table 4.9 shows the permitted values for T_IP2. The T_IP2 value may be different from the T_IP1 value.

| T_IP2 Index | T_IP2 | Mandatory/Optional/Conditional |
|---|---|---|
| 0 | 10 µs | O |
| 1 | 20 µs | O |
| 2 | 30 µs | O |
| 3 | 40 µs | C.1 |
| 4 | 50 µs | O |
| 5 | 60 µs | O |
| 6 | 80 µs | C.1 |

| T_IP2 Index | T_IP2 | Mandatory/Optional/Conditional |
|---|---|---|
| 7 | 145 µs | M |
| C.1: Mandatory if any shorter T_IP2 value is supported, otherwise optional. | | |

*Table 4.9: Permitted values for T_IP2*

After T_IP2, the reflector transmits its CS tone. The duration of the CS tone shall be (T_SW+T_PM)×(N_AP+1), where T_PM is the Phase Measurement period, T_SW is the antenna switch duration, and N_AP is the number of antenna paths. The N_AP parameter is common to the entire CS procedure and is described in [Vol 6] Part A, Section 5.3. An extension transmission slot like the one that was present in the initiator to reflector direction shall immediately follow the last valid N_AP transmission. If a transmission is present, it shall be identical to the transmission of the last T_PM slot and shall use the same antenna element used in that prior slot. If a transmission is not present in the CS tone extension slot, then that T_PM period shall still be present but shall not carry a transmission. The presence of a physical transmission in the extension transmission slot shall be seeded by the CS DRBG described in Section 4.8. The process of determining if a transmission is present is described in Section 4.4. If the reflector has the IPT configuration enabled (see [Vol 6] Part A, Section 6.5), then there is no requirement to compensate the phase of the CS tone over the CS tone extension slot. In this section, both the T_SW and T_SW_IPT parameters are represented by T_SW in Table 4.7 and in the corresponding formulas in Section 4.3.3.

After the transmission from the reflector is complete, a ramp-down window of T_RD shall be present using the same requirements defined for the initiator described above in this section. In the case where a transmission was not present in the CS tone extension slot, a period of T_RD shall still also be appended after the CS tone extension slot and shall also not carry any added transmission from the reflector.

The time duration of a CS step mode-2 can be expressed as:

$$2 \times (T_{PM} + T_{SW}) \times (N_{AP} + 1) + 2 \times T_{RD} + T_{IP2}$$

For CS step mode-2, the time period between reception and transmission of the center of the CS tone at the antenna port is nominally expressed as:

$$T_PM_CENTER_DELTA = (T_{PM} + T_{SW}) \times (N_{AP} + 1) + T_{RD} + T_{IP2}$$

### 4.3.4 Channel Sounding step mode-3

CS step mode-3 is used to measure the phase rotations of the RF signal and the round-trip time between the initiator and the reflector. Devices supporting the CS feature may implement CS step mode-3. Table 4.10 shows the structure defined for CS step mode-3.

| Role/ Duration | T_SY+T_GD+ (T_SW+T_PM)× (N_AP+1) | T_RD | T_IP2 | (T_SW+T_PM)× (N_AP+1) +T_GD+T_SY | T_RD |
|---|---|---|---|---|---|
| Initiator | CS_SYNC followed by a CS tone (CS_SYNC_3_I) | ramp down | | | |
| Reflector | | | | CS tone followed by a CS_SYNC (CS_SYNC_3_R) | ramp down |

*Table 4.10: CS step mode-3*

The initiator transmits a CS_SYNC followed by a CS tone (CS_SYNC_3_I) as described in Section 2.6. The duration of the CS_SYNC (T_SY) corresponds to the definitions described in Section 2 and also depends on the CS_SYNC_PHY being used. The CS_SYNCs for mode-3 may include a sounding sequence or random sequence as part of the CS_SYNC.

The transmission SNR output level may be adjusted during the CS_SYNC transmission. The permitted SNR output levels are described in [Vol 6] Part A, Section 3.1.3.

A guard time of T_GD length, as described in Section 2.6, then follows.

The duration of the CS tone shall be (T_SW+T_PM)×(N_AP + 1), where T_PM is the phase measurement period and N_AP is the number of antenna paths. The valid values for the T_PM parameter are described in Section 4.3.3.

ASK modulation is employed during the CS tone extension slot (see Section 4.4). Here each N_AP set of T_PM length transmissions shall be followed by a single CS tone extension slot of the same T_PM length. This CS tone extension slot can carry a transmission. If a transmission is present, it shall be identical to that of the last T_PM slot and shall use the same antenna element used in that prior slot. If a transmission is not present in the CS tone extension slot, then that T_PM period shall still be present but shall not carry a transmission. The presence of a physical transmission in the extension transmission slot shall be seeded by the CS DRBG described in Section 4.8. The process of determining if a transmission is present is described in Section 4.4.

After the transmission from the initiator has completed, a defined ramp-down window of 5 µs (T_RD) is allowed for the initiator to remove the transmitted energy from the RF channel. After T_RD the output power in the RF channel shall be at least 40 dB less than the output power used during the transmission of the CS_SYNC measured at the initiator's antenna connector. This ramp-down window shall also apply directly after the initiator's transmission is completed in the case where the CS tone extension slot does not carry an actual transmission. In this case, the period of T_RD shall still be present after the CS tone extension slot and shall also not carry any added transmission from the initiator.

T_IP2 represents the idle time between the transmission from the initiator and the transmission from the reflector. Devices may use this time for internal calibrations if needed. The reflector may use this time to ramp up its transmitted signal, if necessary, in advance of the transmission. The duration of T_IP2 is defined in Section 4.3.3.

After T_IP2, the reflector transmits a CS tone followed by a CS_SYNC (CS_SYNC_3_R) as described in Section 2.6. This sequence begins with the transmission of the CS tone. The duration of the CS tone shall be either (T_SW+T_PM) ×N_AP or (T_SW+T_PM)×(N_AP + 1) depending on whether a transmission is selected for the CS tone extension slot. If a transmission is present, it shall be identical to that of the last T_PM slot and shall use the same antenna element used in that prior slot. If a transmission is not present in the CS tone extension slot, then that T_SW+T_PM period shall not be present but shall be compensated for as described below. The presence of a physical transmission in the extension transmission slot shall be seeded by the CS DRBG described in Section 4.8. The process of determining if a transmission is present is described in Section 4.4. There is no requirement for the phase of the reflector's tone during the CS tone extension slot. In this section, both the T_SW and T_SW_IPT values are represented by T_SW in Table 4.10 and in the corresponding formulas in Section 4.3.4.

A guard time of T_GD length as described in Section 2.6 then follows.

The reflector may then transmit its CS_SYNC. Within this CS_SYNC, the presence or absence of a sounding sequence or random sequence, and the length of the sequence if present, shall match that of the initiator’s transmission.

The transmission SNR output level may be adjusted during the CS_SYNC transmission. The permitted SNR output levels are described in [Vol 6] Part A, Section 3.1.3.

A T_PM period without a physical transmission shall be inserted directly after the reflector’s CS_SYNC transmission if the CS tone extension slot was not present during the prior CS tone transmission. This T_PM period shall be accompanied by a T_SW period as if it had occurred before the CS_SYNC transmission.

After the transmission from the reflector is complete, a ramp-down window of T_RD shall be present using the same requirements defined for the initiator described above in this section. In the case where a transmission was not present in the CS tone extension slot, a period of T_RD shall also be appended after the CS tone extension slot and shall also not carry any added transmission from the reflector.

The time duration of a CS step mode-3 can be expressed as:

$$2 \times (T_{SY} + T_{GD}) + 2 \times (T_{PM} + T_{SW}) \times (N_{AP} + 1) + 2 \times T_{RD} + T_{IP2}$$

Section 3.1 describes how round-trip time is computed using CS_SYNC exchanges. When reporting round-trip time to the Host, each device omits nominal known

time offsets from the reported time. This time is equivalent to the time period between the reception and transmission of the center of the CS_SYNC fields, *T_SY_CENTER_DELTA*. For CS step mode-3, *T_SY_CENTER_DELTA* is equal to the following:

- $T_SY_CENTER_DELTA = T_{SY} + T_{RD} + 2 \times T_{GD} + 2 \times (T_{SW} + T_{PM}) \times N_{AP} + 2 \times (T_{SW} + T_{PM}) + T_{IP2}$, if a physical transmission is present in the reflector to initiator transmission extension slot.
- Otherwise, $T_SY_CENTER_DELTA = T_{SY} + T_{RD} + 2 \times T_{GD} + 2 \times (T_{SW} + T_{PM}) \times N_{AP} + (T_{SW} + T_{PM}) + T_{IP2}$.

For CS step mode-3, the time period between reception and transmission of the center of the CS tone at the antenna port is nominally expressed as:

- $T_PM_CENTER_DELTA = (T_{PM} + T_{SW}) \times (N_{AP} + 1) + T_{RD} + T_{IP2}$

## 4.4 Channel Sounding subevent and mode sequencing

CS subevents have a maximum duration of T_SUBEVENT_LEN, as described in [Vol 6] Part B, Section 4.5.18.1. Each subevent contains a varying number of CS steps. The number of steps included in a subevent depends on the varying step combinations used in that subevent.

### 4.4.1 Tone extension slots

For CS tones, each N_AP set of T_PM length transmissions shall be followed by a CS tone extension slot as described in Section 4.3.3 and Section 4.3.4. The presence of an actual transmission in this CS tone extension slot shall be selected at random, seeded by the CS DRBG described in Section 4.8.

CS DRBG shall be invoked to generate a single random bit (see Section 4.8). If this bit is set to 1, then a physical transmission shall be present in the CS tone extension slot. If this bit is set to 0, then no transmission shall be present in the CS tone extension slot. Even without a transmission, the CS tone extension slot shall still be present. Refer to Section 4.3.3 and Section 4.3.4 for information about the CS tone extension slot and its positioning within each step.

Figure 4.4 shows a sequence of four CS steps, each of type mode-2, with the CS tone extension slot transmission determined by the output of the CS DRBG.

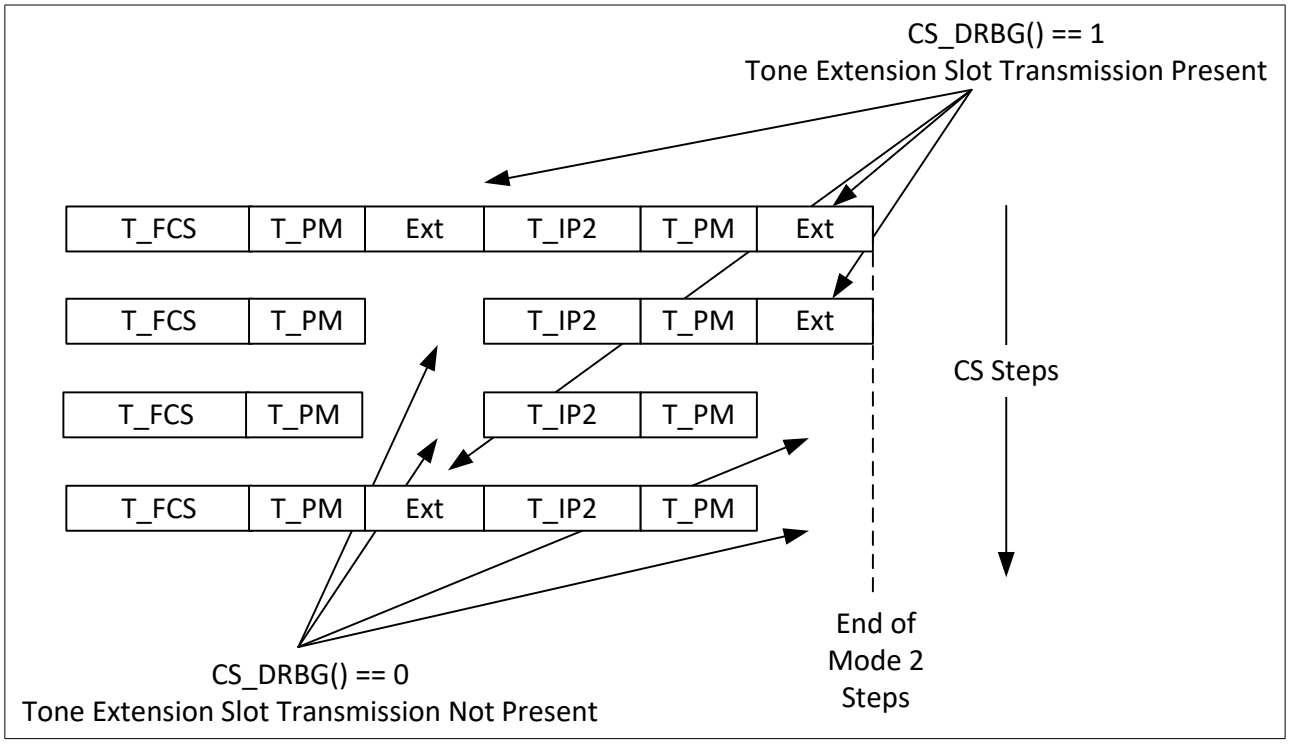



*Figure 4.4: Mode-2 CS steps with CS tone extension slot transmission determined by the CS DRBG*

Likewise, Figure 4.5 shows a sequence of four CS steps, each of type mode-3, with the CS tone extension slot transmission determined by the output of the CS DRBG. The position of the CS tone extension slot for mode-3 steps is described in Section 4.3.4.

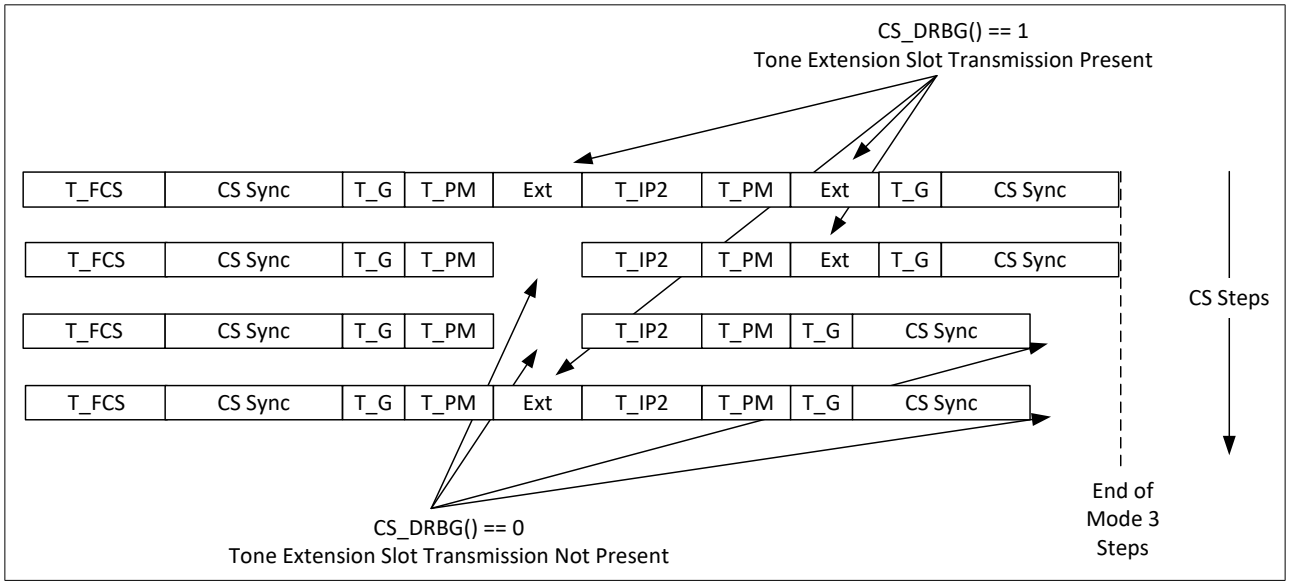



*Figure 4.5: Mode-3 CS steps with CS tone extension slot transmission determined by the CS DRBG*

Refer to Section 4.8 for CS DRBG invocation ordering rules.

### 4.4.2 CS subevent structure

Within a CS subevent, if the cumulative duration of a CS step exceeds T_SUBEVENT_LEN as described in [Vol 6] Part B, Section 4.5.18.1, then the CS subevent shall be closed, and that CS step shall occur in the next scheduled CS subevent directly after the initial mode-0 steps and after any Main_Mode Repetitions (see Section 4.2).

A CS subevent starts with one or more mode-0 steps defined by CSMode0Steps. A series of subsequent non-mode-0 steps shall follow in the same subevent. Any CS procedure shall include the use of no more than two non-mode-0 modes, the Main_Mode and the Sub_Mode. An example of this sequencing is shown in Figure 4.6.

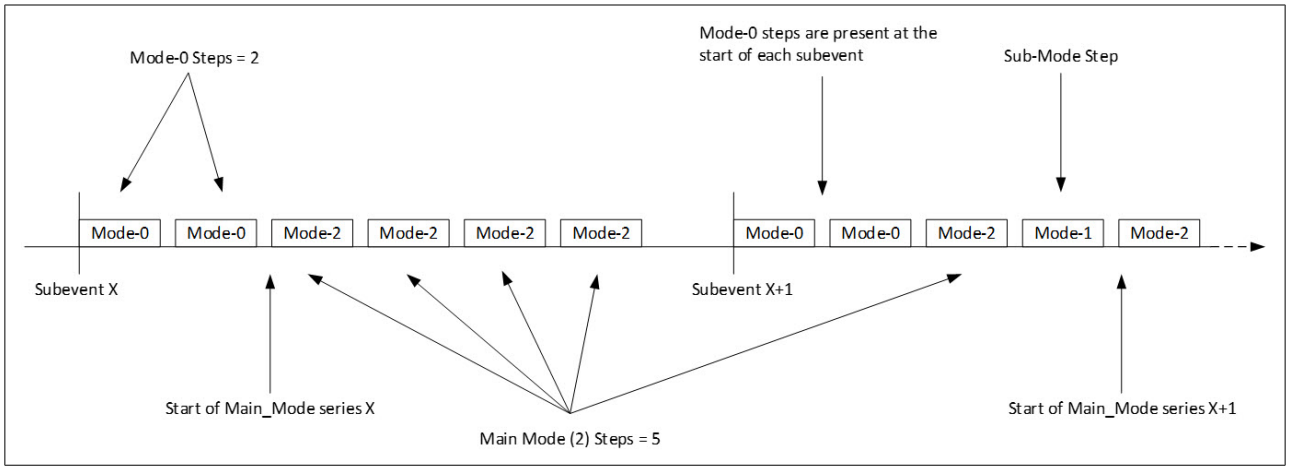



*Figure 4.6: CS mode sequencing example with two mode-0 steps, five Main_Mode steps (of type mode-2), and one Sub_Mode step (of type mode-1)*

For non-mode-0 steps, Main_Mode and Sub_Mode combinations shall be selected from the valid combinations defined in Table 4.11.

| Main_Mode | Sub_Mode |
|---|---|
| Mode-1 | None |
| Mode-2 | None |
| Mode-3 | None |
| Mode-2 | Mode-1 |
| Mode-2 | Mode-3 |
| Mode-3 | Mode-2 |

*Table 4.11: Valid combinations of Main_Mode and Sub_Mode selections for a CS procedure*

### 4.4.3 Sub_Mode insertion

The following rules shall apply for Sub_Mode insertion only when a Sub_Mode has been selected for a CS procedure. These rules shall repeat for the entire CS procedure.

- Sub_Mode insertion is the process of sequencing the occurrence of the Main_Mode steps and the Sub_Mode steps. This sequencing is performed irrespective of the occurrence of CS subevent boundaries and shall use the following parameters:
  - Main_Mode_Min_Steps – the minimum number of Main_Mode steps that shall occur before the occurrence of a single Sub_Mode step.
  - Main_Mode_Max_Steps – the maximum number of Main_Mode steps that shall occur before the occurrence of a single a Sub_Mode step.
- The exact number of Main_Mode steps that occur before a Sub_Mode step insertion shall be randomized for each sequence. The random number generation function hr1 (see Section 4.8.1) is invoked to determine the randomized number of Main_Mode steps that occur in the sequence.
  - The number of Main_Mode steps to execute shall be calculated as follows.
    - hr1( Main_Mode_Max_Steps – Main_Mode_Min_Steps + 1 ) + Main_Mode_Min_Steps
  - The number of Main_Mode steps to execute before a Sub_Mode insertion shall be generated at the CS step where the first Main_Mode step within that sequence occurs. Refer to Section 4.8 for CS DRBG invocation ordering rules.

### 4.4.4 Main_mode repetition

After a CS subevent, a variable number of Main_Mode steps and their respective transmission channels may be repeated in the next CS subevent directly after the mode-0 transmissions. The Main_Mode_Repetition value from the CS configuration selected (see [Vol 6] Part B, Section 2.4.2.45) for a CS procedure shall denote the number of Main_Mode steps that are repeated from the end of one CS subevent to the beginning of the next.

When CS steps are repeated based on the Main_Mode_Repetition value, only the channel index and Main_Mode type shall be repeated in the next CS subevent. All other aspects that involve seeding from the CS DRBG shall be newly generated in the steps that are repeated. If the value of Main_Mode_Repetition is greater than 1, then the repeated steps in the next CS subevent shall be in the same order as in the previous CS subevent.

Only CS steps from the current subevent that were not repeated from the last subevent shall be repeated in the next subevent. If the number of CS steps that are candidates to be repeated from the current subevent into the next subevent is less than the

Main_Mode_Repetition value, then only those CS steps shall be repeated in the next subevent.

Sub_Mode insertion, if present, shall be calculated as if the Main_Mode_Repetition value were specified as zero. This is equivalent to not counting the occurrences resulting from Main_Mode_Repetition when determining where the Sub_Mode insertion occurs. The step placement due to Main_Mode_Repetition shall occur first within a CS subevent in the case of pending step placement due to Sub_Mode insertion.

### 4.4.5 Channel Sounding procedure and procedure repeat desynchronization

A CS subevent will always start with the exchange of one or more mode-0 steps, as described in Section 4.4.2. Within each of these mode-0 steps is a CS_SYNC exchange as described in Section 4.3.1. A mode-0 exchange is considered successful in the case where both Link Layers properly receive a CS_SYNC PDU with an access address quality indication result of 0 or 1, as described in Section 2.2.2. Alternatively, a mode-0 step exchange is considered lost when either link layer fails to receive a CS_SYNC PDU or the resulting access address quality indication result is greater than 1, within that mode-0 step. In the case that all mode-0 steps within a CS subevent are lost, then that CS subevent is also considered lost.

[Vol 6] Part B, Section 4.2.4 describes the window widening recommendations for synchronization of the first CS subevent within a CS event. If the first consecutive CS subevents within a CS event are lost, then these window widening recommendations apply to the next sequential CS subevent within that CS event as if that CS subevent was the first in occurrence within that CS event.

If the first CS subevent(s) within any CS event is lost, then it is implementation dependent if an attempt is made to synchronize with the next CS subevent within that CS event. In this case, if synchronization with a CS event is abandoned, then that CS event is similarly considered lost. In the case that an entire CS procedure is abandoned due to the first consecutive CS events within that procedure being lost, then that CS procedure is also considered lost.

In the case that CS procedure instances are actively running and N_PROCEDURE_LOST=2 consecutive CS procedure repeat instances are lost, then a Link Layer should initiate the Channel Sounding Procedure Repeat Termination procedure, as described in [Vol 6] Part B, Section 5.1.27.

In the case of ACL connection termination or loss (see [Vol 6] Part B, Section 4.5.12), all CS procedures that are actively running as well as all associated CS procedure repeat instances shall be considered lost. In this case, initiating the Channel Sounding Procedure Repeat Termination procedure is not required.

## 4.5 Timing of steps

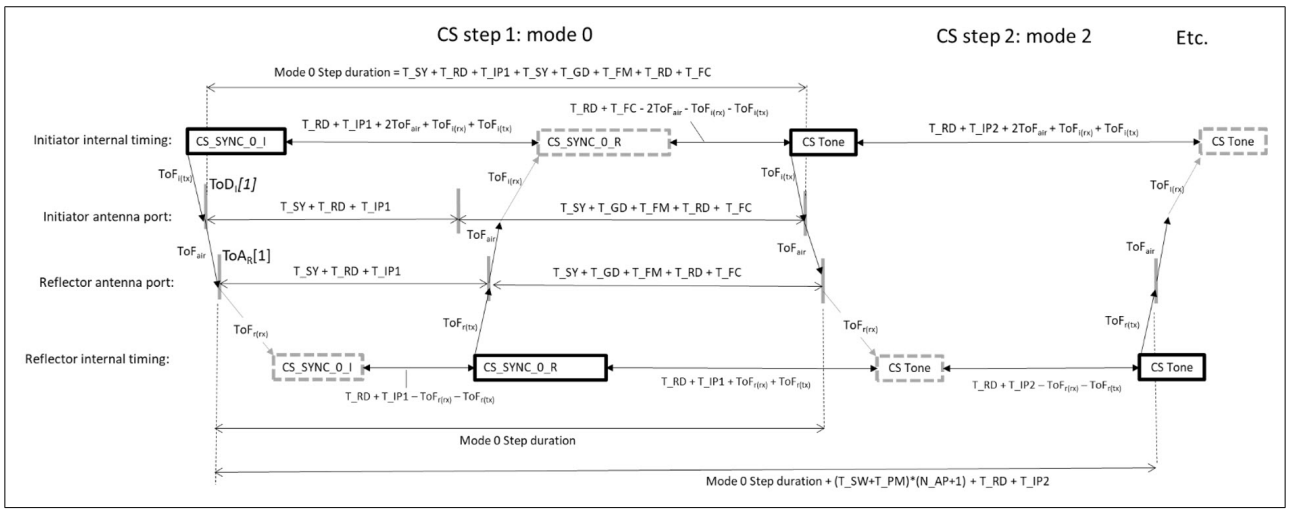



*Figure 4.7: Example timing of CS steps for single antenna mode based on the reflector's timing reference*

Figure 4.7 shows an example of the timing of CS steps when a single antenna is in use. All timings in Figure 4.7 are based on the reflector's timing reference. $ToD_I[k]$ represents the time of departure of the packet at the initiator's antenna port sent in the $k^{th}$ step within a subevent, if such a packet exists within that step. $ToA_R[k]$ represents the time of arrival at the reflector's antenna port of the packet sent in the $k^{th}$ step within a subevent, if such a packet exists within that step. $ToF_{air}$ represents the time of flight between antenna ports. $ToF_I$ and $ToF_R$ represent the initiator and reflector circuit delay, respectively. In Figure 4.7, solid boxes denote transmission times and dashed grey boxes denote reception windows.

The initiator shall align its transmission timings relative to the start of its first mode-0 transmission at its antenna port, $ToD_I[1]$ (see Figure 4.7), based on the reflector's clock reference as follows:

$$\hat{t}_1^{STEP}[k] = \frac{t_1^{STEP}[k]}{1 + FFO[1] \cdot 10^{-6}} + ToD_I[1], k = M + 1, \ldots, K;$$

where $t_1^{STEP}[k]$ is the nominal start time of each transmission according to its local clock reference and FFO[1] is the FFO estimated from the first mode-0 CS tone (see [Vol 6] Part A, Section 3.5.1). The initiator should adjust its CS tone reception window to compensate for the round-trip time of flight between devices and to compensate for any relevant known synchronization errors or internal circuit delays so that its phase center of measurement falls in the center of the received CS tone (see [Vol 6] Part A, Section 6.4).

The reflector shall calculate all transmission timings relative to the time of arrival of a mode-0 packet at its antenna port, using its own timing reference $ToA_r[m]$ (see

Figure 4.7), where m is assumed to be the first successfully received mode-0 packet. The reflector's timing reference, defined by the frequency of the first transmitted mode-0 packet CS tone, shall be constant throughout a subevent, and the reflector shall not perform any timing adjustment of its transmissions to compensate for the clock frequency offset between devices. The reflector should adjust its CS tone reception window to compensate for known synchronization errors or internal circuit delay so that its phase center of measurement falls in the center of the received CS tone (see [Vol 6] Part A, Section 6.4).

Both initiator and reflector shall maintain an accuracy of ±1 µs in the timing of their transmissions as described in this section.

## 4.6 Phase measurements during T_PM

Devices perform phase measurements during T_PM. Devices shall perform N_AP + 1 phase measurements for each of the phase measurement periods during a mode-2 or mode-3 step, one for each of the antenna paths as well as for the CS tone extension slot. The rules for T_PM measurements taken over tone extension slots are described below.

Each T_PM period shall start and end with at least a 1 µs exclusion period, as shown in Figure 4.8, to allow for peer device timing error (see Section 4.5). A longer exclusion period should be introduced as necessary to compensate for local device timing accuracy. These exclusion periods shall not be considered as valid regions for PCT extraction. PCT sampling is described in [Vol 6] Part A, Section 6.4 and Section 4.5.

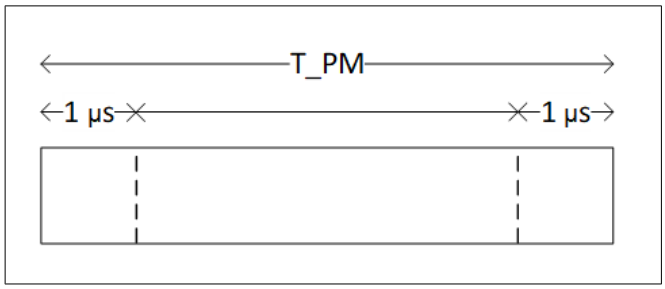



*Figure 4.8: PCT sample window*

Each measurement within T_PM results in a single PCT value expressed as complex number I+jQ. When a CS subevent contains multiple CS steps, devices should maintain receiver characteristics (e.g., receiver's gain and latency across T_PM periods should remain constant across the CS steps). If this is not possible, receivers should be able to make adjustments to the resulting measurements to emulate this constant behavior.

The I and Q terms shall each use 12 bits to encode their value.

If the reflector has the IPT configuration enabled for the current CS procedure (see [Vol 6] Part A, Section 6.5), then it shall adjust the phase of its CS tone such that in the

reported PCT, the imaginary (Q) part is equal to 0 and real (I) part shall be non-negative and should reflect the magnitude of measurement.

In addition to the IQ values, devices shall provide a reference power level (RPL) in dBm. The RPL shall be encoded as an 8-bit signed number. The RPL expressed in dBm shall correspond to the power of a hypothetical PCT value whose amplitude is a constant value of 2048. A single RPL is used for all the PCT values in a CS subevent.

The conversion between IQ values as unitless values to power in dBm can be expressed as:

$$IQ\left[dBm\right] = 20\log_{10}\left(abs\left(\frac{IQ[linear]}{2048}\right)\right) + RPL\left[dBm\right] \quad \text{(EQ 6)}$$

Devices may assign a tone quality indication to each of the PCT measurements. The tone quality indication shall be a 2-bit number with definitions as shown in Table 4.12. Devices that are not able to compute a quality indication shall report a value of 3.

The following definitions pertain to devices that are capable of computing tone quality indications.

| Value | Tone Quality |
|---|---|
| 0 | High quality |
| 1 | Medium quality |
| 2 | Low quality |
| 3 | Quality indication not available / unknown quality |

*Table 4.12: CS tone quality values*

The following shall be tested with a T_PM value of 40 µs.

Devices shall report high quality when the signal at the antenna port is a CS tone of -67 dBm, with a Gaussian noise floor of -151 dBm/Hz or less. This reported value shall be the outcome of the measurement for 90% of the CS tones and under all valid frequency offsets of the input signal, as described in [Vol 6] Part A, Section 3.5.2.

Devices shall report low quality when the signal at the antenna port is a CS tone of -67 dBm with a Gaussian noise floor of -133 dBm/Hz or greater. This reported value shall be the outcome of the measurement for 90% of the CS tones and under all valid frequency offsets of the input signal, as described in [Vol 6] Part A, Section 3.5.2.

Devices may report medium quality when the received signal is between the good quality and low quality thresholds.

The use of Gaussian noise signals to represent channel impairments is intended to represent external sources of wideband interference.

In tone extension slots, no T_PM measurement shall occur in a mode-3 step in the reflector to initiator direction when a CS tone extension slot transmission is not present. Instead, an empty T_PM measurement result shall take the place of the T_PM measurement. In any other case of a CS tone extension, the T_PM measurement over that tone extension slot shall be reported. An empty T_PM measurement shall be indicated by I and Q values each set to zero and with a tone quality indication set to “quality indication not available”.

## 4.7 Phase measurements with antenna switching

During CS tone exchanges, the initiator switches to the intended antenna element and transmits its CS tone first and the reflector performs phase measurements. Then, after a T_RD + T_IP2 period, the reflector switches to the intended antenna element and transmits its CS tone, and the initiator performs phase measurements. The configuration options listed in [Vol 6] Part A, Section 5.3 affect how the switching happens, which antenna is selected during the execution within a CS step, and whether the switch applies to the outgoing signal (i.e., switch on transmitted RF signal), or the incoming signal (i.e., switch on received RF signal), or both.

Antenna switching occurs just before the start of every T_PM period for each configuration with multiple antenna paths. T_SW and T_SW_IPT are the switch period times and are reserved for antenna switching, irrespective of whether the antenna to be switched is a local antenna or a remote antenna. During the period when an antenna is switched, the signal might not be stable and shall not be used for measurements. The T_SW and T_SW_IPT durations are local device specific and each peer shall indicate its own preferred settings, as described in [Vol 6] Part B, Section 5.1.24. When antenna switching is performed, the duration of T_SW and T_SW_IPT shall be 1, 2, 4, or 10 µs. When no antenna switching is performed (e.g., in the 1:1 configuration), the values of T_SW and T_SW_IPT shall be set to 0. Devices switching transmitting antennae must control switching transients according to the requirements specified in Part A.

The values of T_SW and T_SW_IPT used in a CS procedure depend on the antenna switch configuration selected for that procedure, as described in the following subsections.

### 4.7.1 Antenna switching in the 1:1 configuration

There shall not be antenna switching activity in the 1:1 configuration.

### 4.7.2 Antenna switching in the N_AP:1 configuration

In the N_AP:1 configuration, only the initiator performs antenna switching. Figure 4.9 shows the procedure. In the figure, * represents the extension slot that can occur as shown, or can occur after T_SY, specifically for a mode-3 step.

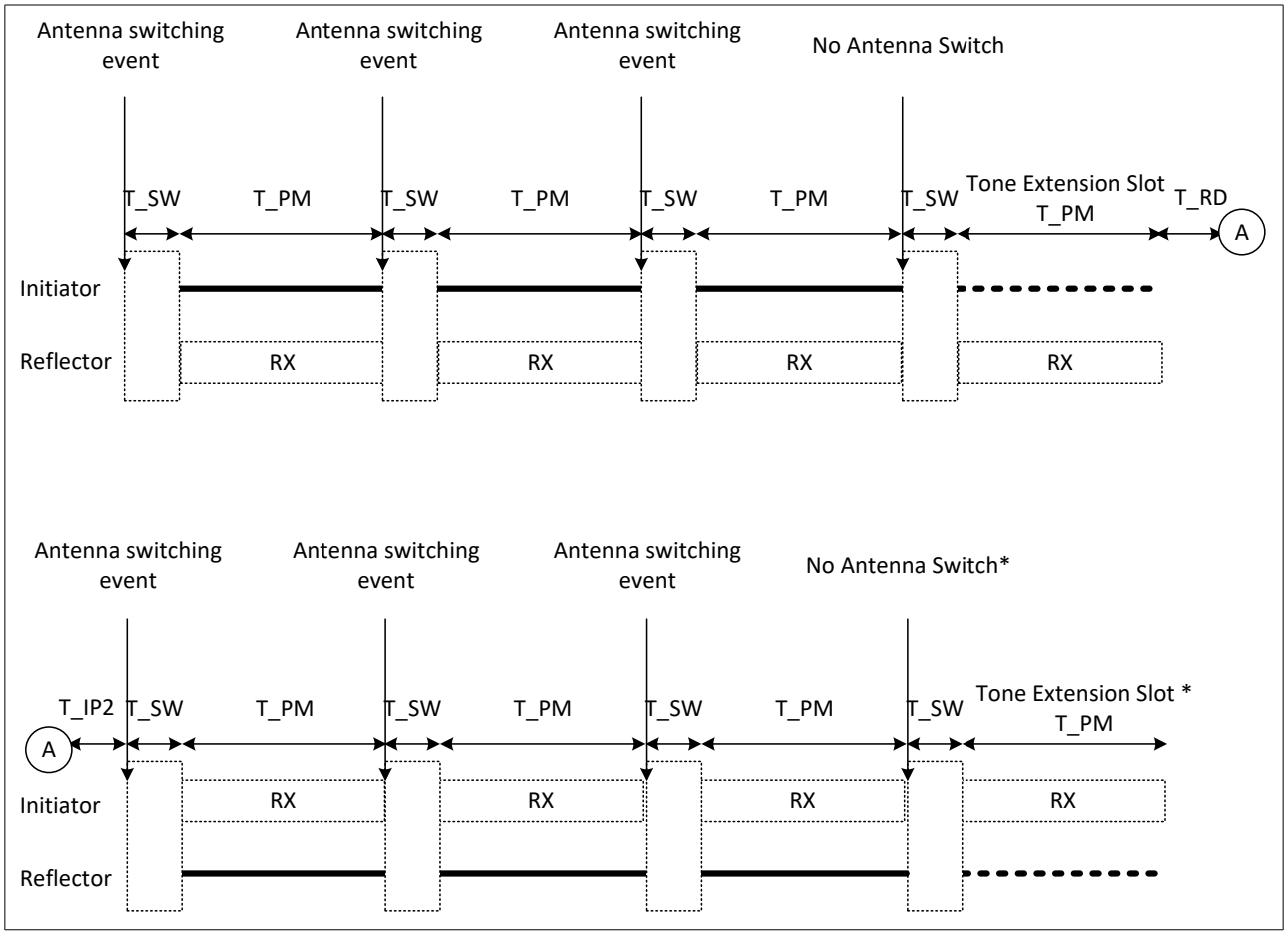



*Figure 4.9: N_AP:1 antenna switch procedure in a 3:1 configuration with the tone extension slot*

In this configuration, with N_AP antennae in the initiator, the initiator shall be the only device performing antenna switching. When IPT is disabled, the selected antenna switch duration shall be the T_SW value of the initiator. When IPT is enabled, the selected antenna switch duration shall be the greater of the initiator's T_SW value and the reflector's T_SW_IPT value. Both the T_SW and T_SW_IPT values are represented by T_SW in Figure 4.9.

During its transmission phase, the initiator shall perform N_AP antenna switches. Each switch shall be performed before the start of the T_PM period.

During its reception phase, the initiator shall perform N_AP antenna switches. Each switch shall be performed before the start of the T_PM period.

The reflector shall not perform antenna switching in this configuration. The reflector shall not perform any measurements during the switch duration of the initiator.

The antenna switching sequence that the initiator selects for each T_PM period during its transmission phase shall be identical to the antenna switching sequence used during its reception phase.

Figure 4.9 shows the antenna switch procedure for a 3:1 antenna switch configuration, including the tone extension slot described in Section 4.4. For a mode-3 step in the reflector-to-initiator direction, the last T_SW period and associated tone extension slot (indicated by * in Figure 4.9) may occur as shown in the figure or may occur after T_SY as described in Section 4.3.4.

### 4.7.3 Antenna switching in the 1:N_AP configuration

In the 1:N_AP configuration, only the reflector performs antenna switching. Figure 4.10 shows the procedure. In this figure, * represents the extension slot that can occur as shown or can occur after T_SY.

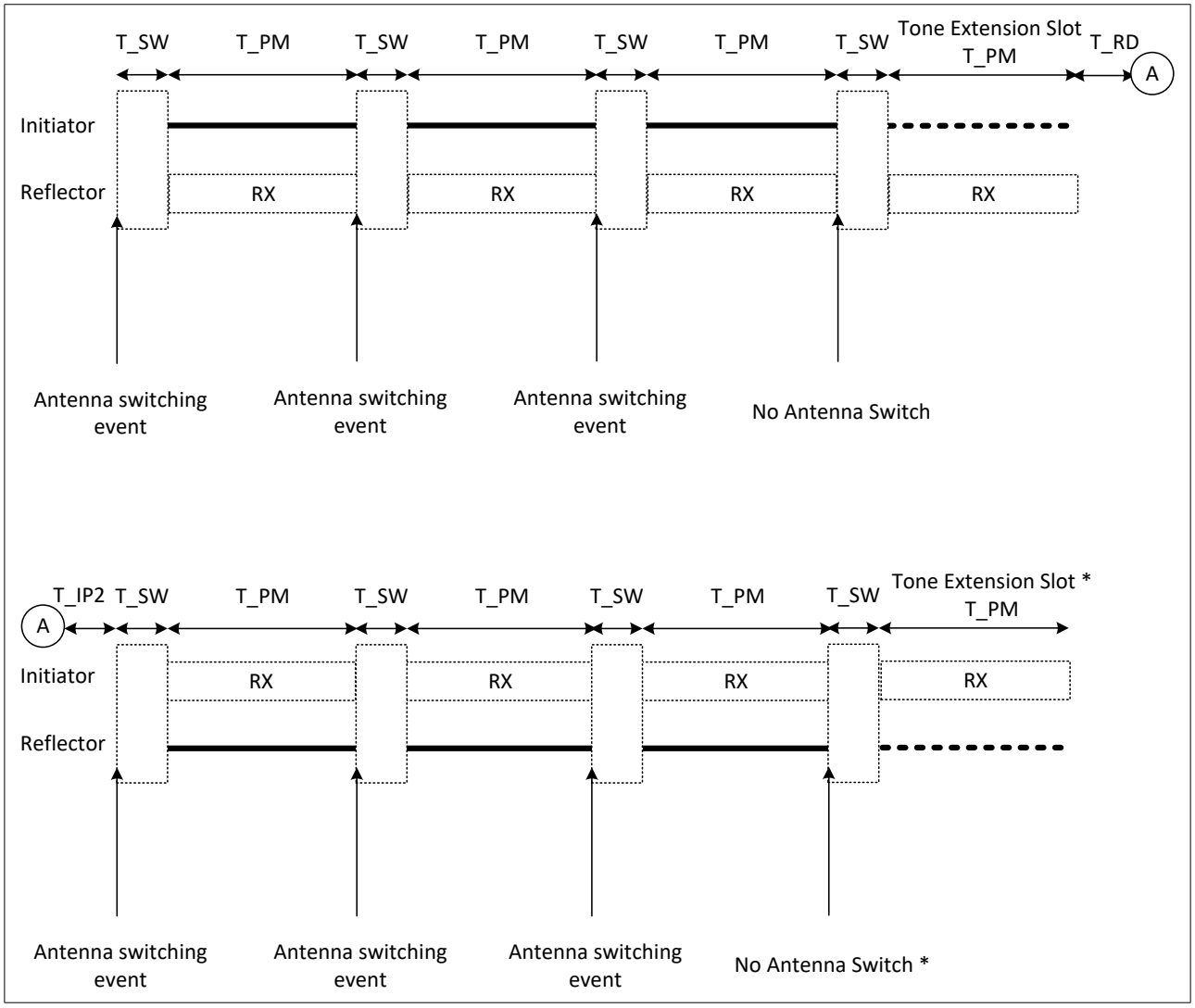



*Figure 4.10: 1:N_AP antenna switch procedure in a 1:3 configuration with the tone extension slot*

In this configuration, with N_AP antennae in the reflector, the reflector shall be the only device performing antenna switching. When IPT is disabled, the selected antenna

switch duration shall be the T_SW value of the reflector. When IPT is enabled, the selected antenna switch duration shall be the reflector's T_SW_IPT value. Both the T_SW and T_SW_IPT values are represented by T_SW in Figure 4.10.

During its reception phase, the reflector shall perform N_AP antenna switches. Each switch shall be performed before the start of the T_PM period.

During its transmission phase, the reflector shall perform N_AP antenna switches. Each switch shall be performed before the start of the T_PM period.

The initiator shall not perform antenna switching in this configuration. The initiator shall not perform any measurements during the switch duration of the reflector.

The antenna switching sequence that the reflector selects for each T_PM period during its transmission phase shall be identical to the antenna switching sequence used during its reception phase.

Figure 4.10 shows the antenna switch procedure for a 1:3 antenna switch configuration, including the tone extension slot as described in Section 4.4. For a mode-3 step in the reflector-to-initiator direction, the last T_SW period and associated tone extension slot (indicated by * in Figure 4.10) may occur as shown in the figure or may occur after T_SY as described in Section 4.3.4.

### 4.7.4 Antenna switching in the 2:2 (N_AP = 4) configuration

In the 2:2 (N_AP = 4) configuration, both devices perform antenna switching. At each antenna switch point, one device or both devices simultaneously may perform antenna switching. Figure 4.11 shows the procedure. In this figure, * represents the extension slot that can occur as shown or can occur after T_SY.

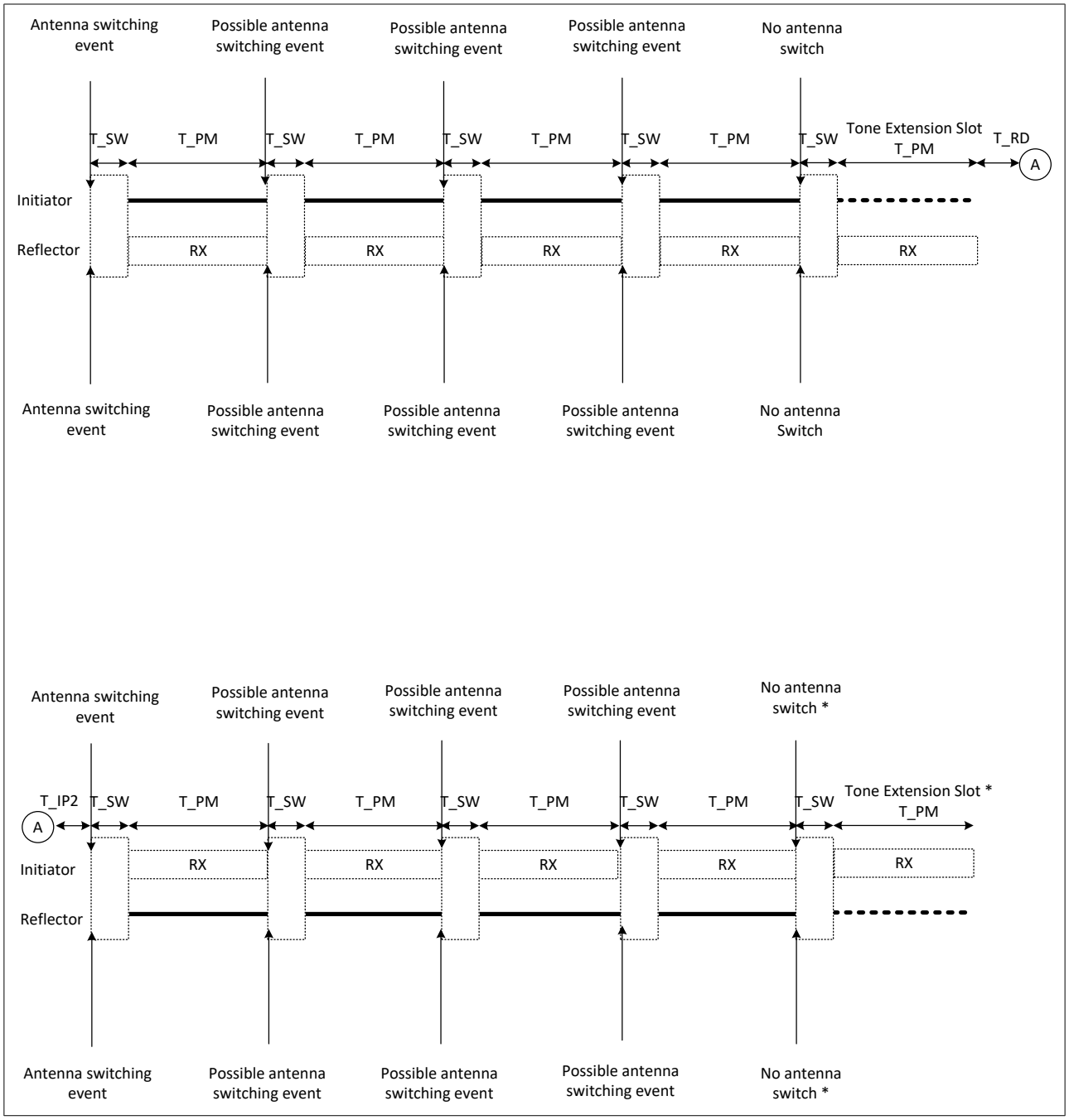



*Figure 4.11: 2:2 antenna switch procedure with the tone extension slot*

For the 2:2 antenna switch procedure, each antenna path consists of a combination of antennae selected by both the initiator and reflector, as described in [Vol 6] Part A, Section 5.3.

During its transmission phase, the initiator shall perform an antenna switch for before the first T_PM interval and then may perform an antenna switch at each subsequent

T_PM interval, depending on the antenna path selected for that interval. Each switch shall be performed immediately before the start of the T_PM period.

Similarly, during its reception phase, the reflector shall perform an antenna switch for the first T_PM interval and then may perform an antenna switch at each subsequent T_PM interval, depending on the antenna path selected for that interval. Each switch shall be performed immediately before the start of the T_PM period.

During its reception phase, the initiator shall perform an antenna switch for the first T_PM interval and then may perform an antenna switch at each subsequent T_PM interval, depending on the antenna path selected for that interval. Each switch shall be performed immediately before the start of the T_PM period.

Similarly, during its transmission phase, the reflector shall perform an antenna switch for the first T_PM interval and then may perform an antenna switch at each subsequent T_PM interval, depending on the antenna path selected for that interval. Each switch shall be performed immediately before the start of the T_PM period.

When IPT is disabled, the selected antenna switch duration for all required antenna element switching shall be the greater of the T_SW values of the initiator and the reflector. When IPT is enabled, it shall be the greater of the initiator's T_SW value and the reflector's T_SW_IPT value. Both the T_SW and T_SW_IPT values are represented by T_SW in Figure 4.11. The device acting as the receiver shall not perform any measurements during the switch duration of the device acting as the transmitter.

The antenna switching sequence that the initiator and reflector select for each T_PM period during the first part of the CS step (initiator to reflector) shall be identical to the antenna switching sequence used during the second part of the CS step (reflector to initiator).

Figure 4.11 shows the antenna switch procedure for a 2:2 antenna switch configuration, including the tone extension slot described in Section 4.4. For a mode-3 step in the reflector-to initiator-direction, the last T_SW period and associated tone extension slot (indicated by * in Figure 4.11) may occur as shown in the figure or may occur after T_SY, as described in Section 4.3.4.

### 4.7.5 Antenna path permutations

During antenna switching, the antenna path position ordering in a CS tone exchange shall be randomized. Each antenna path is described in [Vol 6] Part A, Section 5.3, where the initiator is device A and the reflector is device B.

Each device may switch between up to four antenna paths within a CS step. The randomized antenna path permutation in a CS step shall be selected using the random number generation function hr1 (see Section 4.8.1). The number of antenna paths used

within the permutation is the maximum number of N_AP antenna paths, as described in [Vol 6] Part A, Section 5.3.

When N_AP is greater than 1, the antenna path permutation index shall be calculated as follows:

hr1( N_AP! )

See Section 4.8 for CS DRBG invocation ordering rules.

An antenna path permutation index is used to look up the specific antenna path position order used at each step that contains a CS tone exchange within a CS step. These lookup tables are defined based on the number of antenna paths being used.

If N_AP = 2, then Table 4.13 shall be used to select the antenna path permutation.

| Antenna Path Permutation Index | Antenna Path Positions After Permutation |
|---|---|
| 0 | AP1 AP2 |
| 1 | AP2 AP1 |

*Table 4.13: Antenna path positions after path permutation with N_AP=2*

If N_AP = 3, then Table 4.14 shall be used to select the antenna path permutation.

| Antenna Path Permutation Index | Antenna Path Positions After Permutation |
|---|---|
| 0 | AP1 AP2 AP3 |
| 1 | AP2 AP1 AP3 |
| 2 | AP1 AP3 AP2 |
| 3 | AP3 AP1 AP2 |
| 4 | AP3 AP2 AP1 |
| 5 | AP2 AP3 AP1 |

*Table 4.14: Antenna path positions after path permutation with N_AP=3*

If N_AP = 4, then Table 4.15 shall be used to select the antenna path permutation.

| Antenna Path Permutation Index | Antenna Path Positions After Permutation |
|---|---|
| 0 | AP1 AP2 AP3 AP4 |
| 1 | AP2 AP1 AP3 AP4 |
| 2 | AP1 AP3 AP2 AP4 |
| 3 | AP3 AP1 AP2 AP4 |
| 4 | AP3 AP2 AP1 AP4 |

| Antenna Path Permutation Index | Antenna Path Positions After Permutation |
|---|---|
| 5 | AP2 AP3 AP1 AP4 |
| 6 | AP1 AP2 AP4 AP3 |
| 7 | AP2 AP1 AP4 AP3 |
| 8 | AP1 AP4 AP2 AP3 |
| 9 | AP4 AP1 AP2 AP3 |
| 10 | AP4 AP2 AP1 AP3 |
| 11 | AP2 AP4 AP1 AP3 |
| 12 | AP1 AP4 AP3 AP2 |
| 13 | AP4 AP1 AP3 AP2 |
| 14 | AP1 AP3 AP4 AP2 |
| 15 | AP3 AP1 AP4 AP2 |
| 16 | AP3 AP4 AP1 AP2 |
| 17 | AP4 AP3 AP1 AP2 |
| 18 | AP4 AP2 AP3 AP1 |
| 19 | AP2 AP4 AP3 AP1 |
| 20 | AP4 AP3 AP2 AP1 |
| 21 | AP3 AP4 AP2 AP1 |
| 22 | AP3 AP2 AP4 AP1 |
| 23 | AP2 AP3 AP4 AP1 |

*Table 4.15: Antenna path positions after path permutation with N_AP=4*

Figure 4.12 shows four successive mode-2 CS steps that use a different antenna path permutation index at each step. In this figure, extension slot transmissions are present in the first and third steps.

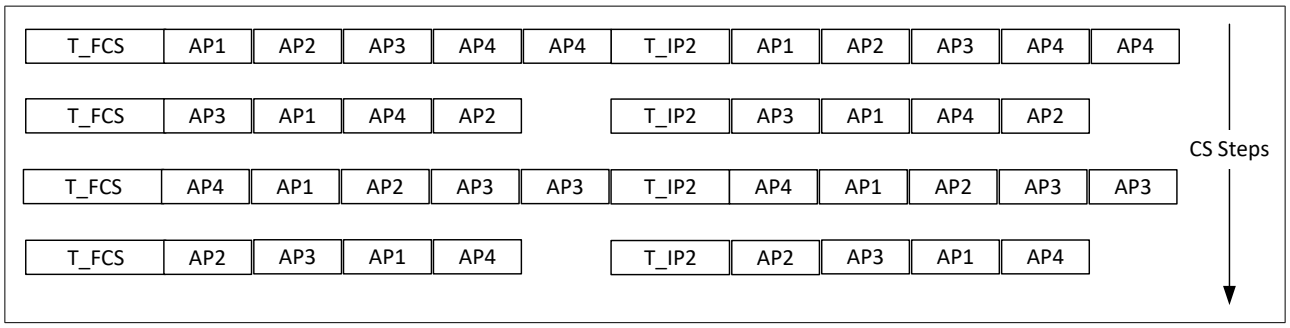



*Figure 4.12: Four mode-2 steps with four antenna paths using antenna path permutation indices 0, 15, 9, and 5*

## 4.8 Channel Sounding random bit generation

A dedicated Channel Sounding Deterministic Random Bit Generator (CS DRBG) is used to coordinate the randomization of several transaction types during a CS procedure. These transaction types are identified by the value of the transaction identifier type, as shown in Table 4.16. *CSTransactionID* represents the transaction identifier being processed by the CS DRBG. The toolbox function CS_DRBG, which is invoked for random bit generation, is described in [Vol 6] Part E, Section 3.1.6.

| **Transaction Identifier (*CSTransactionID*)** | **Transaction Description** |
|---|---|
| 0x00 | Randomization of hop channel set for non-mode-0 steps. Described in Section 4.1. |
| 0x01 | Randomization of hop channel set for mode-0 steps. Described in Section 4.1. |
| 0x02 | Randomization of subevent Sub_Mode (into Main_Mode cycle). Described in Section 4.4. |
| 0x03 | T_PM CS tone extension slot transmission presence. Described in Section 4.4. |
| 0x04 | Antenna path permutation index selection. Described in Section 4.7.5. |
| 0x05 | CS Access Address generation. Described in Section 2.2.1. |
| 0x06 | Sounding sequence marker position randomization. Described in Section 2.4. |
| 0x07 | Sounding sequence marker signal selection. Described in Section 2.4. |
| 0x08 | Random sequence generation. Described in Section 2.5. |
| 0x09 | Backtracking resistance. Described in [Vol 6] Part E, Section 3.1.7. |

*Table 4.16: CS DRBG transaction identifiers (CSTransactionID)*

Each of the transactions listed in Table 4.16 uses a fresh set of random bits at different points within a CS procedure. Each transaction might use a different number of random bits, which might be less than the 128-bit generation output of the CS DRBG. If all bits are not used during a transaction, the remaining bits shall be preserved and used in subsequent transactions until there are not enough remaining bits to complete the transaction. At this point, the remaining bits shall be discarded and a new set of 128 random bits shall be generated.

A newly generated set of random bits might not be needed at each CS step, either because there are still an adequate number of remaining unused bits from a previous DRBG invocation for that specific transaction identifier, or because there is no need for random bit usage for the specific CS mode for that CS step.

For example, if a transaction identifier uses random bits in 8-bit quantities at each step, then at step 0 the DRBG would be invoked to provide 128 bits of random bit output. The first eight bits would be used at step 0, and the remaining 120 bits would be used in 8-bit quantities at each subsequent step through step 15. At step 16, because not enough bits are available to complete the step, a fresh set of 128 random bits is generated and the distribution process of those bits is repeated.

When a new set of 128 random bits is generated, the CS Step_Counter described in [Vol 6] Part E, Section 3.2.2 shall be set to the value of the CS step that is being processed, with the following exceptions:

- When used for Channel Selection Algorithm #3c as described in Section 4.1.4.2, the CS Step_Counter as described in [Vol 6] Part E, Section 3.2.2 shall be set to 0.

A transaction counter represented by *CSTransactionCounter* and used in the CS Transaction_Counter described in [Vol 6] Part E, Section 3.2.2 shall be incremented if a specific transaction ID uses multiple quantities of 128 random bits at any one step. *CSTransactionCounter* shall always begin with a value of 0 the first time a new set of 128 random bits is generated at any CS step for a specific transaction ID.

The order in which the CS DRBG is invoked is critical to maintain synchronization with the peer CS device. The CS DRBG, when used for the purposes of generating random bits for each transaction identifier listed in Table 4.16, shall be invoked with the following ordered rules:

1. By CS steps – In sequential order of CS steps in a CS procedure and subevents, from first to last.
2. By device role – Values used by the initiator shall be generated first, followed by those used by the reflector.
3. By transaction type – In the transmission order of a function using a specific transaction type, if random bits for that transaction type are used more than once within any single CS step.

### 4.8.1 Channel Sounding random number generation function hr1

Within a CS procedure, there are several instances where the CS DRBG is used to seed random number generation with different range sets for the specific transaction types described in Section 4.8. Seeding of random number generation in a uniform distribution within a range is directly supported by the CS DRBG when the range is a power of 2. For other arbitrary ranges, the following description in this section shall be used.

Distribution bias in the random number generation procedure is mitigated with a two-step approach. First, the initial seeding input from the CS DRBG is checked for bias

potential. Next, if a bias potential is detected, a supplemental bit draw from the CS DRBG is performed to extend the granularity of the random number generation process.

N_CS_RANGE_GEN_RANDOMIZED_BITS is the number of randomly generated bits used to perform the randomization seeding process. This value is fixed at 8 bits.

The CS random number generation function hr1 is used to generate random numbers within an arbitrary range. The input to hr1 is an 8-bit unsigned integer R, representing the arbitrary range 0 to R-1 from which a random number is to be generated. This value shall be greater than 0 and less than or equal to 255.

The following value is returned from hr1:

Rout is an 8-bit unsigned integer.

The following temporary value is used:

Trand is a 16-bit unsigned integer.

The input/output format of hr1 is as follows:

Rout = hr1( R )

hr1 processing is as follows:

If ( R == 1 )

then return 0

Trand = R x CS_DRBG( N_CS_RANGE_GEN_RANDOMIZED_BITS )

if ( Trand AND 0xFF ) < ( 256 % R )

$$Rout = \lfloor (( 256 \times CS_DRBG( N_CS_RANGE_GEN_RANDOMIZED_BITS ) \times R ) + Trand )\div 65536 \rfloor$$

else

$$Rout = \lfloor Trand \div 256 \rfloor$$

return Rout

# Wireless Coexistence Signaling and Interfaces

**Specification of the *Bluetooth*® System**

**Volume 7**

**Covered Core Package Version: v6.3**

**Version Date: 2026-05-05**

Bluetooth SIG Proprietary

Wireless Coexistence Signaling
And Interfaces
Part A

# MWS COEXISTENCE LOGICAL SIGNALING SPECIFICATION

*This Part specifies the Mobile Wireless Standards (MWS) coexistence logical interface between the Controller and an MWS device.*

Bluetooth SIG Proprietary

# CONTENTS

**1 Introduction ..... 3898**

**2 Logical interface ..... 3899**
2.1 Coexistence signals ..... 3899
2.1.1 FRAME_SYNC ..... 3899
2.1.2 MWS_RX ..... 3900
2.1.3 BLUETOOTH_RX_PRI ..... 3900
2.1.4 BLUETOOTH_TX_ON ..... 3901
2.1.5 MWS_PATTERN ..... 3901
2.1.6 MWS_TX ..... 3901
2.1.7 802_TX_ON ..... 3901
2.1.8 802_RX_PRI ..... 3902
2.1.9 MWS_INACTIVITY_DURATION ..... 3902
2.1.10 MWS_SCAN_FREQUENCY ..... 3902
2.2 Tolerances for offsets and jitter ..... 3902

# 1 INTRODUCTION

This Part of the specification describes the MWS Coexistence Logical Signaling. A Bluetooth Controller may incorporate a real-time transport interface to transport the logical signals defined in this section between the Bluetooth Controller and an MWS device.

Figure 1.1 depicts the signaling and messaging architecture.

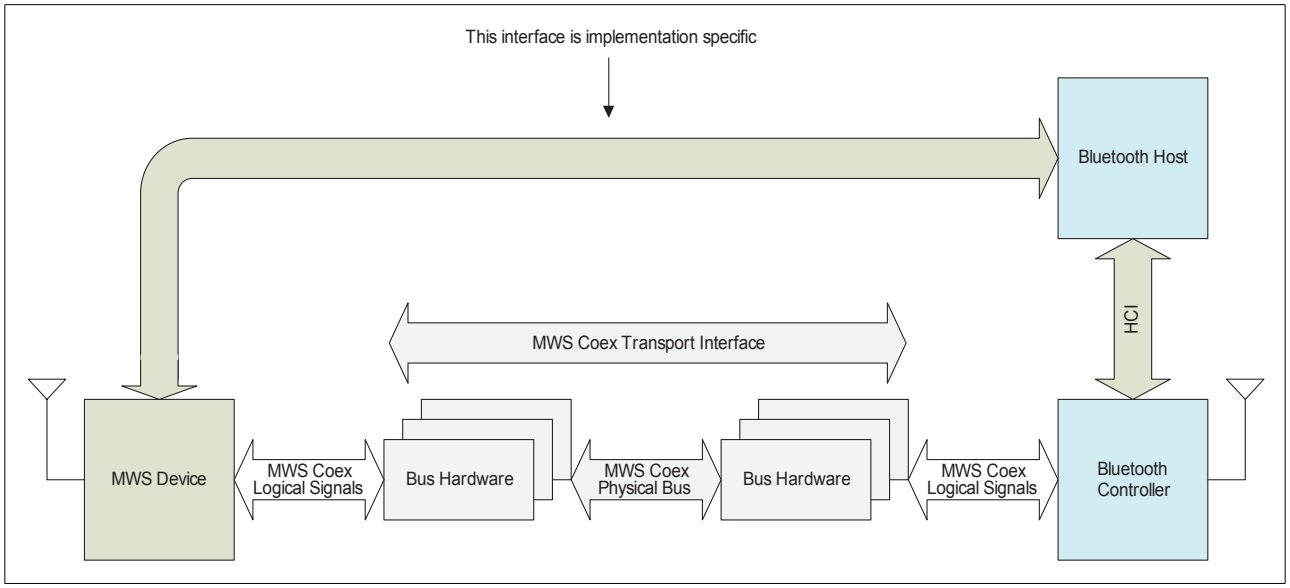



*Figure 1.1: Coexistence signaling/messaging architecture*

MWS Coexistence logical signaling is designed to enable a standard interface to allow an MWS device and a Bluetooth Controller to exchange information and support cooperative coexistence.

MWS Coexistence logical signaling defines a set of signals between the collocated Bluetooth Controller and MWS device. Those signals carry time critical, real-time information such as the start point of an MWS frame. The coexistence logical signaling architecture also includes a transparent data messaging mechanism to enable passing of information between the MWS device and Bluetooth Controller when such information cannot tolerate the long latency (tens of milliseconds) of the signaling path via the Bluetooth Host.

# 2 LOGICAL INTERFACE

## 2.1 Coexistence signals

Table 2.1 defines the logical signals. These logical signals assist in time alignment, protecting the MWS device and the Bluetooth Controller from mutual interference, thus maximizing the usability of the Bluetooth radio.

| Name | Direction | Description |
|---|---|---|
| FRAME_SYNC | MWS → Bluetooth | See Section 2.1.1 |
| MWS_RX | MWS → Bluetooth | See Section 2.1.2 |
| BLUETOOTH_RX_PRI | Bluetooth → MWS | See Section 2.1.3 |
| BLUETOOTH_TX_ON | Bluetooth → MWS | See Section 2.1.4 |
| MWS_PATTERN | MWS → Bluetooth | See Section 2.1.5 |
| MWS_TX | MWS → Bluetooth | See Section 2.1.6 |
| 802_RX_PRI | Bluetooth → MWS | See Section 2.1.7 |
| 802_TX_ON | Bluetooth → MWS | See Section 2.1.8 |
| MWS_INACTIVITY_DURATION | MWS → Bluetooth | See Section 2.1.9 |
| MWS_SCAN_FREQUENCY | MWS → Bluetooth | See Section 2.1.10 |

*Table 2.1: Coexistence signals*

The first 8 of these (FRAME_SYNC, MWS_RX, BLUETOOTH_RX_PRI, BLUETOOTH_TX_ON, MWS_PATTERN, MWS_TX, 802_RX_PRI, and 802_TX_ON) are also referred to as the "real-time coexistence signals".

Many of the signals have associated parameters, which are configured by the Bluetooth Host using HCI commands. There is no requirement for signals that are used internally to be connected to an external interface, although testing requires external control of the FRAME_SYNC signal. For example, a combo-device that integrates a Bluetooth Controller and an MWS radio together does not need to bring out the coexistence signals.

Section 2.2 provides recommended values for the defined offset and jitter parameters.

### 2.1.1 FRAME_SYNC

The FRAME_SYNC signal is sent by the MWS device to indicate the time of the beginning of MWS frames according to the MWS network timing. It provides the anchor point for the Bluetooth Controller to properly align Bluetooth transmission and reception activity with the MWS network frame structure. The time when FRAME_SYNC is sent

over the transport, adjusted for MWS_Frame_Sync_Assert_Offset, indicates the start time of the first Period_Duration parameter in the external frame configuration, which can be set using the HCI_Set_External_Frame_Configuration command.

The MWS device will inform the Bluetooth Controller about changes to the layout and timing of the MWS frame; e.g., by issuing new HCI_Set_External_Frame_Configuration commands.

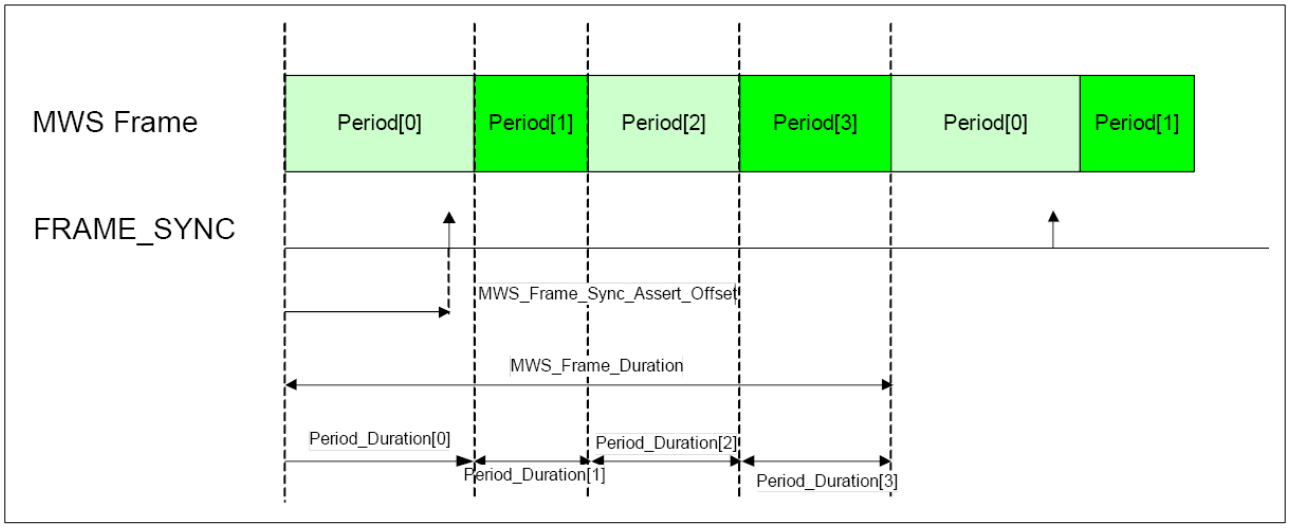



*Figure 2.1: Illustration of FRAME_SYNC*

### 2.1.2 MWS_RX

The MWS_RX signal is sent by the MWS device to indicate that an MWS reception is occurring and to request that the Bluetooth Controller cease ongoing transmission and not start a new transmission. The Bluetooth Controller can occasionally disregard the MWS_RX signal for critical transmissions.

The MWS RX signal should be de-asserted at the time an MWS device stops actively receiving. If there are multiple distinct periods of reception within an MWS downlink duration, the signal may stay asserted until the last period has finished receiving.

### 2.1.3 BLUETOOTH_RX_PRI

The BLUETOOTH_RX_PRI signal is used by the Bluetooth Controller to request that the MWS device cease its transmission and/or refrain from starting a transmission because the Bluetooth Controller is expecting a high priority reception.

The signal should be used minimally by the Bluetooth system so as not to adversely affect the MWS system. There is no guarantee that the MWS device will honor the signal and not transmit or abort an ongoing transmission.

### 2.1.4 BLUETOOTH_TX_ON

The BLUETOOTH_TX_ON signal is sent by the Bluetooth Controller to indicate that it is actively transmitting.

### 2.1.5 MWS_PATTERN

The MWS_PATTERN signal is sent by the MWS device to inform the Bluetooth Controller which MWS_PATTERN is in use.

Up to three different MWS_PATTERNs can be selected: 0, 1, and 2. The MWS device may indicate that the MWS_PATTERN has not changed by setting it to 3. The definitions of the patterns are communicated to the Bluetooth Controller (e.g., by using the HCI_Set_MWS_PATTERN_Configuration HCI command). If the pattern is not currently configured, the behaviour is equivalent to setting a pattern that allows unrestricted activity by the Bluetooth Controller.

At the start of each MWS Frame, as defined by the FRAME_SYNC signal plus the MWS_Frame_Sync_Assert_Offset, the most recent MWS_PATTERN value takes effect as follows.

- If it is 3, the current pattern continues in use.
- If it is the index of the current pattern, then that pattern is restarted.
- Otherwise the indicated pattern is started.

### 2.1.6 MWS_TX

The MWS_TX signal is sent by the MWS device to indicate its transmission state.

The signal should be asserted at the beginning of an MWS transmission and de-asserted at the end of a transmission. If there are multiple transmission periods during an uplink frame, the signal may stay asserted until the end of the last transmission period.

### 2.1.7 802_TX_ON

Bluetooth technology and 802.11 may be collocated and share an interface to coordinate access to the 2.4 GHz ISM band.

The 802_TX_ON is used by the 802.11 device to indicate the state of the 802.11 transmission.

Interference generated by 802.11 to the MWS device will not necessarily be distinguishable from interference created by Bluetooth transmissions.

This signal allows the MWS device to distinguish the interference generated by 802.11 transmissions from the interference generated by the Bluetooth transmissions. The MWS device can use that information to optimize its channel access.

### 2.1.8 802_RX_PRI

This signal requests that the MWS device stop or refrain from any transmission because the WLAN device collocated with the Bluetooth Controller is expecting a high priority reception.

The signal should be used minimally by the 802.11 system so as not to adversely affect the MWS system. There is no guarantee that the MWS device will honor the signal and not transmit or abort an ongoing transmission.

### 2.1.9 MWS_INACTIVITY_DURATION

The MWS_INACTIVITY_DURATION signal provides the time duration until the MWS device is active again. Subsequent MWS_INACTIVITY_DURATION signals override previously sent time durations.

MWS_INACTIVITY_DURATION may be set to zero (i.e. cancel), infinite, or a positive finite duration. The transport layer defines the value for infinite duration and the set of finite durations available.

### 2.1.10 MWS_SCAN_FREQUENCY

The MWS_SCAN_FREQUENCY signal provides an index to a table of RF frequencies during MWS scan operation. If the table is not currently configured, then the behavior during MWS scan operation is vendor-specific.

The MWS device signals MWS_SCAN_FREQUENCY if it starts an inter-frequency scan.

Setting MWS_SCAN_FREQUENCY to a non-zero value indicates the start of the MWS scan period. Setting MWS_SCAN_FREQUENCY to zero indicates the end of the scan period.

The Bluetooth Controller should avoid any transmissions that can interfere with the scan while a scan is active. The Bluetooth Controller can occasionally disregard the signal for critical transmissions.

## 2.2 Tolerances for offsets and jitter

This section lists the recommended tolerances for the signal assertion and de-assertion offsets and jitter for those signals.

The transmitting side should use a value within the range. The receiving side should accept any value within the range and may accept other values.

An offset is a static advance notification or delay between the real physical event and the time when the corresponding signal is issued.

Jitter is variation in the timing of each signal from ideal timing.

Signals that are turned on and off around a period of time have specified values for both assertion and de-assertion offsets and jitter. Signals that represent a single event at a single instant in time have only assertion offset and jitter timing specified.

All jitter values are given as an unsigned value representing the maximum allowable jitter in positive and negative directions. De-assertion and MWS_Frame_Sync_Assert_Offset may be negative (i.e., signal is asserted before the event) or positive (i.e., signal is asserted after the event). All other assertion signals should be negative (i.e., signal is issued before the event).

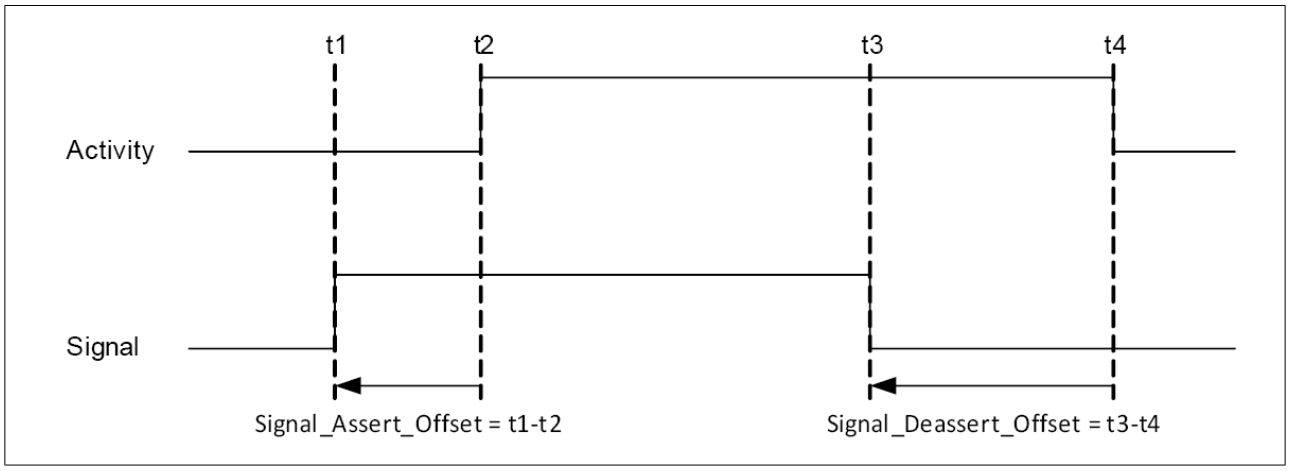



*Figure 2.2: Signal with negative Assert Offset and negative De-assert Offset*

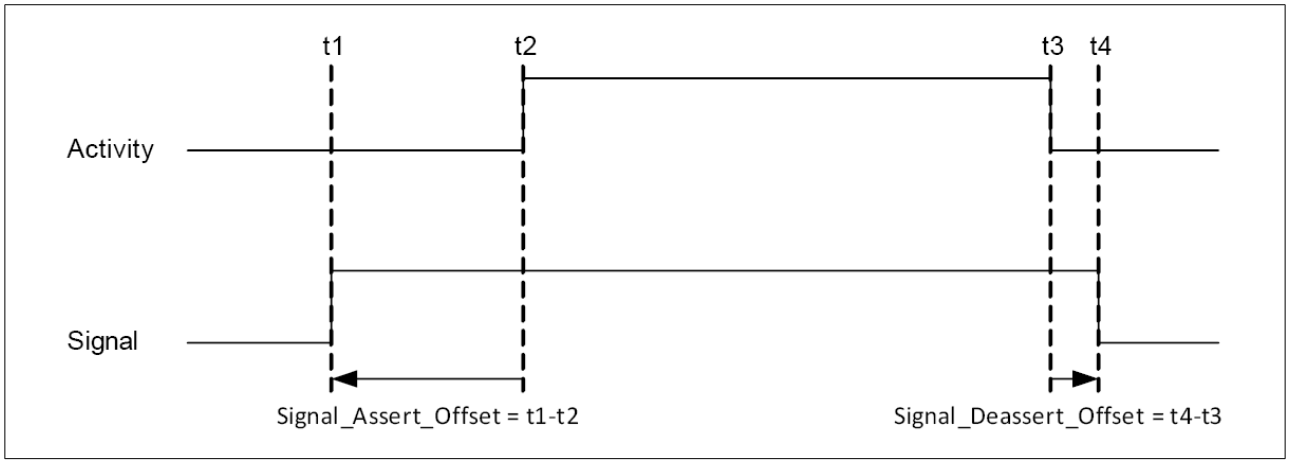



*Figure 2.3: Signal with negative Assert Offset and positive De-assert Offset*

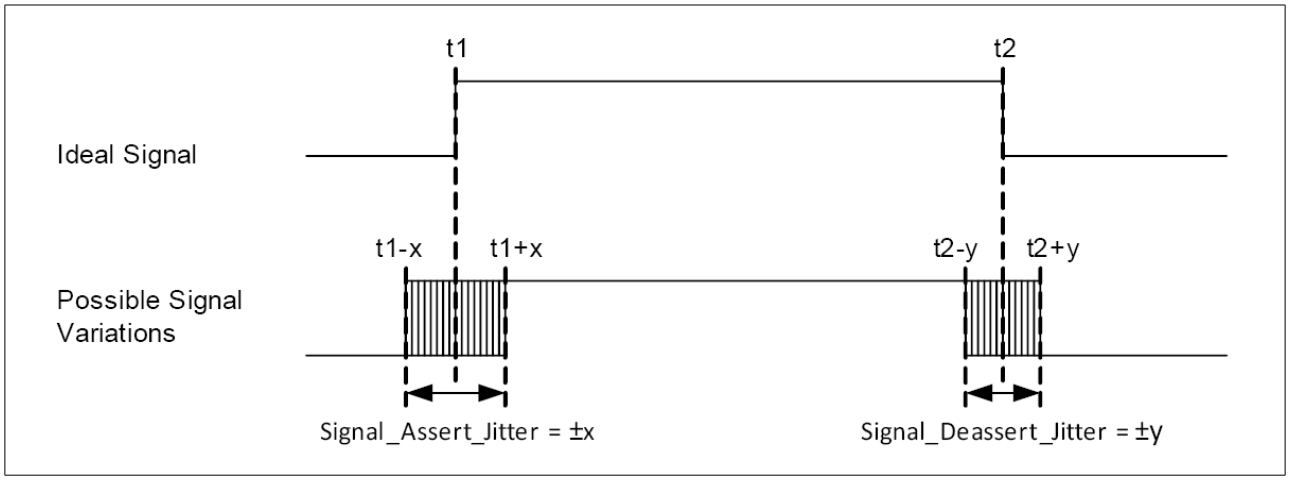



*Figure 2.4: Signal assertion and de-assertion jitter*

| Parameter | Earliest | Latest |
|---|---|---|
| MWS_Frame_Sync_Assert_Offset | -MWS_Frame_Duration | +MWS_Frame_Duration |
| MWS_Rx_Assert_Offset | -2 ms | -20 µs |
| MWS_Rx_Deassert_Offset | -2 ms | 0 |
| MWS_Tx_Assert_Offset | -2 ms | 0 |
| MWS_Tx_Deassert_Offset | -2 ms | 0 |
| MWS_Pattern_Assert_Offset | 0 | +MWS_Frame_Duration |
| MWS_Inactivity_Duration_Assert_Offset | 0 | +MWS_Frame_Duration |
| MWS_Scan_Frequency_Assert_Offset | -2 ms | -20 µs |
| Bluetooth_Rx_Priority_Assert_Offset | -1 ms | 0 |
| Bluetooth_Rx_Priority_Deassert_Offset | -1 ms | 0 |
| 802_Rx_Priority_Assert_Offset | -1 ms | 0 |
| 802_Rx_Priority_Deassert_Offset | -1 ms | 0 |
| Bluetooth_Tx_On_Assert_Offset | -100 µs | 0 |
| Bluetooth_Tx_On_Deassert_Offset | 0 | 100 µs |
| 802_Tx_On_Assert_Offset | -100 µs | 0 |
| 802_Tx_On_Deassert_Offset | 0 | 100 µs |
| MWS_Priority_Assert_Offset_Request | -1 ms | -200 µs |

*Table 2.2: Tolerances for offsets*

| Parameter | Maximum |
|---|---|
| MWS_Frame_Sync_Assert_Jitter | 3 µs |
| MWS_Rx_Assert_Jitter | 5 µs |

| Parameter | Maximum |
|---|---|
| MWS_Rx_Deassert_Jitter | 5 µs |
| MWS_Tx_Assert_Jitter | 5 µs |
| MWS_Tx_Deassert_Jitter | 5 µs |
| MWS_Pattern_Assert_Jitter | 5 µs |
| MWS_Inactivity_Duration_Assert_Jitter | 5 µs |
| MWS_Scan_Frequency_Assert_Jitter | 5 µs |
| Bluetooth_Rx_Priority_Assert_Jitter | 5 µs |
| Bluetooth_Rx_Priority_Deassert_Jitter | 5 µs |
| 802_Rx_Priority_Assert_Jitter | 5 µs |
| 802_Rx_Priority_Deassert_Jitter | 5 µs |
| Bluetooth_Tx_On_Assert_Jitter | 5 µs |
| Bluetooth_Tx_On_Deassert_Jitter | 5 µs |
| 802_Tx_On_Assert_Jitter | 5 µs |
| 802_Tx_On_Deassert_Jitter | 5 µs |

*Table 2.3: Tolerances for jitter*

Wireless Coexistence Signaling
And Interfaces
Part B

# WIRELESS COEXISTENCE INTERFACE 1 (WCI-1) TRANSPORT SPECIFICATION

*This Part specifies the MWS Wireless Coexistence Interface 1 (WCI-1) Transport Interface between the Bluetooth Controller and an MWS device.*

Bluetooth SIG Proprietary

# CONTENTS

**1 Introduction ..... 3908**

**2 Physical layer ..... 3909**

2.1 Physical signal specifications ..... 3910

**3 Transport layer ..... 3912**

3.1 Message types ..... 3912

3.1.1 Real-time Signal message (Type 0) ..... 3913

3.1.2 Transport Control message (Type 1) ..... 3913

3.1.3 Transparent Data message (Type 2) ..... 3914

3.1.4 MWS Inactivity Duration message (Type 3) ..... 3914

3.1.5 MWS Scan Frequency message (Type 4) ..... 3915

# 1 INTRODUCTION

This Part of the specification describes the MWS Wireless Coexistence Interface 1 (WCI-1) Transport for a Bluetooth Controller. It provides a half-duplex UART carrying logical signals framed as UART characters. Only the TXD and RXD UART signals are used.

Note: The physical layers for WCI-1 and WCI-2 (see [Vol 7] Part C) differ but the transport layers are identical.

# 2 PHYSICAL LAYER

The WCI-1 physical layer multiplexes the UART TXD and RXD onto a single wire, using drive strengths to resolve contention. The MWS device uses direct drive to transmit its signals, while the Bluetooth Controller uses a pull up / pull down drive to transmit its signals. The configuration is illustrated in Figure 2.1.

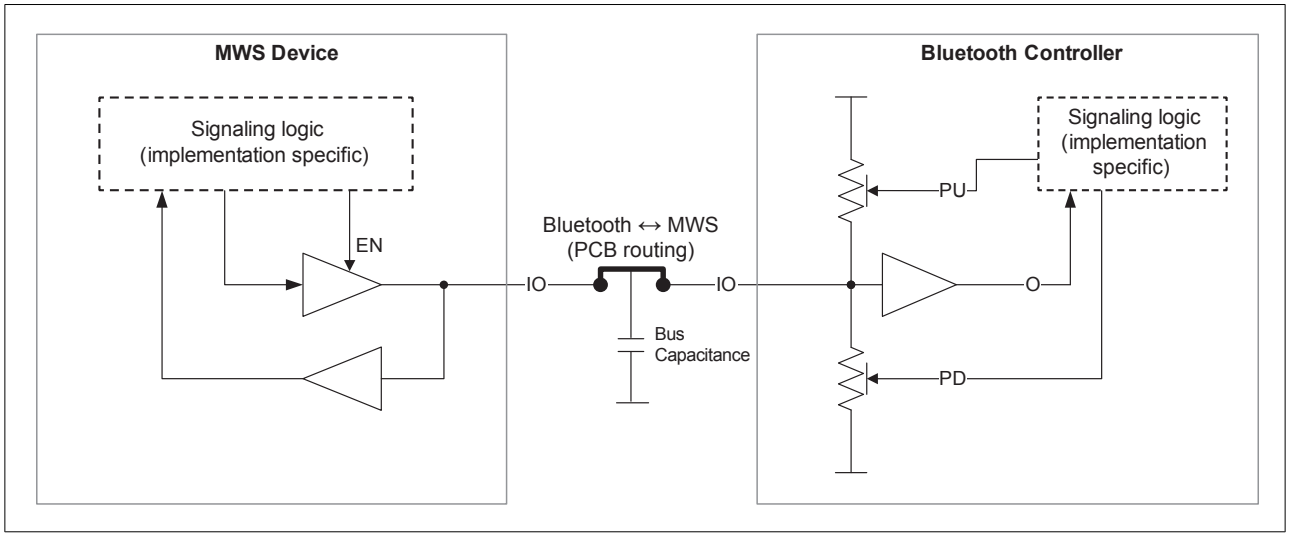



*Figure 2.1: WCI-1 physical interface*

A high voltage on the wire shall be interpreted as a logical 1 and a low voltage shall be interpreted as a logical 0. The actual voltage levels are vendor specific.

The MWS device output buffer shall be in the high impedance state when it is not transmitting. The Bluetooth Controller shall be in the pulled-up state when it is not transmitting.

The MWS device may transmit at any time, using the waveform illustrated in Figure 2.2.

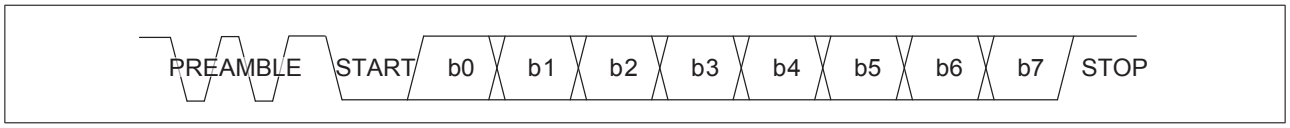



*Figure 2.2: UART waveform for MWS-to-Bluetooth signals*

Every MWS-to-Bluetooth message shall be preceded by a preamble, which consists of 5 bits ‘01011’ (in transmission order). The preamble is sent at a baud rate that is at least twice the baud rate of Bluetooth-to-MWS signals[1]. The nominal drive strength for the preamble should be targeted at no more than 250 Ω equivalent output

[1]The preamble baud rate should be one that is supported by the underlying UART of the Bluetooth device.

resistance. The Bluetooth Controller shall be able to detect the preamble and go into the reception mode to receive the message that follows. If the Bluetooth Controller detects a preamble while it is transmitting, it shall stop the transmission and go into the reception mode. After completion of the reception, it may retransmit the message that was interrupted.

When the Bluetooth Controller is not in the reception mode, it may transmit a message using the pull up / pull down mechanism. The nominal pull strength should be targeted at 4 kΩ ± 1 kΩ equivalent resistance for both pull up and pull down. The Bluetooth-to-MWS waveform is illustrated in Figure 2.3.

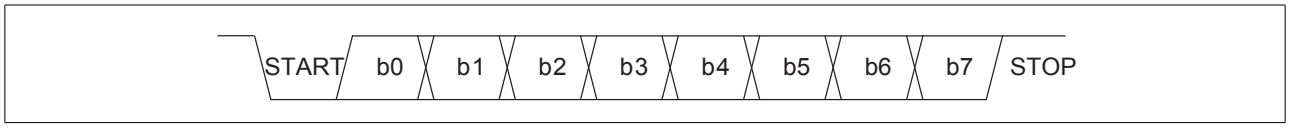



*Figure 2.3: UART waveform for Bluetooth-to-MWS signals*

## 2.1 Physical signal specifications

Table 2.1 and Table 2.2 provide more complete specifications for the physical signals. They are provided as a reference to device makers:

| Symbol | Parameter | Condition | Value | |
|---|---|---|---|---|
| | | | Min | Max |
| $V_{IL}$ | Low level input voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | -0.5 V | 0.2 × $V_{DD}$ |
| $V_{IH}$ | High level input voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | 0.8 × $V_{DD}$ | $V_{DD}$ + 0.5 V |
| $V_{OL}$ | Low level output voltage (with extra pull high $R_B$) | $V_{DD}$ in the range 1.8 V to 2.5 V[1]<br>$R_B$ = 2.5 kΩ | 0 V | 0.1 × $V_{DD}$ |
| $V_{OH}$ | High level output voltage (with extra pull low $R_B$) | $V_{DD}$ in the range 1.8 V to 2.5 V[1]<br>$R_B$ = 2.5 kΩ | 0.9 × $V_{DD}$ | $V_{DD}$ |
| $C_{IO}$ | Capacitance for I/O | *none* | 0 F | 5 pF |
| CB | Capacitive load for bus | *none* | 0 F | 10 pF |
| $R_{ON}$ | Turn on impedance | *none* | 0 Ω | 250 Ω |
| $T_R$ | Rise time (10% to 90% swing time, with extra pull low $R_B$ and capacitive load $C_L$ = $C_B$ + other IO) | $R_B$ = 2.5 kΩ<br>$C_L$ = 15 pF | 0 s | 50 ns |

| Symbol | Parameter | Condition | Value | |
|---|---|---|---|---|
| | | | Min | Max |
| $T_F$ | Fall time (90% to 10% swing time, with extra pull high $R_B$ and capacitive load $C_L = C_B$ + other IO) | $R_B$ = 2.5 kΩ<br>$C_L$ = 15 pF | 0 s | 50 ns |
| $CLK_j$ | Clock jitter | *none* | *none* | 1% |

*Table 2.1: WCI-1 UART physical signal specification for the MWS device*

| Symbol | Parameter | Condition | Value | |
|---|---|---|---|---|
| | | | Min | Max |
| $V_{IL}$ | Low level input voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | -0.5 V | 0.2 × $V_{DD}$ |
| $V_{IH}$ | High level input voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | 0.8 × $V_{DD}$ | $V_{DD}$ + 0.5 V |
| $V_{OL}$ | Low level output voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | 0 V | 0.1 × $V_{DD}$ |
| $V_{OH}$ | High level output voltage | $V_{DD}$ in the range 1.8 V to 2.5 V[1] | 0.9 × $V_{DD}$ | $V_{DD}$ |
| $C_{IO}$ | Capacitance for I/O | *none* | 0 F | 5 pF |
| CB | Capacitive load for bus | *none* | 0 F | 10 pF |
| $R_P$ | Pull up/pull down resistance | *none* | 3 kΩ | 5 kΩ |
| $T_R$ | Rise time (10% to 90% swing time, with capacitive load $C_L = C_B$ + other IO) | $C_L$ = 15 pF | 0 s | 220 ns |
| $T_F$ | Fall time (90% to 10% swing time, with capacitive load $C_L = C_B$ + other IO) | $C_L$ = 15 pF | 0 s | 220 ns |
| $CLK_j$ | Clock jitter | *none* | *none* | 1% |

*Table 2.2: WCI-1 UART physical signal specification for the Bluetooth Controller*

Notes:

1. The voltage levels are vendor specific. Table 2.1 and Table 2.2 do not cover all possible ranges of voltage for all devices, nor is it required that a single device be able to operate in the full range indicated here.
2. The values in these tables are based on a MWS-to-Bluetooth baud rate of 4 megabits per second and a Bluetooth-to-MWS baud rate of 1 megabit per second, in each case ±1%. The actual baud rates used are vendor-specific.

# 3 TRANSPORT LAYER

The transport layer defines the mapping of the logical coexistence signals (see [Vol 7] Part A) onto the physical transport channel.

The 8-bit UART character is divided into two portions with three bits for the message type indicator and five bits for the message body. The bit with index 0 is the LSB and shall be transmitted first.

| b0 | b1 | b2 | b3 | b4 | b5 | b6 | b7 |
|---|---|---|---|---|---|---|---|
| Type[0] | Type[1] | Type[2] | MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |

*Table 3.1: Transport layer format*

## 3.1 Message types

This section describes the message formats for the logical coexistence signals. The message types are listed in Table 3.2.

| Message Type Indicator | Direction | Message Type |
|---|---|---|
| 0 | MWS ↔ Bluetooth | Real-time Signal message |
| 1 | MWS ↔ Bluetooth | Transport Control message |
| 2 | MWS ↔ Bluetooth | Transparent Data message |
| 3 | MWS → Bluetooth<br>Bluetooth → MWS | MWS Inactivity Duration message<br>RFU |
| 4 | MWS → Bluetooth<br>Bluetooth → MWS | MWS Scan Frequency message<br>RFU |
| 5 | MWS → Bluetooth<br>Bluetooth → MWS | RFU<br>RFU |
| 6 | Vendor-specific | |
| 7 | Vendor-specific | |

*Table 3.2: Message types*

The logical coexistence signals are listed in Table 3.3.

| Signal Name | Description |
|---|---|
| FRAME_SYNC | See [Vol 7] Part A, Section 2.1.1 |
| MWS_RX | See [Vol 7] Part A, Section 2.1.2 |
| BLUETOOTH_RX_PRI | See [Vol 7] Part A, Section 2.1.3 |

| Signal Name | Description |
|---|---|
| BLUETOOTH_TX_ON | See [Vol 7] Part A, Section 2.1.4 |
| MWS_PATTERN | See [Vol 7] Part A, Section 2.1.5 |
| MWS_TX | See [Vol 7] Part A, Section 2.1.6 |
| 802_TX_ON | See [Vol 7] Part A, Section 2.1.7 |
| 802_RX_PRI | See [Vol 7] Part A, Section 2.1.8 |
| MWS_INACTIVITY_DURATION | See [Vol 7] Part A, Section 2.1.9 |
| MWS_SCAN_FREQUENCY | See [Vol 7] Part A, Section 2.1.10 |

*Table 3.3: Coexistence signals*

### 3.1.1 Real-time Signal message (Type 0)

The Real-time Signal message is used to transport the real-time coexistence signals (see [Vol 7] Part A) over the WCI-1 transport interface.

The Real-time Signal message conveys all the real-time coexistence signals in one message. The time reference point for the Real-time Signal message is the end of MSG[4] (i.e. the transition to the STOP bit).

Two Real-time Signal messages are defined, one from the Bluetooth Controller to the MWS device and another from the MWS device to the Bluetooth Controller.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| FRAME_SYNC | MWS_RX | MWS_TX | MWS_PATTERN[0] | MWS_PATTERN[1] |

*Table 3.4: Real-time Signal message from MWS device to Bluetooth Controller*

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| BLUETOOTH_RX_PRI | BLUETOOTH_TX_ON | 802_RX_PRI | 802_TX_ON | RFU |

*Table 3.5: Real-time Signal message from Bluetooth Controller to MWS device*

### 3.1.2 Transport Control message (Type 1)

The Transport Control message can request state information from the MWS device's coexistence interface.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| RESEND_REAL_TIME | RFU | RFU | RFU | RFU |

*Table 3.6: Transport Control message*

| Signal Name | Description |
|---|---|
| RESEND_REAL_TIME | This bit is set to request a status update of the real-time coexistence signals. The signal is usually used after wake-up from sleep of the transport interface.<br>If the receiving device's transport interface is awake it shall send a Real-time message with the current status of the real-time coexistence signals within 4 UART character periods. If the signal is not received within 4 UART character periods the device is considered asleep. |

*Table 3.7: Transport Control signals*

### 3.1.3 Transparent Data message (Type 2)

The Transparent Data message can be used to exchange non-time critical signals between the MWS device and the Bluetooth Controller. The interface does not guarantee the delivery of a message. Protocol and content of the message are vendor specific.

Each octet to be transmitted is split into two 4-bit parts, called "nibbles". The least significant nibble consists of bits 0 to 3 of the octet and shall be transmitted first. The most significant nibble consists of bits 4 to 7 of the octet and shall be transmitted after the least significant nibble.

A least significant nibble shall be discarded if the next nibble is a least significant nibble. A most significant nibble shall only be accepted if the preceding nibble was a least significant nibble.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| NIBBLE_POSITION | DATA[0] / DATA[4] | DATA[1] / DATA[5] | DATA[2] / DATA[6] | DATA[3] / DATA[7] |

*Table 3.8: Transparent Data message*

| Signal Name | Description |
|---|---|
| NIBBLE_POSITION | 0 – Least Significant Nibble<br>1 – Most Significant Nibble |
| DATA[n]; n = 0..7 | Data bits of the message octet |

*Table 3.9: Transparent Data bits*

### 3.1.4 MWS Inactivity Duration message (Type 3)

The MWS Inactivity Duration message is used to send the MWS_INACTIVITY_DURATION signal from the MWS device to the Bluetooth Controller.

The message is sent at the beginning of an MWS inactivity period.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| DURATION[0] | DURATION[1] | DURATION[2] | DURATION[3] | DURATION[4] |

*Table 3.10: MWS Inactivity Duration message*

The MWS Inactivity Duration is encoded in 5 bits. DURATION is unsigned.

When DURATION = 0, MWS_INACTIVITY_DURATION is cancelled.

When DURATION = 31, MWS_INACTIVITY_DURATION is infinite.

Otherwise, MWS_INACTIVITY_DURATION is given by the formula:

MWS_INACTIVITY_DURATION = DURATION × 5 ms

### 3.1.5 MWS Scan Frequency message (Type 4)

The MWS Scan Frequency message is used to send the MWS_SCAN_FREQUENCY signal from the MWS device to the Bluetooth Controller.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| FREQ[0] | FREQ[1] | FREQ[2] | FREQ[3] | FREQ[4] |

*Table 3.11: MWS Scan Frequency message*

The MWS Scan Frequency index is encoded in 5 bits. FREQ is unsigned.

Wireless Coexistence Signaling
And Interfaces
Part C

# WIRELESS COEXISTENCE INTERFACE 2 (WCI-2) TRANSPORT SPECIFICATION

*This Part specifies the MWS Wireless Coexistence Interface 2 (WCI-2) Transport Interface between the Bluetooth Controller and an MWS device.*

Bluetooth SIG Proprietary

# CONTENTS

**1 Introduction ..... 3918**

**2 Physical layer ..... 3919**

**3 Transport layer ..... 3920**
3.1 Message types ..... 3920
3.1.1 Real-time Signal message (Type 0) ..... 3921
3.1.2 Transport Control message (Type 1) ..... 3921
3.1.3 Transparent Data message (Type 2) ..... 3922
3.1.4 MWS Inactivity Duration message (Type 3) ..... 3922
3.1.5 MWS Scan Frequency message (Type 4) ..... 3923

# 1 INTRODUCTION

This Part of the specification describes the MWS Wireless Coexistence Interface 2 (WCI-2) Transport Interface for a Bluetooth Controller.

Note: The physical layers for WCI-2 and WCI-1 (see [Vol 7] Part B) differ but the transport layers are identical.

# 2 PHYSICAL LAYER

The WCI-2 Transport is based on a standard full duplex UART carrying logical signals framed as UART characters. Only the TXD and RXD UART signals are used. The interface supports multiple logical channels.

The messaging is based on a standard UART format.

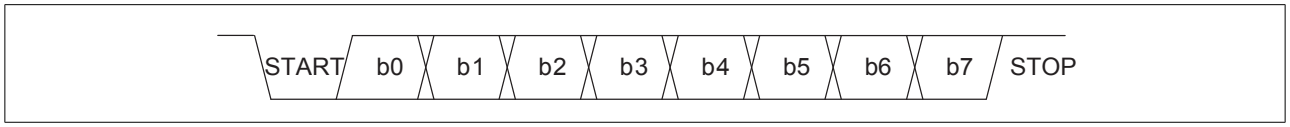



*Figure 2.1: UART waveform*

The UART signals shall be connected in a null-modem fashion; i.e. the local TXD shall be connected to the remote RXD and vice versa.

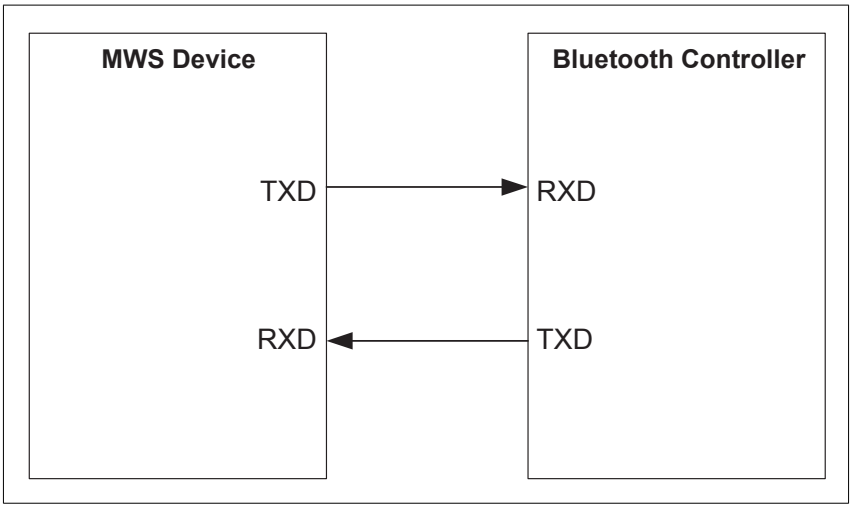



*Figure 2.2: WCI-2 physical interface*

# 3 TRANSPORT LAYER

The transport layer defines the mapping of the logical signals (see [Vol 7] Part A) onto the physical transport channel.

The 8 bit UART character is divided into two portions with three bits for the message type indicator and five bits for the message body. The bit with index 0 is the LSB and shall be transmitted first.

| b0 | b1 | b2 | b3 | b4 | b5 | b6 | b7 |
|---|---|---|---|---|---|---|---|
| Type[0] | Type[1] | Type[2] | MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |

*Table 3.1: Transport layer format*

## 3.1 Message types

This section describes the message formats for the logical coexistence signals. The message types are listed in Table 3.2.

| Message Type Indicator | Direction | Message Type |
|---|---|---|
| 0 | MWS ↔ Bluetooth | Real-time Signal message |
| 1 | MWS ↔ Bluetooth | Transport Control message |
| 2 | MWS ↔ Bluetooth | Transparent Data message |
| 3 | MWS → Bluetooth<br>Bluetooth → MWS | MWS Inactivity Duration message<br>RFU |
| 4 | MWS → Bluetooth<br>Bluetooth → MWS | MWS Scan Frequency message<br>RFU |
| 5 | MWS → Bluetooth<br>Bluetooth → MWS | RFU<br>RFU |
| 6 | Vendor-specific | |
| 7 | Vendor-specific | |

*Table 3.2: Message types*

The logical coexistence signals are listed in Table 3.3.

| Logical Signal Name | Description |
|---|---|
| FRAME_SYNC | See [Vol 7] Part A, Section 2.1.1 |
| MWS_RX | See [Vol 7] Part A, Section 2.1.2 |
| BLUETOOTH_RX_PRI | See [Vol 7] Part A, Section 2.1.3 |

| Logical Signal Name | Description |
|---|---|
| BLUETOOTH_TX_ON | See [Vol 7] Part A, Section 2.1.4 |
| MWS_PATTERN | See [Vol 7] Part A, Section 2.1.5 |
| MWS_TX | See [Vol 7] Part A, Section 2.1.6 |
| 802_TX_ON | See [Vol 7] Part A, Section 2.1.7 |
| 802_RX_PRI | See [Vol 7] Part A, Section 2.1.8 |
| MWS_INACTIVITY_DURATION | See [Vol 7] Part A, Section 2.1.9 |
| MWS_SCAN_FREQUENCY | See [Vol 7] Part A, Section 2.1.10 |

*Table 3.3: Coexistence signals*

### 3.1.1 Real-time Signal message (Type 0)

The Real-time Signal message is used to transport the real-time coexistence signals (see [Vol 7] Part A) over the WCI-2 transport.

The Real-time Signal message conveys all the real-time coexistence signals in one message. The time reference point for the Real-time Signal message is the end of MSG[4] (i.e. the transition to the Stop bit).

Two Real-time Signal messages are defined, one from the Bluetooth Controller to the MWS device and another from the MWS device to the Bluetooth Controller.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| FRAME_SYNC | MWS_RX | MWS_TX | MWS_PATTERN[0] | MWS_PATTERN [1] |

*Table 3.4: Real-time Signal message from MWS device to Bluetooth Controller*

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| BLUETOOTH_RX_PRI | BLUETOOTH_TX_ON | 802_RX_PRI | 802_TX_ON | RFU |

*Table 3.5: Real-time Signal message from Bluetooth Controller to MWS device*

### 3.1.2 Transport Control message (Type 1)

The Transport Control message can request state information from the MWS device's coexistence interface.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| RESEND_REAL_TIME | RFU | RFU | RFU | RFU |

*Table 3.6: Transport Control message*

| Signal Name | Description |
|---|---|
| RESEND_REAL_TIME | This bit is set to request a status update of the real-time coexistence signals. The signal is usually used after wake-up from sleep of the transport interface.<br>If the receiving device's transport interface is awake it shall send a Real-time message with the current status of the real-time coexistence signals within 4 UART character periods. If the signal is not received within 4 UART character periods the device is considered asleep. |

*Table 3.7: Transport Control signals*

### 3.1.3 Transparent Data message (Type 2)

The Transparent Data message can be used to exchange non-time critical messages between the MWS device and the Bluetooth Controller. The interface does not guarantee the delivery of a message. Protocol and content of the message are vendor specific.

Each octet to be transmitted is split into two 4-bit parts, called "nibbles". The least significant nibble consists of bits 0 to 3 of the octet and shall be transmitted first. The most significant nibble consists of bits 4 to 7 of the octet and shall be transmitted after the least significant nibble.

A least significant nibble shall be discarded if the next nibble is a least significant nibble. A most significant nibble shall only be accepted if the preceding nibble was a least significant nibble.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| NIBBLE_POSITION | DATA[0]/<br>DATA[4] | DATA[1]/<br>DATA[5] | DATA[2]/<br>DATA[6] | DATA[3]/<br>DATA[7] |

*Table 3.8: Transparent Data message*

| Signal Name | Description |
|---|---|
| NIBBLE_POSITION | 0 – Least Significant Nibble<br>1 – Most Significant Nibble |
| DATA[n]; n=0 .. 7 | Data bits of the message octet |

*Table 3.9: Transparent Data bits*

### 3.1.4 MWS Inactivity Duration message (Type 3)

The MWS Inactivity Duration message is used to send the MWS_INACTIVITY_DURATION signal from the MWS device to the Bluetooth Controller.

The message is sent at the beginning of the MWS inactivity period.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| DURATION[0] | DURATION[1] | DURATION[2] | DURATION[3] | DURATION[4] |

*Table 3.10: MWS Inactivity Duration message*

The MWS Inactivity Duration is encoded in 5 bits. DURATION is unsigned.

When DURATION = 0, MWS_INACTIVITY_DURATION is cancelled.

When DURATION = 31, MWS_INACTIVITY_DURATION is infinite.

Otherwise, MWS_INACTIVITY_DURATION is given by the formula:

MWS_INACTIVITY_DURATION = DURATION × 5 ms

### 3.1.5 MWS Scan Frequency message (Type 4)

The MWS Scan Frequency message is used to send the MWS_SCAN_FREQUENCY signal from the MWS device to the Bluetooth Controller.

| MSG[0] | MSG[1] | MSG[2] | MSG[3] | MSG[4] |
|---|---|---|---|---|
| FREQ[0] | FREQ[1] | FREQ[2] | FREQ[3] | FREQ[4] |

*Table 3.11: MWS Scan Frequency message*

The MWS Scan Frequency index is encoded in 5 bits. FREQ is unsigned.

**Bluetooth SIG Proprietary**

Version Date: 2026-05-05