Mesh Protocol (MESH)

***Bluetooth*® Test Suite**

- **Revision:** MESH.TS.p19
- **Revision Date:** 2026-11-03
- **Prepared By:** Mesh Working Group
- **Published during TCRL:** TCRL.pkg105

This document, regardless of its title or content, is not a Bluetooth Specification as defined in the Bluetooth Patent/Copyright License Agreement ("PCLA") and Bluetooth Trademark License Agreement. Use of this document by members of Bluetooth SIG is governed by the membership and other related agreements between Bluetooth SIG Inc. ("Bluetooth SIG") and its members, including the PCLA and other agreements posted on Bluetooth SIG's website located at www.bluetooth.com.

THIS DOCUMENT 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 WARRANTY OF MERCHANTABILITY, TITLE, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, THAT THE CONTENT OF THIS DOCUMENT IS FREE OF ERRORS.

TO THE EXTENT NOT PROHIBITED BY LAW, BLUETOOTH SIG, ITS MEMBERS, AND THEIR AFFILIATES DISCLAIM ALL LIABILITY ARISING OUT OF OR RELATING TO USE OF THIS DOCUMENT AND ANY INFORMATION CONTAINED IN THIS DOCUMENT, 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 SUCH DAMAGES.

This document is proprietary to Bluetooth SIG. This document may contain or cover subject matter that is intellectual property of Bluetooth SIG and its members. The furnishing of this document does not grant any license to any intellectual property of Bluetooth SIG or its members.

This document is subject to change without notice.

Copyright © 2015–2026 by Bluetooth SIG, Inc. 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.

Contents

1 Scope . . . 22
2 References, definitions, and abbreviations . . . 23
2.1 References . . . 23
2.2 Definitions . . . 23
2.3 Acronyms and abbreviations . . . 23
3 Test Suite Structure (TSS) . . . 24
3.1 Test Strategy . . . 24
3.1.1 Power-Up Behavior Tests . . . 24
3.1.2 Common Behaviors and Procedures . . . 24
3.2 Test groups . . . 25
4 Test cases (TC) . . . 26
4.1 Introduction . . . 26
4.1.1 Test case identification conventions . . . 26
4.1.2 Conformance . . . 28
4.1.3 Pass/Fail verdict conventions . . . 29
4.2 Setup preambles . . . 29
4.2.1 State: Node PB-ADV Link Established . . . 29
4.2.2 State: Provisioner PB-ADV Link Established . . . 30
4.2.3 State: Node PB-GATT Link Established . . . 30
4.2.4 State: Provisioner PB-GATT Link Established . . . 30
4.2.5 Establish an ATT Bearer Connection . . . 30
4.2.6 Network and Application Security Credentials . . . 31
4.2.7 State: Adding a Bridging Table Entry, IUT as Subnet Bridge . . . 31
4.2.8 Common Initial Test Conditions for Directed Forwarding Tests . . . 31
4.2.9 State: Path Established, as Origin, Unicast Address Destination . . . 32
4.2.10 State: Path Established as Directed Relay, Unicast Address Destination . . . 33
4.2.11 State: Path Established as Destination, Unicast Address Destination . . . 34
4.2.12 Proxy Connection with Proxy Client, IUT as Directed Proxy Server . . . 35
4.2.13 State: Proxy Connection with Directed Proxy Client, IUT as Directed Proxy Server . . . 36
4.3 ADV Provisioning Bearer . . . 38
MESH/NODE/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close] . . . 38
MESH/NODE/PBADV/BV-02-C [Ignore Parallel Link Messages] . . . 39
MESH/NODE/PBADV/BV-03-C [Unprovisioned Device Receives and Sends PB-ADV PDU] . . . 40
MESH/NODE/PBADV/BV-04-C [Unprovisioned Device Receives and Sends PB-ADV PDU with Segmentation] . . . 41
MESH/PVNR/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close – Provisioner] . . . 43
MESH/PVNR/PBADV/BV-02-C [Provisioner Sends and Receives PB-ADV PDU] . . . 44
MESH/PVNR/PBADV/BV-03-C [Provisioner Sends and Receives PB-ADV PDU with Segmentation] . . . 45
MESH/PVNR/PBADV/BI-01-C [PB-ADV Link Establishment Timeout] . . . 46
MESH/NODE/PBADV/BI-01-C [Ignore Link Open Attempts with Incorrect Device UUID] . . . 47
MESH/NODE/PBADV/BI-02-C [Ignore Provisioning Bearer Control PDU with RFU BearerOpcode Value] . . . 48
MESH/NODE/PBADV/BI-03-C [Receive Link Close Message with RFU Reason Value] . . . 49
MESH/NODE/PBADV/BI-04-C [Ignore Connectable Advertisement using Mesh Provisioning AD Type] . . . 49
4.4 Mesh Provisioning Service . . . 50
MESH/NODE/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning] . . . 50
MESH/NODE/MPS/BV-02-C [PB-GATT Link Establishment and Link Close] . . . 51

MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device] . . . 52
4.4.1 Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device . . . 52
MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device] . . . 53
MESH/NODE/MPS/BV-05-C [Expose Mesh Provisioning Data In Characteristic – Unprovisioned Device] . . . 53
MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device] . . . 53
MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device] . . . 54
MESH/NODE/MPS/BV-08-C [Receive Mesh Provisioning Data Out Notifications – Unprovisioned Device] . . . 55
MESH/NODE/MPS/BV-09-C [Unprovisioned Device Sends and Receives Complete Provisioning PDUs] . . . 56
MESH/NODE/MPS/BV-10-C [Unprovisioned Device Sends and Receives Segmented Provisioning PDU Message] . . . 57
MESH/PVNR/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning – Provisioner] . . . 58
MESH/PVNR/MPS/BV-02-C [PB-GATT Link Establishment and Link Close – Provisioner] . . . 59
4.4.2 Mesh Provisioning Service Characteristic Discovery by Provisioner . . . 59
MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] . . . 60
MESH/PVNR/MPS/BV-04-C [Discover Mesh Provisioning Data In Characteristic] . . . 60
MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner] . . . 60
MESH/PVNR/MPS/BV-06-C [Enable Notifications for Mesh Provisioning Data Out - Provisioner] . . . 61
MESH/PVNR/MPS/BV-08-C [Provisioner Sends and Receives Complete Provisioning PDUs] . . . 62
MESH/PVNR/MPS/BV-09-C [Provisioner Sends and Receives Segmented Message] . . . 63
MESH/NODE/MPS/BI-01-C [Mesh Provisioning Data In Characteristic Only Supports Proxy PDU containing Provisioning PDU] . . . 64
4.5 Provisioning Procedure . . . 65
4.5.1 Provisioning Procedure - Unprovisioned Device . . . 65
MESH/NODE/PROV/BV-01-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-02-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-03-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-04-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-05-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-06-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-07-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-08-C [Provisioning Procedure, Public Key OOB, No OOB Authentication, Algorithm 0] . . . 65
MESH/NODE/PROV/BV-13-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-14-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-15-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-16-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-17-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-18-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-19-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-20-C [Provisioning Procedure, Public Key OOB, No OOB Authentication, Algorithm 1] . . . 65
MESH/NODE/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout] . . . 69
MESH/NODE/PROV/BV-10-C [Unprovisioned Device Sends Provisioning Failed Message on Provisioning Error] . . . 70
MESH/NODE/PROV/BV-11-C [Provisioned Node Updates Security Flags Based on the Provisioning Data PDU] . . . 72
MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session] . . . 72
4.5.2 Provisioning Procedure - Provisioner . . . 74
MESH/PVNR/PROV/BV-01-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-02-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication, Algorithm 0] . . . 76

MESH/PVNR/PROV/BV-03-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-04-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-05-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-06-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-07-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-08-C [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication, Algorithm 0] . . . 76
MESH/PVNR/PROV/BV-11-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-12-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-13-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-14-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-15-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-16-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-17-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-18-C [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication, Algorithm 1] . . . 76
MESH/PVNR/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout – Provisioner] . . . 78
MESH/PVNR/PROV/BV-10-C [Provisioner Disconnects Provisioning Bearer on Provisioning Protocol Failure from Unprovisioned Device] . . . 79
MESH/NODE/PROV/BI-01-C [Device Resets Attention Timer State when Provisioning Bearer Dropped] . . . 80
MESH/NODE/PROV/BI-02-C [Reject Provisioning Start PDU with RFU values] . . . 81
MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values] . . . 82
MESH/NODE/PROV/BI-13-C [Reject Invalid Public Keys] . . . 84
MESH/PVNR/PROV/BI-14-C [Reject Invalid Public Keys] . . . 86
MESH/NODE/PROV/BI-15-C [Reject Provisioning PDU, Unsupported Type] . . . 88
MESH/PVNR/PROV/BI-16-C [Reject Provisioning PDU, Unsupported Type] . . . 88
MESH/NODE/PROV/BI-17-C [Reject Invalid Confirmation Value] . . . 89
MESH/PVNR/PROV/BI-18-C [Reject Invalid Confirmation Value] . . . 91
4.6 Private Beacon Model . . . 93
4.6.1 Private Beacon Procedures . . . 93
MESH/SR/PRB/PBS/BV-01-C [Respond to Private Beacon Get/Set Messages] . . . 93
MESH/SR/PRB/PBS/BI-01-C [Ignore Invalid Private Beacon Set Messages] . . . 95
MESH/CL/PRB/PBS/BV-01-C [Send Private Beacon Get/Set messages] . . . 97
4.6.2 Private GATT Proxy . . . 98
MESH/SR/PRB/PGPXY/BV-01-C [Respond to Private GATT Proxy Set/Get Messages] . . . 99
MESH/SR/PRB/PGPXY/BV-02-C [Private Network Identity Advertising on Multiple Subnets] . . . 102
MESH/SR/PRB/PGPXY/BI-01-C [Respond to Private GATT Proxy Set/Get Messages – Invalid Parameters] . . . 104

MESH/CL/PRB/PGPXY/BV-01-C [Send Private GATT Proxy Set/Get Messages] . . . 104
4.6.3 Private Node Identity . . . 106
MESH/SR/PRB/PNID/BV-01-C [Respond to Private Node Identity Set/Get Messages] . . . 106
MESH/SR/PRB/PNID/BV-02-C [Private Node Identity Advertising on Multiple Subnets] . . . 109
MESH/SR/PRB/PNID/BV-03-C [Private Node Identity NetKey Deleted] . . . 112
MESH/SR/PRB/PNID/BI-01-C [Respond to Private Node Identity Set/Get Messages – Invalid Parameters] . . . 113
MESH/SR/PRB/PNID/BI-02-C [Receive Private Node Identity Set Message – Change to Not Supported] . . . 115
MESH/CL/PRB/PNID/BV-01-C [Send Private Node Identity Set/Get Messages] . . . 116
4.7 Mesh Beacons . . . 118
4.7.1 Advertising Mesh Beacons . . . 118
MESH/NODE/BCN/SNB/BV-01-C [Advertising Secure Network Beacons] . . . 119
MESH/NODE/BCN/PRB/BV-01-C [Advertising Mesh Private Beacons] . . . 119
MESH/NODE/BCN/PRB/BV-02-C [Scanning for Mesh Beacons] . . . 119
4.8 Network Layer . . . 120
4.8.1 Transmit Mesh Messages over Advertising Bearer to Different Destination Address Types . . . 121
MESH/NODE/NET/BV-01-C [Send Mesh Message to Unicast Address] . . . 121
MESH/NODE/NET/BV-02-C [Send Mesh Message to Virtual Address] . . . 121
MESH/NODE/NET/BV-03-C [Send Mesh Message to Dynamically Assigned Group Address] . . . 121
MESH/NODE/NET/BV-04-C [Send Mesh Message to All-proxies Address] . . . 121
MESH/NODE/NET/BV-05-C [Send Mesh Message to All-friends Address] . . . 121
MESH/NODE/NET/BV-06-C [Send Mesh Message to All-relays Address] . . . 121
MESH/NODE/NET/BV-07-C [Send Mesh Message to All-nodes Address] . . . 121
4.8.2 Receive Mesh Messages over Advertising Bearer Sent to Different Destination Address Types . . . 123
MESH/NODE/NET/BV-08-C [Receive Mesh Message Sent to Unicast Address] . . . 124
MESH/NODE/NET/BV-09-C [Receive Mesh Message Sent to Virtual Address] . . . 124
MESH/NODE/NET/BV-10-C [Receive Mesh Message sent to Dynamic Group Address] . . . 124
MESH/NODE/NET/BV-11-C [Receive Mesh Message sent to All-proxies Address] . . . 124
MESH/NODE/NET/BV-12-C [Receive Mesh Message sent to All-friends Address] . . . 124
MESH/NODE/NET/BV-13-C [Receive Mesh Message sent to All-relays Address] . . . 124
MESH/NODE/NET/BV-14-C [Receive Mesh Message sent to All-nodes Address] . . . 124
MESH/NODE/NET/BI-01-C [Ignore Connectable Advertisements on ADV Bearer] . . . 125
MESH/NODE/NET/BI-02-C [Discard Messages with Invalid Addresses] . . . 126
MESH/NODE/NET/BI-03-C [Receive Network Packet with Invalid NID Field] . . . 127
4.9 Message Relaying . . . 128
MESH/NODE/RLY/BV-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit] . . . 128
MESH/NODE/RLY/BV-02-C [Network Message Cache] . . . 132
MESH/NODE/RLY/BI-01-C [Ignore Invalid IV Index] . . . 135
MESH/NODE/RLY/BI-02-C [Ignore Invalid SRC] . . . 136
4.10 Transport Layers . . . 136
MESH/NODE/TNPT/BV-01-C [Ignore Message with Unknown Application Key] . . . 136
MESH/NODE/TNPT/BV-02-C [Ignore Message with Unknown Device Key] . . . 137
MESH/NODE/TNPT/BV-03-C [Send an Unsegmented Access Message] . . . 138
MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast] . . . 138
MESH/NODE/TNPT/BV-05-C [Send a Segmented Access Message – Group/Virtual Address] . . . 140
MESH/NODE/TNPT/BV-06-C [Receive an Unsegmented Access Message] . . . 141
MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] . . . 141
MESH/NODE/TNPT/BV-08-C [Receive a Segmented Message – Group/Virtual Address] . . . 143
MESH/NODE/TNPT/BV-09-C [Send Segmented Message to Unicast– Stop When Transaction Rejected] . . . 144
MESH/NODE/TNPT/BV-10-C [Send Segmented Message - Resend Unacknowledged Segments After Segment

Transmission Timer Expires] . . . 145
MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgment Timer Expiration] . . . 147
MESH/NODE/TNPT/BV-12-C [Receive Segmented Message - Receive New Segment After Incomplete Timer Expires] 148
MESH/NODE/TNPT/BV-13-C [Ignore Replayed Message] . . . 150
MESH/NODE/TNPT/BV-22-C [Receive Segmented Message – Discard on New Transaction] . . . 153
MESH/NODE/TNPT/BI-01-C [Ignore Invalid RFU Bits in Segment Acknowledgment Message] . . . 153
MESH/NODE/TNPT/BI-02-C [Ignore Message When Segments Received with Old SeqAuth] . . . 154
4.10.1 Send Segmented Messages Using the SAR Transmitter State Values - Unicast . . . 155
MESH/NODE/TNPT/BV-14-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, Retransmission after Timer Expiration] . . . 157
MESH/NODE/TNPT/BV-15-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, Retransmission after Acknowledgment] . . . 157
MESH/NODE/TNPT/BV-16-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, TTL set to 0] . . . 157
4.10.2 Receive Segmented Messages Using the SAR Receiver State Values - Unicast . . . 159
MESH/NODE/TNPT/BV-17-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, Complete Receival] . . . 159
MESH/NODE/TNPT/BV-18-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, SeqAuth Validation] . . . 159
MESH/NODE/TNPT/BV-19-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, Discard Timeout] . . . 159
MESH/NODE/TNPT/BV-20-C [Send Segmented Messages Using the SAR Transmitter State Values – Group/Virtual Address] . . . 164
MESH/NODE/TNPT/BV-21-C [Receive Segmented Messages Using the SAR Receiver State Values – Group/Virtual Address] . . . 166
MESH/NODE/TNPT/BI-03-C [Ignore Invalid Segment Acknowledgment Messages] . . . 168
4.11 IV Update . . . 171
4.11.1 IV Update Initiated by Receiving Mesh Beacons . . . 171
MESH/NODE/IVU/BV-01-C [IV Update Initiated by Receiving Secure Network Beacons] . . . 171
MESH/NODE/IVU/BV-06-C [IV Update Initiated by Receiving Mesh Private Beacons] . . . 171
4.11.2 IV Update Initiated by Sending Mesh Beacons . . . 175
MESH/NODE/IVU/BV-02-C [IV Update Initiated by Sending Secure Network Beacons] . . . 177
MESH/NODE/IVU/BV-07-C [IV Update Initiated by Sending Mesh Private Beacons] . . . 177
4.11.3 IV Index Recovery . . . 178
MESH/NODE/IVU/BV-03-C [IV Index Recovery using Secure Network Beacons] . . . 178
MESH/NODE/IVU/BV-08-C [IV Index Recovery using Mesh Private Beacons] . . . 178
4.11.4 IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment . . . 180
MESH/NODE/IVU/BV-04-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment using Secure Network Beacons] . . . 181
MESH/NODE/IVU/BV-09-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment using Mesh Private Beacons] . . . 181
4.11.5 IV Update for a Node Not on Primary Subnet . . . 182
MESH/NODE/IVU/BV-05-C [IV Update for a Node Not on Primary Subnet using Secure Network Beacons] . . . 182
MESH/NODE/IVU/BV-10-C [IV Update for a Node Not on Primary Subnet using Mesh Private Beacons] . . . 182
4.11.6 Ignore IV Update Flag When Operating In Normal Operation For Less Than the Minimum Time . . . 183
MESH/NODE/IVU/BI-01-C [Ignore IV Update Flag from Secure Network Beacons When Operating In Normal Operation For Less Than the Minimum Time] . . . 184
MESH/NODE/IVU/BI-07-C [Ignore IV Update Flag from Mesh Private Beacons When Operating In Normal Operation For Less Than the Minimum Time] . . . 184

4.11.7 Ignore IV Update Flag When Operating In Normal Operation For an Abnormal IV Index . . . 185
MESH/NODE/IVU/BI-02-C [Ignore IV Update Flag in Secure Network Beacon When Operating In Normal Operation For an Abnormal IV Index] . . . 185
MESH/NODE/IVU/BI-08-C [Ignore IV Update Flag in Mesh Private Beacon When Operating In Normal Operation For an Abnormal IV Index] . . . 185
4.11.8 Ignore IV Update Flag When Operating In Normal Operation For Message From Other Subnets . . . 187
MESH/NODE/IVU/BI-04-C [Ignore IV Update Flag in Secure Network Beacons When Operating In Normal Operation For Message From Other Subnets] . . . 187
MESH/NODE/IVU/BI-09-C [Ignore IV Update Flag in Mesh Private Beacons When Operating In Normal Operation For Message From Other Subnets] . . . 187
4.11.9 Ignore IV Update Flag When in Normal Operation within 96 Hours of Exiting IV Update In Progress State . . . 188
MESH/NODE/IVU/BI-05-C [Ignore IV Update Flag in Secure Network Beacons When in Normal Operation State within 96 Hours of Exiting IV Update In Progress State] . . . 188
MESH/NODE/IVU/BI-10-C [Ignore IV Update Flag in Mesh Private Beacons When in Normal Operation within 96 Hours of Exiting IV Update In Progress State] . . . 188
4.11.10 IV Update - Invalid IV Index Value . . . 190
MESH/NODE/IVU/BI-06-C [IV Update – Invalid IV Index value in Secure Network beacons] . . . 191
MESH/NODE/IVU/BI-11-C [IV Update – Invalid IV Index value in Mesh Private beacons] . . . 191
4.12 Key Refresh . . . 192
MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via Configuration Client Messages] . . . 192
4.12.1 Key Refresh Procedure as Node via Mesh Beacons . . . 195
MESH/NODE/KR/BV-02-C [Key Refresh Procedure as Node via Secure Network Beacons] . . . 197
MESH/NODE/KR/BV-04-C [Key Refresh Procedure as Node via Mesh Private Beacons] . . . 197
4.12.2 Key Refresh Procedure Skipping Phase 2 . . . 198
MESH/NODE/KR/BV-03-C [Key Refresh Procedure Skipping Phase 2 via Secure Network Beacons] . . . 198
MESH/NODE/KR/BV-05-C [Key Refresh Procedure Skipping Phase 2 via Mesh Private Beacons] . . . 198
MESH/CFGCL/KR/BV-01-C [Key Refresh as Configuration Client via Configuration Messages] . . . 200
4.12.3 Key Refresh as Configuration Client via Mesh Beacons . . . 203
MESH/CFGCL/KR/BV-02-C [Key Refresh as Configuration Client via Secure Network Beacons] . . . 205
MESH/CFGCL/KR/BV-05-C [Key Refresh as Configuration Client via Mesh Private Beacons] . . . 205
MESH/CFGCL/KR/BV-03-C [Key Refresh with Reject Listing via Configuration Messages] . . . 205
4.12.4 Key Refresh with Reject Listing via Mesh Beacons . . . 207
MESH/CFGCL/KR/BV-04-C [Key Refresh with Reject Listing via Secure Network Beacons] . . . 208
MESH/CFGCL/KR/BV-06-C [Key Refresh with Reject Listing via Mesh Private Beacons] . . . 208
MESH/NODE/KR/BI-01-C [Receive Config Key Refresh Phase Set Messages with Valid and Invalid Transition Values] 208
MESH/NODE/KR/BI-02-C [Ignore New Keys During Key Refresh Procedure] . . . 212
MESH/NODE/KR/BI-03-C [Receive Config Key Refresh Phase Get with Invalid NetKeyIndex] . . . 215
MESH/NODE/KR/BI-04-C [Receive Messages with Old Keys During Configuration Client Key Refresh] . . . 215
4.12.5 Receive Messages with Old Keys During Mesh Beacon Key Refresh . . . 217
MESH/NODE/KR/BI-05-C [Receive Messages with Old Keys During Secure Network Beacon Key Refresh] . . . 218
MESH/NODE/KR/BI-06-C [Receive Messages with Old Keys During Mesh Private Beacon Key Refresh] . . . 218
4.13 Friendship . . . 218
4.13.1 Friend Node . . . 219
MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] . . . 219
MESH/NODE/FRND/FN/BV-02-C [Friendship Messaging – Friend Node] . . . 220
MESH/NODE/FRND/FN/BV-03-C [Segment Acknowledgment – Friend Node] . . . 223
MESH/NODE/FRND/FN/BV-04-C [Friend IV Update due to Secure Network Beacon – Friend Node] . . . 225
MESH/NODE/FRND/FN/BV-24-C [Friend IV Update due to Mesh Private Beacon – Friend Node] . . . 225
MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message] . . . 227

MESH/NODE/FRND/FN/BV-06-C [Friendship – Respond to Friend Subscription List Remove Message] . . . 230
MESH/NODE/FRND/FN/BV-07-C [Friendship – Friend Node Friendship Not Established on Missed Friend Poll] . . . 231
MESH/NODE/FRND/FN/BV-08-C [Friendship – Friend Node Circular Queue] . . . 233
MESH/NODE/FRND/FN/BV-09-C [Friendship – Friend Node Cached Cleared on Poll Timeout] . . . 235
MESH/NODE/FRND/FN/BV-10-C [Friendship – Friend Node Config Friend Set Received] . . . 236
MESH/NODE/FRND/FN/BV-11-C [Friendship – Friend Node Friend Clear Send and Retry] . . . 238
MESH/NODE/FRND/FN/BV-12-C [Friend Clear and Friend Request Friendship Termination Scenarios] . . . 240
MESH/NODE/FRND/FN/BV-13-C [Friendship – Friend Node – Friend Offer Delay] . . . 242
MESH/NODE/FRND/FN/BV-14-C [Friend Key Refresh via Secure Network Beacons – Friend Node] . . . 245
MESH/NODE/FRND/FN/BV-25-C [Friend Key Refresh via Mesh Private Beacons – Friend Node] . . . 245
MESH/NODE/FRND/FN/BV-15-C [Friend Key Refresh via Configuration Client Messages – Friend Node] . . . 246
MESH/NODE/FRND/FN/BV-16-C [Friendship – Friend Node - FSN Field Not Changed] . . . 248
MESH/NODE/FRND/FN/BV-17-C [Friendship – Friend Node – SAR Interruption] . . . 251
MESH/NODE/FRND/FN/BV-18-C [Message Caching based on TTL and SRC – Friend Node] . . . 253
MESH/NODE/FRND/FN/BV-19-C [Segment Caching – Friend Node] . . . 255
MESH/NODE/FRND/FN/BV-20-C [Adding another Friend Update upon reception of Secure Network Beacons – Friend Node] . . . 257
MESH/NODE/FRND/FN/BV-26-C [Adding another Friend Update upon reception of Mesh Private Beacons – Friend Node] . . . 257
MESH/NODE/FRND/FN/BV-21-C [Friend Node Respond to Config Low Power Node PollTimeout Get] . . . 259
MESH/NODE/FRND/FN/BV-22-C [Filter Duplicates in the Friend Queue] . . . 260
MESH/NODE/FRND/FN/BV-23-C [Retransmit Message from LPN] . . . 262
MESH/NODE/FRND/FN/BI-01-C [Friendship – Friend Node – Invalid Transport Control OpCode] . . . 263
MESH/NODE/FRND/FN/BI-02-C [Friendship – Friend Node – Friend Poll Invalid Prohibited Bits] . . . 265
MESH/NODE/FRND/FN/BI-03-C [Friendship – Friend Node – Friend Request Invalid RFU Bits] . . . 266
4.13.2 Low Power Node . . . 269
MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node] . . . 269
MESH/NODE/FRND/LPN/BV-02-C [Friendship Messaging – Low Power Node] . . . 271
MESH/NODE/FRND/LPN/BV-03-C [Friend IV Update – Low Power Node] . . . 274
MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message] . . . 276
MESH/NODE/FRND/LPN/BV-05-C [Friendship – Send Friend Subscription List Remove Message] . . . 277
MESH/NODE/FRND/LPN/BV-06-C [Friendship Updates Failed Friend Update Message – Low Power Node] . . . 278
MESH/NODE/FRND/LPN/BV-07-C [Friend Key Refresh – Low Power Node] . . . 280
MESH/NODE/FRND/LPN/BV-08-C [Low Power Node Send Friend Clear Message] . . . 282
MESH/NODE/FRND/LPN/BI-01-C [Friendship Establishment Mode – Low Power Node Friend Poll Retries] . . . 283
MESH/NODE/FRND/LPN/BI-02-C [Invalid Transport Control OpCode – Low Power Node] . . . 285
MESH/NODE/FRND/LPN/BI-03-C [Friendship Low Power Node – Friend Offer Invalid RFU Bits] . . . 286
4.14 Proxy Protocol . . . 288
MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup] . . . 288
4.14.1 Complete Message Network PDU Relay to Proxy Client . . . 289
MESH/SR/PROX/BV-02-C [Complete Message Network PDU Relay to Proxy Client – GATT Proxy Enabled] . . . 290
MESH/SR/PROX/BV-16-C [Complete Message Network PDU Relay to Proxy Client – Private GATT Proxy Enabled] . . . 290
MESH/SR/PROX/BV-03-C [Segmented Message Network PDU Relay to Proxy Client] . . . 290
4.14.2 Send a Complete Message Network PDU to Node from Proxy Client . . . 292
MESH/SR/PROX/BV-04-C [Send a Complete Message Network PDU to Node from Proxy Client – GATT Proxy Enabled] . . . 292
MESH/SR/PROX/BV-17-C [Send a Complete Message Network PDU to Node from Proxy Client – Private GATT Proxy Enabled] . . . 292
MESH/SR/PROX/BV-05-C [Send a Segmented Message Network PDU to Node from Proxy Client] . . . 293

MESH/SR/PROX/BV-06-C [Filter Accept List Addresses] . . . 295
MESH/SR/PROX/BV-07-C [Receive Reject List Filter Setup] . . . 296
MESH/SR/PROX/BV-08-C [Filter Reject List Addresses] . . . 299
MESH/SR/PROX/BV-09-C [Remove Address from Accept List] . . . 300
MESH/SR/PROX/BV-10-C [Remove Address from Reject List] . . . 301
4.14.3 Send Mesh Beacon When Proxy Connection Established . . . 302
MESH/SR/PROX/BV-11-C [Send Secure Network Beacon When Proxy Connection Established] . . . 302
MESH/SR/PROX/BV-18-C [Send Mesh Private Beacon When Proxy Connection Established – using Private Network Identity] . . . 302
MESH/SR/PROX/BV-19-C [Send Mesh Private Beacon When Proxy Connection Established – using Private Node Identity] . . . 302
4.14.4 Send Mesh Beacon When IV Index Updated . . . 303
MESH/SR/PROX/BV-12-C [Send Secure Network Beacon When IV Index Updated] . . . 303
MESH/SR/PROX/BV-20-C [Send Mesh Private Beacon When IV Index Updated] . . . 303
4.14.5 Send Mesh Beacon with Previous IV Index Value . . . 305
MESH/SR/PROX/BV-13-C [Send Secure Network Beacon with Previous IV Index Value] . . . 305
MESH/SR/PROX/BV-21-C [Send Mesh Private Beacon with Previous IV Index Value] . . . 305
4.14.6 IV Update Initiated by Proxy Client . . . 308
MESH/SR/PROX/BV-14-C [IV Update Initiated by Proxy Client using Secure Network Beacons] . . . 308
MESH/SR/PROX/BV-22-C [IV Update Initiated by Proxy Client using Mesh Private Beacons] . . . 308
MESH/SR/PROX/BV-15-C [Disable Network PDU Relay when Proxy Feature is Disabled] . . . 309
4.14.7 Send Mesh Beacon when Secondary NetKey Added . . . 311
MESH/SR/PROX/BV-23-C [Send Secure Network Beacon when Secondary NetKey Added] . . . 312
MESH/SR/PROX/BV-24-C [Send Mesh Private Beacon when Secondary NetKey Added] . . . 312
MESH/SR/PROX/BI-01-C [Ignore Invalid Message Type] . . . 312
MESH/CL/PROX/BV-01-C [Receive Complete Message from Proxy Server] . . . 313
MESH/CL/PROX/BV-02-C [Receive Segmented Message from Proxy Server] . . . 314
MESH/CL/PROX/BV-03-C [Send Complete Message to Proxy Server] . . . 315
MESH/CL/PROX/BV-04-C [Send Segmented Message to Proxy Server] . . . 315
MESH/CL/PROX/BV-05-C [Send Set Filter Type Proxy Configuration Message] . . . 317
MESH/CL/PROX/BV-06-C [Send Add Addresses to Filter Proxy Configuration Message] . . . 318
MESH/CL/PROX/BV-07-C [Send Remove Addresses from Filter Proxy Configuration Message] . . . 319
4.14.8 Receive Mesh Beacon When Proxy Connection Established . . . 320
MESH/CL/PROX/BV-08-C [Receive Secure Network Beacon When Proxy Connection Established] . . . 320
MESH/CL/PROX/BV-11-C [Receive Mesh Private Beacon When Proxy Connection Established] . . . 320
4.14.9 Receive Mesh Beacon When IV Index Updated . . . 320
MESH/CL/PROX/BV-09-C [Receive Secure Network Beacon When IV Index Updated] . . . 321
MESH/CL/PROX/BV-12-C [Receive Mesh Private Beacon When IV Index Updated] . . . 321
4.14.10 Receive Mesh Beacon with Previous IV Index Value . . . 322
MESH/CL/PROX/BV-10-C [Receive Secure Network Beacon with Previous IV Index Value] . . . 323
MESH/CL/PROX/BV-13-C [Receive Mesh Private Beacon with Previous IV Index Value] . . . 323
MESH/CL/PROX/BI-01-C [Ignore Invalid Message Type] . . . 323
4.14.11 Ignore RFU Proxy configuration message . . . 324
MESH/SR/PROX/BI-02-C [Ignore RFU Proxy configuration message] . . . 324
MESH/CL/PROX/BI-02-C [Ignore RFU Proxy configuration message] . . . 324
4.15 Mesh Proxy Service . . . 325
MESH/SR/MPXS/BV-01-C [Advertise Mesh Proxy Service with Network ID] . . . 325
MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service] . . . 325
MESH/SR/MPXS/BV-03-C [Expose Mesh Proxy Data Out Characteristic] . . . 326

MESH/SR/MPXS/BV-04-C [Expose Mesh Proxy Data Out Client Characteristic Configuration Descriptor] . . . 327
MESH/SR/MPXS/BV-05-C [Support Notifications for Mesh Proxy Data Out Characteristic] . . . 328
MESH/SR/MPXS/BV-06-C [Expose Mesh Proxy Data In Characteristic] . . . 328
MESH/SR/MPXS/BV-07-C [Support Write Without Response to Mesh Proxy Data In Characteristic] . . . 329
MESH/SR/MPXS/BV-09-C [Advertise Mesh Proxy Service with Node Identity] . . . 330
MESH/SR/MPXS/BV-10-C [Advertise Mesh Proxy Service with Private Network Identity] . . . 331
MESH/SR/MPXS/BV-11-C [Advertise Mesh Proxy Service with Private Node Identity] . . . 332
MESH/SR/MPXS/BV-12-C [Respond to Private GATT Proxy Solicitation] . . . 332
MESH/SR/MPXS/BV-13-C [Private GATT Proxy Solicitation with RFU Identification Type Value] . . . 335
MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] . . . 336
MESH/CL/MPXS/BV-02-C [Discover Mesh Proxy Service] . . . 337
MESH/CL/MPXS/BV-03-C [Discover Mesh Proxy Data Out Characteristic] . . . 338
MESH/CL/MPXS/BV-04-C [Discover Mesh Proxy Data Out Client Characteristic Configuration Descriptor and Enable and Process Notifications] . . . 338
MESH/CL/MPXS/BV-05-C [Discover Mesh Proxy Data In Characteristic] . . . 339
MESH/CL/MPXS/BV-06-C [Execute Write Without Response to Mesh Proxy Data In Characteristic] . . . 340
MESH/CL/MPXS/BV-07-C [Connect to GATT Proxy Server – Server uses Private Network Identity] . . . 341
MESH/CL/MPXS/BV-08-C [Connect to GATT Proxy Server – Server uses Private Node Identity] . . . 342
MESH/CL/MPXS/BV-09-C [Send GATT Proxy Solicitation] . . . 343
4.16 Configuration Model . . . 344
4.16.1 Secure Network Beacon Procedures . . . 344
MESH/NODE/CFG/SNBP/BV-01-C [Respond to Config Beacon Set/Get Messages] . . . 344
MESH/NODE/CFG/SNBP/BI-01-C [Ignore invalid Config Beacon Set/Get Messages] . . . 345
MESH/CFGCL/CFG/SNBP/BV-01-C [Send Config Beacon Set/Get Messages] . . . 347
4.16.2 Composition Data Procedures . . . 348
MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] . . . 348
MESH/CFGCL/CFG/COMP/BV-01-C [Send Composition Data Get Messages] . . . 351
4.16.3 Default TTL Procedure . . . 352
MESH/NODE/CFG/DTTL/BV-01-C [Respond to Config Default TTL Set/Get Messages] . . . 352
MESH/NODE/CFG/DTTL/BI-01-C [Respond to Config Default TTL Set Message (On Secondary Element or With Invalid TTL)] . . . 353
MESH/CFGCL/CFG/DTTL/BV-01-C [Send Config Default TTL Set/Get Messages] . . . 355
4.16.4 GATT Proxy Procedures . . . 357
MESH/NODE/CFG/GPXY/BV-01-C [Respond to Config GATT Proxy Set/Get Messages] . . . 357
MESH/NODE/CFG/GPXY/BV-02-C [Network ID Advertising on Multiple Subnets] . . . 359
MESH/NODE/CFG/GPXY/BI-01-C [Respond to Config GATT Proxy Set Messages Invalid Parameters] . . . 360
MESH/CFGCL/CFG/GPXY/BV-01-C [Send Config GATT Proxy Set/Get Messages] . . . 361
4.16.5 Friend Procedures . . . 362
MESH/NODE/CFG/CFGF/BV-01-C [Respond to Config Friend Get/Set Messages] . . . 362
MESH/NODE/CFG/CFGF/BI-01-C [Respond to Config Friend Set Messages Invalid Parameters] . . . 365
MESH/CFGCL/CFG/CFGF/BV-01-C [Send Config Friend Get/Set Messages] . . . 366
4.16.6 Relay Procedures . . . 367
MESH/NODE/CFG/CFGR/BV-01-C [Respond to Config Relay Get/Set Messages] . . . 367
MESH/NODE/CFG/CFGR/BI-01-C [Respond to Config Relay Set Messages Invalid Parameters] . . . 370
MESH/CFGCL/CFG/CFGR/BV-01-C [Send Config Relay Get/Set Messages] . . . 371
4.16.7 Model Publication Procedures . . . 372
MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] . . . 372
MESH/NODE/CFG/MP/BV-02-C [Disabling Publishing – AppKey/Config NetKey Deleted] . . . 376

MESH/NODE/CFG/MP/BI-01-C [Respond to Config Model Publication Get/Set/Virtual Address Set Messages (Invalid ElementAddress)] . . . 379
MESH/NODE/CFG/MP/BI-02-C [Respond to Config Model Publication Get/Set/Virtual Address Set Message (Invalid Model Identifier)] . . . 380
MESH/NODE/CFG/MP/BI-03-C [Respond to Config Model Publication Set/Virtual Address Set Messages (Invalid AppKeyIndex)] . . . 382
MESH/CFGCL/CFG/MP/BV-01-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] . . . 383
MESH/NODE/CFG/MP/BV-03-C [No Publishing when PublishTTL is set to 0x01] . . . 386
4.16.8 Subscription List Procedures . . . 388
MESH/NODE/CFG/SL/BV-01-C [Respond to Config Model Subscription Messages] . . . 388
MESH/NODE/CFG/SL/BV-02-C [Respond to Config Model Subscription Virtual Address Messages] . . . 391
MESH/NODE/CFG/SL/BV-03-C [Respond to Config Vendor Model Subscription Get Message] . . . 394
MESH/NODE/CFG/SL/BI-01-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid ElementAddress)] . . . 396
MESH/NODE/CFG/SL/BI-02-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid Model Identifier)] . . . 397
MESH/NODE/CFG/SL/BI-03-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid ElementAddress)] . . . 399
MESH/NODE/CFG/SL/BI-04-C [Respond to Config SIG Model Subscription Get Message (Invalid ElementAddress)] . . . 400
MESH/NODE/CFG/SL/BI-05-C [Respond to Config SIG Model Subscription Get Message (Invalid Model Identifier)] . . . 401
MESH/NODE/CFG/SL/BI-06-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid Model Identifier)] . . . 402
MESH/NODE/CFG/SL/BI-07-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid ElementAddress)] . . . 404
MESH/NODE/CFG/SL/BI-08-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid Model Identifier)] . . . 405
MESH/NODE/CFG/SL/BI-09-C [Respond to Config Vendor Model Subscription Get Message (Invalid ElementAddress)] . . . 407
MESH/NODE/CFG/SL/BI-10-C [Respond to Config Vendor Model Subscription Get Message (Invalid Model Identifier)] 408
MESH/CFGCL/CFG/SL/BV-01-C [Send Config Model Subscription Messages] . . . 409
MESH/CFGCL/CFG/SL/BV-02-C [Send Config Model Subscription Virtual Address Messages] . . . 412
4.16.9 NetKey List Procedures . . . 414
MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] . . . 414
MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message] . . . 416
MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message] . . . 417
MESH/NODE/CFG/NKL/BV-04-C [Receive Config NetKey Delete Message for Last NetKey] . . . 418
MESH/NODE/CFG/NKL/BI-01-C [Respond to Config NetKey Update Message (Invalid NetKeyIndex)] . . . 419
MESH/NODE/CFG/NKL/BI-02-C [Respond to Config NetKey Add Message (NetKeyIndex Already Stored)] . . . 420
MESH/NODE/CFG/NKL/BI-03-C [Respond to Config NetKey Add Message (Insufficient Resources)] . . . 422
MESH/CFGCL/CFG/NKL/BV-01-C [Send Config NetKey Add/Get Messages] . . . 423
MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages] . . . 424
MESH/CFGCL/CFG/NKL/BV-03-C [Send Config NetKey Delete Messages] . . . 425
4.16.10 AppKey List Procedures . . . 426
MESH/NODE/CFG/AKL/BV-01-C [Respond to Config AppKey Add/Get Message] . . . 426
MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message] . . . 428
MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message] . . . 430
MESH/NODE/CFG/AKL/BI-01-C [Respond to Config AppKey Add Message (Invalid NetKey Index)] . . . 431
MESH/NODE/CFG/AKL/BI-02-C [Respond to Config AppKey Update Message (Invalid AppKey Index and NetKey

Index)] . . . 432
MESH/NODE/CFG/AKL/BI-03-C [Respond to Config AppKey Get Message with Invalid NetKey Index] . . . 435
MESH/NODE/CFG/AKL/BI-04-C [Respond to Config AppKey Add Message (NetKeyIndex and AppKeyIndex Already Stored)] . . . 436
MESH/CFGCL/CFG/AKL/BV-01-C [Send Config AppKey Add Messages] . . . 437
MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages] . . . 438
MESH/CFGCL/CFG/AKL/BV-03-C [Send Config AppKey Delete Messages] . . . 439
MESH/CFGCL/CFG/AKL/BV-04-C [Send Config AppKey Get Messages] . . . 440
4.16.11 Model to AppKey List Procedures . . . 440
MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages] . . . 440
MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages] . . . 442
MESH/NODE/CFG/MAKL/BI-01-C [Respond to Config Model App Bind Message with Invalid Values] . . . 444
MESH/NODE/CFG/MAKL/BI-02-C [Respond to Config Model App Unbind Message with Invalid Values] . . . 447
MESH/NODE/CFG/MAKL/BI-03-C [Respond to Config SIG Model App Get Message with Invalid Values] . . . 448
MESH/NODE/CFG/MAKL/BI-04-C [Respond to Config Vendor Model App Get Message with Invalid Values] . . . 450
MESH/CFGCL/CFG/MAKL/BV-01-C [Send Config Model App Bind Messages] . . . 451
MESH/CFGCL/CFG/MAKL/BV-02-C [Send Config Model App Unbind Messages] . . . 452
MESH/CFGCL/CFG/MAKL/BV-03-C [Send Config SIG Model App Get Messages] . . . 453
MESH/CFGCL/CFG/MAKL/BV-04-C [Send Config Vendor Model App Get Messages] . . . 454
4.16.12 Node Identity Procedures . . . 455
MESH/NODE/CFG/NID/BV-01-C [Receive Config Node Identity Get/Set Messages] . . . 455
MESH/NODE/CFG/NID/BV-02-C [Node Identity Advertising on Multiple Subnets] . . . 457
MESH/NODE/CFG/NID/BV-03-C [Node Identity Config NetKey Deleted] . . . 459
MESH/NODE/CFG/NID/BI-01-C [Receive Config Node Identity Get/Set Messages – Invalid Parameter] . . . 461
MESH/NODE/CFG/NID/BI-03-C [Receive Config Node Identity Set Message – Change to Not Supported] . . . 463
MESH/CFGCL/CFG/NID/BV-01-C [Send Config Node Identity Get/Set Messages] . . . 465
4.16.13 Reset Procedures . . . 466
MESH/NODE/CFG/RST/BV-01-C [Receive Config Node Reset Message] . . . 467
MESH/CFGCL/CFG/RST/BV-01-C [Send Config Node Reset Messages] . . . 468
4.16.14 Heartbeat Publication Procedures . . . 468
MESH/NODE/CFG/HBP/BV-01-C [Receive Config Heartbeat Publication Set/Get messages] . . . 468
MESH/NODE/CFG/HBP/BV-02-C [Periodic publishing of Heartbeat messages] . . . 470
MESH/NODE/CFG/HBP/BV-03-C [Triggered publishing of Heartbeat messages – Relay State changes] . . . 473
MESH/NODE/CFG/HBP/BV-04-C [Triggered publishing of Heartbeat messages - Friend state changes] . . . 476
MESH/NODE/CFG/HBP/BV-05-C [Triggered publishing of Heartbeat messages - Low Power State changes] . . . 479
MESH/NODE/CFG/HBP/BV-06-C [Disabling publishing of Heartbeat messages – Config NetKey Deleted] . . . 482
MESH/NODE/CFG/HBP/BV-07-C [Triggered publishing of Heartbeat messages – Proxy state changes] . . . 485
MESH/NODE/CFG/HBP/BI-01-C [Respond to Config Heartbeat Publication Set – Invalid Publication Features RFU Values] . . . 488
MESH/NODE/CFG/HBP/BI-02-C [Respond to Config Heartbeat Publication Get – Wrongly Encrypted by AppKey] . . . 491
MESH/CFGCL/CFG/HBP/BV-01-C [Send Config Heartbeat Publication Set/Get Messages] . . . 492
4.16.15 Heartbeat Subscription Procedures . . . 494
MESH/NODE/CFG/HBS/BV-01-C [Receive Config Heartbeat Subscription Set/Get message] . . . 494
MESH/NODE/CFG/HBS/BV-02-C [Receive Heartbeat messages – Controlled by PeriodLog State] . . . 497
MESH/NODE/CFG/HBS/BV-03-C [Not receiving Heartbeat messages – Unassigned or Mismatched SRC field] . . . 500
MESH/NODE/CFG/HBS/BV-04-C [Receive Heartbeat messages – CountLog and Hops] . . . 503
MESH/NODE/CFG/HBS/BV-05-C [Receive Heartbeat messages – Friendship] . . . 506

MESH/NODE/CFG/HBS/BI-01-C [Ignore Config Heartbeat Subscription Set Message –Invalid Source Address or Destination Address] . . . 508
MESH/CFGCL/CFG/HBS/BV-01-C [Send Config Heartbeat Subscription Set/Get Messages] . . . 511
4.16.16 Network Transmit Procedures . . . 513
MESH/NODE/CFG/NTX/BV-01-C [Respond to Config Network Transmit Get/Set Messages] . . . 513
MESH/CFGCL/CFG/NTX/BV-01-C [Send Config Network Transmit Get/Set Messages] . . . 514
4.16.17 Low Power Node PollTimeout Procedures . . . 514
MESH/NODE/CFG/LPNPT/BI-01-C [Ignore Invalid Config Low Power Node PollTimeout Get Messages] . . . 515
MESH/CFGCL/CFG/LPNPT/BV-01-C [Send Config Low Power Node PollTimeout Get Messages] . . . 515
4.17 Health Model . . . 516
4.17.1 Current Fault State . . . 516
MESH/SR/HM/CFS/BV-01-C [Send Health Current Status Messages - No Faults] . . . 516
MESH/SR/HM/CFS/BV-02-C [Send Health Current Status Messages - With Faults] . . . 518
4.17.2 Registered Fault State . . . 519
MESH/SR/HM/RFS/BV-01-C [Respond to Health Fault Get] . . . 519
MESH/SR/HM/RFS/BV-02-C [Respond to Health Fault Clear and Health Fault Clear Unacknowledged Messages] . . . 520
MESH/SR/HM/RFS/BV-03-C [Respond to Health Fault Test and Health Fault Test Unacknowledged Messages] . . . 522
MESH/SR/HM/RFS/BI-01-C [Ignore Invalid Health Fault Test message] . . . 523
MESH/SR/HM/RFS/BI-02-C [Ignore Invalid Health Fault Test Unacknowledged message] . . . 524
MESH/CL/HM/RFS/BV-01-C [Send Health Messages] . . . 526
4.17.3 Health Period State . . . 528
MESH/SR/HM/HPS/BV-01-C [Respond to Health Period Messages] . . . 528
MESH/CL/HM/HPS/BV-01-C [Send Health Period Messages] . . . 529
4.17.4 Attention Timer State . . . 531
MESH/SR/HM/ATS/BV-01-C [Respond to Attention Messages] . . . 531
MESH/CL/HM/ATS/BV-01-C [Send Attention Messages] . . . 532
4.18 Directed Forwarding . . . 534
4.18.1 Verify Forwarding Table on the primary subnet . . . 534
4.18.2 Directed Forwarding Initialization . . . 538
MESH/NODE/DF/INIT/BV-01-C [Send PATH_REQUEST, IUT as Path Origin] . . . 538
MESH/NODE/DF/INIT/BV-02-C [Send PATH_REQUEST, IUT as Directed Friend Path Origin] . . . 539
MESH/NODE/DF/INIT/BV-03-C [Send PATH_REQUEST, IUT as Path Origin for Proxy Client] . . . 539
MESH/NODE/DF/INIT/BV-04-C [Send PATH_REQUEST, IUT as Path Origin for Directed Proxy Client] . . . 539
MESH/NODE/DF/INIT/BV-11-C [Send PATH_REQUEST, IUT as Subnet Bridge] . . . 539
MESH/NODE/DF/INIT/BV-05-C [Multiple Lanes, IUT as Path Origin] . . . 541
MESH/NODE/DF/INIT/BV-06-C [Path Discovery Retry Interval Timeout, IUT as Path Origin] . . . 543
MESH/NODE/DF/INIT/BV-07-C [Path Monitoring Interval Timeout on Path Expiry, IUT as Path Origin] . . . 545
MESH/NODE/DF/INIT/BV-08-C [Path Monitoring Interval Timeout on Power-Up, IUT as Path Origin] . . . 547
MESH/NODE/DF/INIT/BV-09-C [Transmit Access Message for Existing Non-fixed Path Without PATH_REQUEST] . . . 549
MESH/NODE/DF/INIT/BV-10-C [Multiple Lanes, PATH_REQUEST Uses Discovery Table Values] . . . 550
4.18.3 Directed Forwarding Discovery . . . 552
MESH/NODE/DF/DISC/BV-01-C [Receive PATH_REQUEST, IUT not Path Target] . . . 552
MESH/NODE/DF/DISC/BV-02-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target] . . . 554
MESH/NODE/DF/DISC/BV-03-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Path Target] . . . 555
MESH/NODE/DF/DISC/BV-04-C [Receive PATH_REQUEST - Unicast Address, IUT as Friend Path Target] . . . 561
MESH/NODE/DF/DISC/BV-05-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target for Proxy Client] . . . 561
MESH/NODE/DF/DISC/BV-06-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target for Directed Proxy Client] . . . 561
MESH/NODE/DF/DISC/BV-12-C [Receive PATH_REQUEST - Unicast Address, IUT as Subnet Bridge] . . . 561

MESH/NODE/DF/DISC/BV-07-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Friend Path Target] 564
MESH/NODE/DF/DISC/BV-08-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Proxy Path Target] 564
MESH/NODE/DF/DISC/BV-13-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Subnet Bridge] 564
MESH/NODE/DF/DISC/BV-09-C [Ignore PATH_REQUEST – Directed Relay Disabled] 566
MESH/NODE/DF/DISC/BV-10-C [Ignore PATH_REQUEST – Directed Friend Enabled] 566
MESH/NODE/DF/DISC/BV-11-C [Ignore PATH_REQUEST – Directed Proxy Enabled] 566
MESH/NODE/DF/DISC/BV-14-C [Ignore PATH_REQUEST, Lower Forwarding Number] 566
MESH/NODE/DF/DISC/BV-15-C [Ignore PATH_REQUEST, Neighbor with Low RSSI] 567
MESH/NODE/DF/DISC/BV-16-C [Ignore PATH_REQUEST, Wrong Path Metric] 568
MESH/NODE/DF/DISC/BV-17-C [Multiple Lanes, IUT as Path Target] 569
MESH/NODE/DF/DISC/BI-01-C [Ignore PATH_REQUEST – Invalid Parameters] 571
4.18.4 Directed Forwarding Establishment 573
MESH/NODE/DF/EST/BV-01-C [Receive PATH_REPLY, IUT as Directed Relay] 573
MESH/NODE/DF/EST/BV-02-C [Ignore PATH_REPLY, Empty Discovery Table] 575
MESH/NODE/DF/EST/BV-03-C [Receive PATH_REPLY - Unicast Address, IUT as Path Origin] 576
MESH/NODE/DF/EST/BV-04-C [Receive PATH_REPLY, IUT as Friend Path Origin] 580
MESH/NODE/DF/EST/BV-05-C [Receive PATH_REPLY, IUT as Proxy Path Origin] 580
MESH/NODE/DF/EST/BV-09-C [Receive PATH_REPLY, IUT as Subnet Bridge] 580
MESH/NODE/DF/EST/BV-06-C [Receive PATH_REPLY - Group/Virtual Address, IUT as Directed Relay] 581
MESH/NODE/DF/EST/BV-07-C [Receive PATH_REPLY – Group/Virtual Address, IUT as Path Origin] 583
MESH/NODE/DF/EST/BV-08-C [Multiple Lanes, IUT as Directed Relay] 586
MESH/NODE/DF/EST/BI-01-C [Ignore PATH_REPLY – Invalid Parameters] 587
4.18.5 Directed Forwarding Confirmation 589
MESH/NODE/DF/CONF/BV-01-C [Send PATH_CONFIRMATION, IUT as Path Origin] 589
MESH/NODE/DF/CONF/BV-02-C [Receive PATH_CONFIRMATION, IUT as Directed Relay] 591
MESH/NODE/DF/CONF/BV-03-C [Receive PATH_CONFIRMATION, IUT as Path Target] 592
MESH/NODE/DF/CONF/BV-04-C [Ignore PATH_CONFIRMATION, IUT as Directed Relay] 593
MESH/NODE/DF/CONF/BI-01-C [Ignore PATH_CONFIRMATION – Invalid Parameters] 594
4.18.6 Directed Forwarding Update 596
MESH/NODE/DF/UPDATE/BV-01-C [Node Dependence, IUT as Directed Friend] 598
MESH/NODE/DF/UPDATE/BV-02-C [Node Dependence with Proxy Client, IUT as Directed Proxy] 598
MESH/NODE/DF/UPDATE/BV-03-C [Node Dependence with Directed Proxy Client, IUT as Directed Proxy] 598
MESH/NODE/DF/UPDATE/BV-14-C [Node Dependence, IUT as Subnet Bridge] 598
MESH/NODE/DF/UPDATE/BV-04-C [Remove Node Dependence, IUT as Directed Friend] 599
MESH/NODE/DF/UPDATE/BV-05-C [Remove Node Dependence with Proxy Client, IUT as Directed Proxy] 599
MESH/NODE/DF/UPDATE/BV-06-C [Remove Node Dependence with Directed Proxy Client, IUT as Directed Proxy] . 599
MESH/NODE/DF/UPDATE/BV-07-C [Remove Node Dependence with Directed Proxy Client, IUT as Directed Proxy] . 599
MESH/NODE/DF/UPDATE/BV-08-C [Remove Node Dependence with disconnected Directed Proxy Client, IUT as Directed Proxy] 599
MESH/NODE/DF/UPDATE/BV-09-C [Remove Node Dependence with Directed Proxy Client on filter type change, IUT as Directed Proxy] 599
MESH/NODE/DF/UPDATE/BV-10-C [Remove Node Dependence with disconnected Proxy Client, IUT as Directed Proxy] 599
MESH/NODE/DF/UPDATE/BV-11-C [Remove Node Dependence with Proxy Client on filter type change, IUT as Directed Proxy] 599
MESH/NODE/DF/UPDATE/BV-15-C [Remove Node Dependence, IUT as Subnet Bridge] 599
MESH/NODE/DF/UPDATE/BV-12-C [Receive DEPENDENT_NODE_UPDATE, IUT as Directed Relay] 605
MESH/NODE/DF/UPDATE/BV-13-C [Ignore DEPENDENT_NODE_UPDATE] 608
MESH/NODE/DF/UPDATE/BI-01-C [Receive DEPENDENT_NODE_UPDATE, Invalid Parameters] 609

4.18.7 Directed Forwarding Echo . . . 611
MESH/NODE/DF/ECHO/BV-01-C [Path Echo, IUT as Path Origin] . . . 611
MESH/NODE/DF/ECHO/BV-02-C [Path Echo Failure, IUT as Path Origin] . . . 613
MESH/NODE/DF/ECHO/BV-03-C [Path Echo, IUT as Path Target] . . . 615
MESH/NODE/DF/ECHO/BV-04-C [Two-Way Path Echo, IUT as Path Target] . . . 615
MESH/NODE/DF/ECHO/BV-05-C [Ignore Path Echo Messages] . . . 616
MESH/NODE/DF/ECHO/BV-06-C [Path Echo for Group/Virtual Destination, IUT as Path Origin] . . . 617
MESH/NODE/DF/ECHO/BI-01-C [Ignore Path Echo Messages – Invalid Parameters] . . . 619
MESH/NODE/DF/ECHO/BI-02-C [Ignore Path Echo Reply – One-way Path – Incorrect Security Material] . . . 619
MESH/NODE/DF/ECHO/BI-03-C [Ignore Path Echo Reply – Two-way Path – Incorrect Security Material] . . . 619
MESH/NODE/DF/ECHO/BI-04-C [Ignore Path Echo Reply – Invalid Destination] . . . 621
4.18.8 Directed Forwarding Solicitation . . . 623
MESH/NODE/DF/SOL/BV-01-C [Send PATH_REQUEST_SOLICITATION, IUT as Path Target] . . . 623
MESH/NODE/DF/SOL/BV-02-C [Send PATH_REQUEST_SOLICITATION, IUT as Friend Path Target] . . . 625
MESH/NODE/DF/SOL/BV-03-C [Send PATH_REQUEST_SOLICITATION, IUT as Proxy Path Target] . . . 625
MESH/NODE/DF/SOL/BV-05-C [Send PATH_REQUEST_SOLICITATION, IUT as Subnet Bridge] . . . 625
MESH/NODE/DF/SOL/BV-04-C [Receive PATH_REQUEST_SOLICITATION, IUT as Path Origin] . . . 626
MESH/NODE/DF/SOL/BI-01-C [Ignore PATH_REQUEST_SOLICITATION – Invalid Parameters] . . . 629
4.18.9 Directed Forwarding Using Fixed Paths . . . 631
MESH/NODE/DF/FIX/BV-01-C [Forward Network Message to Unicast Address, IUT as Directed Relay] . . . 631
MESH/NODE/DF/FIX/BV-02-C [Forward Network Message to Group/Virtual Address, IUT as Directed Relay] . . . 633
MESH/NODE/DF/FIX/BV-03-C [Ignore Network Message, IUT as Directed Relay] . . . 634
MESH/NODE/DF/FIX/BV-04-C [Transmit Unsolicited Access Message for Existing Path - Unicast Address] . . . 636
MESH/NODE/DF/FIX/BV-05-C [Transmit Unsolicited Access Message for Existing Path - Group/Virtual Address] . . . 638
MESH/NODE/DF/FIX/BV-06-C [Transmit Access Message Response for Existing Path - Unicast Address] . . . 639
MESH/NODE/DF/FIX/BV-07-C [Transmit Access Message for Existing Path Without PATH_REQUEST] . . . 642
4.18.10 Directed Proxy . . . 643
MESH/SR/DPROX/BV-01-C [Connection with Proxy Client] . . . 643
MESH/SR/DPROX/BV-02-C [Connection with Directed Proxy Client] . . . 644
MESH/SR/DPROX/BV-03-C [Connection with Reject List Client] . . . 645
MESH/SR/DPROX/BV-04-C [Send DIRECTED_PROXY_CAPABILITIES_STATUS on Subnet Configuration] . . . 647
MESH/SR/DPROX/BV-05-C [Send DIRECTED_PROXY_CAPABILITIES_STATUS on Directed Proxy State Configuration] . . . 648
MESH/SR/DPROX/BV-06-C [Multiple Subnet Configuration] . . . 650
MESH/SR/DPROX/BV-07-C [Ignore Unsupported Directed Proxy Messages] . . . 652
MESH/SR/DPROX/BI-01-C [Ignore DIRECTED_PROXY_CONTROL Messages - Invalid Parameters] . . . 653
MESH/CL/DPROX/BV-01-C [Connection with Directed Proxy Server] . . . 654
MESH/CL/DPROX/BV-02-C [Ignore Unsupported Directed Proxy Messages] . . . 655
4.19 Directed Forwarding Configuration Model . . . 656
4.19.1 Common Directed Forwarding Configuration Model Server Behavior . . . 656
4.19.2 Common Directed Forwarding Configuration Model Client Procedures . . . 671
4.19.3 Directed Control State . . . 683
MESH/SR/DFM/DFS/BV-01-C [Directed Control Get/Set/Status Behavior] . . . 683
MESH/SR/DFM/DFS/BV-02-C [Directed Control Set Behavior – Binding with Directed Friend] . . . 683
MESH/SR/DFM/DFS/BV-03-C [Directed Control Set Behavior – Binding with Directed Proxy] . . . 683
MESH/SR/DFM/DFS/BV-04-C [Directed Control State Behavior – IUT as Low Power Node] . . . 685
MESH/SR/DFM/DFS/BV-05-C [Directed Control State Behavior – Binding with GATT Proxy State] . . . 686
MESH/SR/DFM/DFS/BV-06-C [Directed Control State Behavior – Binding with Friend State] . . . 688
MESH/SR/DFM/DFS/BI-01-C [Directed Forwarding State Get/Set Behavior – Invalid Parameters] . . . 689

MESH/CL/DFM/DFS/BV-01-C [Send Directed Forwarding Messages Behavior] . . . 689
MESH/CL/DFM/DFS/BI-01-C [Directed Forwarding Status Procedure – Invalid Parameters] . . . 689
4.19.4 Path Metric State . . . 690
MESH/SR/DFM/PMS/BV-01-C [Path Metric State Get/Set/Status Behavior] . . . 690
MESH/SR/DFM/PMS/BI-01-C [Path Metric State Get/Set Behavior – Invalid Parameters] . . . 690
MESH/CL/DFM/PMS/BV-01-C [Path Metric Get/Set/Status Procedure] . . . 690
MESH/CL/DFM/PMS/BI-01-C [Path Metric Status Procedure – Invalid Parameters] . . . 690
MESH/CL/DFM/PMS/BI-02-C [Path Metric Status Behavior – RFU Bits] . . . 690
4.19.5 Discovery Table Capabilities State . . . 691
MESH/SR/DFM/DTCS/BV-01-C [Discovery Table Capabilities Get/Set/Status Behavior] . . . 691
MESH/SR/DFM/DTCS/BI-01-C [Discovery Table Capabilities State Get/Set Behavior – Invalid Parameters] . . . 691
MESH/CL/DFM/DTCS/BV-01-C [Discovery Table Capabilities Get/Set/Status Procedure] . . . 692
MESH/CL/DFM/DTCS/BI-01-C [Discovery Table Capabilities Status Procedure – Invalid Parameters] . . . 692
4.19.6 Forwarding Table State . . . 692
MESH/SR/DFM/FTS/BV-01-C [Forwarding Table State Add Behavior] . . . 692
MESH/SR/DFM/FTS/BV-02-C [Forwarding Table State Delete Behavior] . . . 695
MESH/SR/DFM/FTS/BV-03-C [Forwarding Table State Dependents Add/Get Behavior] . . . 697
MESH/SR/DFM/FTS/BV-04-C [Forwarding Table State Dependents Delete Behavior] . . . 701
MESH/SR/DFM/FTS/BV-05-C [Forwarding Table State Entries Get Behavior] . . . 704
MESH/SR/DFM/FTS/BV-06-C [Forwarding Table State Add Behavior – Error Condition] . . . 707
MESH/SR/DFM/FTS/BV-07-C [Forwarding Table State Delete Behavior – Error Condition] . . . 709
MESH/SR/DFM/FTS/BV-08-C [Forwarding Table State Dependents Add/Delete Behavior – Error Condition] . . . 710
MESH/SR/DFM/FTS/BV-09-C [Forwarding Table State Get Behavior – Error Condition] . . . 713
MESH/SR/DFM/FTS/BV-10-C [Forwarding Table Add Behavior – Insufficient Resources] . . . 716
MESH/SR/DFM/FTS/BV-11-C [Forwarding Table State – Power-up Behavior] . . . 718
MESH/SR/DFM/FTS/BV-12-C [Forwarding Table State – Binding with Directed Forwarding] . . . 720
MESH/SR/DFM/FTS/BI-01-C [Forwarding Table State Add Behavior – Invalid Parameters] . . . 721
MESH/SR/DFM/FTS/BI-02-C [Forwarding Table State Delete Behavior – Invalid Parameters] . . . 722
MESH/SR/DFM/FTS/BI-03-C [Forwarding Table State Dependent Add/Delete Behavior – Invalid Parameters] . . . 723
MESH/SR/DFM/FTS/BI-04-C [Forwarding Table State Get Behavior – Invalid Parameters] . . . 724
MESH/CL/DFM/FTS/BV-01-C [Forwarding Table State Add Procedure] . . . 725
MESH/CL/DFM/FTS/BV-02-C [Forwarding Table State Get Procedure] . . . 727
MESH/CL/DFM/FTS/BV-03-C [Forwarding Table Delete Procedure] . . . 729
4.19.7 Wanted Lanes State . . . 730
MESH/SR/DFM/WLS/BV-01-C [Wanted Lanes State Get/Set/Status Behavior] . . . 730
MESH/SR/DFM/WLS/BI-01-C [Wanted Lanes Get/Set Behavior – Invalid Parameters] . . . 730
MESH/CL/DFM/WLS/BV-01-C [Send Wanted Lanes Get/Set/Status Procedure] . . . 731
MESH/CL/DFM/WLS/BI-01-C [Wanted Lanes Status Procedure – Invalid Parameters] . . . 731
4.19.8 Two Way Path State . . . 731
MESH/SR/DFM/TWPS/BV-01-C [Two Way Path State Get/Set Behavior] . . . 731
MESH/SR/DFM/TWPS/BI-01-C [Two Way Path State Get/Set Messages – Invalid Parameters] . . . 731
MESH/CL/DFM/TWPS/BV-01-C [Two Way Path Get/Set/Status Procedure] . . . 731
MESH/CL/DFM/TWPS/BI-01-C [Two Way Path Status Procedure – Invalid Parameters] . . . 731
4.19.9 Path Echo Interval State . . . 732
MESH/SR/DFM/PEIPS/BV-01-C [Path Echo Interval State Get/Set/Status Behavior, Change Both Intervals] . . . 732
MESH/SR/DFM/PEIPS/BV-02-C [Path Echo Interval State Get/Set/Status Behavior, Change Only Unicast] . . . 732
MESH/SR/DFM/PEIPS/BV-03-C [Path Echo Interval State Get/Set/Status Behavior, Change Only Multicast] . . . 732
MESH/SR/DFM/PEIPS/BI-01-C [Path Echo Interval State Get/Set Behavior – Invalid Parameters, Invalid Intervals] . . . 732
MESH/SR/DFM/PEIPS/BI-02-C [Path Echo Interval State Get/Set Behavior – Invalid Parameters, Invalid NetKeyIndex] 732

MESH/CL/DFM/PEIPS/BV-01-C [Path Echo Interval Get/Set/Status Procedure] . . . 733
MESH/CL/DFM/PEIPS/BI-01-C [Path Echo Interval Status Procedure – Invalid Parameters] . . . 733
4.19.10 Directed Network Transmit State . . . 733
MESH/SR/DFM/DNTS/BV-01-C [Directed Network Transmit State Get/Set Behavior] . . . 733
MESH/CL/DFM/DNTS/BV-01-C [Send Directed Network Transmit Get/Set Behavior] . . . 733
4.19.11 Directed Relay Retransmit State . . . 733
MESH/SR/DFM/DRTS/BV-01-C [Directed Relay Retransmit State Get/Set Behavior] . . . 734
MESH/CL/DFM/DRTS/BV-01-C [Send Directed Relay Retransmit Get/Set Behavior] . . . 734
4.19.12 RSSI Threshold State . . . 734
MESH/SR/DFM/RTS/BV-01-C [RSSI Threshold State Get/Set Behavior] . . . 734
MESH/SR/DFM/RTS/BI-01-C [RSSI Threshold State Set Behavior – Invalid Parameters] . . . 734
MESH/CL/DFM/RTS/BV-01-C [Send RSSI Threshold Get/Set Behavior] . . . 735
MESH/CL/DFM/RTS/BI-01-C [RSSI Threshold Status Procedure – Invalid Parameters] . . . 735
4.19.13 Directed Paths State . . . 735
MESH/SR/DFM/DPS/BV-01-C [Directed Paths State Get/Status Behavior] . . . 735
MESH/CL/DFM/DPS/BV-01-C [Send Directed Paths Get Behavior] . . . 735
4.19.14 Directed Publish Policy State . . . 735
MESH/SR/DFM/PPS/BV-01-C [Directed Publish Policy State Get/Set Behavior] . . . 735
MESH/SR/DFM/PPS/BI-01-C [Directed Publish Policy State Get/Set Behavior – Invalid Parameters] . . . 737
MESH/CL/DFM/PPS/BV-01-C [Send Directed Publish Policy Get/Set Behavior] . . . 738
MESH/CL/DFM/PPS/BI-01-C [Directed Publish Policy Status Procedure – Invalid Parameters] . . . 738
4.19.15 Path Discovery Timing Control State . . . 738
MESH/SR/DFM/DTS/BV-01-C [Path Discovery Timing Control State Get/Set Behavior] . . . 738
MESH/SR/DFM/DTS/BI-01-C [Path Discovery Timing Control State Get/Set Behavior – Invalid Parameters] . . . 738
MESH/CL/DFM/DTS/BV-01-C [Path Discovery Timing Control State Get/Set Behavior] . . . 738
4.19.16 Directed Control Network Transmit State . . . 739
MESH/SR/DFM/DCNTS/BV-01-C [Directed Control Network Transmit State Get/Set Behavior] . . . 739
MESH/CL/DFM/DCNTS/BV-01-C [Send Directed Control Network Transmit Get/Set Behavior] . . . 739
4.19.17 Directed Control Relay Retransmit State . . . 739
MESH/SR/DFM/DCRTS/BV-01-C [Directed Control Relay Retransmit State Get/Set Behavior] . . . 739
MESH/CL/DFM/DCRTS/BV-01-C [Send Directed Control Relay Retransmit Get/Set Behavior] . . . 739
4.20 Subnet Bridge . . . 739
4.20.1 Network Layer Messages . . . 740
MESH/NODE/SBR/NET/BV-01-C [Receive Network Layer Message, DST as Unicast Address, IUT as Subnet Bridge] . . . 740
MESH/NODE/SBR/NET/BV-02-C [Ignore Replayed Network Layer Message] . . . 742
4.21 Bridge Configuration Models . . . 744
4.21.1 Subnet Bridge State . . . 744
MESH/SR/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Behavior] . . . 744
MESH/SR/BCM/SBS/BI-01-C [Subnet Bridge State Set Behavior – Invalid Parameters] . . . 745
MESH/CL/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Procedure] . . . 746
4.21.2 Bridging Table State . . . 747
MESH/SR/BCM/BDS/BV-01-C [Bridging Table State Add Behavior] . . . 747
MESH/SR/BCM/BDS/BV-02-C [Bridging Table State Remove Behavior] . . . 749
MESH/SR/BCM/BDS/BV-03-C [Bridging Table State Get Behavior] . . . 751
MESH/SR/BCM/BDS/BV-04-C [Bridging Table State Subnets Get Behavior] . . . 752
MESH/SR/BCM/BDS/BV-05-C [Bridging Table State Behavior – Error Condition] . . . 754
MESH/SR/BCM/BDS/BV-06-C [Bridging Table State – Binding with NetKey List State] . . . 755
MESH/SR/BCM/BDS/BI-01-C [Bridging Table State Add Behavior – Invalid Parameters] . . . 756
MESH/SR/BCM/BDS/BI-02-C [Bridging Table State Remove Behavior – Invalid Parameters] . . . 757

MESH/SR/BCM/BDS/BI-03-C [Bridging Table State Subnets Get Behavior – Invalid Parameters] . . . 758
MESH/CL/BCM/BDS/BV-01-C [Bridging Table State Procedure] . . . 758
4.21.3 Bridging Table Size State . . . 760
MESH/SR/BCM/BTS/BV-01-C [Bridging Table Size Get Behavior] . . . 760
MESH/CL/BCM/BTS/BV-01-C [Bridging Table Size State Procedure] . . . 761
4.22 Remote Provisioning . . . 762
4.22.1 Scanning . . . 762
MESH/SR/RPR/SCN/BV-01-C [Scan Any Device UUID] . . . 762
MESH/SR/RPR/SCN/BV-02-C [Scan Specific Device UUID] . . . 766
MESH/SR/RPR/SCN/BV-03-C [Process Consecutive Scan Requests] . . . 768
MESH/SR/RPR/SCN/BV-04-C [Extended Scanning of Unprovisioned Device Beacons] . . . 770
MESH/SR/RPR/SCN/BV-05-C [Extended Scanning of Mesh Provisioning Service] . . . 770
MESH/SR/RPR/SCN/BV-06-C [Extended Scanning on the Remote Provisioning Server] . . . 773
MESH/SR/RPR/SCN/BV-07-C [Scan Items Limit and Timeout] . . . 775
MESH/SR/RPR/SCN/BV-08-C [Extended Scanning with Bad Hash or Different BD_ADDR] . . . 777
MESH/SR/RPR/SCN/BI-01-C [Ignore Invalid Messages] . . . 779
MESH/SR/RPR/SCN/BI-02-C [Reject Scan Requests with Invalid Parameters] . . . 781
MESH/CL/RPR/SCN/BV-01-C [Send Scan Commands and Process Scan Reports] . . . 782
4.22.2 Link Management . . . 784
MESH/SR/RPR/LNK/BV-01-C [Open and Close the Link – PB-ADV] . . . 786
MESH/SR/RPR/LNK/BV-02-C [Open and Close the Link – PB-GATT] . . . 786
MESH/SR/RPR/LNK/BV-03-C [Respond to Redundant Link Requests with Unprovisioned Device – PB-ADV] . . . 787
MESH/SR/RPR/LNK/BV-04-C [Respond to Redundant Link Requests with Unprovisioned Device – PB-GATT] . . . 787
MESH/SR/RPR/LNK/BV-05-C [Outbound Packet Transfer State – Handle Link Open] . . . 793
MESH/SR/RPR/LNK/BV-06-C [Outbound Packet Transfer State – Handle Link Close] . . . 793
MESH/SR/RPR/LNK/BV-07-C [Outbound Packet Transfer State – Handle Link Closed by Device] . . . 793
MESH/SR/RPR/LNK/BV-09-C [Outbound Packet Transfer State – Handle Resending the Outbound PDU] . . . 793
MESH/SR/RPR/LNK/BV-10-C [Outbound Packet Transfer State – Handle PDU Transfer Complete] . . . 793
MESH/SR/RPR/LNK/BV-11-C [Outbound Packet Transfer State – Handle Provisioning Link Timeout] . . . 793
MESH/SR/RPR/LNK/BI-01-C [Outbound Packet Transfer State – Handle Unexpected Outbound PDU] . . . 793
MESH/SR/RPR/LNK/BV-13-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Link Close] . . . 797
MESH/SR/RPR/LNK/BV-14-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Link Close] . . . 797
MESH/SR/RPR/LNK/BV-15-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Link Closed by Device] . . . 797
MESH/SR/RPR/LNK/BV-16-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Link Closed by Device] . . . 797
MESH/SR/RPR/LNK/BV-17-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Resending the Outbound PDU] . . . 797
MESH/SR/RPR/LNK/BV-18-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Resending the Outbound PDU] . . . 797
MESH/SR/RPR/LNK/BV-19-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Network Key Removal] . . . 797
MESH/SR/RPR/LNK/BV-22-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Network Key Removal] . . . 797
MESH/SR/RPR/LNK/BI-02-C [Link Active State with Unprovisioned Device – Handle Invalid Outbound PDU] . . . 797
MESH/SR/RPR/LNK/BV-20-C [Respond to Redundant Link Requests with Device Key Interface Procedures] . . . 799
MESH/SR/RPR/LNK/BV-21-C [Link Active State with Device Key Interface Procedures – Handle Link Close] . . . 802
MESH/SR/RPR/LNK/BV-23-C [Link Active State with Device Key Interface Procedures – Handle Resending the Outbound PDU] . . . 802

MESH/SR/RPR/LNK/BV-25-C [Link Active State with Device Key Interface Procedures – Handle Power Reset] . . . 802
MESH/SR/RPR/LNK/BV-26-C [Link Active State with Device Key Interface Procedures – Handle Network Key Removal] . . . 802
MESH/SR/RPR/LNK/BI-03-C [Link Active State with Device Key Interface Procedures – Handle Invalid Outbound PDU] . . . 802
MESH/SR/RPR/LNK/BI-04-C [Ignore Invalid Link Requests] . . . 807
MESH/SR/RPR/LNK/BI-05-C [Reject Parallel Provisioning and Node Provisioning Protocol Interface Procedure] . . . 808
MESH/CL/RPR/LNK/BV-01-C [Send Link Commands and Process Link Reports] . . . 810
4.22.3 Provisioning over PB-Remote . . . 812
MESH/SR/RPR/PDU/BV-01-C [Provision an Unprovisioned Device over PB-ADV] . . . 812
MESH/SR/RPR/PDU/BV-05-C [Provision an Unprovisioned Device over PB-GATT] . . . 812
MESH/SR/RPR/PDU/BV-02-C [Device Key Refresh Procedure] . . . 817
MESH/SR/RPR/PDU/BV-03-C [Node Address Refresh Procedure] . . . 817
MESH/SR/RPR/PDU/BV-04-C [Node Composition Refresh Procedure] . . . 817
MESH/SR/RPR/PDU/BI-01-C [Device Key Refresh Failures] . . . 817
MESH/SR/RPR/PDU/BI-02-C [Node Address Refresh Failures] . . . 817
MESH/SR/RPR/PDU/BI-03-C [Node Composition Refresh Failures] . . . 817
MESH/CL/RPR/PDU/BV-01-C [Execute a Complete Provisioning Procedure over PB-Remote] . . . 823
4.23 On-demand Proxy Models . . . 825
4.23.1 On-demand Proxy State . . . 825
MESH/SR/ODP/BV-01-C [On-Demand Private GATT Proxy State Get/Set Behavior] . . . 825
MESH/CL/ODP/BV-01-C [On-Demand Private GATT Proxy State Get/Set Procedure] . . . 827
4.24 SAR Configuration Models . . . 828
4.24.1 SAR Transmitter State . . . 829
MESH/SR/SAR/BV-01-C [SAR Transmitter State Get/Set Behavior] . . . 829
MESH/CL/SAR/BV-01-C [SAR Transmitter State Get/Set Procedure] . . . 830
4.24.2 SAR Receiver State . . . 831
MESH/SR/SAR/BV-02-C [SAR Receiver State Get/Set Behavior] . . . 831
MESH/CL/SAR/BV-02-C [SAR Receiver State Get/Set Procedure] . . . 833
4.25 Opcodes Aggregator Models . . . 834
MESH/SR/AGG/BV-01-C [Message Request List Processing Behavior – Success] . . . 834
MESH/SR/AGG/BV-02-C [Message Request List Processing Behavior – Errors] . . . 837
MESH/SR/AGG/BI-01-C [Message Request List Processing Behavior – Invalid Parameters] . . . 839
MESH/CL/AGG/BV-01-C [Message Request List Processing Procedure] . . . 839
4.26 Large Composition Data Models . . . 841
4.26.1 Large Composition Data Procedures . . . 841
MESH/SR/LCD/LCMP/BV-01-C [Respond to LARGE_COMPOSITION_DATA_GET Messages] . . . 841
MESH/CL/LCD/LCMP/BV-01-C [Send LARGE_COMPOSITION_DATA_GET Messages] . . . 843
4.26.2 Models Metadata Procedures . . . 844
MESH/SR/LCD/MMD/BV-01-C [Respond to MODELS_METADATA_GET Messages] . . . 845
MESH/CL/LCD/MMD/BV-01-C [Send MODELS_METADATA_GET Messages] . . . 847
4.27 Solicitation PDU RPL Configuration Models . . . 848
MESH/SR/SRPL/BV-01-C [RPL Item Clear Behavior] . . . 849
MESH/CL/SRPL/BV-01-C [RPL Item Clear Procedure] . . . 850
4.28 Certificate-Based Provisioning . . . 851
4.28.1 Get provisioned using X.509 Certificates retrieved out-of-band . . . 851
MESH/NODE/CBP/BV-01-C [Get Provisioned using a X.509 Certificate over PB-ADV] . . . 853
MESH/NODE/CBP/BV-02-C [Get Provisioned using a X.509 Certificate over PB-GATT] . . . 853

4.28.2 Send X.509 Certificates over the Provisioning Bearer . . . 853
MESH/NODE/CBP/BV-03-C [Send X.509 Certificates over PB-ADV] . . . 855
MESH/NODE/CBP/BV-04-C [Send X.509 Certificates over PB-GATT] . . . 855
MESH/NODE/CBP/BI-01-C [Handle Invalid Provisioning Record Request PDUs] . . . 855
4.28.3 Retrieve X.509 Certificates over HTTPS and Provision Device . . . 857
MESH/PVNR/CBP/BV-01-C [Retrieve X.509 Certificates over HTTPS and Provision Device over PB-ADV] . . . 858
MESH/PVNR/CBP/BV-02-C [Retrieve X.509 Certificates over HTTPS and Provision Device over PB-GATT] . . . 858
4.28.4 Retrieve X.509 Certificates over the Provisioning Bearer . . . 859
MESH/PVNR/CBP/BV-03-C [Retrieve X.509 Certificates over PB-ADV] . . . 860
MESH/PVNR/CBP/BV-04-C [Retrieve X.509 Certificates over PB-GATT] . . . 860
MESH/PVNR/CBP/BI-01-C [Reject Invalid X.509 Certificates] . . . 861
5 Test case mapping . . . 864
6 Annex: Supplementary Mesh Test Cases . . . 886
6.1 Mesh Node role supplementary tests . . . 886
MESH/NODE/BV-01-C [Provisioning and reprovisioning] . . . 886
MESH/NODE/BV-02-C [Mesh device composition data] . . . 886
MESH/NODE/BV-03-C [Network message cache and replay protection] . . . 887
7 Revision history and acknowledgments . . . 888

1 Scope

This Bluetooth document contains the Test Suite Structure (TSS) and test cases to test the implementation of the Bluetooth Mesh Protocol Specification with the objective to provide a high probability of air interface interoperability between the tested implementation and other manufacturers' Bluetooth devices.

2 References, definitions, and abbreviations

2.1 References

This document incorporates provisions from other publications by dated or undated reference. These references are cited at the appropriate places in the text, and the publications are listed hereinafter. Additional definitions and abbreviations can be found in [1] and [2].

[1] Bluetooth Core Specification, Version 4.0 or later

[2] Test Strategy and Terminology Overview

[3] Mesh Protocol Specification, Version 1.0 or later

[4] ICS Proforma for Mesh Protocol

[5] GATT Test Suite, GATT.TS

[6] IXIT Proforma for Mesh Protocol

[7] Mesh Erratum 10395: Mandating ECDH Public key validation

[8] Appropriate Language Mapping Tables document

[9] Mesh Erratum 16350: Remove security claim

[10] Mesh Protocol Specification, Version 1.1

[11] Mesh Model Specification, Version 1.0 or later

[12] Mesh Model (MMDL) Test Suite, MMDL.TS

[13] ICS Proforma for Mesh Model (MMDL)

Note: Previous versions of the Mesh Specification (Versions 1.0 and 1.0.1) were known as “Mesh Profile Specification.” The name of the specification has changed to “Mesh Protocol Specification” in the 1.1 version, and any mentions of the Mesh Specification herein reflect the new title.

2.2 Definitions

In this Bluetooth document, the definitions from [1] and [2] apply.

Certain terms that were identified as inappropriate have been replaced. For a list of the original terms and their replacement terms, see the Appropriate Language Mapping Tables document [8].

2.3 Acronyms and abbreviations

In this Bluetooth document, the definitions, acronyms, and abbreviations from [1] and [2] apply.

3 Test Suite Structure (TSS)

3.1 Test Strategy

The test objectives are to verify the functionality of the Mesh Protocol within a Bluetooth Host and enable interoperability between Bluetooth Hosts on different devices. The testing approach covers mandatory and optional requirements in the specification and matches these to the support of the IUT as described in the ICS. Any defined test herein is applicable to the IUT if the ICS logical expression defined in the Test Case Mapping Table (TCMT) evaluates to true.

The test equipment provides an implementation of the Radio Controller and the parts of the Host needed to perform the test cases defined in this Test Suite. A Lower Tester acts as the IUT's peer device and interacts with the IUT over-the-air interface. The configuration, including the IUT, needs to implement similar capabilities to communicate with the test equipment. For some test cases, it is necessary to stimulate the IUT from an Upper Tester. In practice, this could be implemented as a special test interface, a Man Machine Interface (MMI), or another interface supported by the IUT.

This Test Suite contains Valid Behavior (BV) tests complemented with Invalid Behavior (BI) tests where required. The test coverage mirrored in the Test Suite Structure is the result of a process that started with catalogued specification requirements that were logically grouped and assessed for testability enabling coverage in defined test purposes.

This specification uses features common in many GATT-based specifications although it includes an advertising-based protocol that will introduce behavior per the data contained in the advertising packet.

3.1.1 Power-Up Behavior Tests

Some test cases verify that the IUT maintains specific data in the non-volatile memory and that it restores the data correctly during a power-up sequence, as configured by other Mesh nodes.

To run such test cases, the IUT must be capable to reset when commanded by the Upper Tester and to continue the test procedure after the power-up sequence.

3.1.2 Common Behaviors and Procedures

This Test Suite defines common behaviors and procedures in the form of common test cases.

These common test cases are described using parametrized message names and values, known as macros. Macros are written in a unique underscore-and-uppercase notation, for example:

- _SET_ stands for an acknowledged SET message name.
- _PARAM_SET_ stands for the parameters of a SET message.

For each Model-state combination that defines a behavior or procedure that can be described using a common test case, an associated table defines the message names and lists of parameters.

A table in a certain subsection is always associated with the common test procedure described in its enclosing section. For example, the tables from Sections 4.19.1.1.1 to 4.19.1.1.8 are associated with the common test

procedure from Section 4.19.1.1.

Each table is then referenced by a Model test case that needs to be executed as part of the test plan. To execute such a test case, the common test case is run by replacing the macros with the messages and parameters described by the associated table.

3.2 Test groups

The following test groups have been defined:

- ADV Provisioning Bearer
- Mesh Provisioning Service
- Provisioning Procedure
- Beacons
- Network Layer
- Relay
- Transport Layers
- IV Update
- Key Refresh
- Friendship
- Access Layer
- Proxy Protocol
- Mesh Proxy Service
- Configuration Model
- Health Model
- Subnet Bridge
- Directed Forwarding

4 Test cases (TC)

4.1 Introduction

4.1.1 Test case identification conventions

Test cases are assigned unique identifiers per the conventions in [2]. The convention used here is: **<spec abbreviation>/<IUT role>/**<class>/**<feat>**/<func>/<subfunc>/<cap>**/<xx>-<nn>-<y>**.

Identifier Abbreviation	**Spec Identifier <spec abbreviation>**
MESH	Mesh Protocol
Identifier Abbreviation	**Role Identifier <IUT role>**
CFGCL	Configuration Client role
CL	Client role
NODE	Node role
PVNR	Provisioner role
SR	Server role
Identifier Abbreviation	**Feature Identifier <feat>**
ACC	Access Layer
BCM	Bridge Configuration Model
BCN	Beacons
CFG	Configuration Model
DF	Directed Forwarding
DFM	Directed Forwarding Configuration Model
FRND	Friendship
HM	Health Model
IVU	IV Update
KR	Key Refresh
MPS	Mesh Provisioning Service
MPXS	Mesh Proxy Service
NET	Network Layer
PBADV	PB-ADV Provisioning
PRB	Private Beacon Model
PROV	Provisioning Procedure
PROX	Proxy Protocol
RLY	Relay
SBR	Subnet Bridge
TNPT	Transport Layers

Identifier Abbreviation	Capability Identifier <cap>
AKL	Configuration Model - AppKey List Procedures
ATS	Health Model - Attention Timer state
BDS	Bridging Table State
BTS	Bridging Table Size State
CFGF	Configuration Model - Friend Procedure
CFGR	Configuration Model - Relay Procedure
CFS	Health Model - Current Fault state
COMP	Configuration Model - Composition Data Procedure
CONF	Directed Forwarding - Confirmation
DCNTS	Directed Forwarding Configuration Model - Directed Control Network Transmit State
DCRTS	Directed Forwarding Configuration Model - Directed Control Relay Retransmit State
DFS	Directed Forwarding Configuration Model - Directed Control State
DISC	Directed Forwarding - Discovery
DNTS	Directed Forwarding Configuration Model - Directed Network Transmit State
DPROX	Directed Forwarding - Directed Proxy
DPS	Directed Forwarding Configuration Model - Directed Paths State
DRTS	Directed Forwarding Configuration Model - Directed Relay Retransmit State
DTCS	Directed Forwarding Configuration Model - Discovery Table Capabilities State
DTS	Directed Forwarding Configuration Model - Path Discovery Timing Control State
DTTL	Configuration Model - Default TTL Procedure
ECHO	Directed Forwarding - Echo
EST	Directed Forwarding - Establishment
FIX	Directed Forwarding - Forwarding using Fixed Paths
FN	Friendship - Friend Node Procedures
FTS	Directed Forwarding Configuration Model - Forwarding Table State
GPXY	Configuration Model - GATT Proxy Procedures
HBP	Configuration Model - Heartbeat Publication Procedures
HBS	Configuration Model - Heartbeat Publication Procedures
HPS	Health Model - Health Period state
INIT	Directed Forwarding - Initialization
KRP	Configuration Model - Key Refresh Procedures

Identifier Abbreviation	Capability Identifier <cap>
LPN	Friendship - Low Power Node Procedures
LPNPT	Configuration Model - Low Power Node PollTimeout Procedures
MAKL	Configuration Model - Model to AppKey List Procedures
MP	Configuration Model - Model Publication Procedures
NID	Configuration Model - Node Identity Procedures
NKL	Configuration Model - NetKey List Procedures
NTX	Configuration Model - Network Transmit Procedures
PBS	Private Beacon state
PEIPS	Directed Forwarding Configuration Model - Path Echo Interval State
PGPXY	Private GATT Proxy state
PMS	Directed Forwarding Configuration Model - Path Metric State
PNID	Private Node Identity state
PPS	Directed Forwarding Configuration Model - Directed Publish Policy State
RFS	Health Model - Registered Fault state
RTS	Directed Forwarding Configuration Model - RSSI Threshold State
RST	Configuration Model - Reset Procedures
SBS	Subnet Bridge State
SL	Configuration Model - Subscription List Procedures
SNBP	Configuration Model - Secure Network Beacon Procedure
SOL	Directed Forwarding - Solicitation
TWPS	Directed Forwarding Configuration Model - Two Way Path State
UPDATE	Directed Forwarding - Update
WLS	Directed Forwarding Configuration Model - Wanted Lanes State

Table 4.1: Mesh TC feature naming conventions

4.1.2 Conformance

When conformance is claimed for a particular specification, all capabilities are to be supported in the specified manner. The mandated tests from this Test Suite depend on the capabilities to which conformance is claimed.

The Bluetooth Qualification Program may employ tests to verify implementation robustness. The level of implementation robustness that is verified varies from one specification to another and may be revised for cause based on interoperability issues found in the market.

Such tests may verify:

- That claimed capabilities may be used in any order and any number of repetitions not excluded by the specification

- That capabilities enabled by the implementations are sustained over durations expected by the use case
- That the implementation gracefully handles any quantity of data expected by the use case
- That in cases where more than one valid interpretation of the specification exists, the implementation complies with at least one interpretation and gracefully handles other interpretations
- That the implementation is immune to attempted security exploits

A single execution of each of the required tests is required to constitute a Pass verdict. However, it is noted that to provide a foundation for interoperability, it is necessary that a qualified implementation consistently and repeatedly pass any of the applicable tests.

In any case, where a member finds an issue with the test plan generated by the Bluetooth SIG qualification tool, with the test case as described in the Test Suite, or with the test system utilized, the member is required to notify the responsible party via an erratum request such that the issue may be addressed.

4.1.3 Pass/Fail verdict conventions

Each test case has an Expected Outcome section. The IUT is granted the Pass verdict when all the detailed pass criteria conditions within the Expected Outcome section are met.

The convention in this Test Suite is that, unless there is a specific set of fail conditions outlined in the test case, the IUT fails the test case as soon as one of the pass criteria conditions cannot be met. If this occurs, the outcome of the test is a Fail verdict.

4.2 Setup preambles

4.2.1 State: Node PB-ADV Link Established

1. The IUT is induced to send Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT's Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
3. The Lower Tester begins to periodically advertise PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.
4. The Lower Tester receives a Link Ack message from the IUT before the link establishment timeout. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Ack message has a BearerOpcode value of 0x01, the GPCF field is set to 0b11, the Generic Provisioning Payload is zero length, the Link ID is set to the value sent in the Link Open message, the Transaction Number value is set to 0x00, and the Link Ack message has no parameters.

4.2.2 State: Provisioner PB-ADV Link Established

1. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Open message has a Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the Lower Tester as supplied in the IXIT [6].
2. The Lower Tester detects the PB-ADV PDU advertised by the IUT.
3. The Lower Tester periodically advertises a Link Ack message containing the newly allocated Link ID sent in Step 1. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.

4.2.3 State: Node PB-GATT Link Established

1. IUT is induced to advertise the Mesh Provisioning Service in connectable advertising PDUs; if the IUT supports PB-ADV, it interleaves these connectable advertising PDUs with Unprovisioned Device beacons.
2. The Lower Tester scans for connectable advertisements from the IUT until it receives a service advertisement containing the Mesh Provisioning Service and with a Service Data field containing the IUT's Device UUID.
3. Execute the steps in setup preamble 4.2.5.
4. The Lower Tester discovers the handles for the Mesh Provisioning Data Out characteristic and the Mesh Provisioning Data In characteristic by executing the procedure in GATT.TS [5] Discover All Characteristics of a Service - by Client, GATT/CL/GAD/BV-04-C.
5. The Lower Tester sends an ATT_Write_Request to the IUT with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.

4.2.4 State: Provisioner PB-GATT Link Established

1. Execute the steps in setup preamble 4.2.5.
2. The IUT is induced to discover the handles for the Mesh Provisioning Data In characteristic and the Mesh Provisioning Data out characteristic exposed by the Mesh Provisioning Service on the Lower Tester by executing the procedures in GATT.TS [5] Discover All Characteristics of a Service - by Server, GATT/SR/GAD/BV-04-C.
3. The IUT is induced to send an ATT_Write_Request to the Lower Tester with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.

4.2.5 Establish an ATT Bearer Connection

1. Establish an LE transport connection between the IUT and the Lower Tester.
2. Establish an L2CAP channel 0x0004 between the IUT and the Lower Tester over that LE transport.

4.2.6 Network and Application Security Credentials

Some test cases require, as initial condition, that the IUT has been provisioned and that it shares a common network key and an IV Index with the Lower Tester. This data is denoted as network security credentials.

Some test cases require the IUT and the Lower Tester to also share application security credentials, which consist of an application key defined in the IXIT file.

Some test cases require the IUT and the Lower Tester to encrypt Network PDUs with directed security credentials.

4.2.7 State: Adding a Bridging Table Entry, IUT as Subnet Bridge

- Preamble Purpose
  This preamble procedure specifies how to add bridging table entries on a Subnet Bridge IUT.
- Reference

  [10] 4.4.9.2.2
- Preamble Procedure

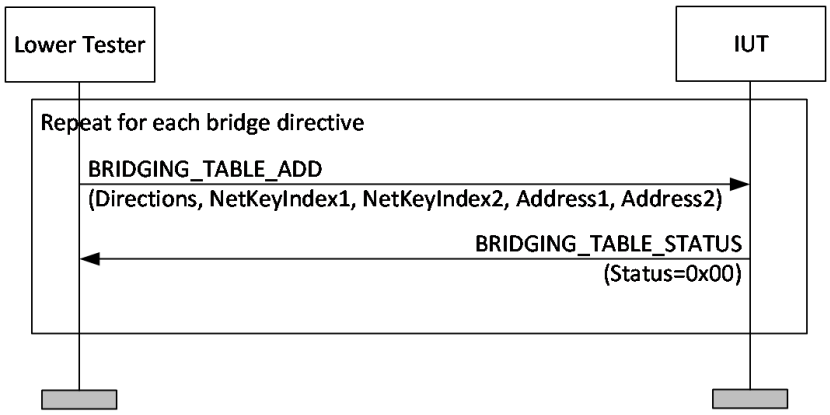


Figure 4.1: Adding a bridging table entry, IUT as Subnet Bridge MSC

For each bridging table entry (Directions, NetKeyIndex1, NetKeyIndex2, Address1, Address2), repeat Steps 1 and 2.

1. The Lower Tester sends a BRIDGING_TABLE_ADD message with Directions, NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to the corresponding values from the bridging table entry.
2. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to Directions field from Step 1, and NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to the values in Step 1.

4.2.8 Common Initial Test Conditions for Directed Forwarding Tests

Unless stated otherwise in individual test cases, the following applies throughout the Directed Forwarding test in Section 4.18:

- The IUT and the Lower Testers share established network security material and directed security material.
- The Lower Testers have access to the device key of the IUT defined in the IXIT [6] parameters.
- The Discovery Table on the IUT is empty.
- Unless otherwise stated, the following states are configured on the IUT for all subnets:
  - Directed Forwarding is enabled.
  - Two Way Path is set to 0b0.
  - The Path Discovery Retry Interval state is set to 0x0000.
  - Directed Proxy, Directed Proxy Use Directed Default, Directed Relay, Directed Friend, Path Metric, Discovery Table Capabilities, Wanted Lanes, Path Echo Interval, RSSI Threshold, Path Discovery Interval, Path Monitoring Interval, and Lane Discovery Guard Interval states are set to the corresponding default values.
  - The Forwarding Table is empty.
- The primary element address of the IUT is marked as _IUT_ADDR_. The unicast address range of the IUT (see [10] Section 4.3.1.6) is marked as _IUT_UAR_ and sets the RangeStart field to _IUT_ADDR_. If the IUT supports secondary elements, the LengthPresent field is set to 0b1 and RangeLength is set to the number of addresses in the range of unicast addresses, otherwise the LengthPresent field is set to 0b0.
- The primary element address of the Lower Tester is marked as _LT_ADDR_. If more than one Lower Tester is used, the addresses are marked as _LT1_ADDR_, _LT2_ADDR_, and so on. The unicast address range values of the Lower Testers are marked as _LT1_UAR_, _LT2_UAR_, and so on, and have the LengthPresent field set to 0b1 and the RangeLength set to 0x02.
- _PO_ADDR_, _DO_ADDR_, _DT_ADDR_, and _PT_ADDR_ are random valid unicast addresses not matching the element addresses of the IUT or the Lower Testers. These addresses are specified in IXIT items TSPX_po_addr and TSPX_pt_addr, respectively. The corresponding unicast address range values are marked as _PO_UAR_, _DO_UAR_, _DT_UAR_, and _PT_UAR_, respectively, and have the LengthPresent field set to "1" and the RangeLength set to 0x02.
- _GR_ADDR_ is a random valid group address, for which the IUT is not subscribed, unless otherwise specified.
- _VR_ADDR_ is a random valid virtual address, for which the IUT is not subscribed, unless otherwise specified. This virtual address is computed based on the label UUID specified in IXIT item TSPX_mdf_label_uuid_for_virtual_address.
- _FN_ is marked as a random valid value used by the Lower Tester for the Path Origin Forwarding Number, unless otherwise specified.

4.2.9 State: Path Established, as Origin, Unicast Address Destination

- Preamble Purpose

  This preamble procedure specifies how an IUT establishes a path to a unicast target destination as a path origin.

- Reference

  [10] 3.6.8.2.1, 3.6.8.2.3

- Preamble Procedure

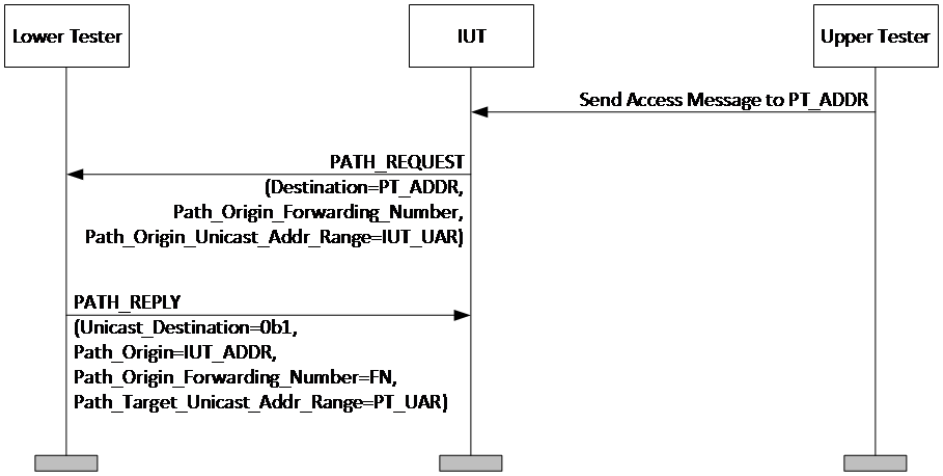


Figure 4.2: Preamble Procedure for State: Path Established, as Origin, Unicast Address Destination MSC

1. The Upper Tester commands the IUT to send an access message to _PT_ADDR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to the Path_Origin_Path_Metric_Type field value in the IUT's Discovery Table, Path_Origin_Path_Lifetime set to the Path_Origin_Path_Lifetime field value in the IUT's Discovery Table, Path_Discovery_Interval set to the Path_Discovery_Interval field value in the IUT's Discovery Table, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_. The Lower Tester stores the Path_Origin_Forwarding_Number value as _FN_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.

4.2.10 State: Path Established as Directed Relay, Unicast Address Destination

- Preamble Purpose

  This preamble procedure specifies how an IUT establishes a path to a unicast target destination as a directed relay. The path may be established as one-way or two-way.

- Reference

  [10] 3.6.8.2.2, 3.6.8.2.3

- Preamble Procedure

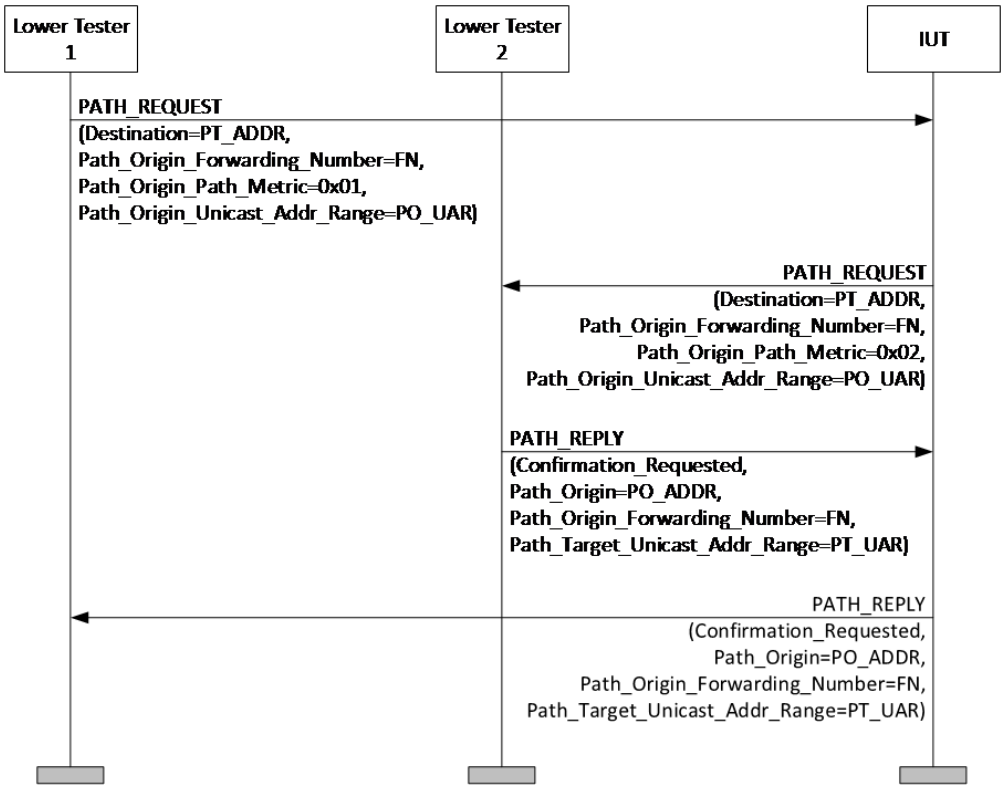


Figure 4.3: Preamble procedure for State: Path Established as Directed Relay, Unicast Address Destination MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and the other fields set to the same values as in Step 1.
3. Lower Tester 2 sends a PATH_REPLY message to _IUT_ADDR_ with On_Behalf_Of_Dependent_Target set to 0b0; Confirmation_Request set to 0b1 if the path is two-way, otherwise set to 0b0; Path_Origin set to _PO_ADDR_; Path_Origin_Forwarding_Number set to _FN_; and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
4. The IUT sends a PATH_REPLY message to _LT1_ADDR_ with all fields set to the same values as in Step 3.

4.2.11 State: Path Established as Destination, Unicast Address Destination

- Preamble Purpose

This preamble procedure specifies how an IUT establishes a path as a target destination.

- Reference

  [10] 3.6.8.2.3, 3.6.8.2.4

- Preamble Procedure

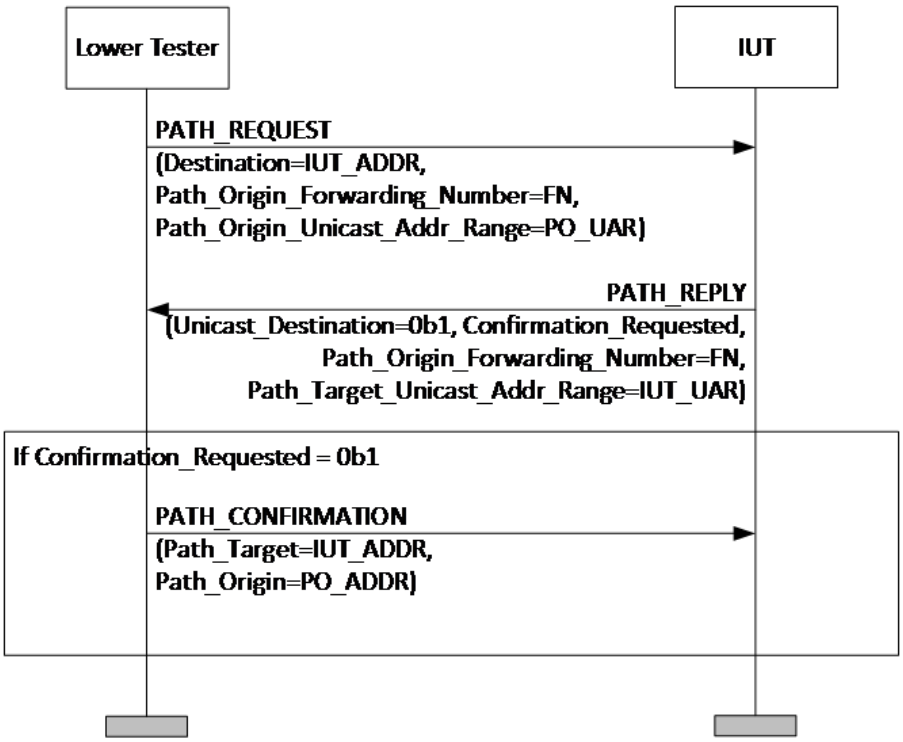


Figure 4.4: Preamble procedure for State: Path Established as Destination, Unicast Address Destination MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Destination set to _IUT_ADDR_, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Origin_Path_Metric set to 0x00.
2. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to the value of the Two Way Path state on the IUT, Path_Origin, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _IUT_UAR_.
3. If the Confirmation_Request field from Step 2 is set to 0b1, the Lower Tester sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _IUT_ADDR_ and Path_Origin set to _PO_ADDR_.

4.2.12 Proxy Connection with Proxy Client, IUT as Directed Proxy Server

- Preamble Purpose

  This preamble procedure specifies how an IUT establishes a path as a target destination.

- Reference

  [10] 4.4.8.2.1, 6.2.1.1

- Preamble Procedure

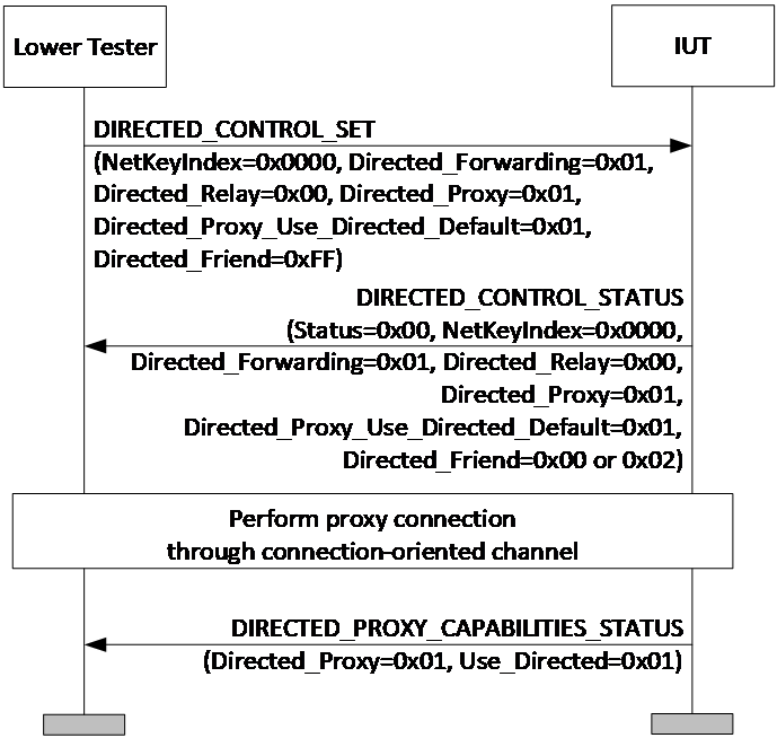


Figure 4.5: Preamble procedure for State: Proxy connection with Proxy client, IUT as Directed Proxy Server MSC

1. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x01 (Enabled), and Directed_Friend set to 0xFF (Ignore).
2. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x01 (Enabled), and Directed_Friend set to 0x00 or 0x02.
3. The Lower Tester performs a proxy connection to the IUT through a connection-oriented channel.
4. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x01 (Enabled).

4.2.13 State: Proxy Connection with Directed Proxy Client, IUT as Directed Proxy Server

- Preamble Purpose

  This preamble procedure specifies how an IUT establishes a path as a target destination.

- Reference

  [10] 4.4.8.2.1, 6.2.1.1

- Preamble Procedure

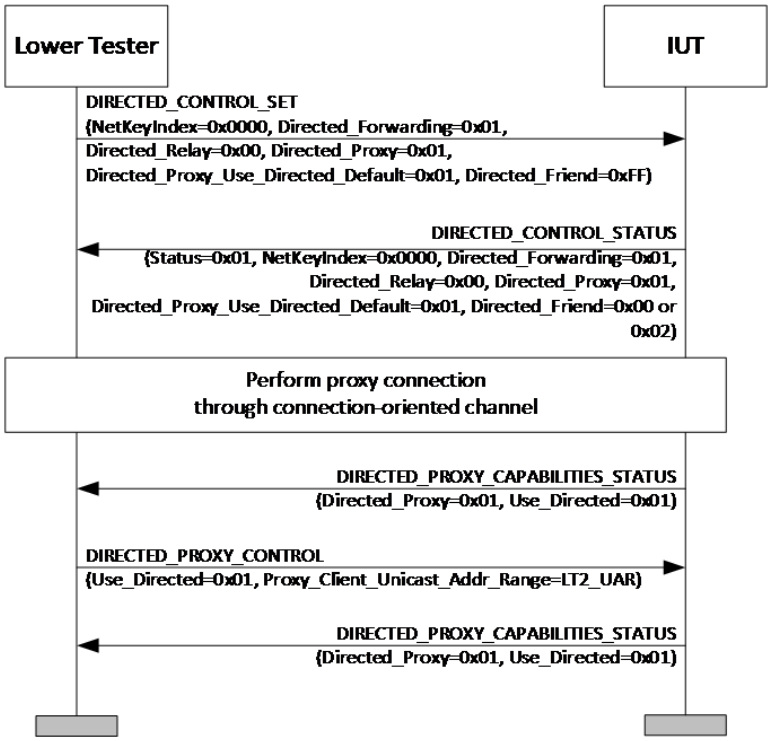


Figure 4.6: Preamble procedure for State: Proxy connection with Directed Proxy client, IUT as Directed Proxy Server MSC

1. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x01 (Enabled), and Directed_Friend set to 0xFF (Ignore).
2. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x01 (Enabled), and Directed_Friend set to 0x00 or 0x02.
3. The Lower Tester performs a proxy connection to the IUT through a connection-oriented channel.
4. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x01 (Enabled).
5. The Lower Tester sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed set to 0x01 (Enabled) and Proxy_Client_Unicast_Addr_Range set to _LT2_UAR_.
6. The IUT responds with a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x01 (Enabled).

4.3 ADV Provisioning Bearer

The test group objective is to verify behavior specific to the PB-ADV provisioning bearer. These include the testing of Link Establishment and PB-ADV PDUs. These test cases also serve as common procedures for the general provisioning procedures when PB-ADV is the selected bearer.

MESH/NODE/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close]

- Test Purpose

  Verify that an unprovisioned device IUT can establish a PB-ADV provisioning bearer link. Verify that the IUT sends correctly formatted PB-ADV packets.

- Reference

  [3] 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure
  1. The Upper Tester orders the IUT to send Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT's Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
  2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
  3. The Lower Tester begins to periodically advertise PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.
  4. The Lower Tester receives a Link Ack message from the IUT before the link establishment timeout. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Ack message has a BearerOpcode value of 0x01, the GPCF field is set to 0b11, the Generic Provisioning Payload is zero length, the Link ID is set to the value sent in the Link Open message, the Transaction Number value is set to 0x00, and the Link Ack message has no parameters.
  5. The Lower Tester begins to periodically advertise a Link Close message with a Reason value of 0x00 to terminate the link. The Lower Tester sends this message until the IUT reports that the PB-ADV link has been closed.

- Expected Outcome

  Pass verdict

  The IUT broadcasts an Unprovisioned Device beacon that is formatted correctly and contains valid values.

  The IUT ignores the value of the Transaction Number field in the Generic Provisioning PDU containing the Link Open message.

  The IUT responds to the Link Open message with a Link Ack message and establishes a PB-ADV link.

  The IUT receives a Link Close message from the Lower Tester and reports that the PB-ADV link is closed.

  All Generic Provisioning PDUs are sent using the Mesh Provisioning AD Type («Mesh Provisioning»).

  All bits of the Transaction Number field are set to 0 in each PB-ADV PDU sent by the IUT. All PB-ADV PDUs are formatted correctly with the correct values and are non-connectable.

MESH/NODE/PBADV/BV-02-C [Ignore Parallel Link Messages]

- Test Purpose

  Verify that an unprovisioned device IUT ignores multiple parallel Link Open attempts and only sends a Link Ack on the first received Link Open attempt and on subsequent Link Open messages with the same Link ID. Verify that after Link Close the IUT will respond again to future Link Open attempts. Verify that the IUT accepts new Link Open attempts after 60 seconds have passed without receiving any provisioning PDUs. Verify that the IUT ignores all other messages on parallel links.

- Reference

  [3] 3.3.1, 5.2.1, 5.3.1.4.1, 5.3.2

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure

  1. Execute the procedure in Section 4.2.1 to establish a link on the PB-ADV bearer.
  2. The Lower Tester immediately begins to periodically advertise a new Link Open message containing a different (2nd) random Link ID and the Device UUID of the IUT.
  3. The Lower Tester periodically scans for a Link Ack message containing the 2nd allocated Link ID sent in Step 2. If the Lower Tester receives a Link Ack containing the 2nd allocated link ID, the test case ends in a Fail verdict.
  4. The Lower Tester resends the Link Open message with the 1st Link ID and expects the IUT to respond with a Link Ack.
  5. The Lower Tester begins to periodically advertise a PB-ADV PDU with the 1st Link ID containing a Link Close message to terminate the link. The Link Close message field Info is set to 0b000010 and Control set to 0b11. The Lower Tester continues to advertise for at least 30 seconds to ensure the

message is received by the IUT.

6. Execute the procedure in Section 4.2.1 to establish a link on the PB-ADV bearer with a new random Link ID.
7. The Lower Tester begins to periodically advertise a Link Close message using the 3rd Link ID to terminate the link. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.
8. The Lower Tester sends a Link Open message with a new (4th) Link ID and expects the IUT to respond with a Link Ack.
9. The Lower Tester waits for 65 seconds, then it sends a Link Open message with a new (5th) Link ID and expects the IUT to respond with a Link Ack.
10. The Lower Tester sends a PB-ADV PDU containing a Provisioning Invite PDU, but using a different (6th) Link ID, and the IUT ignores the PB-ADV PDUs.
11. The Lower Tester advertises a Link Close message with the 6th Link ID, for at least 5 seconds.
12. The Lower Tester sends a PB-ADV PDU containing a Provisioning Invite PDU, using the correct (5th) Link ID, and the IUT acknowledges the PDU and continues the procedure by sending a PB-ADV PDU containing a Provisioning Capabilities PDU.
13. The Lower Tester acknowledges the PB-ADV PDU from the IUT using the incorrect (6th) Link ID, and the IUT ignores the acknowledgment and continues to retransmit the PDU.
14. After at least 5 seconds, the Lower Tester advertises a Link Close message with the 5th Link ID, and the IUT stops retransmitting the PDU from the previous step.

- Expected Outcome

  Pass verdict

  The IUT broadcasts an Unprovisioned Device beacon that is formatted correctly and contains valid values.

  The IUT sends a Link Ack message to the Lower Tester after receiving a Link Open message and establishes a link over PB-ADV. The Link Ack message is properly formatted.

  The IUT ignores the Link Open message(s) sent by the Lower Tester while the 1st link is still established and does not send a Link Ack message containing the 2nd allocated Link ID.

  The IUT responds to the same Link Open message that has opened the link (containing the same Link ID).

  The IUT successfully closes the 1st link and responds to the Link Open sent by the Lower Tester containing the 3rd allocated Link ID. The Link Ack message is properly formatted with the expected Info and Control values.

  The IUT accepts a Link Open message with a different Link ID after 60 seconds have passed since sending the Link Ack.

  The IUT ignores Link Close messages and PB-ADV PDUs (containing Provisioning PDUs or acknowledgments) with an incorrect Link ID.

MESH/NODE/PBADV/BV-03-C [Unprovisioned Device Receives and Sends PB-ADV PDU]

- Test Purpose

  Verify that an unprovisioned device IUT can send and receive PB-ADV PDUs that do not require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within a PB-ADV PDU.

- Reference

  [3] 3.3.1, 4.3.1.4, 5.2.1, 5.3.3, 5.4.1

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Node PB-ADV Link Established after execution of setup preamble 4.2.1.

- Test Procedure

  1. The Lower Tester sends a Provisioning PDU containing a Provisioning Invite. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains the Link ID set to the value used in link establishment and the Transaction Number is set to 0, and the Generic Provisioning PDU containing the Provisioning Invite message. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x00 ("Provisioning Invite"), and the Parameters field contains the Attention Duration field set to a value greater than 0.
  2. The IUT receives the Provisioning PDU and sends a Provisioning PDU containing a Provisioning Capabilities PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x01 ("Provisioning Capabilities"), and the Parameters field set to any valid Provisioning Capabilities field values.

- Expected Outcome

  Pass verdict

  The IUT receives a Provisioning PDU containing a Provisioning Invite PDU.

  The IUT sends a Provisioning PDU containing a Provisioning Capabilities PDU.

  All Mesh Provisioning AD Type messages are non-connectable.

MESH/NODE/PBADV/BV-04-C [Unprovisioned Device Receives and Sends PB-ADV PDU with Segmentation]

- Test Purpose

  Verify that an unprovisioned device IUT can send and receive PB-ADV PDUs that require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within any PB-ADV PDUs. Verify that segmented messages are transmitted in the correct order and Transaction Acknowledgment PDUs are sent when

required.

- Reference

  [3] 3.1.2, 3.3.1, 4.3.1.4, 5.2.1, 5.3.1.2, 5.3.1.3, 5.3.3, 5.4.1

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Node PB-ADV Link Established after execution of setup preamble 4.2.1.
- Test Procedure
  1. The Lower Tester sends a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid the Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the Provisioning PDU, and FCS set to the value calculated by the IUT. Each subsequent segment is transmitted as a Transaction Continuation PDU with the SegmentIndex set to the correct sequential value and the GPCF set to 0b10.
  2. Upon receiving the PDU with the SegmentIndex value of SegN from the Transaction Start PDU, the IUT reassembles the Provisioning PDU. The IUT calculates the FCS for the reassembled Provisioning PDU.
  3. The IUT compares the calculated FCS value with the FCS field value from the Transaction Start PDU. If the values match, the IUT sends a Transaction Acknowledgment PDU after a random delay between 20 ms and 50 ms with the RFU field bits set to 0, and the GPCF field set to 0b01. If the values do not match, the test case ends in failure.
  4. The IUT sends a Provisioning PDU containing a Provisioning Public Key PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values.
  5. The Lower Tester receives the Provisioning PDU, reassembles the message, and responds by sending a Transaction Acknowledgment PDU if FCS validation is successful.
- Expected Outcome

  Pass verdict

  The IUT sends a Transaction Acknowledgment PDU with the GPCF field set to 0b01.

  The IUT receives a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments, reassembles the message, and validates the FCS value. The IUT ignores any RFU bits that are set to 1.

  The IUT sends a Provisioning PDU containing a Provisioning Public Key PDU. The initial segment (Segment 0) is sent using a Transaction Start PDU. The subsequent segments are sent using Transaction Continuation PDUs. Each Provisioning PDU is the length of the negotiated full MTU for the bearer (23

bytes), except the final segment which may be smaller. The number of segments received by the IUT matches the SegN value from the Transaction Start PDU.

All Mesh Provisioning AD Type messages are non-connectable.

Each Provisioning PDU is the length of the negotiated full MTU (23 bytes) for the bearer, except the final segment which may be smaller.

MESH/PVNR/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close – Provisioner]

- Test Purpose

  Verify that a Provisioner IUT can establish a PB-ADV provisioning bearer link with an unprovisioned device. Verify that the IUT sends correctly formatted PB-ADV packets.

- Reference

  [3] 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester emulates an unprovisioned device.
  - The IUT has been triggered to provision the Lower Tester.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].

- Test Procedure

  1. The Lower Tester sends Unprovisioned Device beacons with the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value set to the IUT's Device UUID, and the OOB Information field has the bit set to a non-RFU value supported by the manufacturer.
  2. The IUT is induced to scan for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the Lower Tester.
  3. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning». The PB-ADV PDU containing the Link Open message has a Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the Lower Tester as supplied in the IXIT [6].
  4. The Lower Tester detects the PB-ADV PDU advertised by the IUT.
  5. The Lower Tester periodically advertises a Link Ack message containing the newly allocated Link ID sent in Step 1. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.
  6. The IUT is induced to send a Link Close message to terminate the link. The message is sent at least three times to ensure the message is received by the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT and the Lower Tester establish a link on the PB-ADV bearer.

  The IUT advertises a PB-ADV PDU containing a Link Open message. The GPCF field is set to 0b11.

  All bits of the Transaction Number field are set to 0 in each PB-ADV PDU transmitted and sent by the IUT. All PB-ADV PDUs are formatted correctly with the correct values.

  The IUT sends a properly formatted Link Close message with the Reason field set to a valid value different than 0x00.

MESH/PVNR/PBADV/BV-02-C [Provisioner Sends and Receives PB-ADV PDU]

- Test Purpose

  Verify that a Provisioner IUT can send and receive PB-ADV PDUs that do not require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within a PB-ADV PDU.

- Reference

  [3] 3.1.2, 3.3.1, 4.3.1.4, 5.2.1, 5.3.3, 5.4.1, 5.4.2

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Provisioner PB-ADV Link Established after execution of setup preamble 4.2.2.

- Test Procedure

  1. The IUT is induced to send a Provisioning PDU containing a Provisioning Invite PDU. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains the Link ID set to the value used in link establishment and the Transaction Number is set to 0, and the Generic Provisioning PDU containing the Provisioning Invite message. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x00 ("Provisioning Invite"), and the Parameters field contains the Attention Duration field set to a value greater than 0.
  2. The Lower Tester receives the Provisioning PDU and sends a Provisioning PDU containing a Provisioning Capabilities PDU in response. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x01 ("Provisioning Capabilities"), and the Parameters field set to valid Provisioning Capabilities field values.

- Expected Outcome

  Pass verdict

The IUT sends the Provisioning PDU containing the Provisioning Invite PDU.

The IUT receives the Provisioning PDU containing the Provisioning Capabilities PDU.

All Mesh Provisioning AD Type messages are non-connectable.

MESH/PVNR/PBADV/BV-03-C [Provisioner Sends and Receives PB-ADV PDU with Segmentation]

- Test Purpose

  Verify that a Provisioner IUT can send and receive PB-ADV PDUs that require segmentation. Verify the formatting and values of the Mesh Provisioning AD Type, any PB-ADV PDUs contained in this AD Type, and any Generic Provisioning PDUs contained within any PB-ADV PDUs. Verify that segmented messages are transmitted in the correct order and Transaction Acknowledgment PDUs are sent when required.

- Reference

  [3] 3.3.1, 5.2.1, 5.3.1, 5.3.3, 5.4.1, 5.4.2

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Provisioner PB-ADV Link Established after execution of setup preamble 4.2.2.

- Test Procedure

  1. The Upper Tester sends a command to the IUT to send a Provisioning PDU containing a Provisioning Public Key PDU in multiple segments. The Generic Provisioning PDU contains two RFU bits set to 0, the Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the Provisioning PDU, and FCS set to the value calculated by the IUT.
  2. The Lower Tester receives the Provisioning PDU, reassembles the message, and responds by sending a Transaction Acknowledgment PDU if FCS validation is successful.
  3. The Lower Tester sends a Provisioning PDU containing a Provisioning Public Key PDU to the IUT. The Provisioning PDU contains a 2-bit RFU field with all bits set to 0, a 6-bit Type field set to 0x03 ("Provisioning Public Key"), and the Parameters field set to valid Provisioning Public Key field values. The first segment of the message is a Transaction Start PDU with SegN set to the last segment number, GPCF set to 0b00, TotalLength set to the number of octets in the Provisioning PDU, and FCS set to the value calculated by the IUT. Each subsequent segment is transmitted as a Transaction Continuation PDU with the SegmentIndex set to the correct sequential value and the GPCF set to 0b10.
  4. Upon receiving the Transaction Continuation PDU with the SegmentIndex value equal to the value of SegN in the Transaction Start PDU, the IUT reassembles the Provisioning PDU. The IUT calculates the FCS for the Provisioning PDU once all segments are received.

5. The IUT compares the calculated FCS to the FCS field value from the Transaction Start PDU. If the values match, the IUT sends a Transaction Acknowledgment PDU after a random delay between 20 ms and 50 ms with all bits of the RFU field set to 0 and the GPCF field set to 0b01. If the values do not match, the test case ends in failure.

- Expected Outcome

  Pass verdict

  The IUT sends the Provisioning PDU containing the Provisioning Public Key PDU. The initial segment (Segment 0) is sent using a Transaction Start PDU. The subsequent segments are sent using Transaction Continuation PDUs. Each Provisioning PDU is the length of the negotiated full MTU for the bearer (23 bytes), except the final segment which may be smaller. All segments are sent in sequence.

  The IUT receives the Provisioning PDU containing the Provisioning Capabilities PDU in multiple segments, reassembles the message, and validates the FCS value.

  The IUT sends a Transaction Acknowledgment PDU with the RFU field bits set to 0 and the GPCF field set to 0b01 after calculating the FCS.

  All Mesh Provisioning AD Type messages are non-connectable.

MESH/PVNR/PBADV/BI-01-C [PB-ADV Link Establishment Timeout]

- Test Purpose

  Verify that a Provisioner IUT properly handles a timeout while establishing a PB-ADV link.

- Reference

  [3] 3.3.1, 5.2.1, 5.2.4, 5.3.3, 5.4.4

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The IUT has been triggered to provision the Lower Tester.
  - The Device UUID for the Lower Tester has been supplied by the manufacturer via IXIT [6].

- Test Procedure

1. The IUT is induced to periodically advertise a PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents and the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU contains a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU field contains the Provisioning Bearer Control PDU containing the Link Open message with the BearerOpcode set to 0x00, the GPCF field to 0b11, and the Parameters set to the Device UUID of the Lower Tester.
2. The Lower Tester detects the PB-ADV PDU advertised by the IUT.

3. The Lower Tester does not send a Link Ack message to the IUT within the timeout period (at least 60 seconds).
4. The Lower Tester does not receive a Provisioning Failure PDU from the IUT after the timeout period is exceeded.

- Expected Outcome

  Pass verdict

  The IUT transmits a PB-ADV PDU containing a Link Open message. All fields are correctly formatted and contain valid values.

  The IUT does not send a Provisioning Failure PDU after the timeout is exceeded without receiving a Link Ack message from the Lower Tester.

MESH/NODE/PBADV/BI-01-C [Ignore Link Open Attempts with Incorrect Device UUID]

- Test Purpose

  Verify that an unprovisioned device IUT ignores Link Open attempts containing the incorrect Device UUID and only sends a Link Ack when the Device UUID matches.

- Reference

  [3] 3.3.1, 5.2.1. 5.3.1.4

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The IUT is in the unprovisioned state.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure
  1. The Upper Tester sends a command to the IUT to send Unprovisioned Device beacons with the following field values: the Beacon Type set to 0x00 (Unprovisioned Device), the Device UUID value as supplied in the IXIT [6], and the OOB Information field and URI Hash field set to valid values.
  2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
  3. The Lower Tester attempts to execute setup preamble 4.2.2 using a Device UUID different than the Device UUID contained in the Unprovisioned Device beacon. The Lower Tester expects the setup to fail with the IUT not sending a Link ACK message.
  4. The Lower Tester periodically scans for 100 ms for a Link Ack message containing the 1st allocated Link ID sent in Step 3. If the Lower Tester receives a Link Ack containing the 1st allocated link ID, the test case ends in a Fail verdict.
  5. The Lower Tester executes the procedure setup preamble 4.2.2 using the Device UUID in the IUT's Unprovisioned Device beacons.

6. The Lower Tester begins to periodically advertise a Link Close message using the 2nd Link ID to terminate the link. The Lower Tester continues to advertise for at least 5 seconds to ensure the message is received by the IUT.

- Expected Outcome

  Pass verdict

  The IUT broadcasts an Unprovisioned Device beacon containing its Device UUID.

  The IUT ignores the Link Open message(s) sent by the Lower Tester with the incorrect Device UUID and does not send a Link Ack message containing the 1st allocated Link ID.

  The IUT responds to the Lower Tester's Link Open message containing the correct Device UUID with a Link Ack message establishing the PB-ADV link. The PB-ADV PDU contains the expected 2nd Link ID and Transaction Number.

MESH/NODE/PBADV/BI-02-C [Ignore Provisioning Bearer Control PDU with RFU BearerOpcode Value]

- Test Purpose

  Verify that an unprovisioned device IUT will ignore a message that has the BearerOpcode field set to an RFU value.

- Reference

  [3] 3.1.2, 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure

  1. The Lower Tester begins advertising a PB-ADV PDU containing a random Link ID and the Transaction Number set to 0. The Generic Provisioning PDU field contains a Provisioning Bearer Control PDU, with the BearerOpcode set to an RFU value, the GPCF field to 0b11, and the Parameters set to the Device UUID of the IUT.
  2. The Upper Tester confirms that the IUT receives the message from the Lower Tester.
  3. The Lower Tester does not expect any message in response from the IUT.

- Expected Outcome

  Pass verdict

  The IUT does not send any message in response to detecting the PDU from the Lower Tester.

MESH/NODE/PBADV/BI-03-C [Receive Link Close Message with RFU Reason Value]

- Test Purpose

  Verify that an unprovisioned device IUT will accept a Link Close message that has the Reason field set to an RFU value.

- Reference

  [3] 3.1.2, 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
- Test Procedure
  1. The steps in setup preamble 4.2.1 are executed to establish a link over the advertising bearer.
  2. The Lower Tester begins to periodically advertise a Link Close message with the Reason field set to any RFU value. The Lower Tester sends this message until the IUT reports that the PB-ADV link has been closed.
- Expected Outcome

  Pass verdict

  In Step 2, The IUT closes the PB-ADV link.

MESH/NODE/PBADV/BI-04-C [Ignore Connectable Advertisement using Mesh Provisioning AD Type]

- Test Purpose

  Verify that an unprovisioned device IUT can ignore a connectable advertisement using the Mesh Provisioning AD Type.

- Reference

  [3] 3.3.1, 5.2.1, 5.3.1.4, 5.3.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
- Test Procedure
  1. The IUT is induced to send Unprovisioned Device beacons with the Device UUID value set to the

IUT's Device UUID.

2. The Lower Tester scans for Unprovisioned Device beacons until it receives a beacon matching the Device UUID of the IUT.
3. The Lower Tester begins to periodically advertise a connectable PB-ADV PDU containing a Link Open message. The mesh packet containing the PB-ADV PDU has the Length field set to the length of the AD Type and Contents, the AD Type set to «Mesh Provisioning», and the Contents set to the PB-ADV PDU. The PB-ADV PDU containing the Link Open message has the Link ID value set to a newly allocated random Link ID, the Transaction Number set to 0, and the Generic Provisioning PDU containing the Link Open message. The Generic Provisioning PDU has a BearerOpcode value of 0x00, the GPCF field is set to 0b11, and the Link Open message parameters contain the Device UUID field equal to the Device UUID of the IUT.

- Expected Outcome

  Pass verdict

  The IUT ignores the connectable advertisement sent by the Lower Tester.

4.4 Mesh Provisioning Service

The test group objective is to the Mesh Provisioning Service. This includes tests covering GATT requirements of the Mesh Provisioning Service and functionality related to segmentation and reassembly for Proxy PDUs received and sent by the Mesh Provisioning Service.

MESH/NODE/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning]

- Test Purpose

  Verify that an unprovisioned device that supports GATT provisioning indicates this in an advertisement with a List of Service UUIDs that includes the Mesh Provisioning Service. Verify that the advertising payload Service Data is properly formatted with the expected Device UUID. Verify that a connectable advertisement is utilized.

- Reference

  [3] 3.3.2, 4.3.1.4, 4.3.1.5, 5.2.2, 5.4.1, 5.4.2, 7.1.2.1, 7.1.2.2.1

- Initial Condition
  - The IUT is an unprovisioned device.
  - The IUT supports PB-GATT provisioning.
  - The IUT is in either Limited Discoverable mode or General Discoverable mode.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
  - The Lower Tester is a Provisioner.
  - Support for optional AD Types is supplied by the manufacturer via IXIT [6].
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and

the Lower Tester.

- Test Procedure

  1. The Upper Tester orders the IUT to advertise the Mesh Provisioning Service as a «Primary Service» in connectable advertising PDUs. The advertisements include the following AD Types: a 3-octet «Flags» type, either a 4-octet «Incomplete List of 16-bit Service UUIDs» type or a 4-octet «Complete List of 16-bit Service UUIDs» type that includes «Mesh Provisioning Service», and a 22-octet «Service Data» type, including the Device UUID and OOB Information. Any of the following optional AD Types are included if manufacturer support is indicated: URI, Appearance, TX Power Level, Local Name.
  2. The Lower Tester receives the advertisements from the IUT.

- Expected Outcome

  Pass verdict

  The IUT sends connectable service advertisements containing the Mesh Provisioning Service UUID.

  The included Device UUID matches the expected value for the IUT.

  The IUT's service advertisement uses a connectable advertising PDU and contains the expected AD Types as described in Step 1. The list of Service Class UUIDs contains a Service UUID of «Mesh Provisioning Service». The Service Data field must contain the expected Device UUID and OOB Information field values.

MESH/NODE/MPS/BV-02-C [PB-GATT Link Establishment and Link Close]

- Test Purpose

  Verify that an unprovisioned device IUT can establish a PB-GATT provisioning bearer link with a Provisioner.

- Reference

  [3] 3.3.2, 5.2.2

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].

- Test Procedure

  1. The steps in Setup Preamble 4.2.3 are executed using the IUT's Device UUID.
  2. The Lower Tester terminates the ATT bearer connection.

- Expected Outcome

  Pass verdict

  The IUT advertises the Mesh Provisioning Service in connectable advertising PDUs containing its Device UUID.

The Lower Tester can establish an ATT bearer connection with the IUT, discover the Provisioning Data characteristic, and configure it for notifications.

MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device]

- Test Purpose

  Verify that an unprovisioned device can support discovery of the Mesh Provisioning Service as a Primary Service. Also, verify that the Mesh Proxy Service is not discoverable.

- Reference

  [3] 3.3.2, 5.2.2, 7.1, 7.1.2.1

- Initial Condition
  - The IUT is an unprovisioned device.
  - The IUT is in either Limited Discoverable mode or General Discoverable mode.
  - The Lower Tester is a Provisioner.
  - The IUT and the Lower Tester support the GAP Peripheral and GAP Central roles.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- Test Procedure
  1. The Lower Tester executes either the procedure in GATT.TS [5] Discover Primary Services by Service UUID - from Server, GATT/SR/GAD/BV-02-C, with the Service UUID of «Mesh Provisioning Service» or the procedure in GATT.TS [5] Discover All Primary Services - from Server, GATT/SR/GAD/BV-01-C.
  2. The Lower Tester executes either the procedure in GATT.TS [5] Discover Primary Services by Service UUID - From Server, GATT/SR/GAD/BV-02-C, with the Service UUID of «Mesh Proxy Service» or the procedure in GATT.TS [5] Discover All Primary Services - from Server, GATT/SR/GAD/BV-01-C.

- Expected Outcome

  Pass verdict

  The IUT returns one attribute handle range containing the starting handle and the ending handle of the instantiation of the Mesh Provisioning Service. The Attribute Type in that service declaration is «Primary Service».

  The IUT returns zero attribute handle ranges in response to discovery requests of the Mesh Proxy Service.

4.4.1 Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device

This test group is for generic use and contains one or more test cases to verify compliant operation in response to characteristic discovery as IUT acting as an unprovisioned device. The verification is done one value at a

time, as enumerated in the test cases in Table 4.2 below, using this generic test procedure.

- Reference

  [3] 3.3.2, 7.1.2, 7.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The Lower Tester has discovered the Mesh Provisioning Service and has saved the handle range for an instantiation of the Mesh Provisioning Service that exposes the characteristic specified in Table 4.2 (e.g., by executing test case MESH/NODE/MPS/BV-03-C [Expose Mesh Provisioning Service – Unprovisioned Device]).
- Test Procedure
  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service, GATT/SR/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Provisioning Service, or Discover Characteristic by UUID, GATT/SR/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Provisioning Service and UUID set to the Characteristic UUID in Table 4.2. In the selected procedure, only one pass is needed with the server database defined in Initial Condition.

Test Case	Characteristic UUID
MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device]	«Mesh Provisioning Data Out»
MESH/NODE/MPS/BV-05-C [Expose Mesh Provisioning Data In Characteristic – Unprovisioned Device]	«Mesh Provisioning Data In»

Table 4.2: Mesh Provisioning Service Characteristic Discovery as Unprovisioned Device test cases

- Expected Outcome

  Pass verdict

  One attribute handle/value pair is returned containing the specified Characteristic UUID.

MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device]

- Test Purpose

  Verify that an unprovisioned device IUT allows discovery of the Mesh Provisioning Data Out Client Characteristic Configuration descriptor.

- Reference

  [3] 3.3.2, 7.1.2, 7.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The IUT has discovered the attribute handle/value pair of the Mesh Provisioning Data Out characteristic (e.g., by executing test case MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device]).

- Test Procedure

  The Lower Tester executes one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors - from Server, GATT/SR/GAD/BV-06-C using the specified handle range.

- Expected Outcome

  Pass verdict

  The IUT returns one attribute handle/value pair containing the UUID «Client Characteristic Descriptor».

  The characteristic descriptor does not require authentication or encryption for access.

MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device]

- Test Purpose

  Verify that an unprovisioned device IUT can have notifications enabled for the Mesh Provisioning Data Out characteristic by a Provisioner.

- Reference

  [3] 3.3.2, 7.1.2, 7.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The handle of the Mesh Provisioning Data Out characteristic is known to the Lower Tester by executing the test procedure in MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device] or is known to the IUT by other means.
  - The handle of the Mesh Provisioning Data Out Client Characteristic Configuration descriptor has been previously discovered by the IUT by executing the test procedure in MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned

Device] or is known to the IUT by other means.

- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- Test Procedure

  1. The Lower Tester disables notification by sending an *ATT_Write_Request* with the handle of the Mesh Provisioning Data Out characteristic and the value 0x0000.
  2. The Lower Tester sends a correctly formatted *ATT_Write_Request* (Code = 0x12) with the handle of the characteristic descriptor and a value of 0x0001 to the Lower Tester.

- Expected Outcome

  Pass verdict

  The characteristic descriptor is successfully written.

MESH/NODE/MPS/BV-08-C [Receive Mesh Provisioning Data Out Notifications – Unprovisioned Device]

- Test Purpose

  Verify that an unprovisioned device IUT can send notifications of the Mesh Provisioning Data Out characteristic.

- Reference

  [3] 3.3.2, 5.2.2, 7.1.2, 7.1.3

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The handle of the Mesh Provisioning Data Out characteristic is known to the Lower Tester by executing the test procedure in MESH/NODE/MPS/BV-04-C [Expose Mesh Provisioning Data Out Characteristic – Unprovisioned Device] or is known to the IUT by other means.
  - The handle of the Client Characteristic Configuration descriptor of the Mesh Provisioning Data Out characteristic has been previously discovered by the IUT by executing the test procedure in MESH/NODE/MPS/BV-06-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor – Unprovisioned Device] or is known to the IUT by other means.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- Test Procedure

  1. The Lower Tester configures the IUT for notifications (e.g., by executing test case MESH/NODE/MPS/BV-07-C [Enable Notifications for Mesh Provisioning Data Out – Unprovisioned Device]).

2. The Lower Tester sends a Provisioning message to the IUT (e.g., a Provisioning Invite PDU) and the IUT responds with a Provisioning message appropriate for the message received (e.g., a Provisioning Capabilities PDU) using Handle Value Notification ATT PDU.

- Expected Outcome

  Pass verdict

  The IUT responds to a Provisioning message from the Lower Tester with a Provisioning message after notifications are enabled for Mesh Provisioning Data Out.

  Characteristics are transmitted in "little endian".

MESH/NODE/MPS/BV-09-C [Unprovisioned Device Sends and Receives Complete Provisioning PDUs]

- Test Purpose

  Verify that an unprovisioned device IUT can send and receive complete Provisioning PDUs within a single Handle Value Notification ATT PDU.

- Reference

  [3] 3.3.2, 5.2.2, 5.3.1, 5.3.3, 7.1.2, 7.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The IUT is running the Mesh Provisioning Service.
  - The Lower Tester is a Provisioner.
  - The IUT is in State: Node PB-GATT Link Established after execution of setup preamble 4.2.3.

- Test Procedure

  1. The Lower Tester writes to the Mesh Provisioning Data In characteristic exposed by the IUT to send a single Proxy PDU containing a complete Provisioning PDU in a single Handle Value Notification smaller than the negotiated ATT_MTU to the IUT (e.g., by executing the procedure in GATT.TS [5] GATT/SR/GAW/BV-01-C with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU containing a Provisioning PDU). The SAR field is set to 0b00 (Complete higher level message), the MessageType field is set to 0x03 (Provisioning PDU), and the Data field contains the Provisioning PDU Parameters for the Provisioning PDU sent.
  2. The IUT receives the message.
  3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data Out characteristic exposed by the Lower Tester to send a Provisioning PDU smaller than the negotiated ATT_MTU (e.g., by executing the procedure in GATT.TS [5]. GATT/CL/GAW/BV-01-C). The Proxy PDU containing the Provisioning PDU has the SAR field set to 0b00 ("Complete higher level message"), the MessageType field set to 0x03 ("Provisioning PDU"), and a complete proxy message to the Lower Tester.
  4. The Lower Tester receives a Proxy Protocol message from the IUT with the exact data contained in the Proxy Protocol message sent from the IUT.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT receives a single Proxy PDU message in one Handle Value Notification ATT PDU.

  All values received in the Handle Value Notification and Proxy PDU are formatted correctly and contain valid values.

  In Step 3, The IUT sends the message using one Handle Value Notification ATT PDU containing a single Proxy message to the Lower Tester with SAR field set to b00 ("Complete higher level message"), RFU set to zero, MessageType equal to 0x3 ("Provisioning PDU"), and the complete Proxy message.

MESH/NODE/MPS/BV-10-C [Unprovisioned Device Sends and Receives Segmented Provisioning PDU Message]

- Test Purpose

  Verify that an unprovisioned device IUT can send and receive a Provisioning PDU that is segmented into multiple Proxy PDUs when the required Proxy PDU size is larger than the negotiated ATT_MTU.

- Reference

  [3] 5.2.2, 5.3.3, 5.4.1, 7.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT has been supplied by the manufacturer via IXIT [6].
  - The IUT is in State: Node PB-GATT Link Established after execution of setup preamble 4.2.3.

- Test Procedure
  1. The Lower Tester writes to the Mesh Provisioning Data In characteristic exposed by the IUT to send a Provisioning PDU that is larger than the negotiated ATT_MTU to the IUT. The Provisioning PDU is split into multiple Proxy PDUs and transmitted via multiple Handle Value Notification ATT PDUs.
  2. The IUT receives the complete Provisioning PDU and reassembles the fragments before processing the message. The Upper Tester receives confirmation from the IUT that the message was received.
  3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data Out characteristic exposed by the Lower Tester to send a Provisioning PDU larger than the negotiated ATT_MTU.
  4. The Lower Tester receives the Provisioning PDU segmented in multiple Proxy PDUs. The first Proxy PDU has the SAR field set to b01 ("Data field contains the first segment of a message"), the MessageType field set to 0x03 ("Provisioning PDU"), and the Data field filled with the first segment of the Provisioning PDU message. Each intermediate Proxy PDU that is not the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with each subsequent segment of the Provisioning PDU. The Proxy PDU containing the final segment of the Provisioning PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message. (For example, Provisioning Public Key PDU. If

necessary, start with Provisioning Invite PDU, Provisioning Capabilities PDU, Provisioning Start PDU as in the generic test procedure under Provisioning Procedure - Unprovisioned Device).

- Expected Outcome

  Pass verdict

  In Step 2, the IUT receives the complete Provisioning PDU in multiple Handle Value Notification ATT PDUs.

  Each Proxy PDU received by the IUT is prepended with a 1-octet header containing the expected SAR value and MessageType of 0x03. The first fragment received has the SAR field set to 0b01, the last fragment received has the SAR field set to 0b11, and any intermediate fragments have the SAR field set to 0b10.

  In Step 3, the IUT sends the complete Provisioning PDU segmented into multiple Proxy PDUs to the Lower Tester. Each Proxy PDU contains the expected fields and values. Each segment is sent using a Handle Value Notification ATT PDU.

MESH/PVNR/MPS/BV-01-C [Advertising Unprovisioned State for GATT Provisioning – Provisioner]

- Test Purpose

  Verify that a Provisioner IUT can receive Mesh Provisioning Service advertisements.

- Reference

  [3] 3.3.2, 5.2.2, 5.4.1, 5.4.2, 7.1.2.1, 7.1.2.2.1

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester is in either Limited Discoverable mode or General Discoverable mode.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service.

- Test Procedure
  1. The Lower Tester advertises the Mesh Provisioning Service in connectable advertising PDUs. The connectable advertising PDUs contain the following AD Types: Flags, List of Service Class UUIDs, and Service Data. The List of Service Class UUIDs contains the Mesh Provisioning Service UUID. The Service Data field contains the Lower Tester’s 16-octet Device UUID.
  2. The Upper Tester expects the IUT to report the discovery of an unprovisioned device with the Lower Tester’s Device UUID.

- Expected Outcome

  Pass verdict

  The IUT reports the discovery of an unprovisioned device with the Lower Tester’s Device UUID to the Upper Tester.

MESH/PVNR/MPS/BV-02-C [PB-GATT Link Establishment and Link Close – Provisioner]

- Test Purpose

  Verify that a Provisioner IUT can establish a PB-GATT provisioning bearer link with an unprovisioned device. Verify that the IUT sends correctly formatted PB-GATT packets.

- Reference

  [3] 3.3.2, 5.2.2

- Initial Condition

  - The IUT is a Provisioner.
  - The Lower Tester emulates an unprovisioned device.
  - The IUT has been triggered to provision the Lower Tester.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service.

- Test Procedure

  1. The Lower Tester advertises the Mesh Provisioning Service in connectable advertising PDUs. The IUT establishes an LE connection to the Lower Tester.
  2. The IUT is induced to discover the handles for the Mesh Provisioning Data In characteristic and the Mesh Provisioning Data Out characteristic exposed by the Mesh Provisioning Service on the Lower Tester by executing the procedure in GATT.TS [5] Discover All Characteristics of a Service - by Server, GATT/SR/GAD/BV-04-C.
  3. The IUT is induced to send an ATT_Write_Request to the Lower Tester with the handle of the Mesh Provisioning Data Out characteristic and the value set to 0x0001 to enable notifications.
  4. The Lower Tester terminates the ATT bearer connection.

- Expected Outcome

  Pass verdict

  The IUT successfully establishes an ATT bearer connection with the Lower Tester, discovers the Provisioning Data characteristic, and configures it to enable notifications.

  The ATT bearer connection is terminated between the IUT and the Lower Tester.

4.4.2 Mesh Provisioning Service Characteristic Discovery by Provisioner

This test group is for generic use and contains one or more test cases to verify compliant operation in response to characteristic discovery by an IUT acting as a Provisioner. The verification is done one value at a time, as enumerated in the test cases in Table 4.3, using this generic test procedure.

- Test Purpose

  Verify that a Provisioner IUT can discover characteristics exposed by the Mesh Provisioning Service on an unprovisioned device.

- Reference

  [3] 3.3.2, 7.1.3.1, 7.1.3.2

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service including all defined characteristics.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The IUT has previously discovered the Mesh Provisioning Service and has saved the handle range for an instantiation of the Mesh Provisioning Service that exposes the characteristic specified in Table 4.3 by executing the procedure in GATT.TS [5] GATT/CL/GAD/BV-02-C with the service UUID of «Mesh Provisioning Service» or by other means.
- Test Procedure
  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service - by Client, GATT/CL/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Provisioning Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Provisioning Service and UUID set to the value in Table 4.3. In the selected procedure, only one pass is needed with the server database defined in Initial Condition.

Test Case	Characteristic UUID
MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner]	«Mesh Provisioning Data Out»
MESH/PVNR/MPS/BV-04-C [Discover Mesh Provisioning Data In Characteristic]	«Mesh Provisioning Data In»

Table 4.3: Mesh Provisioning Service Characteristic Discovery by Provisioner test cases

- Expected Outcome

  Pass verdict

  One attribute handle/value pair is returned containing the expected characteristic UUID.

MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner]

- Test Purpose

  Verify that a Provisioner IUT can discover the Mesh Provisioning Data Out Client Characteristic Configuration descriptor on an unprovisioned device.

- Reference

  [3] 3.3.2, 7.1.2, 7.1.3

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service in which the Mesh Provisioning Data Out characteristic and its associated Client Characteristic Configuration descriptor are exposed.
  - The IUT has discovered the handle range of the Mesh Provisioning Data Out characteristic, by executing the procedure in MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] or by other means.

- Test Procedure

  The Upper Tester orders the IUT to execute one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors, GATT/CL/GAD/BV-06-C using the specified handle range for Mesh Provisioning Data Out, with the server database defined in Initial Condition.

- Expected Outcome

  Pass verdict

  One attribute handle/value pair is returned containing the UUID «Client Characteristic Descriptor».

  The Lower Tester does not require authentication or encryption to access the characteristic descriptor.

MESH/PVNR/MPS/BV-06-C [Enable Notifications for Mesh Provisioning Data Out - Provisioner]

- Test Purpose

  Verify that a Provisioner IUT can enable notifications to the Mesh Provisioning Data Out Client Characteristic Configuration Descriptor.

- Reference

  [3] 3.3.2, 7.1.2, 7.1.3

- Initial Condition
  - The Lower Tester has one instantiation of the Mesh Provisioning Service in which the Mesh Provisioning Data Out characteristic and its associated Client Characteristic Configuration descriptor are exposed.
  - The handle of the Mesh Provisioning Data Out characteristic has been previously discovered by the

IUT by executing the test procedure in MESH/PVNR/MPS/BV-03-C [Discover Mesh Provisioning Data Out Characteristic - Provisioner] or is known to the IUT by other means.

- The handle of the Client Characteristic Configuration descriptor has been previously discovered by the IUT by executing the test procedure in MESH/PVNR/MPS/BV-05-C [Discover Mesh Provisioning Data Out Client Characteristic Configuration Descriptor - Provisioner] or is known to the IUT by other means.
- Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- Test Procedure

  1. The Upper Tester orders the IUT to send a correctly formatted *ATT_Write_Request* (Code = 0x12) with the handle of the characteristic descriptor and a value of 0x0001 to the Lower Tester.

- Expected Outcome

  Pass verdict

  The characteristic descriptor is successfully written.

MESH/PVNR/MPS/BV-08-C [Provisioner Sends and Receives Complete Provisioning PDUs]

- Test Purpose

  Verify that a Provisioner IUT can send and receive provisioning data contained in a single Provisioning PDU.

- Reference

  [3] 3.2.2, 5.2.2, 5.3.1, 5.3.3, 7.1.3

- Initial Condition

  - The IUT is a Provisioner.
  - The IUT is running the Mesh Provisioning Service.
  - The Lower Tester is an unprovisioned device.
  - The Lower Tester has one instantiation of the Mesh Provisioning Service including all defined characteristics.
  - The IUT is in State: Provisioner PB-GATT Link Established after execution of setup preamble 4.2.4.

- Test Procedure

  1. The Lower Tester writes to the Mesh Provisioning Data Out characteristic exposed by the IUT to send a Provisioning PDU smaller than the negotiated ATT_MTU (e.g., by executing the procedure in GATT.TS [5] GATT/CL/GAW/BV-01-C, Write Without Response - by Client with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU containing a Provisioning PDU). The Proxy PDU containing the Provisioning PDU has the SAR field set to 0b00 ("Complete higher level message"), the MessageType field set to 0x03 ("Provisioning

PDU"), and the Data field set to the complete Provisioning PDU message.
2. The Upper Tester confirms that the IUT receives the message from the Lower Tester.
3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data In characteristic exposed by the Lower Tester to send a single Proxy PDU containing a complete Provisioning PDU in a single Handle Value Notification. The SAR field is set to 0b00 (Complete higher level message), the MessageType field is set to 0x03 (Provisioning PDU), and the Data field contains the Provisioning PDU Parameters for the Provisioning PDU sent.
4. The Lower Tester receives the complete message from the IUT in a single PDU.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT receives a single Proxy PDU containing a complete Provisioning PDU from the Lower Tester with the expected values.

  In Step 3, the IUT sends a single Proxy PDU containing the entire Provisioning PDU to the Lower Tester in one Handle Value Notification ATT PDU. All fields are formatted correctly and contain valid values.

MESH/PVNR/MPS/BV-09-C [Provisioner Sends and Receives Segmented Message]

- Test Purpose

  Verify that a Provisioner IUT can send and receive a message when the complete message does not fit into one PDU.

- Reference

  [3] 5.2.2, 5.3.3, 5.4.1, 7.1.3

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - The IUT is in State: Provisioner PB-GATT Link Established (4.2.4).

- Test Procedure

  1. The Lower Tester writes to the Mesh Provisioning Data Out characteristic exposed by the IUT to send a Provisioning PDU larger than the negotiated ATT_MTU. The Provisioning PDU is segmented into multiple Proxy PDUs. The first Proxy PDU sent has the SAR field set to b01 ("Data field contains the first segment of a message"), the MessageType field set to 0x3 ("Provisioning PDU"), and the Data field containing the first segment of the Provisioning PDU message. Each intermediate Proxy PDU that is not the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with each subsequent segment of the Provisioning PDU. The Proxy PDU containing the final segment of the Provisioning PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message. (For example, Provisioning Public Key PDU. If necessary, start with Provisioning Invite PDU, Provisioning Capabilities PDU, Provisioning Start PDU as in the generic test procedure under Provisioning

Procedure - Provisioner).

2. The Upper Tester confirms that the IUT receives all Proxy PDUs from the Lower Tester and that the reassembled message is valid.
3. The Upper Tester orders the IUT to write to the Mesh Provisioning Data In characteristic exposed by the Lower Tester to send a Provisioning PDU message larger than the negotiated MTU.
4. The Lower Tester receives all Proxy PDUs and reassembles the expected message from the segments.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT receives all Proxy PDUs containing the segmented Provisioning PDU message and reassembles the expected message.

  In Step 3, the IUT sends the complete Provisioning PDU message segmented among multiple Proxy PDUs. The initial Proxy PDU has the SAR field set to 0b01 ("Data field contains the first segment of a message"), the MessageType field set to 0x03 ("Provisioning PDU"), and the first segment of the Provisioning PDU message. Each subsequent Proxy PDU that does not contain the final segment has the SAR field set to 0b10, the MessageType field set to 0x03, and the Data field filled with that segment of the Provisioning PDU message. The final Proxy PDU has the SAR field set to 0b11, the MessageType field set to 0x03, and the Data field containing the end of the Provisioning PDU message.

MESH/NODE/MPS/BI-01-C [Mesh Provisioning Data In Characteristic Only Supports Proxy PDU containing Provisioning PDU]

- Test Purpose

  Verify that the Mesh Provisioning Data In characteristic only supports Proxy PDU type messages that contain Provisioning PDUs and does not support other Proxy PDU type messages.

- Reference

  [3] 7.1.3.1.1

- Initial Condition
  - The IUT is an unprovisioned device.
  - The IUT is running the Mesh Provisioning Service.
  - The Lower Tester is a Provisioner.
  - The IUT is in State: Node PB-GATT Link Established (4.2.3).

- Test Procedure

  The procedure in GATT.TS [5] GATT/SR/GAW/BV-01-C, Write Without Response - to Server, is performed with the handle of the Mesh Provisioning Data In characteristic and the value containing a properly formatted Proxy PDU type that does not contain a Provisioning PDU.

- Expected Outcome

  Pass verdict

  The IUT does not accept the Proxy PDU written to the Mesh Provisioning Data In characteristic.

4.5 Provisioning Procedure

The test group objective is to verify the provisioning procedure. Provisioning relies on a common provisioning protocol and common procedures that may be transported by multiple different provisioning bearers. To leverage the common aspects of these procedures, the bearer-specific test cases have been isolated from the common procedures.

This section contains tests which are common to all provisioning bearers and that test the overall provisioning protocol and procedures used over all supported bearers. These tests are written to be bearer-agnostic, and are executed using each supporter bearer for the IUT.

4.5.1 Provisioning Procedure - Unprovisioned Device

- Test Purpose

  This is a generic procedure to verify the behavior of an unprovisioned device in the Provisioning procedure. The verification is done for one combination of authentication values at a time, as enumerated in the test cases in Table 4.4, using this generic procedure.

- Reference

  [3] 5.1, 5.3.3, 5.4.1, 5.4.2, 5.4.3

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - Field values of the Provisioning Capabilities PDU are specified in the IXIT [6].

- Test Procedure

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
  3. The Lower Tester receives a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT. The Number of Elements field is set to a value between 0x01-0xFF and the Algorithms field has bits 0 and 1 set only for the supported algorithms and all other bits set to 0. The rest of the fields are set as follows:
     a. Public Key Type field bit 0 is set to 0 (zero) if Public Key Available OOB is "No", and set to 1 otherwise. If Public Key Available OOB is “Yes”, public_key and private_key are supplied in the IXIT [6].

b. If the Authentication Method is:
   i. Output OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size field is set to a non-RFU value greater than 0x00 in the range indicated in Table 4.4, the Output OOB Action field bit is set to the actions supported by the IUT, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.
   ii. Input OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size is set to 0x00, the Output OOB Action is set to 0x00, the Input OOB Size is set to a non-RFU value greater than 0x00 in the range indicated in Table 4.4 and the Input OOB Action field bit is set.
   iii. Static OOB, then the OOB Type field bit 0 is set to 1, the Output OOB Size is set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0. OOB data is supplied in the IXIT [6].
   iv. No OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size is set to 0x00, the Output OOB Action is set to 0x00, the Input OOB Size field bits are set to 0, and the Input OOB Action field bits are set to 0.

4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) with the field values set as follows:
   a. The Algorithm field set to the value indicated in Table 4.4.
   b. The Public Key field set to 0x00 if Public Key Available OOB is "No" and set to 0x01 otherwise.
   c. If the Authentication Method is:
      i. "No OOB", then the Authentication Method field is set to 0x00, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
      ii. "Static OOB", then the Authentication Method field is set to 0x01, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.
      iii. "Output OOB", then the Authentication Method field is set to 0x02, the Authentication Action field is set to a value between 0x00-0x04 as declared in Step 3, and the Authentication Size field is set to a value in the range declared in Step 3.
      iv. "Input OOB", then the Authentication Method field is set to 0x03, the Authentication Action field set to a value between 0x00-0x04 as declared in Step 3, and the Authentication Size field set to a value in the range declared in Step 3.

5. The Lower Tester sends a Provisioning Public Key PDU (PDU Type 0x03) to the IUT with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm, and expects a Provisioning Public Key PDU in the response containing the IUT's public key, with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.
6. If the Authentication Method is:
   a. "Input OOB", then the Lower Tester generates a random number and displays/reports this value so it can be entered on the IUT. The value is entered into the IUT using its Input OOB mechanism. The Lower Tester expects a Provisioning Input Complete PDU (PDU Type 0x04) from the IUT once input has been completed.
   b. "Output OOB", then the IUT is expected to internally generate a random number and utilize its Output OOB mechanism to communicate this generated value. This value is inputted into the

Lower Tester.

c. "Static OOB", then no action is taken in this step.

d. "No OOB", then no action is taken in this step.

7. The Lower Tester sends a Provisioning Confirmation PDU (PDU Type 0x05) to the IUT with the Confirmation field set to the calculated confirmation value. The Lower Tester expects a Provisioning Confirmation PDU in response containing the confirmation value calculated by the IUT.
8. The Lower Tester sends a Provisioning Random PDU (PDU Type 0x06) to the IUT containing the random value and expects to receive a Provisioning Random PDU in response from the IUT.
9. The Lower Tester authenticates the IUT based on the received data. If the authentication fails, the test case ends with a Fail verdict and the Lower Tester terminates the provisioning bearer link.
10. The Lower Tester calculates the session key and sends a Provisioning Data PDU (PDU Type 0x07) to the IUT with the Encrypted Provisioning Data field containing an encrypted and authenticated network key, NetKey Index of the current NetKey, current value of the IV Index, and unicast address of the primary Element, and the Provisioning Data MIC field containing a PDU Integrity Check value. The Lower Tester expects a Provisioning Complete PDU (PDU Type 0x08) from the IUT in response.
11. The Lower Tester terminates the provisioning bearer link on receipt of the Provisioning Complete PDU.
12. If PB-GATT was used as provisioning bearer, then after the provisioning bearer link has been terminated, the Lower Tester expects the IUT to start advertising with Node Identity.

Test Case	Algorithm	Public Key Available OOB	Authentication Method and Size
MESH/NODE/PROV/BV-01-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Output OOB 0x01-0x08
MESH/NODE/PROV/BV-02-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Input OOB 0x01-0x08
MESH/NODE/PROV/BV-03-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Static OOB
MESH/NODE/PROV/BV-04-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Output OOB 0x01-0x08
MESH/NODE/PROV/BV-05-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Input OOB 0x01-0x08
MESH/NODE/PROV/BV-06-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Static OOB

Test Case	Algorithm	Public Key Available OOB	Authentication Method and Size
MESH/NODE/PROV/BV-07-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	No OOB
MESH/NODE/PROV/BV-08-C [Provisioning Procedure, Public Key OOB, No OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	No OOB
MESH/NODE/PROV/BV-13-C [Provisioning Procedure, No Public Key OOB, Output OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Output OOB 0x01-0x20
MESH/NODE/PROV/BV-14-C [Provisioning Procedure, No Public Key OOB, Input OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Input OOB 0x01-0x20
MESH/NODE/PROV/BV-15-C [Provisioning Procedure, No Public Key OOB, Static OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Static OOB
MESH/NODE/PROV/BV-16-C [Provisioning Procedure, Public Key OOB, Output OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Output OOB 0x01-0x20
MESH/NODE/PROV/BV-17-C [Provisioning Procedure, Public Key OOB, Input OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Input OOB 0x01-0x20
MESH/NODE/PROV/BV-18-C [Provisioning Procedure, Public Key OOB, Static OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Static OOB
MESH/NODE/PROV/BV-19-C [Provisioning Procedure, No Public Key OOB, No OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	No OOB
MESH/NODE/PROV/BV-20-C [Provisioning Procedure, Public Key OOB, No OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	No OOB

Table 4.4: Provisioning Procedure - Unprovisioned Device test cases

- Expected Outcome

  Pass verdict

The IUT and the Lower Tester successfully establish a provisioning bearer link.

The IUT sends a correctly formatted Provisioning Capabilities PDU with all valid values. If bit 1 of the OOB Type field is set to 1, then either bit 0 is also set, or at least one of the Output OOB Size and Input OOB Size field values is greater than 0, and bit 0 of the Algorithms field is set to 0. The supported algorithms match the corresponding ICS entries.

The Lower Tester successfully authenticates the IUT using the selected authentication method.

If the Output OOB Action is Blink, Beep, or Vibrate (bits 0, 1, or 2 is set), the IUT outputs a random number as the selected sequence of events with a gap of at least 3 seconds between sequences. If the Output OOB Action is Output Numeric (bit 3 is set), the value is output using digits. If the Output OOB Action is Output Alphanumeric (bit 4 is set), the value is output using ASCII digits and uppercase letters.

If the public key is not available via an OOB mechanism, the public key sent by the IUT is not the one from the sample data.

All multiple-octet numeric values in Provisioning PDUs are sent as "big endian".

All RFU bits are ignored for each field in each Generic Provisioning PDU.

Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

If PB-ADV is used, the IUT increases the value of the Transaction Number field of the PB-ADV PDU by one for each new Provisioning PDU it sends.

In Step 12, if PB-GATT was used as provisioning bearer, the IUT starts advertising with Node Identity, and the Lower Tester verifies that the advertising data meets the requirements from the Pass verdict of MESH/SR/MPXS/BV-09-C [Advertise Mesh Proxy Service with Node Identity].

MESH/NODE/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout]

- Test Purpose

  Verify that an unprovisioned device IUT can successfully respond to a provisioning procedure after experiencing a procedure timeout.

- Reference

  [3] 5.3.3, 5.4.1, 5.4.2, 5.4.4

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- Test Procedure

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).

2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) selecting any method supported by the IUT.
5. The Lower Tester does not respond further to the IUT and waits for the duration of the provisioning procedure timeout.
6. After a period equal to 2 times the provisioning timeout has passed, the Lower Tester attempts to reestablish the provisioning bearer and initiate provisioning procedure by repeating Steps 1–3.
7. Lower Tester confirms that the IUT can respond to a new provisioning attempt following procedure timeout.

- Expected Outcome

  Pass verdict

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT sends a Provisioning Capabilities PDU that is correctly formatted with valid values.

  The IUT and the Lower Tester successfully re-establish the provisioning bearer link after the provisioning procedure timeout.

MESH/NODE/PROV/BV-10-C [Unprovisioned Device Sends Provisioning Failed Message on Provisioning Error]

- Test Purpose

  Verify that an unprovisioned device IUT sends a Provisioning Failed PDU when it encounters a provisioning error in the provisioning protocol and waits for the Provisioner to close the provisioning bearer.

- Reference

  [3] 5.3.3, 5.4.1, 5.4.2, 5.4.4

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- Test Procedure

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
3. The Lower Tester receives a Provisioning Capabilities PDU in response.

4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) to the IUT. The Algorithm field is set to 0x00 (FIPS P-256), Public Key is set to 0x00 ("Public key is not available via OOB"), Output OOB set to 0x00, the Input OOB Size set to a non-RFU, and Static OOB set to 0x00.
5. The Lower Tester sends any valid Provisioning PDU to the IUT, other than the Provisioning Public Key PDU.
6. The Lower Tester expects a Provisioning Failed PDU (PDU Type 0x09) from the IUT with the Value field set to 0x03 ("Unexpected PDU").
7. The Lower Tester sends a valid Provisioning Public Key PDU to the IUT.
8. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
9. The Lower Tester sends a valid Provisioning Invite PDU to the IUT.
10. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
11. The Lower Tester disconnects the provisioning bearer.
12. Repeat Steps 1–4, then continue the provisioning protocol to the point where the Lower Tester needs to send the Provisioning Random PDU.
13. The Lower Tester sends a Provisioning Random PDU containing an incorrect Provisioner Random value, i.e., a value different than the one used to compute the Provisioner Confirmation value sent in the Provisioning Confirmation PDU in the previous step.
14. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x04 ("Confirmation Failed").
15. The Lower Tester sends a Provisioning Random PDU containing the correct Provisioner Random value, i.e., the same value used to compute the Provisioner Confirmation value sent in the Provisioning Confirmation PDU in the previous step.
16. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
17. The Lower Tester sends a valid Provisioning Invite PDU to the IUT.
18. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
19. The Lower Tester disconnects the provisioning bearer.
20. Repeat Steps 1–4, then continue the provisioning protocol to the point where the Lower Tester needs to send the Provisioning Data PDU.
21. The Lower Tester sends a Provisioning Data PDU containing an incorrect Provisioning Data MIC field value.
22. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x06 ("Decryption Failed").
23. The Lower Tester sends a Provisioning Data PDU containing the correct Provisioning Data MIC field value.
24. The Lower Tester expects a Provisioning Failed PDU from the IUT with the Value field set to 0x03 ("Unexpected PDU").
25. The Lower Tester disconnects the provisioning bearer.

- Expected Outcome

  Pass verdict

The IUT sends a Provisioning Failed PDU in with the appropriate error code whenever a protocol error occurs.

Notes:

Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

MESH/NODE/PROV/BV-11-C [Provisioned Node Updates Security Flags Based on the Provisioning Data PDU]

- Test Purpose

  Verify that an IUT that has just been provisioned correctly updates the IV Update procedure state and the Key Refresh Phase based on the Flags field of the Provisioning Data PDU.

- Reference

  [3] 5.3.3, 5.4.1.8, 5.4.2.5

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester is a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- Test Procedure
  1. The Lower Tester provisions the IUT by executing any of the procedures described in Provisioning Procedure - Unprovisioned Device. When sending the Provisioning Data PDU, the Lower Tester sets the Key Refresh flag and the IV Update flag of the Flags field to 1.
  2. The Lower Tester disconnects the provisioning bearer.
  3. The Upper Tester orders the IUT (or the IUT is triggered) to send a Secure Network beacon containing the IUT's security flags.
  4. The Lower Tester expects the IUT to send a Secure Network beacon containing the current IV Index, with the IV Update flag and the Key Refresh flag set to 1.

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends the Secure Network beacon with the IV Update flag and the Key Refresh flag set to 1.

MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session]

- Test Purpose

  Verify that the IUT, as an unprovisioned device, generates a new private-public key pair for each new provisioning session before sending the public key over the air.

- Reference

  [3], [7] 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition
  - The IUT is an unprovisioned device with the Device UUID specified in the IXIT [6].
  - The Lower Tester is a Provisioner.
- Test Procedure

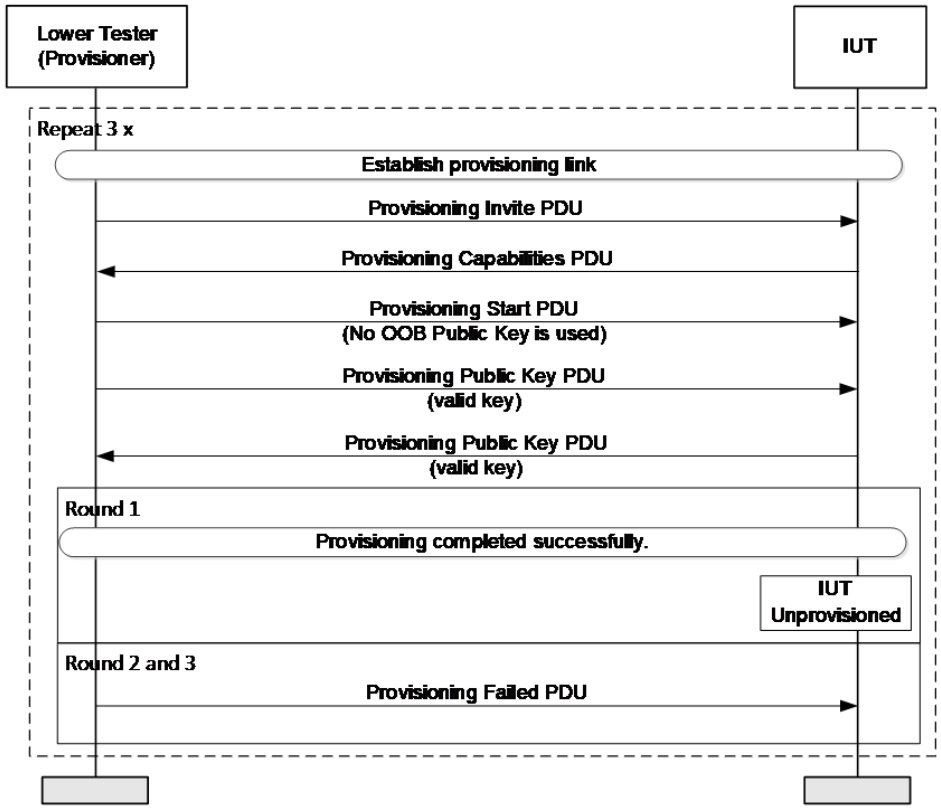


Figure 4.7: Test procedure for MESH/NODE/PROV/BV-12-C [Generate New Public Key on Each Provisioning Session] MSC

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU to the IUT, with the Attention Duration field set to any value.
3. The Lower Tester expects the IUT to respond with a Provisioning Capabilities PDU with valid fields.
4. The Lower Tester sends a Provisioning Start PDU with the Public Key field set to 0x00 (No OOB Public Key is used), and all other fields set to valid values.
5. The Lower Tester sends a Provisioning Public Key PDU with valid values.
6. The Lower Tester expects the IUT to send a Provisioning Public Key PDU with valid values. The Lower

Tester saves the public key of the IUT.

7. The Lower Tester successfully completes the provisioning protocol with the IUT, then the IUT is reset to the unprovisioned state.
8. Repeat Steps 1–6, expecting the IUT to send a public key value different than the one received in the previous iteration. The Lower Tester saves the new public key of the IUT.
9. The Lower Tester fails the provisioning protocol before it is completed.
10. Repeat Steps 1–6, expecting the IUT to send a public key value different from those received in the previous iterations.
11. The Lower Tester terminates the provisioning session.

- Expected Outcome

  Pass verdict

  In each new provisioning session, the IUT sends a public key that is different than the public keys sent in previous sessions.

4.5.2 Provisioning Procedure - Provisioner

- Test Purpose

  This is a generic procedure to verify the behavior of a Provisioner in the Provisioning procedure. The verification is done for one combination of authentication values at a time, as enumerated in the test cases in Table 4.5, using this generic procedure.

- Reference

  [3] 5.1, 5.3.3, 5.4.1, 5.4.2, 5.4.3

- Initial Condition

  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device.
  - Field values of the Provisioning Capabilities PDU are specified in the IXIT [6].

- Test Procedure

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
  2. The Upper Tester orders the IUT to send a Provisioning Invite PDU (PDU Type 0x00) to the Lower Tester with the Attention Duration set to a value greater than 0.
  3. Upon receipt of the Provisioning Invite PDU, the Lower Tester sends a Provisioning Capabilities PDU (PDU Type 0x01) to the IUT. The Number of Elements field is set to a value between 0x01-0xFF and the Algorithms field has set to 1 only bit 0 and the bit corresponding to the algorithm indicated in the Algorithm column in Table 4.5 (unless this is already bit 0); all other bits are set to 0. The rest of the fields are set as follows:
     a. Public Key Type field bit 0 is set to 0 (zero) if Public Key Available OOB is "No", and set to 1

otherwise.

b. If the Authentication Method is:

i. Output OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size field is set to a random non-RFU value greater than 0x00 in the range indicated in Table 4.5, the Output OOB Action field bit is set to a set of randomly selected actions, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.

ii. Input OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size is set to 0x00, the Output OOB Action is set to 0x00, the Input OOB Size is set to a non-RFU value greater than 0x00 in the range indicated in Table 4.5 as supplied in the IXIT [6], and the Input OOB Action is set to a set of randomly selected actions.

iii. Static OOB, then the OOB Type field bit 0 is set to 1, the Output OOB Size field is set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.

iv. No OOB, then the OOB Type field bit 0 is set to 0, the Output OOB Size is set to 0x00, the Output OOB Action field bits are set to 0, the Input OOB Size field is set to 0x00, and the Input OOB Action field bits are set to 0.

4. The IUT is induced to send a Provisioning Start PDU (PDU Type 0x02) to the Lower Tester with the Algorithm field value set to the value indicated in Table 4.5, and the rest of the field values are as follows:

a. The Public Key field set to 0x00 if Public Key Available OOB is "No" and set to 0x01 otherwise.

b. If the Authentication Method is:

i. "No OOB", then the Authentication Method field is set to 0x00, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.

ii. "Static OOB", then the Authentication Method field is set to 0x01, the Authentication Action field is set to 0x00, and the Authentication Size field is set to 0x00.

iii. "Output OOB", then the Authentication Method field is set to 0x02, the Authentication Action field is set to a value between 0x00-0x04 as declared in Step 3, and the Authentication Size field is set to a value in the range declared in Step 3.

iv. "Input OOB", then the Authentication Method field is set to 0x03, the Authentication Action field set to a value between 0x00-0x03 as declared in Step 3, and the Authentication Size field set to a value in the range declared in Step 3.

5. The IUT is induced to send a Provisioning Public Key PDU (PDU Type 0x03) to the Lower Tester with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm, and expects a Provisioning Public Key PDU in response from the Lower Tester containing the Lower Tester's public key, with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.
6. If the Authentication Method is:

a. "Input OOB", then the IUT generates a random number and displays/reports this value so it can be entered on the Lower Tester. The value is entered into the Lower Tester using its Input OOB mechanism. The Lower Tester sends a Provisioning Input Complete PDU (PDU Type 0x04) to the IUT once input has been completed.

b. "Output OOB", then the Lower Tester is expected to internally generate a random number and utilize its Output OOB mechanism to communicate this generated value. This value is inputted into the IUT.
c. "Static OOB", then no action is taken in this step.
d. "No OOB", then no action is taken in this step and the value is zero.

7. The IUT is induced to send a Provisioning Confirmation PDU (PDU Type 0x05) to the Lower Tester with the Confirmation field set to any values exchanged between IUT and the Lower Tester based on the OOB Authentication method. The Lower Tester sends a Provisioning Confirmation PDU in response containing the confirmation value calculated by the Lower Tester.
8. The IUT is induced to send a Provisioning Random PDU (PDU Type 0x06) to the Lower Tester containing the random value and expects to receive a Provisioning Random PDU in response from the Lower Tester.
9. The IUT authenticates the Lower Tester based on the received data. If the authentication fails, the test case ends with a Fail verdict and the IUT terminates the provisioning bearer link.
10. The IUT calculates the session key and sends a Provisioning Data PDU (PDU Type 0x07) to the Lower Tester with the Encrypted Provisioning Data field containing an encrypted and authenticated network key, NetKey Index of the current NetKey, the IV Index field set to 0x00000000, and unicast address of the primary Element, and the Provisioning Data MIC field containing a PDU Integrity Check value. Upon receipt of the Provisioning Data PDU, the Lower Tester sends a Provisioning Complete PDU (PDU Type 0x08) to the IUT.
11. The IUT terminates the provisioning bearer link on receipt of the Provisioning Complete PDU.

Test Case	Algorithm	Public Key Available OOB	Authentication Method and Size
MESH/PVNR/PROV/BV-01-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Output OOB 0x01-0x08
MESH/PVNR/PROV/BV-02-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Input OOB 0x01-0x08
MESH/PVNR/PROV/BV-03-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	Static OOB
MESH/PVNR/PROV/BV-04-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Output OOB 0x01-0x08

Test Case	Algorithm	Public Key Available OOB	Authentication Method and Size
MESH/PVNR/PROV/BV-05-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Input OOB 0x01-0x08
MESH/PVNR/PROV/BV-06-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	Static OOB
MESH/PVNR/PROV/BV-07-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	No	No OOB
MESH/PVNR/PROV/BV-08-C [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication, Algorithm 0]	BTM_ECDH_P256_CMAC_AES128_AES_CCM	Yes	No OOB
MESH/PVNR/PROV/BV-11-C [Provisioning Procedure as Provisioner, No Public Key OOB, Output OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Output OOB 0x01-0x20
MESH/PVNR/PROV/BV-12-C [Provisioning Procedure as Provisioner, No Public Key OOB, Input OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Input OOB 0x01-0x20
MESH/PVNR/PROV/BV-13-C [Provisioning Procedure as Provisioner, No Public Key OOB, Static OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	Static OOB
MESH/PVNR/PROV/BV-14-C [Provisioning Procedure as Provisioner, Public Key OOB, Output OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Output OOB 0x01-0x20
MESH/PVNR/PROV/BV-15-C [Provisioning Procedure as Provisioner, Public Key OOB, Input OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Input OOB 0x01-0x20
MESH/PVNR/PROV/BV-16-C [Provisioning Procedure as Provisioner, Public Key OOB, Static OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	Static OOB

Test Case	Algorithm	Public Key Available OOB	Authentication Method and Size
MESH/PVNR/PROV/BV-17-C [Provisioning Procedure as Provisioner, No Public Key OOB, No OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	No	No OOB
MESH/PVNR/PROV/BV-18-C [Provisioning Procedure as Provisioner, Public Key OOB, No OOB Authentication, Algorithm 1]	BTM_ECDH_P256_HMAC_SHA256_AES_CCM	Yes	No OOB

Table 4.5: Provisioning Procedure - Provisioner test cases

- Expected Outcome

  Pass verdict

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT sends a correctly formatted Provisioning Invite PDU with valid values.

  The IUT sends a correctly formatted Provisioning Start PDU with valid values in response to the Provisioning Capabilities PDU.

  The public key sent by the IUT is not the one from the sample data.

  The IUT successfully authenticates the Lower Tester using the selected authentication method.

  If the Input OOB Action is Push or Twist, the IUT selects a random number to input by the designated action.

  The IUT sends a correctly formatted Provisioning Data PDU with valid values to the Lower Tester and receives a Provisioning Complete PDU from the Lower Tester in response. The NetKey from the provisioning data is not the one from the sample data.

  The IUT successfully terminates the provisioning bearer link.

  All multiple-octet numeric values in Provisioning PDUs are sent as "big endian".

  All RFU bits are ignored for each field in each Generic Provisioning PDU.

  Each Generic Provisioning PDU is sent after a random delay between 20 ms and 50 ms.

  If PB-ADV is used, the IUT increases the value of the Transaction Number field of the PB-ADV PDU by one for each new Provisioning PDU it sends.

MESH/PVNR/PROV/BV-09-C [Provisioning Procedure, Procedure Timeout – Provisioner]

- Test Purpose

Verify that a Provisioner IUT can initiate the provisioning procedure after experiencing a procedure timeout in a previous attempt.

- Reference

  [3] 5.3.3, 5.4.1, 5.4.2, 5.4.4

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned node.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].
- Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
  2. The IUT is induced to send a Provisioning Invite PDU with the Attention Duration set to a value greater than 0.
  3. The Lower Tester sends a Provisioning Capabilities PDU in response.
  4. The Lower Tester expects a Provisioning Start PDU selecting any method supported by the IUT.
  5. The Lower Tester expects a Provisioning Public Key PDU from the IUT.
  6. The Lower Tester does not respond further to the IUT and waits for the duration of the provisioning procedure timeout. After the timeout has expired, the IUT disconnects the provisioning bearer.
  7. After a period, equal to 2 times the provisioning timeout has passed, the IUT attempts to reestablish the provisioning bearer and initiate provisioning by repeating Steps 1–3.
  8. The Lower Tester confirms that the IUT can initiate a new provisioning attempt following procedure timeout.
- Expected Outcome

  Pass verdict

  The IUT and the Lower Tester successfully establish a provisioning bearer link.

  The IUT terminates the procedure when the timeout has expired.

  The IUT does not transmit a Provisioning Failed PDU.

  The IUT can reestablish a provisioning bearer and restart the provisioning procedure following procedure timeout.

MESH/PVNR/PROV/BV-10-C [Provisioner Disconnects Provisioning Bearer on Provisioning Protocol Failure from Unprovisioned Device]

- Test Purpose

  Verify that a Provisioner IUT disconnects the provisioning bearer when it encounters a provisioning protocol

failure from an unprovisioned device.

- Reference

  [3] 5.2, 5.3.3, 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition

  - The IUT is a Provisioner.
  - The Lower Tester is an Unprovisioned Device.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- Test Procedure

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.2 State: Provisioner PB-ADV Link Established, or for PB-GATT use 4.2.4 State: Provisioner PB-GATT Link Established).
  2. The Upper Tester sends a command to the IUT to send a Provisioning Invite PDU (PDU Type 0x00) with the Attention Duration set to a value greater than 0.
  3. The Lower Tester sends a Provisioning Capabilities PDU (PDU Type 0x01) in response, indicating the Lower Tester supports No OOB authentication. The Number of Elements field is set to a value between 0x01-0xFF, the Algorithms field has bit 0 ("FIPS-P256 Elliptic Curve") set to 1 and all other bits set to 0, the Public Key Type field is set to 0x00, the Static OOB Type field bit 0 is set to 0, the Output OOB Size field is set to 0x00, the Output OOB Action field is set to 0x00, the Input OOB Size field is set to 0x00, the Input OOB Action field is set to 0x00.
  4. The Lower Tester expects a Provisioning Start PDU (PDU Type 0x02) with the Algorithm field value set to 0x00, the Public Key field set to 0x00 (No OOB Public Key), the Authentication Method field set to 0x00 (No OOB), the Authentication Action field set to 0x00, and the Authentication Size field set to 0x00.
  5. The Lower Tester expects a Provisioning Public Key PDU (PDU Type 0x03) from the IUT with the fields Public Key X and Public Key Y containing the X and Y components of its public key for the FIPS P-256 algorithm.
  6. The Lower Tester sends any Provisioning PDU that does not follow the correct Provisioning procedure.
  7. The IUT disconnects the provisioning bearer on receipt of the Provisioning PDU sent in Step 6.

- Expected Outcome

  Pass verdict

  In Step 7, the IUT disconnects the provisioning bearer.

MESH/NODE/PROV/BI-01-C [Device Resets Attention Timer State when Provisioning Bearer Dropped]

- Test Purpose

  Verify that an unprovisioned device IUT resets the attention timer state to 0x00 when the provisioning bearer is dropped unexpectedly before the provisioning protocol is complete.

- Reference

  [3] 3.1.2, 3.9.2, 3.10.1, 5.4.1, 5.4.2.2

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester acts as a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].
- Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.
  3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
  4. The Lower Tester closes the link on the provisioning bearer.
  5. The Upper Tester verifies that the IUT sets the Attention Timer state of its primary Element to 0x00.
- Expected Outcome

  Pass verdict

  The IUT sets the Attention Timer state of its primary Element to 0x00.

MESH/NODE/PROV/BI-02-C [Reject Provisioning Start PDU with RFU values]

- Test Purpose

  Verify that an unprovisioned device IUT can reject a Provisioning Start PDU containing RFU values.

- Reference

  [3] 3.1.2, 3.9.2, 3.10.1, 5.4.1.3

- Initial Condition
  - The IUT is an unprovisioned device.
  - The Lower Tester acts as a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].
- Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU (PDU Type 0x00) to the IUT with the Attention Duration set to a value greater than 0.

3. The Lower Tester expects a Provisioning Capabilities PDU (PDU Type 0x01) in response from the IUT.
4. The Lower Tester sends a Provisioning Start PDU (PDU Type 0x02) with the Algorithm field set to any RFU value.
5. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
6. Repeat Steps 1–3.
7. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00 and the Public Key set to any RFU value.
8. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
9. Repeat Steps 1–3.
10. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x00, and the Authentication Method field set to any RFU value.
11. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
12. Repeat Steps 1–3.
13. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x01, the Authentication Method field set to 0x00, and the Authentication Action field set to any RFU value.
14. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.
15. Repeat Steps 1–3.
16. The Lower Tester sends a Provisioning Start PDU with the Algorithm field set to 0x00, the Public Key field set to 0x01, the Authentication Method field set to 0x00, the Authentication Action field set to 0x00, and the Authentication Size field set to any RFU value.
17. The IUT sends a Provisioning Failed PDU with the Error Code field set to 0x02 to the Lower Tester. The Lower Tester closes the provisioning bearer link.

- Expected Outcome

  Pass verdict

  In Steps 5, 8, 11, 14, and 17, the IUT sends a Provisioning Failed PDU with the appropriate error code.

MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values]

- Test Purpose

  Verify that an unprovisioned device IUT rejects Provisioning Start PDUs with values that are inconsistent with its capabilities.

- Reference

  [3] 5.4.1.10

- Initial Condition

  - The IUT is an unprovisioned device.

- The Lower Tester acts as a Provisioner.
- The Device UUID of the IUT is specified in the IXIT [6].

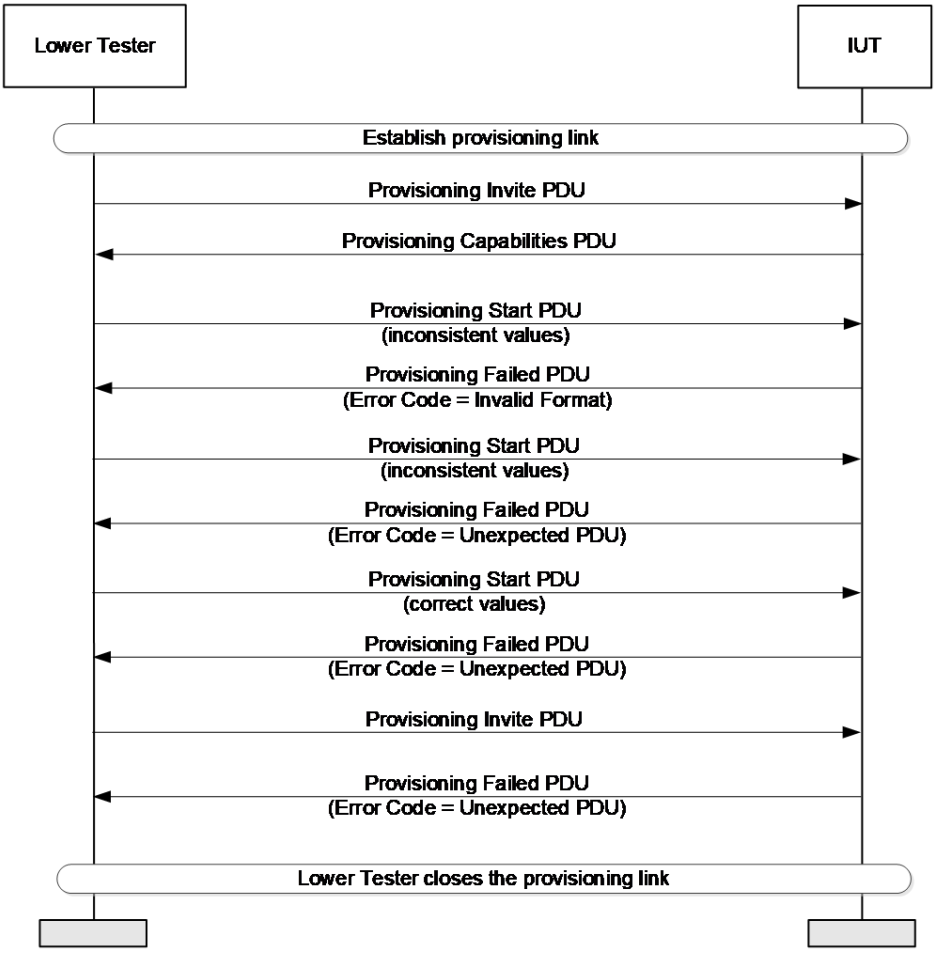


Figure 4.8: Test procedure for MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values] MSC

- Test Procedure

  Execute Steps 1–11 for each round from Table 4.6.

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning Invite PDU to the IUT and expects the IUT to respond with a Provisioning Capabilities PDU.
  3. If the condition from Table 4.6, column "Condition" for this round, is met by the IUT as determined from the Provisioning Capabilities PDU, then the Lower Tester sends a Provisioning Start PDU with the values indicated in Table 4.6, column "Provisioning Start PDU" for this round. Otherwise, the current round ends with a successful verdict (skip Steps 4–11 and move to the next round, if any).

4. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Invalid Format.
5. The Lower Tester sends the same Provisioning Start PDU from Step 3.
6. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.
7. The Lower Tester sends a new Provisioning Start PDU with field values that are consistent with the IUT capabilities.
8. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.
9. The Lower Tester sends a new Provisioning Invite PDU.
10. The Lower Tester expects the IUT to send a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.
11. The Lower Tester closes the provisioning bearer link.

- Test Case Configuration

Round	Condition	Provisioning Start PDU
1	IUT does not support an OOB public key	Use an OOB Public Key
2	Output OOB is not supported or Output OOB Size is less than 8	Use Output OOB with size equal to 8
3	Input OOB is not supported or Input OOB Size is less than 8	Use Input OOB with size equal to 8
4	IUT does not support all provisioning algorithms.	Use an algorithm not supported by the IUT.
5	Bit 1 of the OOB Type field is set to 1.	Use Authentication Method 0x00.

Table 4.6: Rounds for MESH/NODE/PROV/BI-03-C [Reject Provisioning Start PDU with Inconsistent Values]

- Expected Outcome

  Pass verdict

  The IUT rejects the inconsistent Provisioning Start PDU with the Invalid Format error code.

  After sending the Provisioning Failed PDU, the IUT rejects any other PDU with the Unexpected PDU error code.

MESH/NODE/PROV/BI-13-C [Reject Invalid Public Keys]

- Test Purpose

  Verify that the IUT, as an unprovisioned device, detects and rejects an invalid public key during provisioning.

- Reference

  [3], [7] 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition

- The IUT is an unprovisioned device with the Device UUID specified in the IXIT [6].
- The Lower Tester is a Provisioner.

• Test Procedure

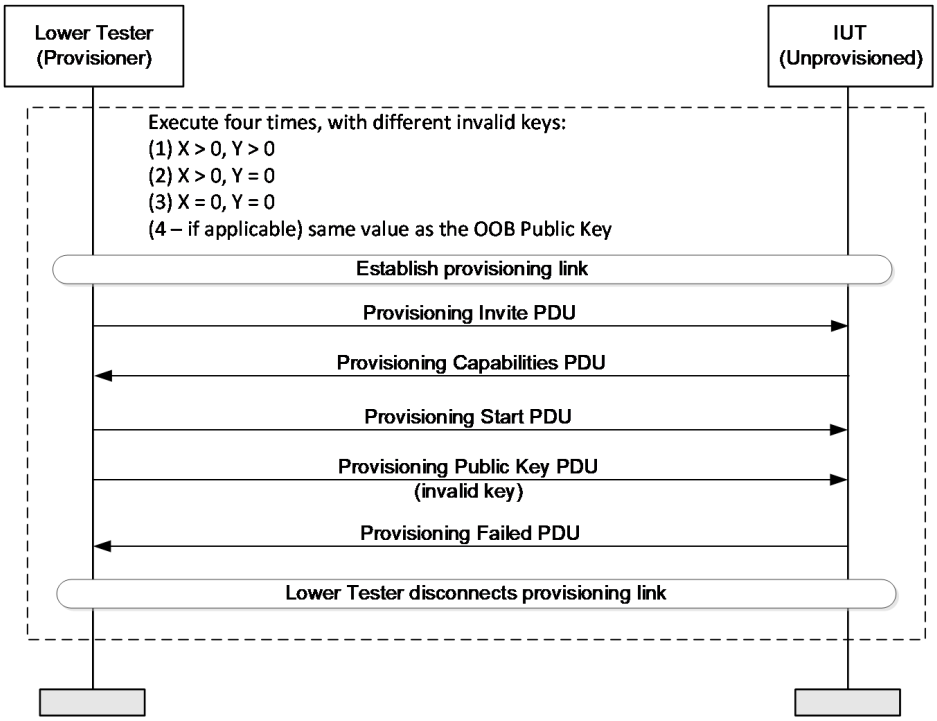


Figure 4.9: Test procedure for MESH/NODE/PROV/BI-13-C [Reject Invalid Public Keys] MSC

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU to the IUT, with the Attention Duration field set to any value, and expects the IUT to respond with a Provisioning Capabilities PDU with valid fields.
3. The Lower Tester sends a Provisioning Start PDU with parameters that are valid with respect to the IUT capabilities received in Step 2.
4. The Lower Tester sends a Provisioning Public Key PDU containing invalid public key coordinates that are not on the P-256 curve. To obtain the invalid public key, the Lower Tester generates a valid public key, then flips a single bit from the Y-coordinate (in the Public Key Y field of the Provisioning Public Key PDU) before sending it over the air. The Lower Tester verifies that these new coordinates are not on the curve before sending them over the air; if the new coordinates happen to be valid, then the generation procedure is repeated.
5. The Lower Tester expects a Provisioning Failed PDU from the IUT.
6. The Lower Tester disconnects the provisioning bearer.
7. Repeat Steps 1–6 using, in Step 4, another invalid public key obtained by generating a valid public key and changing the Y coordinate to 0 (the Public Key Y field of the Provisioning Public Key PDU is set to all-zeroes).

8. Repeat Steps 1–6 using, in Step 4, the invalid public key coordinates (0, 0). Both the Public Key X and the Public Key Y fields of the Provisioning Public Key PDU are set to all-zeroes.
9. If the IUT supports an OOB Public Key, then repeat Steps 1–6 using, in Step 4, the same value as the IUT's public key value that has been obtained out of band.

- Expected Outcome

  Pass verdict

  The IUT detects each invalid public key and sends a Provisioning Failed PDU on each iteration.

MESH/PVNR/PROV/BI-14-C [Reject Invalid Public Keys]

- Test Purpose

  Verify that the IUT, as a Provisioner, detects and rejects an invalid public key during provisioning.

- Reference

  [3], [7] 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device with the Device UUID specified in the IXIT [6].
- Test Procedure

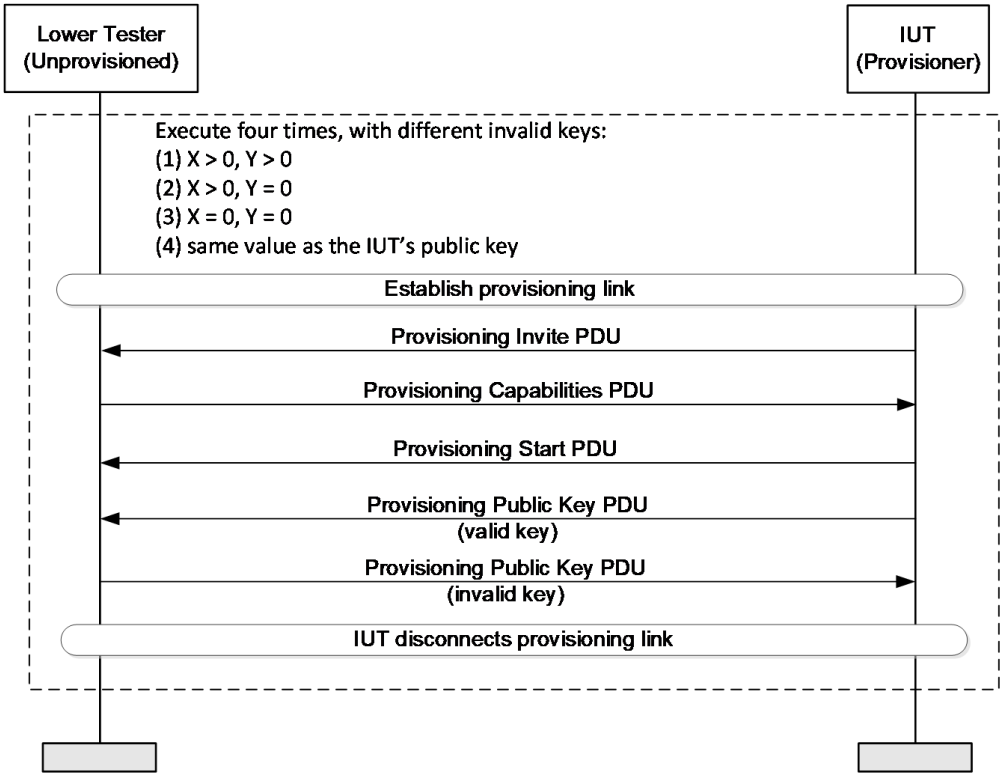


Figure 4.10: Test procedure for MESH/PVNR/PROV/ BI-14-C [Reject Invalid Public Keys] MSC

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The IUT is induced to begin the provisioning procedure. The Lower Tester expects a Provisioning Invite PDU from the IUT, with the Attention Duration field set to any value, and responds with a Provisioning Capabilities PDU with any valid fields, indicating that a public key is not available OOB (i.e., setting bit 0 of the Public Key Type field to 0).
3. The Lower Tester expects a Provisioning Start PDU with parameters that are valid with respect to the capabilities sent in Step 2, followed by a Provisioning Public Key PDU containing a valid public key. The Lower Tester saves the public key of the IUT.
4. The Lower Tester sends a Provisioning Public Key PDU containing invalid public key coordinates that are not on the P-256 curve. To obtain the invalid public key, the Lower Tester generates a valid public key, then flips a single bit from the Y-coordinate (in the Public Key Y field of the Provisioning Public Key PDU) before sending it over the air. The Lower Tester verifies that these new coordinates are not on the curve before sending them over the air; if the new coordinates happen to be valid, then the generation procedure is repeated.
5. The Lower Tester expects the IUT to disconnect the provisioning bearer.
6. Repeat Steps 1–5 using, in Step 4, another invalid public key obtained by generating a valid public key and changing the Y coordinate to 0 (the Public Key Y field of the Provisioning Public Key PDU is set to all-zeroes).
7. Repeat Steps 1–5 using, in Step 4, the invalid public key coordinates (0, 0). Both the Public Key X and the Public Key Y fields of the Provisioning Public Key PDU are set to all-zeroes.

8. Repeat Steps 1–5 using, in Step 4, the same public key value that was received from the IUT in Step 3.

- Expected Outcome

  Pass verdict

  The IUT detects each invalid public key and disconnects the provisioning bearer on each iteration.

  In each repetition of Step 3, the IUT sends a public key value different that in the previous provisioning session.

MESH/NODE/PROV/BI-15-C [Reject Provisioning PDU, Unsupported Type]

- Test Purpose

  Verify that an unprovisioned device IUT can reject a Provisioning PDU of an unsupported type.

- Reference

  [3] 5.4.1, 5.4.4

- Initial Condition

  - The IUT is an unprovisioned device.
  - The Lower Tester acts as a Provisioner.
  - The Device UUID for the IUT is specified by the manufacturer in the IXIT [6].

- Test Procedure

  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, use 4.2.1 State: Node PB-ADV Link Established, or for PB-GATT use 4.2.3 State: Node PB-GATT Link Established).
  2. The Lower Tester sends a Provisioning PDU with the Type field set to the lowest unsupported or RFU value and the Parameters field set to a random valid value.
  3. The IUT sends a Provisioning Failed PDU with the Error Code field set to Invalid PDU (0x01) to the Lower Tester.
  4. The Lower Tester closes the provisioning bearer link.

- Expected Outcome

  Pass verdict

  The IUT sends a Provisioning Failed PDU with the appropriate error code.

MESH/PVNR/PROV/BI-16-C [Reject Provisioning PDU, Unsupported Type]

- Test Purpose

  Verify that a Provisioner IUT can reject a Provisioning PDU of an unsupported type.

- Reference

  [3] 5.4.1, 5.4.4

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device with the Device UUID specified in the IXIT [6].
- Test Procedure
  1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, 4.2.1 State: Node PB-ADV Link Established or for PB-GATT, 4.2.3 State: Node PB-GATT Link Established).
  2. The IUT sends a Provisioning Invite PDU with the Attention Duration field set to a random valid value.
  3. The Lower Tester sends a Provisioning PDU with the Type field set to the lowest unsupported or RFU value and the Parameters field set to a random valid value.
  4. The IUT disconnects the provisioning bearer.
- Expected Outcome

  Pass verdict

  The IUT disconnects the provisioning bearer.

MESH/NODE/PROV/BI-17-C [Reject Invalid Confirmation Value]

- Test Purpose

  Verify that the IUT, as an unprovisioned device, detects and rejects an invalid confirmation value during provisioning.

- Reference

  [3], [7] 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition
  - The IUT is an unprovisioned device with the Device UUID specified in the IXIT [6].
  - The Lower Tester is a Provisioner.
- Test Procedure

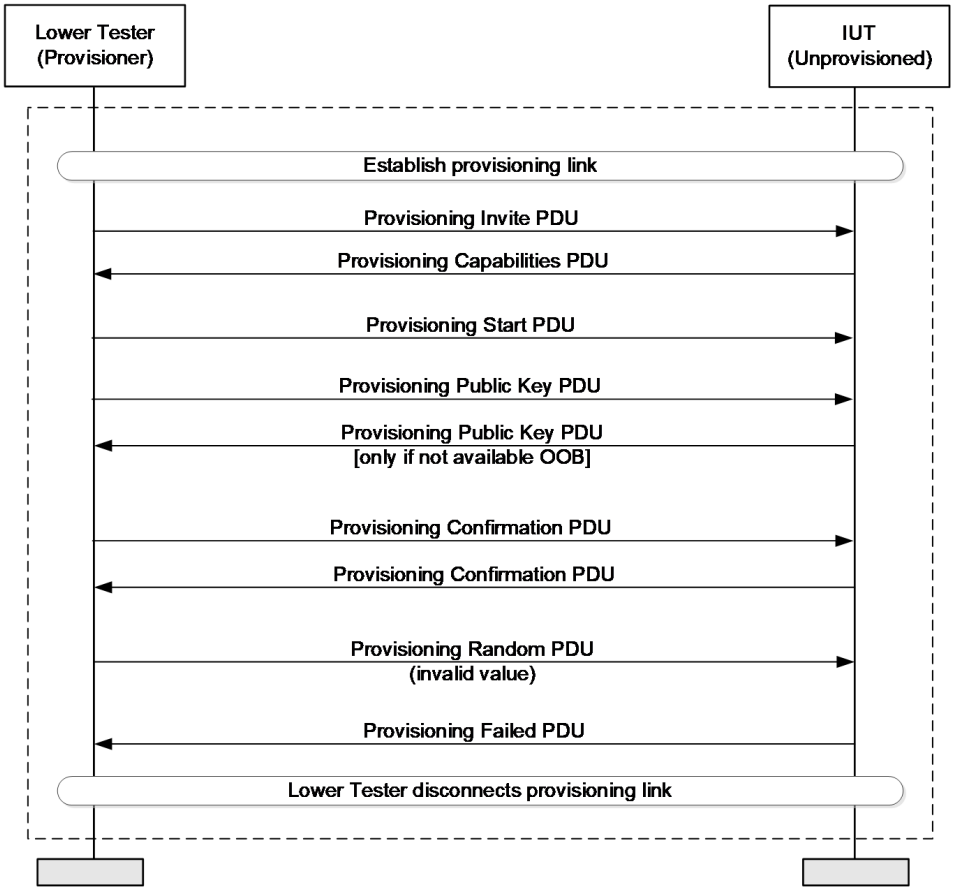


Figure 4.11: Test procedure for MESH/NODE/PROV/BI-17-C [Reject Invalid Confirmation Value] MSC

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The Lower Tester sends a Provisioning Invite PDU to the IUT, with the Attention Duration field set to any value, and the IUT responds with a Provisioning Capabilities PDU with valid fields.
3. The Lower Tester sends a Provisioning Start PDU with parameters that are valid with respect to the IUT capabilities received in Step 2.
4. The Lower Tester sends a Provisioning Public Key PDU containing a valid public key.
5. If an OOB Public Key is used, then skip this step; otherwise, the IUT sends a Provisioning Public Key PDU.
6. The Lower Tester sends a Provisioning Confirmation PDU.
7. The IUT sends a Provisioning Confirmation PDU.
8. The Lower Tester sends a Provisioning Random PDU with a value different than the one used to compute the confirmation value sent in Step 6.
9. The IUT sends a Provisioning Failed PDU.
10. The Lower Tester disconnects the provisioning bearer.

- Expected Outcome

Pass verdict

The IUT detects the invalid confirmation-random pair and sends a Provisioning Failed PDU.

MESH/PVNR/PROV/BI-18-C [Reject Invalid Confirmation Value]

- Test Purpose

  Verify that the IUT, as a Provisioner, detects and rejects an invalid confirmation value during provisioning.

- Reference

  [3], [7] 5.4.1, 5.4.2, 5.4.3, 5.4.4

- Initial Condition
  - The IUT is a Provisioner.
  - The Lower Tester is an unprovisioned device with the Device UUID specified in the IXIT [6].
- Test Procedure

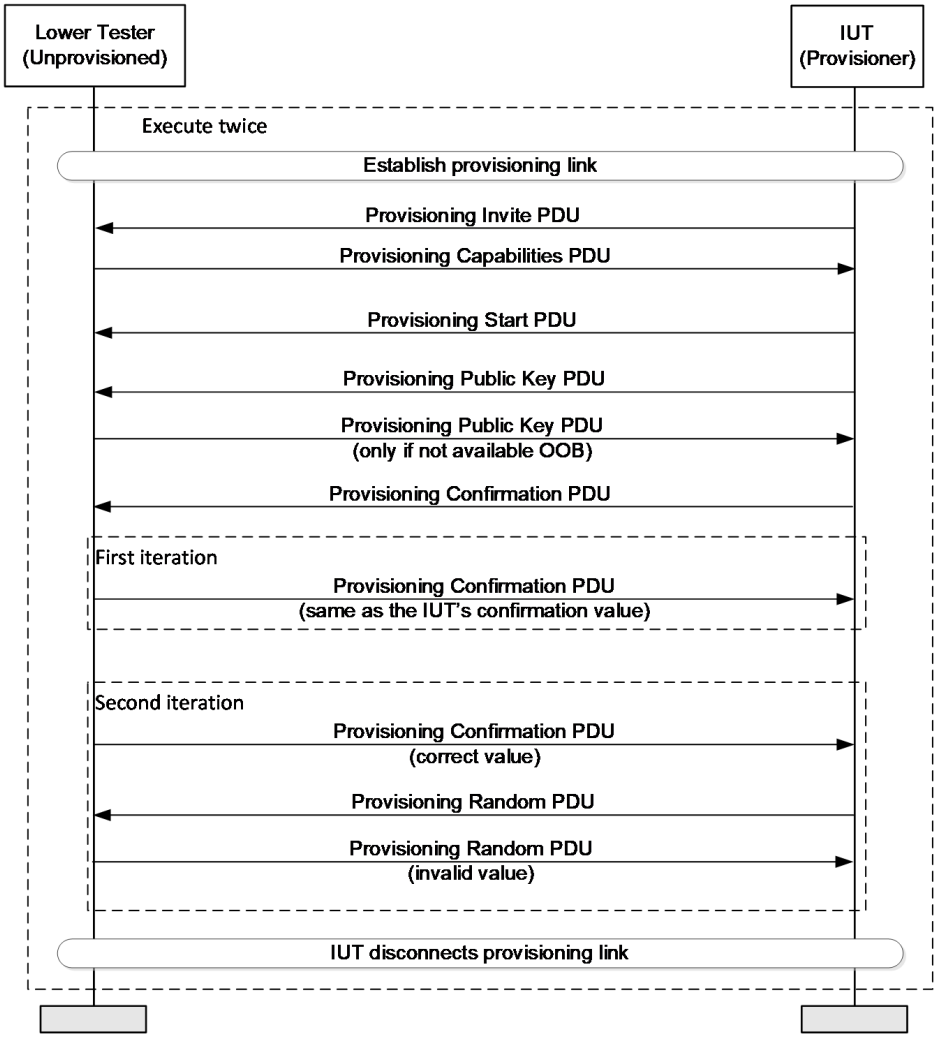


Figure 4.12: Test procedure for MESH/PVNR/PROV/ BI-18-C [Reject Invalid Confirmation Value] MSC

1. The IUT and the Lower Tester establish a provisioning bearer link using the appropriate Setup Preamble for the supported provisioning bearer being tested (e.g., for PB-ADV, State: Node PB-ADV Link Established 4.2.1 or for PB-GATT, State: Node PB-GATT Link Established 4.2.3).
2. The IUT is induced to begin the provisioning procedure. The Lower Tester expects a Provisioning Invite PDU from the IUT, with the Attention Duration field set to any value, and responds with a Provisioning Capabilities PDU with any valid fields, indicating that a public key is not available OOB (i.e., setting bit 0 of the Public Key Type field to 0).
3. The IUT sends a Provisioning Start PDU with parameters that are valid with respect to the capabilities sent in Step 2, followed by a Provisioning Public Key PDU containing a valid public key. The Lower Tester saves the public key of the IUT.
4. If an OOB Public Key is used, then skip this step; otherwise, the Lower Tester sends a Provisioning Public Key PDU containing a valid public key.
5. The IUT sends a Provisioning Confirmation PDU.
6. The Lower Tester sends a Provisioning Confirmation PDU with the same value received in Step 5.

7. The IUT disconnects the provisioning bearer.
8. Repeat Steps 1–6, but in Step 6, send a confirmation value computed correctly based on the generated random value.
9. The IUT sends a Provisioning Random PDU with a valid value.
10. The Lower Tester sends a Provisioning Random PDU with a value different than the one used to compute the confirmation value sent in Step 8.
11. The IUT disconnects the provisioning bearer.

- Expected Outcome

  Pass verdict

  The IUT detects each invalid confirmation value and disconnects the provisioning bearer on each iteration.

  In each repetition of Step 3, the IUT sends a public key value different than in the previous provisioning session.

4.6 Private Beacon Model

The test objective is to verify functionality specific to the Private Beacon Model for both the Private Beacon Server Model and the Private Beacon Client Model.

4.6.1 Private Beacon Procedures

This section validates the Private Beacons procedures for both the Private Beacon Server Model and the Private Beacon Client Model.

MESH/SR/PRB/PBS/BV-01-C [Respond to Private Beacon Get/Set Messages]

- Test Purpose

  Verify that the Private Beacon Server IUT can respond to Private Beacon Get and Set messages.

- Reference

  [3] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model on the primary Element.
  - The Lower Tester supports the Private Beacon Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.
  - The Private Beacon state and the Random Update Interval Steps state are set to their default values after provisioning.
  - The Upper Tester sets the Private Beacon Interval to 10s on the IUT.

- Test Procedure

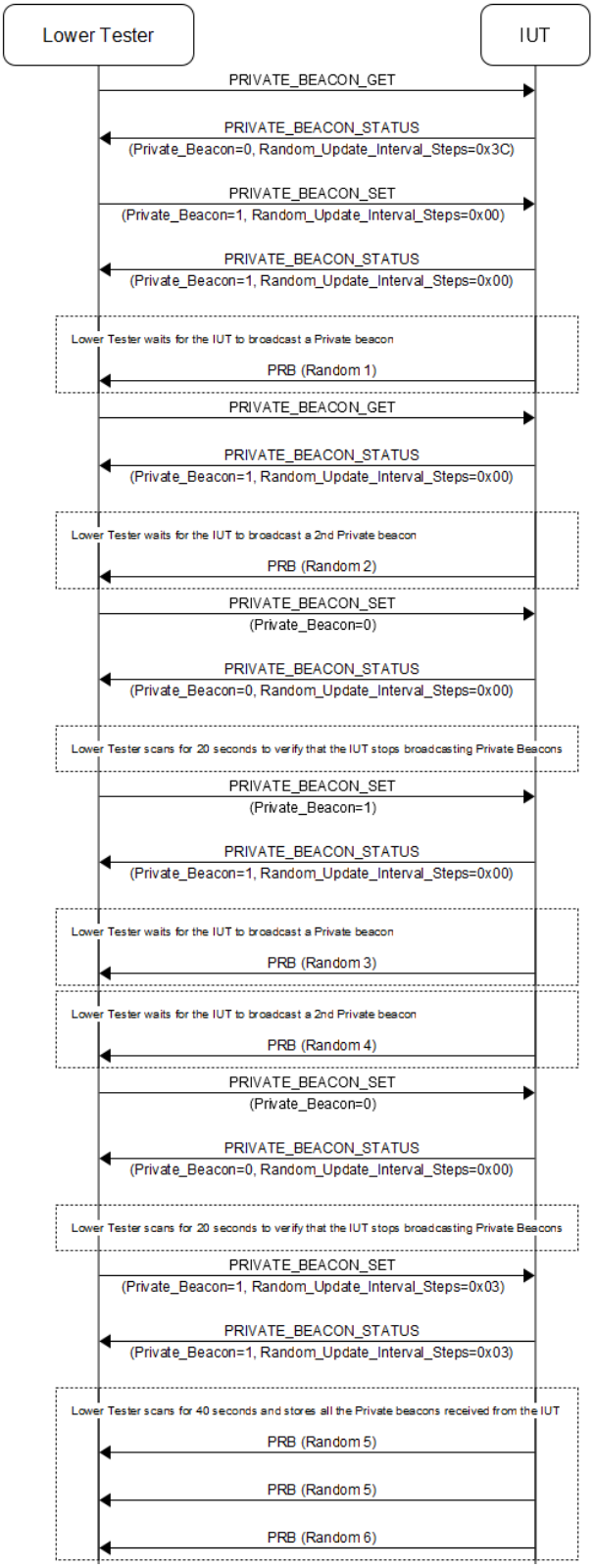


Figure 4.13: Test procedure for MESH/SR/PRB/PBS/BV-01-C MSC

1. The Lower Tester sends a PRIVATE_BEACON_GET message to the IUT.
2. The IUT responds with a PRIVATE_BEACON_STATUS with Private_Beacon set to 0 and Random_Update_Interval_Steps set to 0x3C.
3. The Lower Tester sends a PRIVATE_BEACON_SET message to the IUT with Private_Beacon set to 1 and Random_Update_Interval_Steps set to 0x00.
4. The IUT responds with a PRIVATE_BEACON_STATUS message with the fields set as requested in Step 3.
5. The Lower Tester waits for the IUT to broadcast a Private beacon on the shared subnet and stores the Random field.
6. The Lower Tester sends a PRIVATE_BEACON_GET message to the IUT.
7. The IUT responds with a PRIVATE_BEACON_STATUS with the parameters set as specified in Step 3.
8. The Lower Tester waits for the IUT to broadcast a second Private beacon on the shared subnet and verifies that the received Random field is different than the stored one. The Lower Tester stores the Random field of the most recently received Private beacon.
9. The Lower Tester sends a PRIVATE_BEACON_SET message to the IUT with Private_Beacon set to 0 and with no Random_Update_Interval_Steps parameter.
10. The IUT responds with a PRIVATE_BEACON_STATUS message with Private_Beacon set to 0 and Random_Update_Interval_Steps set to 0x00.
11. The Lower Tester scans for 20 seconds to verify that the IUT stops broadcasting Private Beacons.
12. Repeat Steps 3–5 and 8-11, using in Step 3, Private_Beacon set to 1 and no Random_Update_Interval_Steps field.
13. Repeat Steps 3 and 4, using in Step 3, Private_Beacon set to 1 and Random_Update_Interval_Steps set to 0x03.
14. The Lower Tester scans for 40 seconds and stores all the Private beacons received from the IUT. From all the received private beacons, at least two use the same Random field and there is at least one beacon with a different Random field.

- Expected Outcome

  Pass verdict

  The IUT responds correctly to PRIVATE_BEACON_GET/SET messages.

  In Step 2, the IUT returns the correct default values of the Private Beacon state and of the Random Update Interval Steps state.

  PRIVATE_BEACON_GET/SET and PRIVATE_BEACON_STATUS messages are encrypted using the device key of the IUT.

  The IUT uses correct format and security for Mesh Private beacons.

  The advertising PDUs containing beacons with different Random field values also have different advertising addresses.

MESH/SR/PRB/PBS/BI-01-C [Ignore Invalid Private Beacon Set Messages]

- Test Purpose

Verify that the Private Beacon Server IUT ignores invalid Private Beacon Set messages.

- Reference

  [3] 4.3.12.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model on the primary Element.
  - The Lower Tester supports the Private Beacon Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

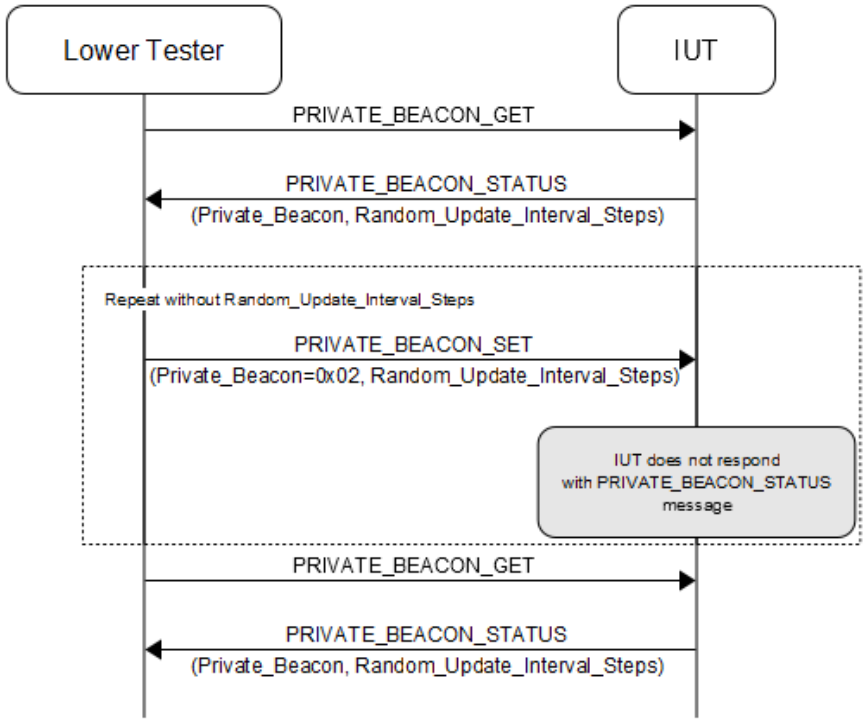


Figure 4.14: Test procedure for MESH/SR/PRB/PBS/BI-01-C MSC

1. The Lower Tester sends a PRIVATE_BEACON_GET message to the IUT.
2. The IUT responds with a PRIVATE_BEACON_STATUS message.
3. The Lower Tester sends a PRIVATE_BEACON_SET message to the IUT with a Private_Beacon field value of 0x02 (value is Prohibited) and any value for the Random_Update_Interval_Steps.
4. The IUT does not respond with any PRIVATE_BEACON_STATUS message.
5. Repeat Steps 3 and 4, using in Step 3, the same Private_Beacon field value and no Random_Update_Interval_Steps.
6. Repeat Steps 1 and 2.

- Expected Outcome

Pass verdict

In Steps 4 and 5, the IUT does not send any PRIVATE_BEACON_STATUS messages.

In Step 6, the Private_Beacon and the Random_Update_Interval_Steps fields in the PRIVATE_BEACON_STATUS message have the same value as the ones obtained in Step 2.

All configuration messages are encrypted using the device key of the IUT.

MESH/CL/PRB/PBS/BV-01-C [Send Private Beacon Get/Set messages]

- Test Purpose

  Verify that the Private Beacon Client IUT can send Private Beacon Get and Set messages.

- Reference

  [3] 4.4.12.2

- Initial Condition
  - The IUT supports the Private Beacon Client Model on the primary Element.
  - The Lower Tester supports the Private Beacon Server Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

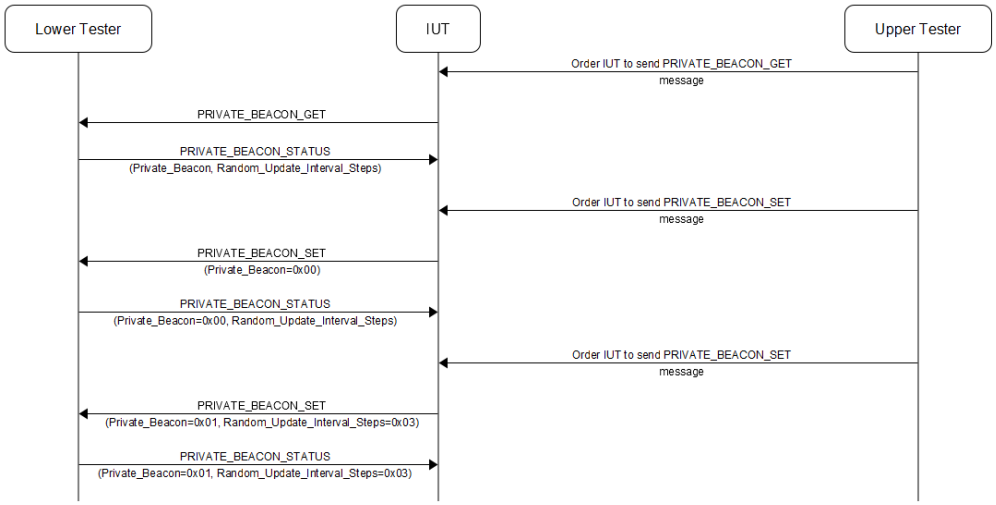


Figure 4.15: Test procedure for MESH/CL/PRB/PBS/BV-01-C MSC

1. The Upper Tester commands the IUT to send a PRIVATE_BEACON_GET message to the Lower Tester.
2. The IUT sends PRIVATE_BEACON_GET message and receives from the Lower Tester a PRIVATE_BEACON_STATUS message.
3. The Upper Tester commands the IUT to send a PRIVATE_BEACON_SET message to the Lower Tester with the Private_Beacon field value set to 0x00 and no Random_Update_Interval_Steps field.
4. The IUT sends PRIVATE_BEACON_SET message and receives from the Lower Tester a PRIVATE_BEACON_STATUS message with the Private_Beacon field value set to 0x00 and the Random_Update_Interval_Steps field value set to the same value as in Step 2.
5. The Upper Tester commands the IUT to send a PRIVATE_BEACON_SET message to the Lower Tester with the Private_Beacon field value set to 0x01 and the Random_Update_Interval_Steps field value set to 0x03.
6. The IUT sends PRIVATE_BEACON_SET message and receives from the Lower Tester a PRIVATE_BEACON_STATUS message with the Private_Beacon field value set to 0x01 and the Random_Update_Interval_Steps field value set to the same value as in Step 5.

- Expected Outcome

  Pass verdict

  The IUT sends correctly formatted Private Beacon Get/Set messages and reports the Private Beacon Status message field values to the Upper Tester.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.6.2 Private GATT Proxy

This section validates the Private GATT Proxy procedures for both the Private Beacon Server Model and the Private Beacon Client Model.

MESH/SR/PRB/PGPXY/BV-01-C [Respond to Private GATT Proxy Set/Get Messages]

- Test Purpose

  Verify that the Private Beacon Server IUT can respond to Private GATT Proxy Set and Get messages.

- Reference

  [10] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model and the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Private Beacon Client Model and the Configuration Client Model.
  - If the IUT supports the Proxy feature, the GATT Proxy state is set to 0x00 ("The Proxy feature is supported and disabled").
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
  - The Private GATT Proxy state is set to its default value after provisioning.
- Test Procedure
  1. The Lower Tester sends a PRIVATE_GATT_PROXY_GET message to the IUT.
  2. The IUT responds with a PRIVATE_GATT_PROXY_STATUS message with the Private_GATT_Proxy field set to:
     a. 0x00 ("The Private Proxy feature is supported and disabled"), if the IUT supports the Proxy feature.
     b. 0x02 ("The Proxy feature is not supported"), if the IUT does not support the Proxy feature.
  3. The Lower Tester sends a PRIVATE_GATT_PROXY_SET message to the IUT with the Private_GATT_Proxy field set to 0x00 ("The Private Proxy feature is supported and disabled").
  4. The IUT responds with a PRIVATE_GATT_PROXY_STATUS message with the Private_GATT_Proxy field set to:
     a. 0x00 ("The Private Proxy feature is supported and disabled"), if the IUT supports the Proxy feature.
     b. 0x02 ("The Proxy feature is not supported"), if the IUT does not support the Proxy feature.
  5. Repeat Step 3, using Private_GATT_Proxy field set to 0x01 ("The Private Proxy feature is supported and enabled").
  6. The IUT responds with a PRIVATE_GATT_PROXY_STATUS message with the Private_GATT_Proxy field set to:
     a. 0x01 ("The Private Proxy feature is supported and enabled"), if the IUT supports the Proxy feature. The IUT begins advertising with Private Network Identity.
     b. 0x02 ("The Proxy feature is not supported"), if the IUT does not support the Proxy feature.

7. The Lower Tester sends:
    a. A Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01 ("The Proxy feature is supported and enabled"), if the IUT supports the Proxy feature.
    b. A Config GATT Proxy Get message to the IUT, if the IUT does not support the Proxy feature.
8. The IUT responds with a Config GATT Proxy Status with the GATTProxy field set to:
    a. 0x01 ("The Proxy feature is supported and enabled"), if the IUT supports the Proxy feature.
    b. 0x02 ("The Proxy feature is not supported"), if the IUT does not support the Proxy feature.
9. If the IUT responds as specified in 8a, repeat Steps 1 and 2, otherwise skip this step.
10. If Step 9 is executed, the Lower Tester expects the IUT to stop advertising with Private Network Identity and start advertising with Network ID.

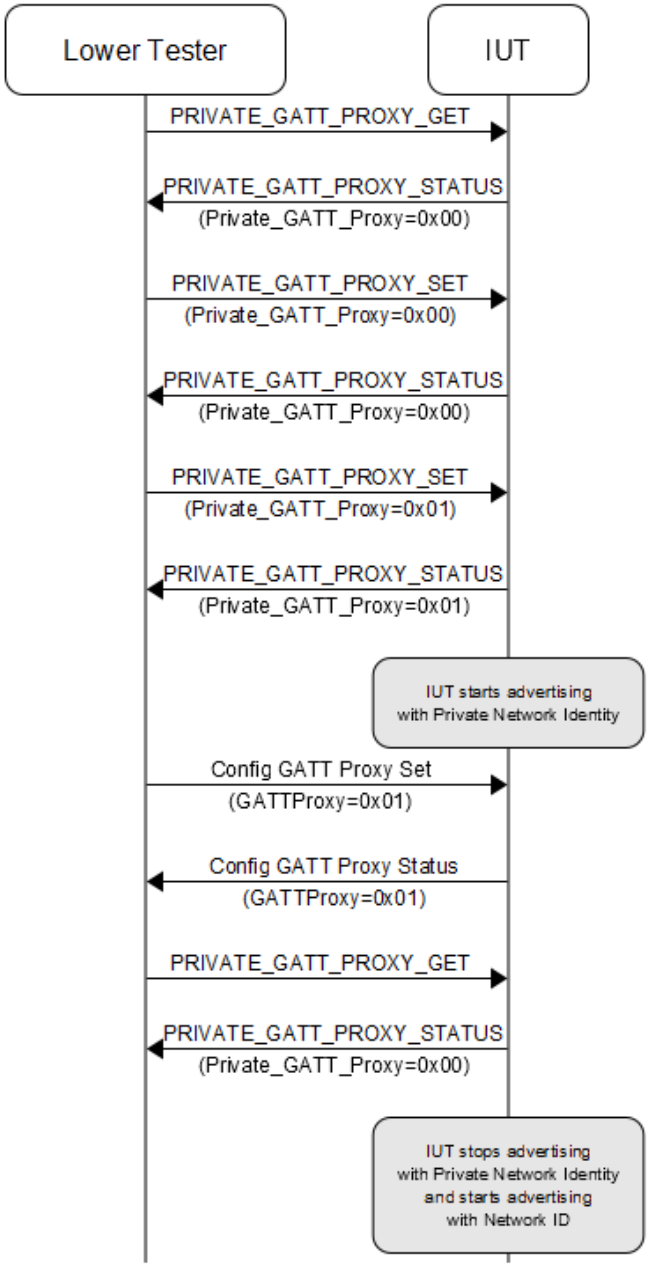


Figure 4.16: Test procedure for MESH/SR/PRB/PGPXY/BV-01-C -Proxy feature supported MSC

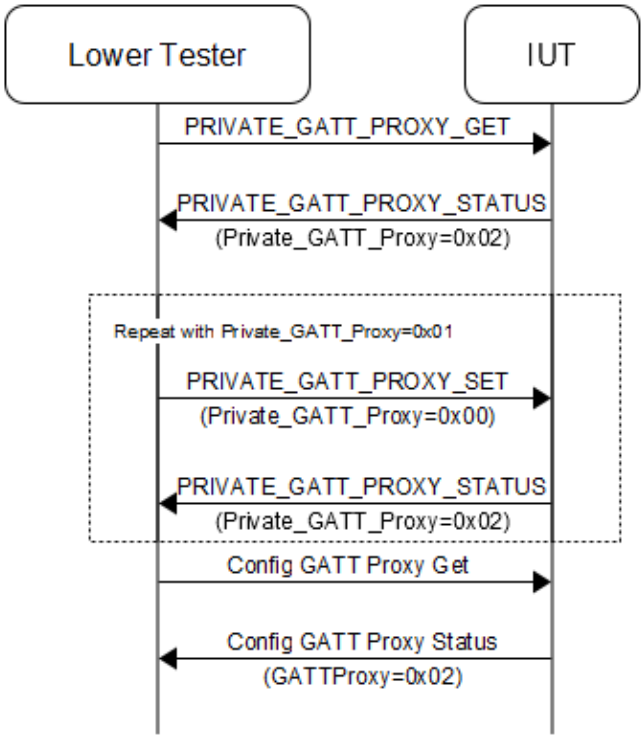


Figure 4.17: Test procedure for MESH/SR/PRB/PGPXY/BV-01-C -Proxy feature not supported MSC

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages.

  In Step 2, the IUT responds with the default value of the Private GATT Proxy state.

  If Step 6a is executed, the IUT uses correct format and security for Private Network Identity advertising.

  If Step 9 is executed, the IUT responds with a PRIVATE_GATT_PROXY_STATUS message with the Private_GATT_Proxy field value set to 0x00, stops advertising with Private Network Identity, and starts advertising with Network ID.

  All configuration messages are encrypted using the device key of the IUT.

MESH/SR/PRB/PGPXY/BV-02-C [Private Network Identity Advertising on Multiple Subnets]

- Test Purpose

  Verify that the Private Beacon Server IUT can trigger Private Network Identity advertising based on the Private GATT Proxy state change and can interleave advertising for multiple Subnets.

- Reference

  [10] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model on the Primary Element.
  - The Lower Tester supports the Private Beacon Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Proxy feature.
  - The GATT Proxy state is set to 0x00.
  - The Private GATT Proxy state is set to 0x00.
- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The Lower Tester sends a PRIVATE_GATT_PROXY_SET message to the IUT with the Private_GATT_Proxy field set to 0x01.
  3. The IUT responds with a PRIVATE_GATT_PROXY_STATUS message and begins sending interleaved Private Network Identity advertising PDUs for both Subnets it supports.

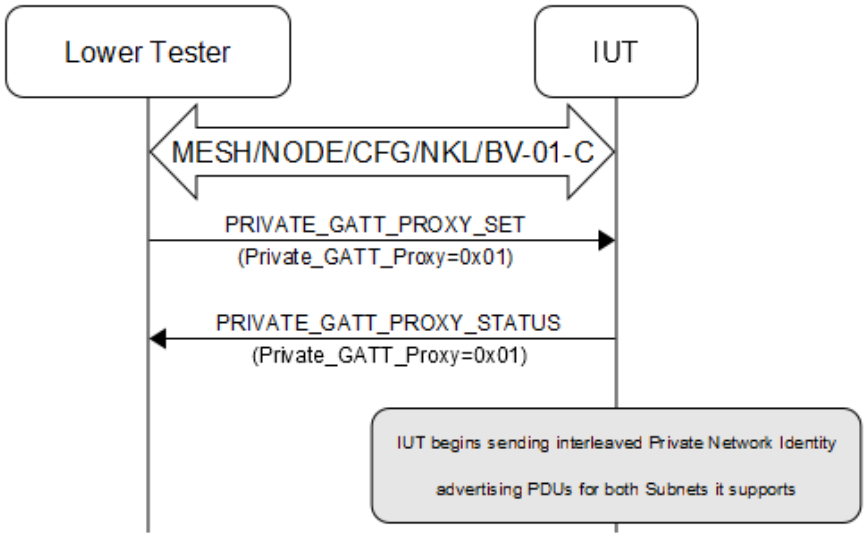


Figure 4.18: MESH/SR/PRB/PGPXY/BV-02-C MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a PRIVATE_GATT_PROXY_STATUS message with the Private_GATT_Proxy field set to 0x01. The IUT then begins advertising with Private Network Identity with interleaved packets for both Subnets it supports.

MESH/SR/PRB/PGPXY/BI-01-C [Respond to Private GATT Proxy Set/Get Messages – Invalid Parameters]

- Test Purpose

  Verify that the Private Beacon Server IUT ignores Private GATT Proxy Set messages with invalid field values.

- Reference

  [10] 4.4.11.2

- Initial Condition

  - The IUT supports the Private Beacon Server Model on the Primary Element.
  - The Lower Tester supports the Private Beacon Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a PRIVATE_GATT_PROXY_SET message to the IUT with the Private_GATT_Proxy field set to a Prohibited value (in the range 0x02-0xFF).
  2. The IUT ignores the message and does not send any response.

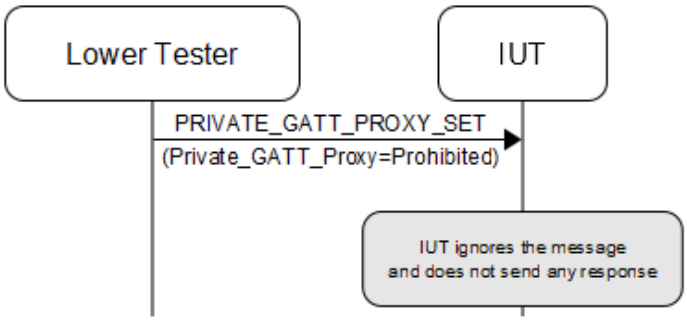


Figure 4.19: MESH/SR/PRB/PGPXY/BI-01-C MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any message in response to the PRIVATE_GATT_PROXY_SET message with invalid parameters sent by the Lower Tester in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CL/PRB/PGPXY/BV-01-C [Send Private GATT Proxy Set/Get Messages]

- Test Purpose

Verify that the Private Beacon Client IUT can send Private GATT Proxy Get and Set messages to a Private Beacon Server.

- Reference

  [10] 4.4.12.2

- Initial Condition
  - The IUT supports the Private Beacon Client Model.
  - The Lower Tester supports the Private Beacon Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a PRIVATE_GATT_PROXY_GET message to the Lower Tester.
  2. The Lower Tester receives PRIVATE_GATT_PROXY_GET message.
  3. The Lower Tester sends a PRIVATE_GATT_PROXY_STATUS message in response to the IUT with a valid Private_GATT_Proxy field value.
  4. The Upper Tester orders the IUT to send a PRIVATE_GATT_PROXY_SET message to the Lower Tester with a Private_GATT_Proxy field value of 0x01.
  5. The Lower Tester receives a PRIVATE_GATT_PROXY_SET message with the Private_GATT_Proxy field value set to the value requested in Step 4.
  6. The Lower Tester sends a PRIVATE_GATT_PROXY_STATUS message in response to the IUT with the Private_GATT_Proxy field value matching the value from the request.

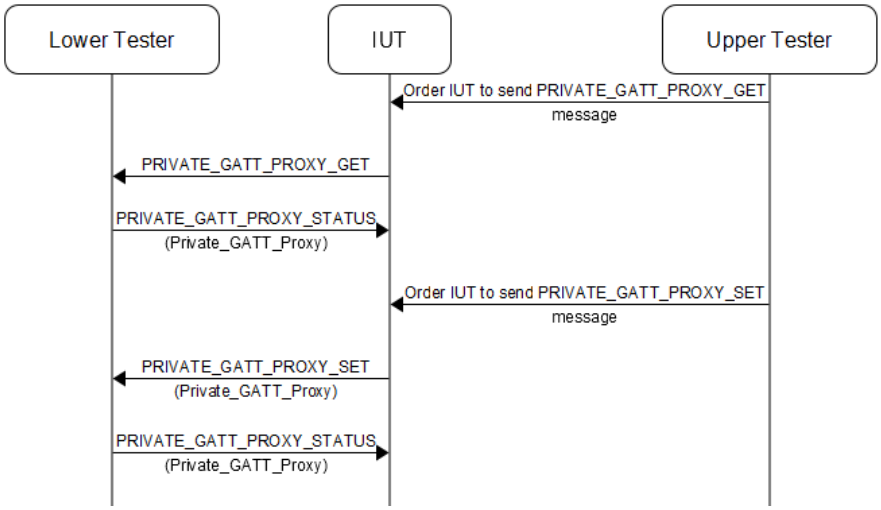


Figure 4.20: MESH/CL/PRB/PGPXY/BV-01-C MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a PRIVATE_GATT_PROXY_GET message to the Lower Tester.

  In Step 5, the IUT sends a PRIVATE_GATT_PROXY_SET message to the Lower Tester with the Private_GATT_Proxy field value set to the value requested in Step 4.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.6.3 Private Node Identity

This section validates the Private Node Identity procedures for both the Private Beacon Server Model and the Private Beacon Client Model.

MESH/SR/PRB/PNID/BV-01-C [Respond to Private Node Identity Set/Get Messages]

- Test Purpose

  Verify that the Private Beacon Server IUT can respond to Private Node Identity Get and Set messages.

- Reference

  [3] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model and the Configuration Server Model on the Primary Element.
  - The IUT supports the Proxy feature and exposes the Mesh Proxy Service.
  - The Lower Tester supports the Private Beacon Client Model and the Configuration Client Model.

- The Node Identity state on the IUT is set to 0x00.
- The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The Private Identity state is set to its default value after provisioning.

• Test Procedure

1. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message to the IUT with the NetKeyIndex field set to 0x000.
2. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the NetKeyIndex field set to 0x000, and the Private_Identity field set to 0x00 and setting the Status field to 0x00.
3. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message with the NetKeyIndex field set to 0x000 and the Private_Identity field set to 0x00, which means Mesh Proxy Service is exposed, but Private Node Identity for a Subnet is stopped.
4. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the NetKeyIndex field set to 0x000, the Private_Identity field set to 0x00, and the Status field set to 0x00.
5. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message to the IUT with the same NetKeyIndex used in Step 1.
6. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Private_Identity field and the NetKeyIndex field used in Step 3 and setting the Status field to 0x00.
7. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message with the NetKeyIndex field set to 0x000 and the Private_Identity field set to 0x01, which means Mesh Proxy Service is supported, and Private Node Identity for a Subnet is running.
8. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Private_Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
9. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message with the same NetKeyIndex used in Step 7 to the IUT.
10. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Private_Identity field and the NetKeyIndex field used in Step 7 and setting the Status field to 0x00.
11. The Lower Tester sends a Config Node Identity Set message with the NetKeyIndex field set to 0x000 and the Identity field set to 0x01, which means Mesh Proxy Service is supported, and Node Identity for a Subnet is running.
12. The IUT responds with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
13. Repeat Steps 1 and 2.

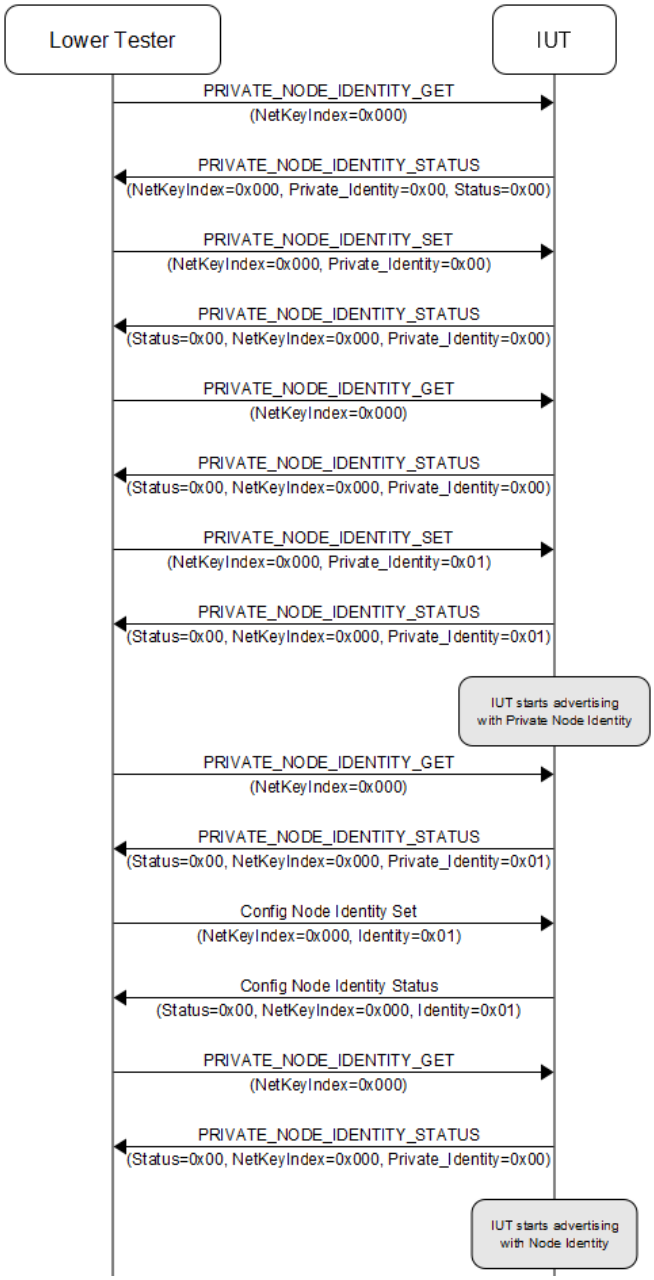


Figure 4.21: MESH/SR/PRB/PNID/BV-01-C MSC

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages.

  In Step 2, the IUT responds with the default value of the Private Node Identity state.

  After Step 8, the IUT starts advertising with Private Node Identity.

  In Step 13, the IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with Private_Identity field value set to 0x00 due to the binding between Private Node Identity state and the Node Identity state.

After Step 12, the IUT stops advertising with Private Node Identity and starts advertising with Node Identity.

All messages are encrypted using the device key of the IUT.

The IUT uses correct format and security for Private Node Identity advertising.

MESH/SR/PRB/PNID/BV-02-C [Private Node Identity Advertising on Multiple Subnets]

- Test Purpose

  Verify that Private Beacon Server IUT can trigger Private Node Identity advertising based on the Private Node Identity state change and can interleave advertising for multiple Subnets.

- Reference

  [3] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model and the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Private Beacon Client Model and the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT exposes the Mesh Proxy Service.
  - The Node Identity state on the IUT is set to 0x00.
  - The IUT supports multiple subnets.

- Test Procedure

  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message to the IUT with the NetKeyIndex field set randomly to 0x000 or 0x001 and the Private_Identity field set to 0x01.
  3. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message indicating the state change and begins advertising with Private Node Identity only for the Subnet identified by the NetKeyIndex sent in Step 2.
  4. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message to the IUT with the NetKeyIndex field set to the index of the network key other than the one used in Step 2.
  5. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Private_Identity field set to 0x00.
  6. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message to the IUT with the NetKeyIndex field set to the same value as in Step 2 and the Private_Identity field set to 0x00.
  7. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message indicating the state change and stops advertising with Private Node Identity.
  8. The Upper Tester orders the IUT to begin advertising with Private Node Identity, simulating user

interaction.

9. The Lower Tester expects the IUT to begin sending interleaved Private Node Identity advertising PDUs for both Subnets it supports.
10. The Lower Tester reads the Private Node Identity states for both network keys, and the IUT responds with the Private_Identity field value of 0x01 in both cases.
11. After 60 seconds of advertising, the IUT stops advertising with Private Node Identity.
12. The Lower Tester reads the Private Node Identity states for both network keys and the IUT responds with the Private_Identity field value of 0x00 in both cases.

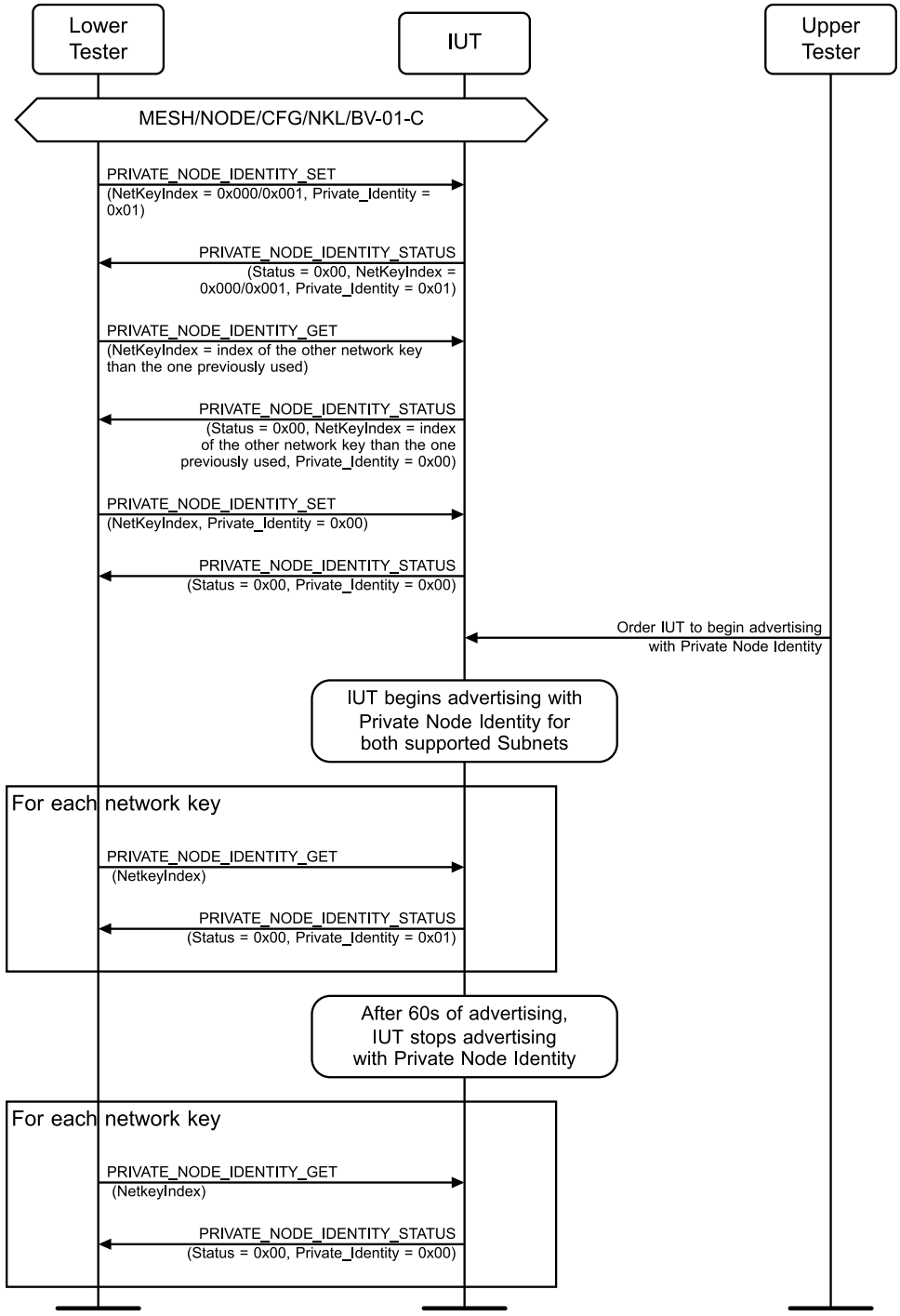


Figure 4.22: MESH/SR/PRB/PNID/BV-02-C MSC

- Expected Outcome

  Pass verdict

  The IUT responds to all messages as expected and advertises on the correct subnets.

MESH/SR/PRB/PNID/BV-03-C [Private Node Identity NetKey Deleted]

- Test Purpose

  Verify that a Private Beacon Server IUT sets the Private Node Identity state to 0x00 when NetKey is deleted.

- Reference

  [3] 4.4.11.2

- Initial Condition
  - The IUT supports the Private Beacon Server Model and the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Private Beacon Client Model and the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The Node Identity state on the IUT is set to 0x00 for all the subnets.
  - The IUT exposes the Mesh Proxy Service.
  - The IUT supports multiple subnets.
- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT with NetKeyIndex 1.
  2. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message to the IUT with the NetKeyIndex field set to 0x001 and the Private_Identity field set to 0x01.
  3. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Status field set to 0x00, the NetKeyIndex field set to 0x001, and the Private_Identity field set to 0x01.
  4. The Lower Tester deletes a NetKey from IUT's NetKey List, following the procedure described in MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]. The NetKeyIndex field is set to 0x001.
  5. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message to the IUT with the NetKeyIndex field set to 0x001.
  6. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Status field set to 0x04 (Invalid NetKey Index), the NetKeyIndex field set to 0x001, and the Private_Identity field set to 0x00.

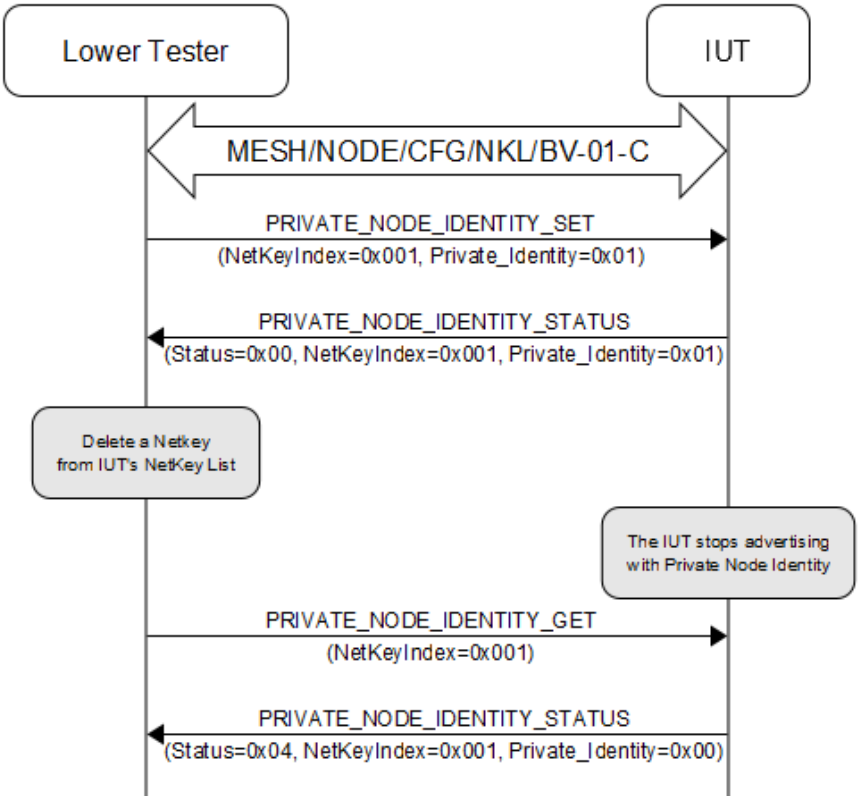


Figure 4.23: MESH/SR/PRB/PNID/BV-03-C MSC

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages.

  After Step 3, the IUT starts advertising with Private Node Identity for the subnet with NetKeyIndex 0x001.

  After Step 4, the IUT stops advertising with Private Node Identity for the subnet with NetKeyIndex 0x001.

  All messages are encrypted using the device key of the IUT.

MESH/SR/PRB/PNID/BI-01-C [Respond to Private Node Identity Set/Get Messages – Invalid Parameters]

- Test Purpose

  Verify that a Private Beacon Server IUT can respond to Private Node Identity Get and Set messages with invalid parameters.

- Reference

  [3] 4.3.12.7, 4.3.12.8

- Initial Condition

- The IUT supports the Private Beacon Server Model on the Primary Element.
- The Lower Tester supports the Private Beacon Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT exposes the Mesh Proxy Service.

• Test Procedure

1. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message with a NetKeyIndex which doesn't exist in the IUT's NetKey list, and Private_Identity field set to 0x00.
2. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Status field set to 0x04 (Invalid NetKey Index), and the NetKeyIndex and Private_Identity fields as defined in the incoming message in Step 1.
3. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message with the same invalid NetKeyIndex used in Step 1 to the IUT.
4. Repeat Step 2.
5. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message with a valid NetKeyIndex and the Private_Identity field set to a random number between 0x02 and 0xFF (Prohibited).
6. The IUT does not respond.
7. The Lower Tester sends a PRIVATE_NODE_IDENTITY_SET message with NetKeyIndex set to a random number between 0x8000 and 0xFFFF (Prohibited) and the Private_Identity field set to a valid value.
8. The IUT does not respond.
9. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message with NetKeyIndex set to a random number between 0x8000 and 0xFFFF (Prohibited).
10. The IUT does not respond.

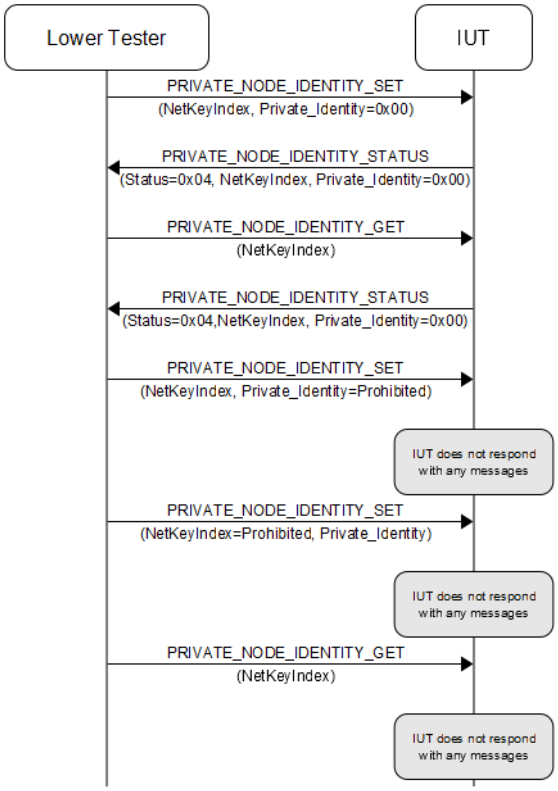


Figure 4.24: MESH/SR/PRB/PNID/BI-01-C MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, the IUT responds as expected.

  In Steps 6, 8, and 10, the IUT does not respond.

  All messages are encrypted using the device key of the IUT.

MESH/SR/PRB/PNID/BI-02-C [Receive Private Node Identity Set Message – Change to Not Supported]

- Test Purpose

  Verify that a Private Beacon Server IUT supporting the Private Node Identity feature ignores the Private Node Identity Set message to change the Private Node Identity to be not supported.

- Reference

  [3] 4.3.12.8

- Initial Condition

  - The IUT supports the Private Beacon Server Model on the Primary Element.

- The Lower Tester supports the Private Beacon Client Model.
- The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT exposes the Mesh Proxy Service and has the Node Identity and Private Node Identity states set to 0x00 for the supported Subnet.

- Test Procedure

  1. The Lower Tester sends PRIVATE_NODE_IDENTITY_SET message with a NetKeyIndex field identifying the supported Subnet and Private_Identity field set to 0x02, which means Private Node Identity feature is not supported.
  2. The IUT does not respond to any messages.
  3. The Lower Tester sends a PRIVATE_NODE_IDENTITY_GET message with the same NetKeyIndex used in Step 1 to the IUT.
  4. The IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Private_Identity field set to 0x00, the NetKeyIndex field as defined by the incoming message and the Status field set to 0x00.

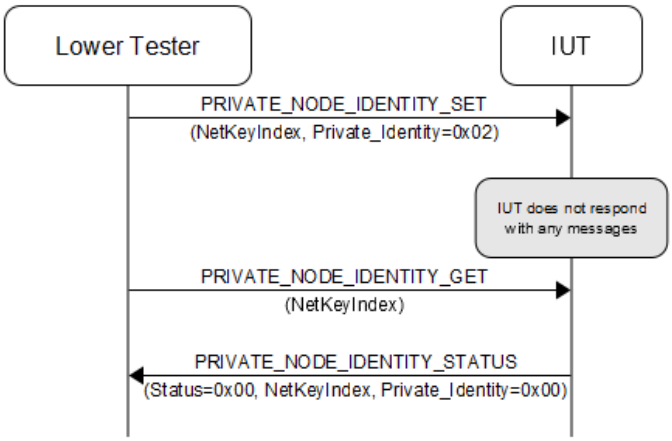


Figure 4.25: MESH/SR/PRB/PNID/BI-02-C MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not respond to any messages.

  In Step 4, the IUT responds with a PRIVATE_NODE_IDENTITY_STATUS message with the Status field set to 0x00, the NetKeyIndex field set to the value used in Step 1, and the Private_Identity field set to 0x00.

  All messages are encrypted using the device key of the IUT.

MESH/CL/PRB/PNID/BV-01-C [Send Private Node Identity Set/Get Messages]

- Test Purpose

Verify that an IUT node supporting the Private Beacon Client Model can send Private Node Identity Get and Set messages to a Private Beacon Server.

- Reference

  [3] 4.4.12.2

- Initial Condition
  - The IUT supports the Private Beacon Client Model.
  - The Lower Tester supports the Private Beacon Server Model.
  - The Lower Tester supports the Private Node Identity feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester commands the IUT to send a PRIVATE_NODE_IDENTITY_SET message to the Lower Tester with a NetKeyIndex field identifying the Subnet to which the IUT is provisioned, and the Private_Identity field set to 0x00.
  2. The Lower Tester receives a PRIVATE_NODE_IDENTITY_SET message with the fields set with the values defined in Step 1.
  3. The Lower Tester responds with a PRIVATE_NODE_IDENTITY_STATUS message.
  4. Repeat Steps 1–3 using, in Step 1, a Private_Identity field value of 0x01.
  5. The Upper Tester orders the IUT to send a PRIVATE_NODE_IDENTITY_GET message to the Lower Tester with the same NetKeyIndex value as in Step 1.
  6. The Lower Tester receives a PRIVATE_NODE_IDENTITY_GET message with the fields set with the values defined in Step 5.
  7. The Lower Tester responds with a PRIVATE_NODE_IDENTITY_STATUS message.

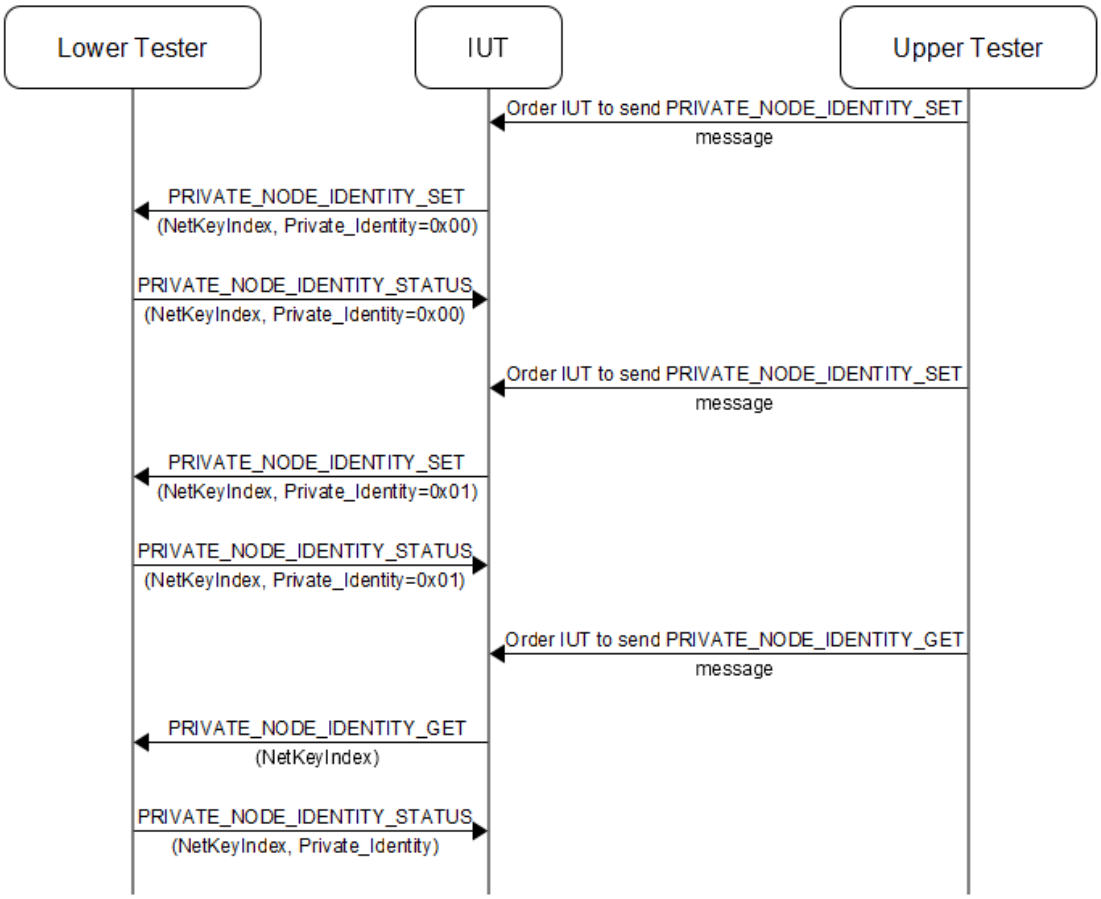


Figure 4.26: MESH/CL/PRB/PNID/BV-01-C MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a properly formatted PRIVATE_NODE_IDENTITY_SET message with a valid NetKeyIndex field and the Private_Identity field value set to 0x00.

  In Step 4, the IUT sends a properly formatted PRIVATE_NODE_IDENTITY_SET message with the same NetKeyIndex field value as in Step 1 and the Private_Identity field value set to 0x01.

  In Step 6, the IUT sends a properly formatted PRIVATE_NODE_IDENTITY_GET message with the same NetKeyIndex field value as in Step 1.

  All messages are encrypted using the device key of the Lower Tester.

4.7 Mesh Beacons

The test group objective is to verify the behavior of Secure Network beacons and Mesh Private beacons.

4.7.1 Advertising Mesh Beacons

- Test Purpose

Verify that a Node IUT generates a valid Secure Network beacon or a valid Mesh Private beacon. Confirm that the beacon is advertised and can be received by the Lower Tester. Confirm that the advertising payload is properly formatted. Confirm that a non-connectable advertising packet type is utilized.

- Reference

  [3] 3. 10.4

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - IUT and the Lower Tester share previously established network security credentials.
- Test Procedure
  1. The Upper Tester orders the IUT to generate and advertise a beacon with AD Type set to «Mesh Beacon», the Beacon Type field set per the Beacon Type column in Table 4.7, and the Flag AD Type is not included in Beacon Data (non-connectable).
  2. The Lower Tester scans for and receives the advertising packets sent by the IUT.
  3. The Lower Tester observes the advertisement by the IUT and validates the payload.
- Test Case Configuration

Test Case	Beacon Type
MESH/NODE/BCN/SNB/BV-01-C [Advertising Secure Network Beacons]	0x01 (Secure Network beacon)
MESH/NODE/BCN/PRB/BV-01-C [Advertising Mesh Private Beacons]	0x02 (Mesh Private beacon)

Table 4.7: Advertising Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  The IUT sends a beacon with the AD Type set to «Mesh Beacon» and the Beacon Type field set as defined in Table 4.7, and the Flag AD Type is not included in Beacon Data (non-connectable).

  All multiple-octet numeric values in mesh beacons are sent in "network byte order" defined as "big endian".

MESH/NODE/BCN/PRB/BV-02-C [Scanning for Mesh Beacons]

- Test Purpose

  Verify that a Node IUT receives Secure Network beacons and Mesh Private beacons irrespective of the Secure Network Beacon and Mesh Private Beacon states.

- Reference

  [3] 3.3.1, 3.9.1, 3.9.3, 3.9.3.1, 3.10.4

- Initial Condition
  - The IUT supports the Configuration Server Model and the Private Beacon Server Model.
  - The Lower Tester supports the Configuration Client Model and the Private Beacon Client Model.
  - The IUT and the Lower Tester share previously established network security credentials.
- Test Procedure

Repeat Steps 1–5 for each round in Table 4.8.

1. The Lower Tester configures the Secure Network Beacon state and the Mesh Private Beacon state on the IUT as described in Table 4.8.
2. The Lower Tester generates and advertises a Secure Network beacon with AD Type set to «Mesh Beacon» and the Beacon Type field set to 0x01 (Secure Network beacon), and the Flag AD Type is not included in Beacon Data (non-connectable).
3. The IUT scans for and receives the advertising packets sent by the Lower Tester. The IUT validates the Secure Network beacon and reports it to the Upper Tester.
4. The Lower Tester generates and advertises a Mesh Private beacon with AD Type set to «Mesh Beacon» and the Beacon Type field set to 0x02 (Mesh Private beacon), the Flag AD Type is not included in Beacon Data (non-connectable), and the Random field is set to a unique value in each round. The beacon is secured using the shared network security credentials.
5. The IUT scans for and receives the advertising packets sent by the Lower Tester. The IUT validates the Mesh Private beacon and reports it to the Upper Tester.

Round	States
1	Secure Network Beacon state = 0x01. Mesh Private Beacon state = 0x01.
2	Secure Network Beacon state = 0x01. Mesh Private Beacon state = 0x00.
3	Secure Network Beacon state = 0x00. Mesh Private Beacon state = 0x00.
4	Secure Network Beacon state = 0x00. Mesh Private Beacon state = 0x01.

Table 4.8: Scanning for Mesh Beacons rounds

- Expected Outcome

  Pass verdict

  In Steps 4 and 6, the IUT receives and authenticates the beacons sent by the Lower Tester.

4.8 Network Layer

The test group objective is to verify functionality of the network layer. Test functionality specific to the ADV bearer for the network layer.

4.8.1 Transmit Mesh Messages over Advertising Bearer to Different Destination Address Types

- Test Purpose

  This test group is for generic use and contains seven test cases to verify that a node can transmit non-connectable mesh message AD Type advertisements over the advertising bearer in the context of a mesh Subnet, using previously established network security credentials, to each of the possible Destination Address types (unicast, group, or virtual). The verification is done one value at a time, as enumerated in the test cases in Table 4.9.

  The test cases include verification that a Node can send advertisements using TTL values varying from minimum to maximum allowable.

- Reference

  [3] 3.3.1, 3.4.2, 3.4.4, 3.4.6.3, 3.8.6.2, 3.8.6.3

- Initial Condition
  - The IUT is a Node.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a Node.
  - The Lower Tester supports the GAP Observer role.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT's output filter allows any messages specified to pass.
  - The Network Transmit state on the IUT is equal to 0x00.
  - Condition specified for this test case in Table 4.9.

- Test Procedure

  1. The Lower Tester begins scanning for advertising packets from the IUT.
  2. The Upper Tester sends a command to the IUT to initiate advertising using non-connectable advertising packets with a payload that consists an AD Type value of «mesh message», a payload length of 29 bytes, and a valid mesh packet as payload. The fields of the packet are set as follows, with all multiple-octet values sent in "big endian", with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from shared Network Key), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IVI fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from shared Network Key) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):

     - IVI set to the least significant bit of the current IV Index
     - NID set to the least 7 significant bits of the Network Key Identifier of the Network Key
     - CTL set to 0
     - TTL set to 0x00

- SEQ set to any value between 0x000000 and 0xFFFFFF inclusive
- SRC set to one of the IUT's local unicast addresses
- DST set to the specified value in Table 4.9 for this Test Case
- TransportPDU set to between 1 and 16 inclusive octets of data
- NetMIC set to a 32-bit value generated by the IUT

3. The Lower Tester expects an advertising packet from the IUT containing the expected AD Type and payload. The Lower Tester examines the IVI and NID fields to confirm they are set using the expected security credentials. The Lower Tester confirms the packet is authenticated using the previously established network security credentials and the NetMIC field is set correctly.
4. Repeat Steps 2–3 with a TTL value of 10 in Step 2.
5. Repeat Steps 2–3 with a TTL value of 127 in Step 2.
6. The Lower Tester sends a Config Network Transmit Set message to the IUT with the NetworkTransmitCount set to 0b010 and the NetworkTransmitIntervalSteps set to 0b10000.
7. The Lower Tester expects the IUT to respond with a Config Network Transmit Status message containing the field values requested in Step 6.
8. The Upper Tester orders the IUT to send a Network PDU with parameters as described in Step 2, using any valid DST field value defined in Table 4.9.
9. The Lower Tester expects the IUT to send the requested Network PDU 3 times.

Test Case	Initial Condition	DST Value
MESH/NODE/NET/BV-01-C [Send Mesh Message to Unicast Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a unicast address.	any value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
MESH/NODE/NET/BV-02-C [Send Mesh Message to Virtual Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a virtual address based on a virtual label UUID.	any value between 0x8000 and 0xBFFF inclusive
MESH/NODE/NET/BV-03-C [Send Mesh Message to Dynamically Assigned Group Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to a dynamically assigned group address.	any value between 0xC000 and 0xFEFF inclusive
MESH/NODE/NET/BV-04-C [Send Mesh Message to All-proxies Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-proxies address.	0xFFFC
MESH/NODE/NET/BV-05-C [Send Mesh Message to All-friends Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-friends address.	0xFFFD

Test Case	Initial Condition	DST Value
MESH/NODE/NET/BV-06-C [Send Mesh Message to All-relays Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-relays address.	0xFFFE
MESH/NODE/NET/BV-07-C [Send Mesh Message to All-nodes Address]	The layer above the Network Layer on the IUT (the Transport Layer) has attempted to send a message to the all-nodes address.	0xFFFF

Table 4.9: Transmit Mesh Messages over Advertising Bearer test cases

- Expected Outcome

  Pass verdict

  The IUT sends an advertising packet to the Lower Tester with the correct AD type and the expected payload, with all fields containing valid values and formatted correctly.

  Any message sent by the IUT is delivered to all network interfaces supported by the node. The IVI and NID fields are set using the appropriate network security credentials.

  The NetMIC field is set correctly using the appropriate network security credentials.

  The SEQ field value is unique for each Network PDU sent by the IUT.

  All multiple-octet numeric values are in "big endian" byte order.

  In Step 7, the IUT sends with a Config Network Transmit Status message containing the field values requested in Step 7.

  In Step 9, the IUT sends the Network PDU 3 times.

4.8.2 Receive Mesh Messages over Advertising Bearer Sent to Different Destination Address Types

- Test Purpose

  This test group is for generic use and contains seven test cases to verify that a node can receive authenticated non-connectable mesh message AD Type advertisements over the advertising bearer in the context of a mesh Subnet, using previously established network security credentials, to each of the possible Destination Address types (unicast, group, or virtual). The verification is done one value at a time, as enumerated in the test cases in Table 4.10.
  The test cases include verification that a Node can receive advertisements using TTL values varying from minimum to maximum allowable.

- Reference

  [3] 3.3.1, 3.4.2, 3.4.4, 3.4.6.2, 3.8.6.2, 3.8.6.3

- Initial Condition
  - The IUT is a Node.
  - The IUT supports the GAP Observer role.
  - The Lower Tester is a Node.
  - The Lower Tester supports the GAP Broadcaster role.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT's network layer input filter allows any messages specified to pass.
  - Condition specified for this Test Case in Table 4.10.
- Test Procedure
  1. The Lower Tester initiates advertising using non-connectable advertising packets with a payload that consists of the mesh message AD Type value, and the Contents containing a Network PDU of maximum size (29 bytes).
  2. The Upper Tester sends a command to the IUT to scan for advertising packets and receives the advertising packets sent by the Lower Tester.
  3. The Upper Tester confirms that the IUT receives one or more non-connectable advertising packets containing the expected AD Type and payload and notifies the Upper Tester of packet receipt.
  4. The Lower Tester sends a network packet with fields set as follows, with all multiple-octet values sent in "big endian", with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from shared Network Key), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IVI fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from shared Network Key) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):
     - IVI set to the least significant bit of the current IV Index
     - NID set to the least 7 significant bits of the Network Key Identifier of the Network Key
     - CTL set to 0
     - TTL set to 0x00
     - SEQ set to any value between 0x000000 and 0xFFFFFF inclusive
     - SRC set to the Lower Tester's address, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
     - DST set to the DST Value for this test case in Table 4.10
     - TransportPDU set to between 1 and 16 inclusive octets of data
     - NetMIC set to a 32-bit value generated by the Lower Tester
  5. Repeat Step 4 with a TTL value of 1.
  6. Repeat Step 4 with a TTL value of 10.
  7. Repeat Step 4 with a TTL value of 127.

Test Case	Initial Condition	DST Value
MESH/NODE/NET/BV-08-C [Receive Mesh Message Sent to Unicast Address]	N/A	one of the IUT's local unicast addresses, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address
MESH/NODE/NET/BV-09-C [Receive Mesh Message Sent to Virtual Address]	The Network Layer on the IUT knows a virtual address based on a virtual label UUID.	a virtual address the IUT knows, a value between 0x8000 and 0xBFFF inclusive (a valid virtual address), where the 14 least significant bits are set to the 14 least significant bits of the result generated by AES-CMAC (Virtual Label UUID, "vtad")
MESH/NODE/NET/BV-10-C [Receive Mesh Message sent to Dynamic Group Address]	The Network Layer on the IUT knows a dynamic group address.	a group address the IUT knows, a value between 0xC000 and 0xFEFF inclusive.
MESH/NODE/NET/BV-11-C [Receive Mesh Message sent to All-proxies Address]	The IUT supports the proxy feature and it is enabled.	0xFFFC
MESH/NODE/NET/BV-12-C [Receive Mesh Message sent to All-friends Address]	The IUT supports the friend feature and it is enabled.	0xFFFD
MESH/NODE/NET/BV-13-C [Receive Mesh Message sent to All-relays Address]	The IUT supports the relay feature and it is enabled.	0xFFFE
MESH/NODE/NET/BV-14-C [Receive Mesh Message sent to All-nodes Address]	N/A	0xFFFF

Table 4.10: Receive Mesh Messages over Advertising Bearer test cases

- Expected Outcome

  Pass verdict

  The IUT receives an advertising packet from the Lower Tester with the correct AD Type and the expected payload, with all fields containing valid values and formatted correctly. All encrypted fields are authenticated successfully.

  The NID field value matches an NID known by the IUT.

  If the mesh message received is sent to a unicast address, it is processed by only one Element.

  If the mesh message received is sent to a group or a virtual address, it is processed by all Elements subscribed to this address.

MESH/NODE/NET/BI-01-C [Ignore Connectable Advertisements on ADV Bearer]

- Test Purpose

Verify that a Node IUT silently ignores mesh packets sent in connectable advertisement PDUs.

- Reference

  [3] 3.3.1

- Initial Condition
  - The IUT is Observer.
  - The Lower Tester is Broadcaster.
  - The IUT and the Lower Tester share previously established network security credentials.
- Test Procedure
  1. The Lower Tester initiates advertising using connectable advertising packets with a payload that consists of the Mesh Packet AD Type value, a payload length of 29 bytes, and a valid mesh packet as payload.
  2. The IUT scans for and receives the advertising packet(s) sent by the Lower Tester.
  3. The IUT silently ignores the mesh packets received in connectable advertising PDUs and does not notify the Upper Tester of their receipt.
- Expected Outcome

  Pass verdict

  The IUT ignores any messages received without delivering the Transport PDU to the lower transport layer.

MESH/NODE/NET/BI-02-C [Discard Messages with Invalid Addresses]

- Test Purpose

  Verify that a Node IUT can discard messages received with an invalid SRC or DST address.

- Reference

  [3] 3.4.4, 3.4.6.2

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.
  - If the IUT supports message relaying, relaying is enabled.
- Test Procedure
  1. The Lower Tester sends a message to the IUT with SRC set to a valid address and DST set to 0b0000000000000000.
  2. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.

3. The Lower Tester sends a message to the IUT with SRC set to the unassigned address and DST set to the IUT's address.
4. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
5. The Lower Tester sends a message to the IUT with SRC set to a virtual address the IUT knows and DST set to the IUT's address.
6. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
7. The Lower Tester sends a message to the IUT with SRC set to a virtual address the IUT does not know and DST set to the IUT's address.
8. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
9. The Lower Tester sends a message to the IUT with SRC set to a group address the IUT knows and DST set to the IUT's address.
10. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.
11. The Lower Tester sends a message to the IUT with SRC set to a group address the IUT does not know and DST set to the IUT's address.
12. The Upper Tester confirms that the message is ignored. If the IUT supports relaying, the Lower Tester expects to receive no relayed copy of the packet.

- Expected Outcome

  Pass verdict

  When the IUT receives a Mesh Packet with an invalid SRC or DST address, the IUT discards the message without delivering the Transport PDU to the lower transport layer.

  If the IUT supports message relaying, the message with invalid SRC or DST is not relayed.

MESH/NODE/NET/BI-03-C [Receive Network Packet with Invalid NID Field]

- Test Purpose

  Verify that if a Node IUT can ignore a message containing a NID that does not match any known NIDs.

- Reference

  [3] 3.3.1, 3.4.4, 3.4.6.2

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester emulates a Node.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Lower Tester sends a network packet with fields set as follows, with all multiple-octet values sent

in "big endian", but authenticated and encrypted using a Network Key that the IUT does not know; with DST and TransportPDU encrypted with AES-CCM using the Encryption Key (derived from a Network Key the IUT does not know), a new Network Nonce (Nonce Type set to 0x00, and the CTL and TTL, SEQ, SRC, and IV Index fields as set below), and with CTL, TTL, SEQ, and SRC obfuscated with AES-ECB using the Privacy Key (derived from a Network Key the IUT does not know) and a Privacy Random (generated from 0x0000000000, IV Index, encrypted DST, and encrypted TransportPDU):

- IVI set to the least significant bit of the current IV Index
- NID set to a value that does not match any NID values known to the IUT
- CTL set to 0
- TTL set to any value between 0x00 and 0x3F inclusive
- SEQ set to any value between 0x000000 and 0xFFFFFF inclusive
- SRC set to the Lower Tester's address, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
- DST set to one of the IUT's local unicast addresses, a value between 0x0001 and 0x7FFF inclusive (a valid unicast address)
- TransportPDU set to between 1 and 16 inclusive octets of data
- NetMIC set to the 4 most significant octets of the MIC generated by using AES-CCM to encrypt the DST and TransportPDU fields using the Encryption Key and Network Nonce

2. The Upper Tester expects the IUT's Network Layer to discard the packet without delivering the Transport PDU to the next upper layer (the Transport Layer).

- Expected Outcome

  Pass verdict

  The IUT's Network Layer receives the network packet, processes it, and discards it without delivering the Transport PDU to the Lower Transport layer.

4.9 Message Relaying

The test group objective is to verify functionality related to message relaying.

MESH/NODE/RLY/BV-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit]

- Test Purpose

  Verify that an IUT acting as Relay node forwards valid network PDUs selectively, based on the SRC, DST and TTL field values and on the Relay Retransmit state value.

- Reference

  [3] 3.8.4, 3.4.2.4, 3.4.4, 3.4.6.1, 3.4.6.3, 4.2.20, 4.3.2, 13, 4.3.2.14

- Initial Condition

  - The IUT and the Lower Tester share previously established network security credentials and a virtual

address.

- The IUT supports the Relay feature and the feature is enabled.
- The Relay Retransmit Count state on the IUT is equal to 0b000.
- The Lower Tester is configured to generate valid network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the SEQ field starting from 0 and increasing on each new PDU, and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. All network PDUs are encrypted using the shared network key.

- Test Procedure

1. The Lower Tester sends a network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to a unicast address not matching the Element addresses of the IUT.
2. The Lower Tester expects the IUT to send the network PDU received in Step 1 with the TTL field decremented by 1 and re-encrypted using the same shared network key.
3. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to 1, and the DST field set to a unicast address not matching the Element addresses of the IUT.
4. The Lower Tester expects the IUT to not forward the received network PDU.
5. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to 0, and the DST field set to a unicast address not matching the Element addresses of the IUT.
6. The Lower Tester expects the IUT to not forward the received network PDU.
7. Repeat Steps 1–6, using a new network PDU on each round and setting the DST as indicated in Table 4.11.

Round	DST value
1	Random value in the range 0xC000 to 0xFEFF
2	Random value in the range 0x8000 to 0xBFFF
3	Random Reserved for Future Use fixed group address value

Table 4.11: Additional DST values tested for MESH/NODE/RLY/BV-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit]

8. The Lower Tester sends a new network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to the unicast address of the IUT.
9. The Lower Tester expects the IUT to not forward the received network PDU.
10. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01, the RelayRetransmitCount field set to 0b001, and the RelayRetransmitIntervalSteps field set to 0b10000.
11. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b10000.
12. The Lower Tester sends a network PDU with the SRC field set to its own unicast address, the TTL field set to a random valid value greater than or equal to 2, and the DST field set to a unicast address not

matching the Element addresses of the IUT.

13. The Lower Tester expects the IUT to forward the network PDU received in Step 12, with the TTL field value decremented by 1, two times.

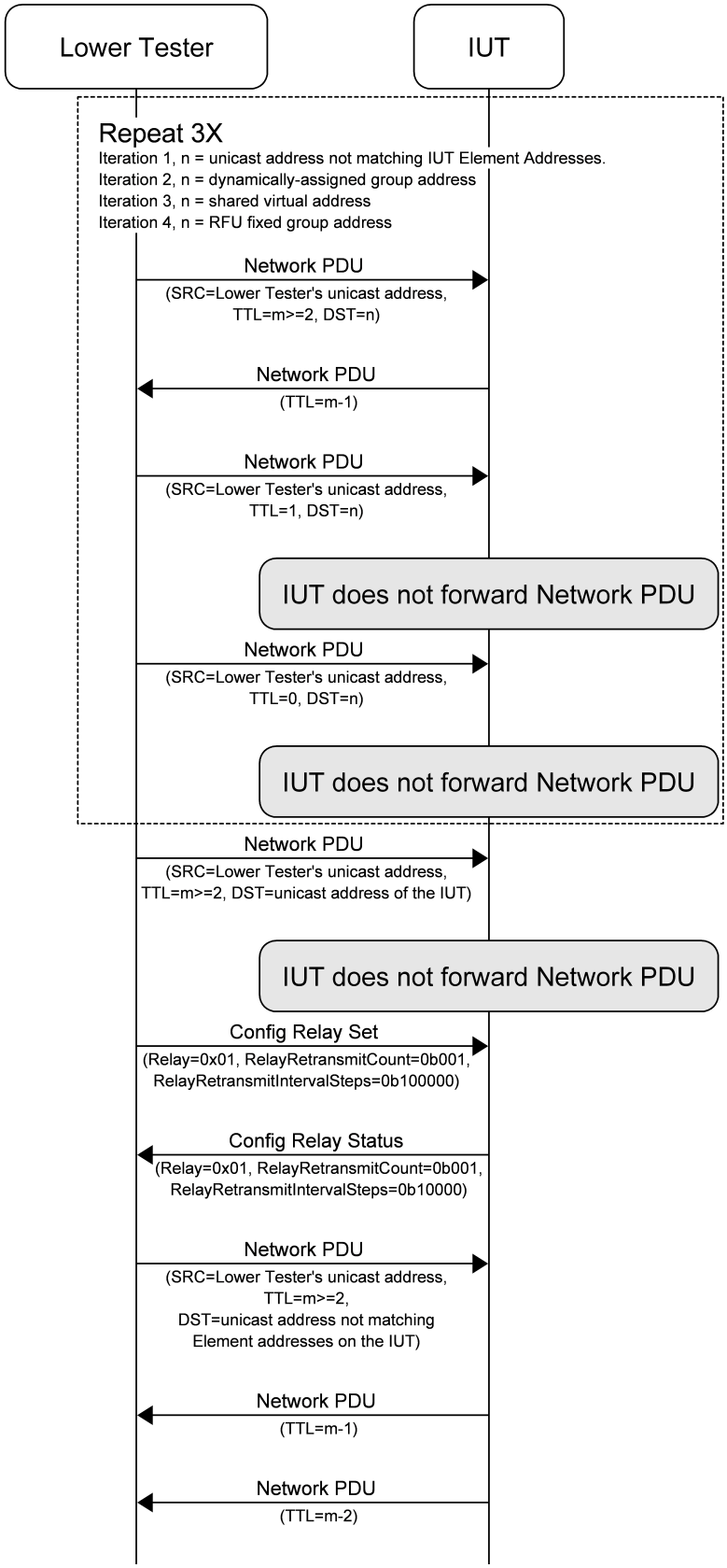


Figure 4.27: MESH/NODE/RLY/BV-01-C [Relay Network PDUs based on SRC, DST, TTL and Relay Retransmit] MSC

- Expected Outcome

  Pass verdict

  In each iteration through Step 2, the IUT sends a copy of the received network PDU, without re-transmit, with the TTL field value decremented by 1 and all other plain field values unchanged. The DST and the TransportPDU fields are re-encrypted and the NetMIC field contains the new generated MIC. The CTL, TTL, SEQ and SRC fields are properly obfuscated.

  In each iteration through Steps 4, 6, and 9, the IUT does not forward the received network PDU.

  In Step 13, the IUT sends a copy of the received network PDU, with the TTL field value decremented by 1, two times.

MESH/NODE/RLY/BV-02-C [Network Message Cache]

- Test Purpose

  Verify that an IUT acting as Relay node maintains a properly implemented message cache.

- Reference

  [3] 2.2, 3.4.4.4, 3.4.4.6, 3.4.6.1, 3.4.6.4

- Initial Condition
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT supports the Relay feature and the feature is enabled.
  - The maximum number of entries supported by the network message cache on the IUT has been supplied in the IXIT [6]. This number is denoted as N and is greater than or equal to 2.
  - The Lower Tester is configured to generate valid network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the TTL field set to a random valid value greater than or equal to 2 on each new PDU, the SRC field set to a unique unicast address (not matching the Element addresses of the IUT) on each new PDU, the SEQ field set randomly on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

- Test Procedure

  1. The Lower Tester sends a new network PDU.
  2. The Lower Tester expects the IUT to send the network PDU received in Step 1 with the TTL field decremented by 1.
  3. The Lower Tester resends the same network PDU as in Step 1.
  4. The Lower Tester expects the IUT to not forward the same network PDU a second time.
  5. The Lower Tester repeats Steps 1–2 an additional (N - 1) times.
  6. The Lower Tester repeats Steps 3–4 an additional N times, iterating through all N network PDUs that have been sent in the N passes through Step 1.
  7. The Lower Tester repeats Steps 1–4 one time.

8. The Lower Tester resends all the (N - 1) network PDUs sent in Step 5 and expects the IUT to not forward these PDUs.
9. The Lower Tester resends the same network PDU that it has sent in the first iteration through Step 1.
10. The Lower Tester expects the IUT to resend the network PDU that it sent in the first iteration through Step 2.
11. The Lower Tester sends a new network PDU with the defined field value generation procedure, but with the TTL field set to a random valid value greater than or equal to 4.
12. The Lower Tester expects the IUT to send the network PDU received in Step 11 with the TTL field decremented by 1.
13. The Lower Tester forwards the network PDU received in Step 12 after decrementing the TTL by 1.
14. The Lower Tester expects the IUT to not forward the network PDU received in Step 13.
15. The Upper Tester orders the IUT to send a valid network PDU with the DST field set to the all nodes address and the TTL field set to a random valid value greater than or equal to 4.
16. The Lower Tester expects the IUT to send a valid network PDU with the requested field values.
17. The Lower Tester forwards the network PDU received in Step 16 after decrementing the TTL by 1.
18. The Lower Tester expects the IUT to not forward the network PDU received in Step 17.

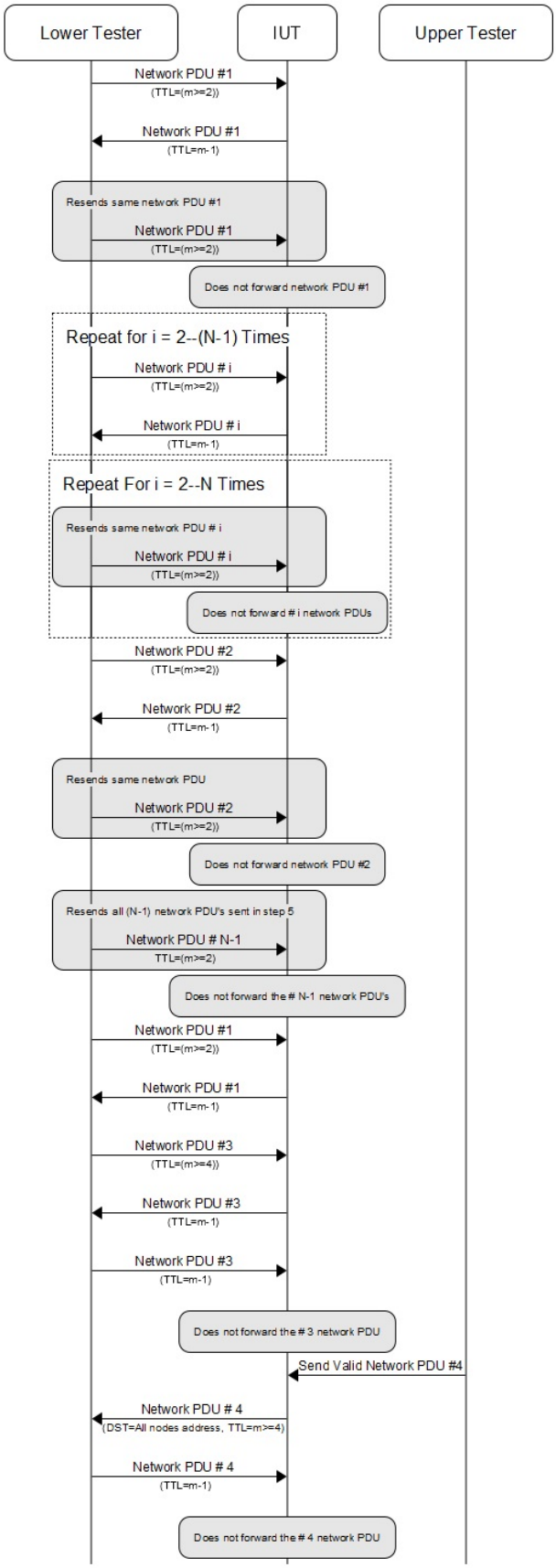


Figure 4.28: MESH/NODE/RLY/BV-02-C [Network Message Cache] MSC

- Expected Outcome

  Pass verdict

  In all passes through Step 2, the IUT forwards the received network PDU with the TTL field decremented by 1 and re-encrypted using the shared network key.

  In all passes through Step 4, the IUT does not forward the received network PDU.

  In Step 8, the IUT does not forward the received network PDUs, confirming that those are still in the cache.

  In Step 10, the IUT forwards the received network PDU, confirming that it has been removed from the cache in Step 7.

  In Step 12, the IUT forwards the received network PDU.

  In Step 14, the IUT does not forward the received network PDU, confirming that the cache filters the same network PDU with lower TTL values.

  In Step 16, the IUT sends a valid network PDU with the DST and the TTL fields set to the requested values.

  In Step 18, the IUT does not forward the received network PDU, confirming that the cache filters the network PDUs generated by the IUT and relayed by other nodes.

MESH/NODE/RLY/BI-01-C [Ignore Invalid IV Index]

- Test Purpose

  Verify that an IUT acting as Relay node does not relay network PDUs encrypted using an invalid IV Index.

- Reference

  [3] 2.2, 3.4.4, 3.4.4.1, 3.4.6.3

- Initial Condition
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT supports the Relay feature and the feature is enabled.
  - The Lower Tester is configured to generate network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the TTL field set to a random valid value greater than or equal to 2 on each new PDU, the SRC field set to a unique unicast address (not matching the Element addresses of the IUT) on each new PDU, the SEQ field set randomly on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

- Test Procedure
  1. The Lower Tester sends a new network PDU encrypted with the current IV index plus one.
  2. The Lower Tester expects the IUT to not forward the received network PDU.
  3. The Lower Tester sends a new network PDU encrypted with the current IV index minus two.
  4. The Lower Tester expects the IUT to not forward the received network PDU.

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, the IUT does not forward the received network PDUs.

MESH/NODE/RLY/BI-02-C [Ignore Invalid SRC]

- Test Purpose

  Verify that an IUT acting as Relay node does not relay network PDUs with an invalid SRC field value.

- Reference

  [3] 3.4.4, 3.4.6.3

- Initial Condition
  - The IUT and the Lower Tester share previously established network security credentials and a virtual address.
  - The IUT supports the Relay feature and the feature is enabled.
  - The Lower Tester is configured to generate network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the SEQ field set randomly on each new PDU, the DST set to the all nodes address and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU. The network PDUs are encrypted using the shared network key.

- Test Procedure
  1. The Lower Tester sends a network PDU, with the SRC field set to the unicast address of the IUT and the TTL field set to a random valid value greater than or equal to 2.
  2. The Lower Tester expects the IUT to not forward the received network PDU.
  3. The Lower Tester repeats Steps 1–2 three more times, using a new network PDU on each iteration through Step 1, and setting the SRC field to the unassigned address, a dynamically assigned group address, and to the shared virtual address, respectively.

- Expected Outcome

  Pass verdict

  In each iteration through Step 2, the IUT does not forward the received network PDUs.

4.10 Transport Layers

Test functionality of the Upper Transport Layer and the Lower Transport Layer of the Mesh specification.

MESH/NODE/TNPT/BV-01-C [Ignore Message with Unknown Application Key]

- Test Purpose

  Verify that an IUT node ignores messages encrypted with an unknown application key.

- Reference

  [3] 3.5.4.1

- Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with one of the IUT's application keys.
- Test Procedure
  1. The Lower Tester sends an acknowledged message to the IUT encrypted with an application key different than all the IUT's application keys.
  2. The Lower Tester expects to receive no response from the IUT.
- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any response.

MESH/NODE/TNPT/BV-02-C [Ignore Message with Unknown Device Key]

- Test Purpose

  Verify that an IUT node ignores messages encrypted with an unknown device key.

- Reference

  [3] 3.5.4.2

- Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with the IUT's device key.
- Test Procedure
  1. The Lower Tester sends an acknowledged message to the IUT encrypted with a device key different than the IUT's device key.
  2. The Lower Tester expects to receive no response from the IUT.
- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send a response.

MESH/NODE/TNPT/BV-03-C [Send an Unsegmented Access Message]

- Test Purpose

  Verify that an IUT node can send Unsegmented Access Messages.

- Reference

  [3] 3.5.2.3, 3.5.2.3.1, 3.5.3

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to send an Unsegmented Access Message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key and a shared application key.
  2. The Lower Tester expects to receive a valid Network PDU, with the field values specified in Step 1 and properly encrypted with the specified keys.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Network PDU containing the expected field values. The Access PDU has been encrypted with the specified application key and the Lower Transport PDU has been encrypted with the specified network key. The CTL field is set to 0, the SEG field is set to 0, the AKF field is set to 1, the AID field is set to the correct identifier of the shared application key, and the TransMIC has a size of 4 octets.

MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast]

- Test Purpose

  Verify that an IUT node can send Segmented Access Messages to a unicast address.

- Reference

  [3] 3.5.2.3.1, 3.5.2.4, 3.5.2.2, 3.5.2.3

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key, a shared application key and with a specified TransMIC size. The destination address is a unicast address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access

Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.

2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1.
3. The Lower Tester sends a Segment Acknowledgment Message to the IUT with the SeqZero field set to the same value from the Segmented Access Message received in Step 2, and the BlockAck field indicating that not all segments have been received from the IUT.
4. The Lower Tester expects the IUT to resend the segments that have not been acknowledged in Step 3.
5. The Lower Tester sends another Segment Acknowledgment Message to the IUT with the BlockAck field indicating that all segments have been received.
6. The Lower Tester expects the IUT to not send any more segments from this transaction.

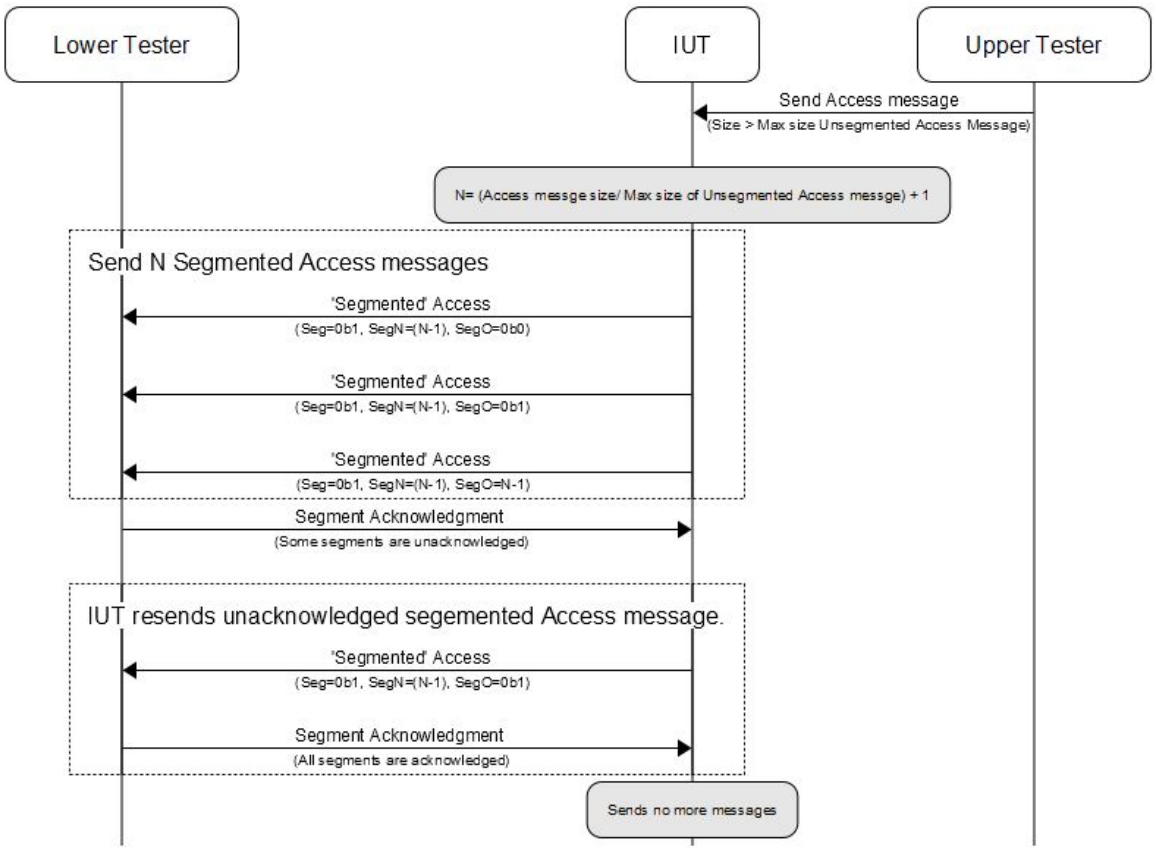


Figure 4.29: MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends all Segmented Access Messages. For all these messages, the CTL field is set to 0, the SEG field is set to 1, the AKF field is set to 1, the AID field is set to the correct identifier of the shared

application key, the TransMIC has the requested size, the SZMIC field is set to 1 only if the size of the TransMIC is 8 octets, the SeqZero field is set to the least significant 13 bits of the value of the SEQ field from the segment with the SegO field set to 0, the SegN field is set to N-1 and the SegO fields have all the values from 0 to N-1.

In Step 4, the IUT sends only the segments that have not been acknowledged in Step 3.

In Step 6, the IUT does not send any other segments.

Each message is formatted correctly, and the Lower Tester can decrypt the reassembled message and confirm it is the same message requested by the Upper Tester in Step 1.

MESH/NODE/TNPT/BV-05-C [Send a Segmented Access Message – Group/Virtual Address]

- Test Purpose

  Verify that an IUT node can send Segmented Access Messages to a group address, and a virtual address.

- Reference

  [3] 3.5.2.3.1, 3.5.2.4, 3.5.2.2, 3.5.2.3, 3.7.1, 3.5.3.3

- Initial Condition
  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - A known virtual address has been programed into the IUT.

- Test Procedure
  1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and Access PDU fields, and to encrypt the message using a shared network key and a shared application key. The destination address is a group address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.
  2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1.
  3. Repeat Steps 1–2 using a virtual address as destination address.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends all Segmented Access Messages. For all these messages, the CTL field is set to 0, the SEG field is set to 1, the AKF field is set to 1, the AID field is set to the correct identifier of the shared application key, the TransMIC has the requested size, the SZMIC field is set to 1 only if the size of the TransMIC is 8 octets, the SeqZero field is set to the least significant 13 bits of the value of the SEQ field

from the segment with the SegO field set to 0, the SegN field is set to N-1 and the SegO fields have all the values from 0 to N-1.

Each message is formatted correctly, and the Lower Tester can decrypt the reassembled message and confirm it is the same message requested by the Upper Tester in Step 1.

MESH/NODE/TNPT/BV-06-C [Receive an Unsegmented Access Message]

- Test Purpose

  Verify that an IUT node can receive and process an Unsegmented Access Message.

- Reference

  [3] 3.5.2, 3.5.2.1

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Lower Tester sends a small access message as an Unsegmented Access Message to the IUT, encrypted with a shared network key and a shared application key.
  2. The Upper Tester expects an indication from the IUT containing the same access message data sent by the Lower Tester in Step 1.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT indicates to the Upper Tester that it has received an access message from the Lower Tester. The message data is the same as the data sent by the Lower Tester in Step 1.

MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast]

- Test Purpose

  Verify that an IUT node can receive and reassemble Segmented Access and Control Messages directed to a unicast address.

- Reference

  [3] 3.1.2, 3.5.2.3.1, 3.5.2.4, 3.5.2.3, 3.5.2.2

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the

IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values.

2. The Lower Tester expects a Segment Acknowledgment Message from the IUT with the same SeqAuth value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received.
3. The Lower Tester repeats Steps 1–2 until the Segment Acknowledgment Message received in Step 2 indicates that all segments have been received. When repeating Step 1, the Lower Tester sends only the segments that have not been acknowledged by the IUT in Step 2.
4. The Upper Tester expects an indication from the IUT containing the data of the large message segmented by the Lower Tester in the first iteration through Step 1.
5. The Lower Tester resends the segments from the last iteration through Step 1.
6. The Lower Tester expects the IUT to immediately send the Segment Acknowledgment Message indicating that all segments have been received.
7. Repeat Steps 1–6 but with a large control message, sending it in a series of Segmented Control Messages. If the IUT does not support any Transport Control Message opcode that allows segmentation, then use an RFU value (the IUT will still report the reception of the complete message in Step 4 but will not respond to the control opcode).

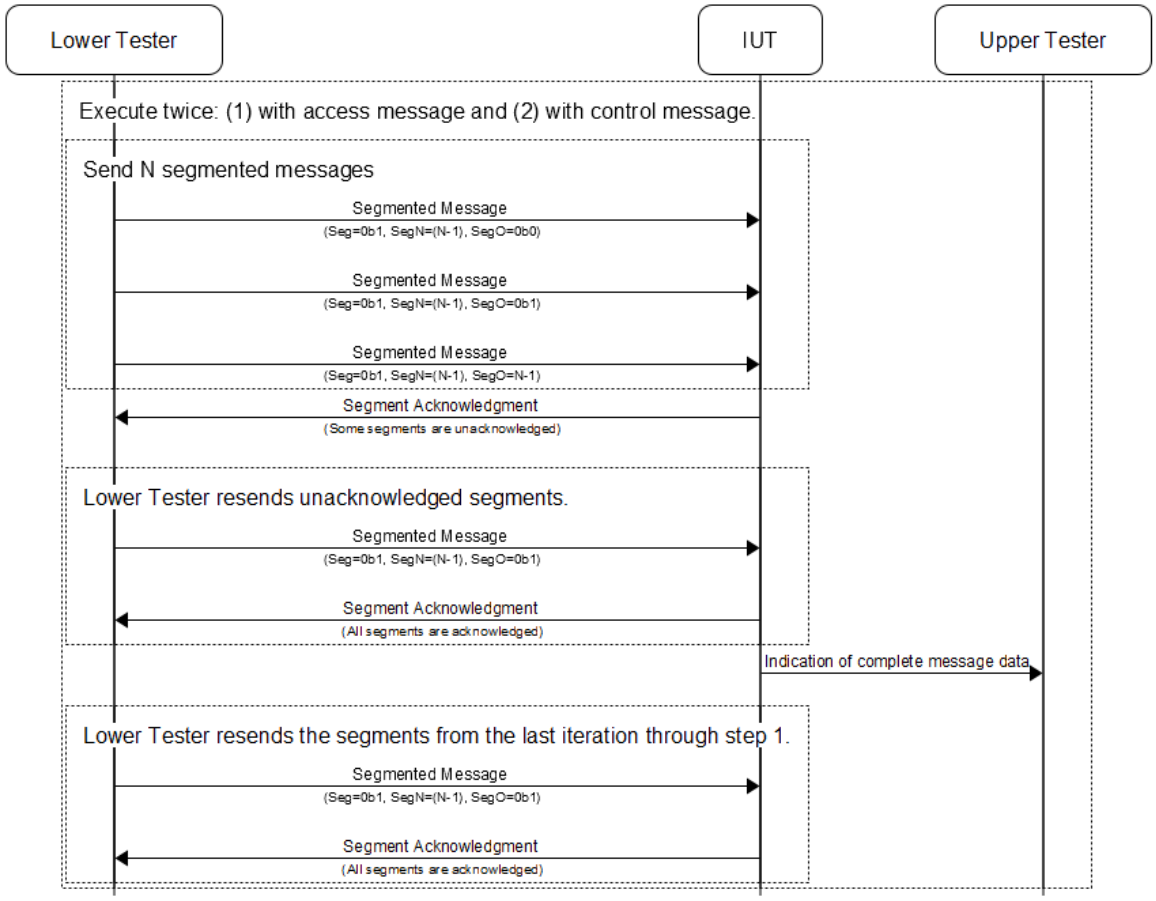


Figure 4.30: MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] MSC

- Expected Outcome

Pass verdict

In both iterations (for access and control messages):

In Step 2, the IUT sends a Segment Acknowledgment Message. The CTL field is set to 1, the SEG field is set to 0, the Opcode field is set to 0x00, the OBO field is set to 0, the SeqZero field is set to the SeqZero field from the Segmented Access or Control Messages sent by the Lower Tester in Step 1, the RFU bits are set to 0 and the BlockAck field has a valid value. In the final iteration through Steps 1–2, the IUT sends the message with the BlockAck field indicating all segments have been received. If more than one iteration through Steps 1–2 is executed, then each BlockAck field value from the message sent by the IUT acknowledges all the segments form the previous pass.

In Step 4, after the IUT has acknowledged all segments, the IUT reassembles the complete message (and decrypts the access message, if applicable) and indicates to the Upper Tester that it has received a message from the Lower Tester. The message data is the same as the data of the large message segmented by the Lower Tester in the first pass of Step 1.

In Step 6, the IUT sends the Segment Acknowledgment Message immediately after Step 5, i.e., faster than the acknowledgment timer period.

MESH/NODE/TNPT/BV-08-C [Receive a Segmented Message – Group/Virtual Address]

- Test Purpose

  Verify that an IUT node can receive and reassemble Segmented Access and Control Messages directed to a group address or a virtual address.

- Reference

  [3] 3.5.2.3.1, 3.5.2.4, 3.5.2.3, 3.5.2.2

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.
  - The IUT is subscribed to a shared group address and to a shared virtual address.

- Test Procedure

  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the shared group address, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends the complete set of Segmented Access Messages three times.
  2. The Upper Tester expects an indication from the IUT containing the data of the large message segmented by the Lower Tester in Step 1.
  3. The Lower Tester repeats Steps 1–2 replacing the shared group address from the DST field with the shared virtual address.
  4. Repeat Steps 1–3 but with a large control message, sending it in a series of Segmented Control Messages. If the IUT does not support any Transport Control Message opcode that allows segmentation, then use an RFU value (the IUT will still report the reception of the complete message

in Step 2 but will not respond to the control opcode).

- Expected Outcome

  Pass verdict

  In both iterations (for access and control messages):

  In Step 2, the IUT reassembles (and decrypts, if applicable) the complete message and indicates to the Upper Tester that it has received a message from the Lower Tester. The message data is the same as the data of the large message segmented by the Lower Tester in the first pass of Step 1.

MESH/NODE/TNPT/BV-09-C [Send Segmented Message to Unicast– Stop When Transaction Rejected]

- Test Purpose

  Verify that an IUT node sending Segmented Access Messages to a unicast address stops resending segments when the transaction is rejected by the receiver.

- Reference

  [3] 3.4.4.4, 3.5.3.3

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Upper Tester starts the procedure described in MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast], executing only Steps 1–2.
  2. The Lower Tester sends a Segment Acknowledgment Message with the BlockAck field set to 0.
  3. The Lower Tester expects the IUT to stop resending the segments.

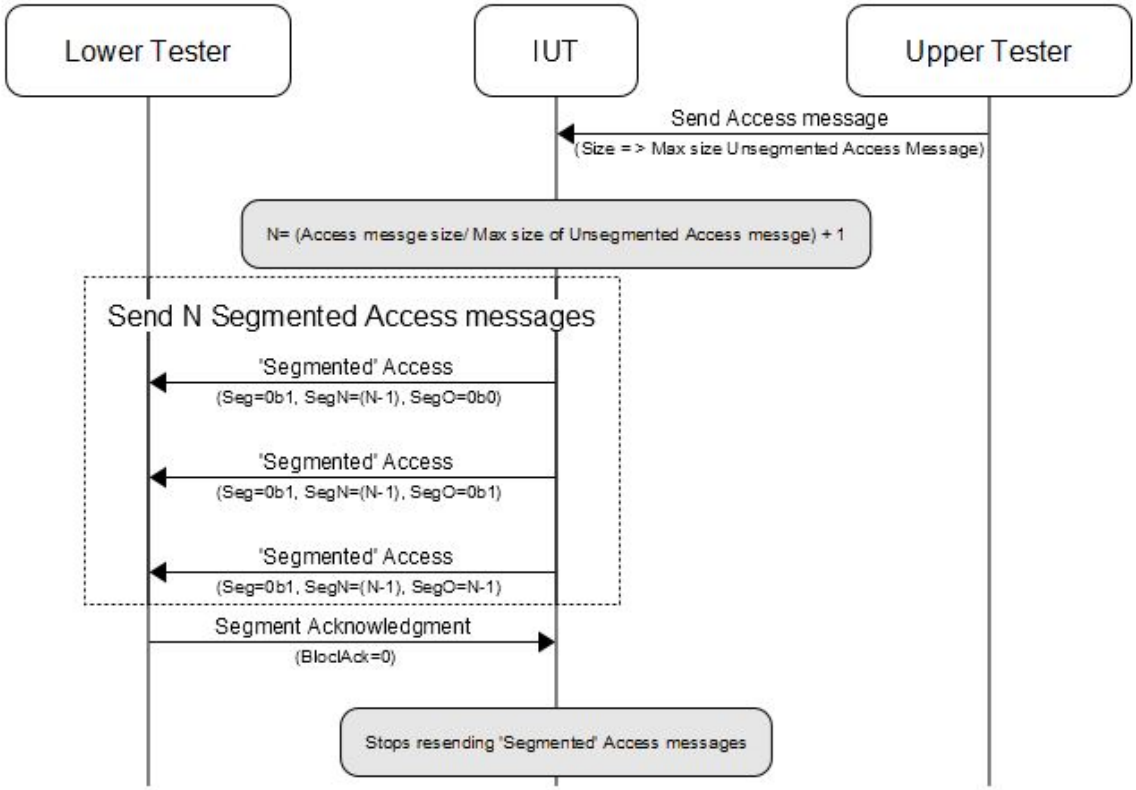


Figure 4.31: MESH/NODE/TNPT/BV-09-C [Send Segmented Message to Unicast– Stop When Transaction Rejected] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT does not resend any segment.

MESH/NODE/TNPT/BV-10-C [Send Segmented Message - Resend Unacknowledged Segments After Segment Transmission Timer Expires]

- Test Purpose

  Verify that an IUT node sending Segmented Access Messages to a unicast address resends unacknowledged segments after the segment transmission timer expires with no Segment Acknowledgment Message received.

- Reference

  [3] 3.5.3.3, 3.4.4.3

- Initial Condition

  - The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- The timer period is greater than or equal to 200 + 50 * TTL.

- Test Procedure

1. The Upper Tester orders the IUT to send an access message with specified valid values for the TTL, DST and access payload fields, and to encrypt the message using a shared network key, a shared application key and with a specified TransMIC size. The destination address is a unicast address. The size of the access message is greater than the maximum size that can fit in an Unsegmented Access Message and a number of Segmented Access Messages, denoted as N, are needed to send the access message.
2. The Lower Tester expects the IUT to send N Segmented Access Messages with the SEG field set to 1, the SegN field set to N-1 and the same value for the, AKF, AID, SZMIC and SeqZero fields. The SegO fields contain all values from 0 to N-1. All messages carry the maximum payload, except for the segment with the SegO field set to N-1.
3. The Lower Tester waits for the segment transmission timer to expire without sending a message to the IUT.
4. The Lower Tester expects the IUT to resend all segments from Step 2.
5. Repeat Steps 3–4.

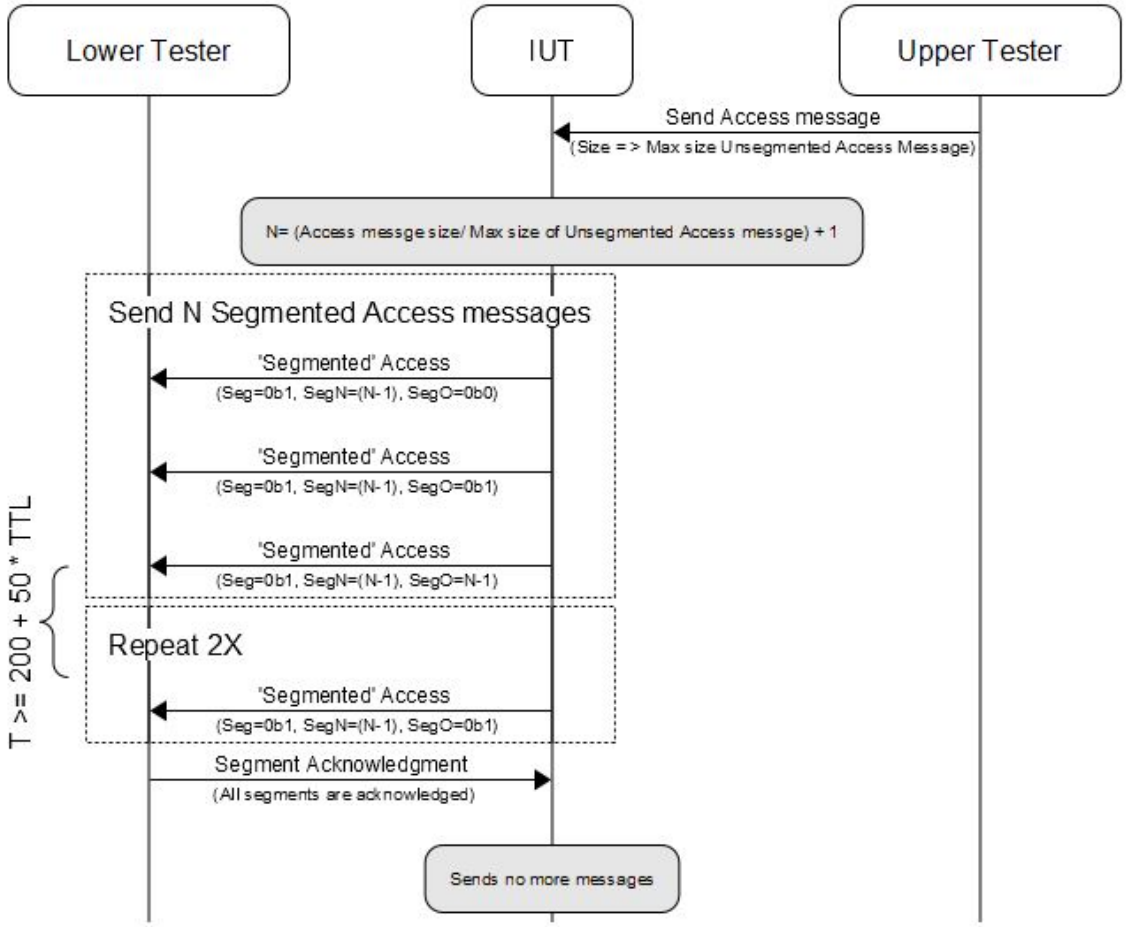


Figure 4.32: MESH/NODE/TNPT/BV-10-C [Send Segmented Message - Resend Unacknowledged Segments After Segment Transmission Timer Expires] MSC

- Expected Outcome

  Pass verdict

  The IUT successfully executes the segmentation procedure by resending all segments each time the segment transmission timer expires.

MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgment Timer Expiration]

- Test Purpose

  Verify that an IUT node receiving a segmented message sends the Segment Acknowledgment Message when the acknowledgment timer expires.

- Reference

  [3] 3.4.4.4, 3.5.3.3

- Initial Condition

- The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends all but the last segment.
  2. When the acknowledgment timer expires, the IUT sends a Segment Acknowledgment Message with the same SeqAuth field value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received and that the last segment has not been received.

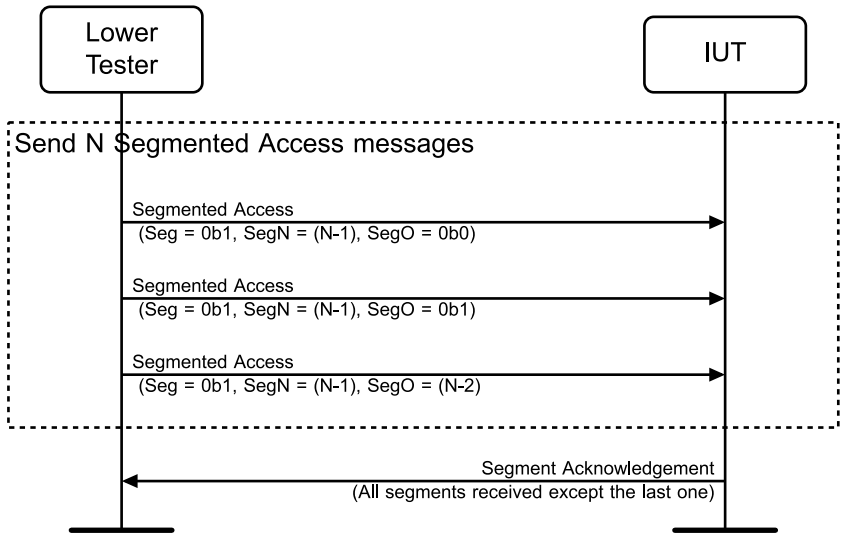


Figure 4.33: MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgment Timer Expiration] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Segment Acknowledgment Message with the expected parameters.

MESH/NODE/TNPT/BV-12-C [Receive Segmented Message - Receive New Segment After Incomplete Timer Expires]

- Test Purpose

  Verify that an IUT node receiving a segmented message cancels the transaction when the incomplete timer expires.

- Reference

  [3] 3.8.7

- Initial Condition

- The IUT and the Lower Tester are Nodes that share previously established network and application security credentials.

• Test Procedure

1. The Lower Tester sends a large access message in a series of Segmented Access Messages to the IUT, encrypted with a shared network key and a shared application key. All messages contain valid field values. The Lower Tester sends all but the last segment.
2. The Lower Tester expects a Segment Acknowledgment Message from the IUT with the same SeqAuth field value as the Segmented Access Messages, and the BlockAck field value indicating at least one segment has been received and that the last segment has not been received.
3. The Lower Tester waits for the incomplete timer on the IUT to expire, ignoring all Segment Acknowledgment Messages from the IUT.
4. The Lower Tester resends the unacknowledged segments to the IUT.
5. The Lower Tester expects no response from the IUT.

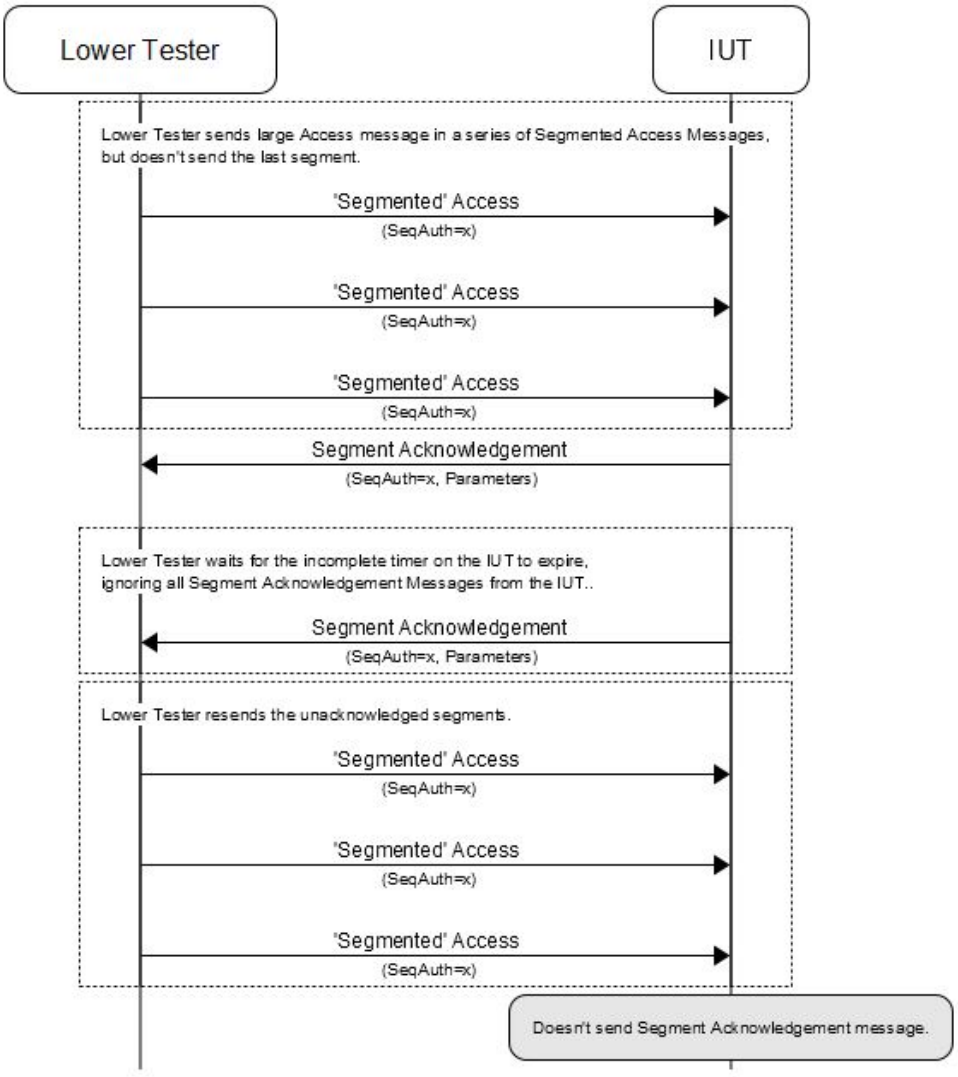


Figure 4.34: MESH/NODE/TNPT/BV-12-C [Receive Segmented Message - Receive New Segment After Incomplete Timer Expires] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Segment Acknowledgment Message with the expected parameters.

  In Step 5, the IUT does not send a Segment Acknowledgment Message.

MESH/NODE/TNPT/BV-13-C [Ignore Replayed Message]

- Test Purpose

  Verify that an IUT node performs replay protection on both the previous and the current IV Index.

- Reference

  [3] 3.8.7

- Initial Condition

  - The IUT, Lower Tester 1, and Lower Tester 2 are mesh nodes that share previously established network and application security credentials.
  - An application on the IUT expects to receive acknowledged messages encrypted with one of the IUT's application keys.
  - The replay protection list on the IUT is empty.
  - Lower Tester 1 has obtained the minimum replay protection list size on the IUT by reading the composition data. The list size is denoted as RPLS for this test.

- Test Procedure

  1. Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and a sequence number greater than 0.
  2. The IUT sends a response.
  3. Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in Step 1.
  4. The IUT sends a response.
  5. Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in Step 1.
  6. The IUT does not send a response.
  7. Perform a power reset on the IUT and wait until the IUT has complete the reset cycle.
  8. Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and the same sequence number as in Step 3.
  9. The IUT does not send a response.
  10. Lower Tester 1 sends an acknowledged message to the IUT using the previous IV Index and a smaller sequence number than in Step 1.
  11. The IUT does not send a response.
  12. Lower Tester 2 sends an acknowledged message to the IUT using the previous IV Index and a smaller sequence number than in Step 3.

13. The IUT does not send a response.
14. The Lower Testers repeat Steps 1–13 using the current IV Index for all messages. For the rest of the test procedure, only the current IV Index will be used.
15. Fill in the replay protection list on the IUT by executing the following procedure:
    a. Lower Tester 1 sends acknowledged messages with a new SRC address for each message, until either the IUT stops responding, RPLS messages have been sent, or the SRC address space has been exhausted.
    b. Lower Tester 1 sends an acknowledged message using its original source address and the last sequence number that was acknowledged by the IUT. If the IUT responds, the test ends with a Fail verdict.
    c. If Step 15a has completed because RPLS messages had been sent and the SRC address space had not been exhausted, then go back to Step 15a (Lower Tester 1 continues to use new SRC addresses from the available space). Otherwise (i.e., the IUT had stopped responding to new source addresses or the SRC address space has been exhausted), proceed to Step 16.
16. Lower Tester 1 sends an acknowledged message to the IUT using its original source address and the last sequence number that was acknowledged by the IUT.
17. The IUT does not send a response.
18. Perform a power reset on the IUT and wait until the IUT has complete the reset cycle.
19. Repeat Steps 16 and 17.

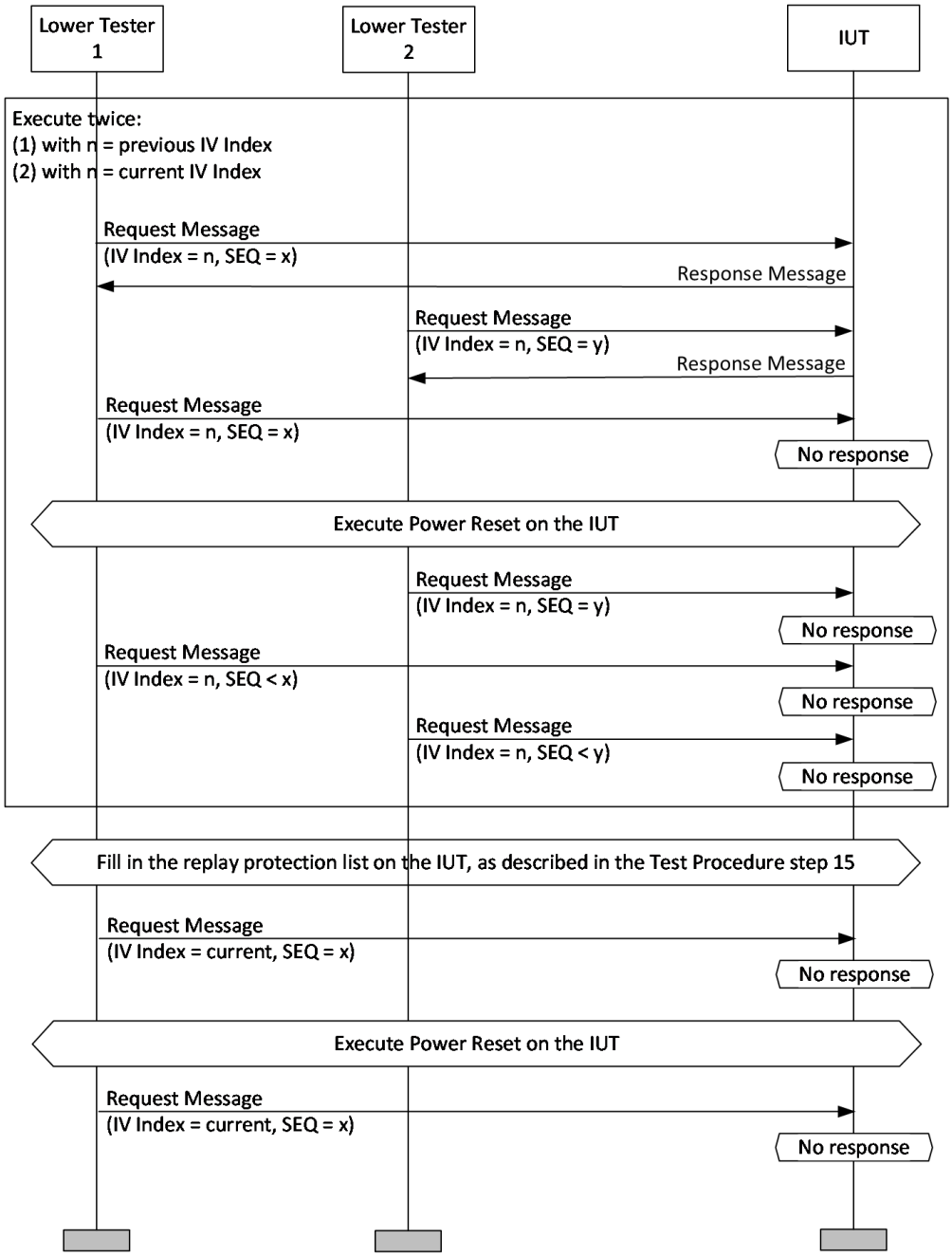


Figure 4.35: MESH/NODE/TNPT/BV-13-C [Ignore Replayed Message] MSC

- Notes

  To verify that the IUT responds to an acknowledged message, the Lower Tester needs to take into account that the response may be lost, and therefore it must repeat the message two more times (each time with an incremented SEQ number), at a reasonable interval (e.g., 3 seconds), before concluding that the IUT fails to respond.

  To verify that the IUT does not respond to an acknowledged message, the Lower Tester must send the message (with identical fields) three times at a reasonable interval (e.g., 3 seconds).

- Expected Outcome

  Pass verdict

  When the IUT receives a message with a sequence number that is not greater than the last saved sequence number for that SRC address, the IUT ignores the message.

  When the IUT does not have enough resources to perform replay protection for that SRC address, the IUT ignores the message.

MESH/NODE/TNPT/BV-22-C [Receive Segmented Message – Discard on New Transaction]

- Test Purpose

  Verify that the IUT discards an ongoing received segmented message transaction when a new segmented message is received.

- Reference

  [3] 3.5.3.4

- Initial Condition

  - The IUT and the Lower Tester are nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Lower Tester starts sending a new multi-segment access message to the IUT but does not send all segments, and the IUT acknowledges at least one of the received segments.
  2. The Lower Tester sends a new 3-segment access message to the IUT, and the IUT acknowledges the received segments for this new transaction.
  3. The Lower Tester sends segments of the large message from Step 1 that had previously not been sent, and the IUT does not acknowledge these new segments.
  4. Repeat Steps 1-3, but in Step 2 the Lower Tester sends a short message as a single-segment transaction.

- Expected Outcome

  Pass verdict

  In both iterations of Step 3 the IUT does not acknowledge the segments, confirming that it has discarded the previous transaction when receiving a new segmented message.

MESH/NODE/TNPT/BI-01-C [Ignore Invalid RFU Bits in Segment Acknowledgment Message]

- Test Purpose

  Verifies that an IUT node ignores invalid RFU bits that are set to 1 in the Segment Acknowledgment Message while sending Segmented Access Messages.

- Reference

  [3] 3.1.2, 3.5.2.3

- Initial Condition

  - The IUT and the Lower Tester are mesh nodes that share previously established network and application security credentials.

- Test Procedure

  The Lower Tester executes the procedure described in MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast], but sets all the RFU bits to 1 in all Segment Acknowledgment Messages.

- Expected Outcome

  Pass verdict

  The IUT behaves as described in the referenced test case, ignoring the RFU bits that are set to 1.

MESH/NODE/TNPT/BI-02-C [Ignore Message When Segments Received with Old SeqAuth]

- Test Purpose

  Verifies that an IUT node ignores Segmented Access Messages with older SeqAuth values.

- Reference

  [3] 3.5.3.4

- Initial Condition

  - The IUT and the Lower Tester are mesh nodes that share previously established network and application security credentials.

- Test Procedure

  1. The Lower Tester executes Steps 1 and 2 of the procedure described in MESH/NODE/TNPT/BV-11-C [Receive Segmented Message - Acknowledgment Timer Expiration].
  2. The Lower Tester resends the unacknowledged segments but with an older SeqAuth value.
  3. The Lower Tester expects the IUT not to respond to the segments with the old SeqAuth value.

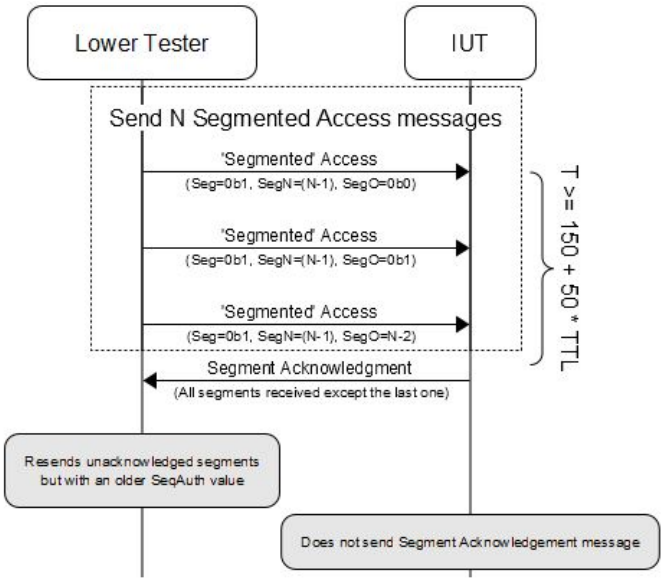


Figure 4.36: MESH/NODE/TNPT/BI-02-C [Ignore Message When Segments Received with Old SeqAuth] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT does not send any Segment Acknowledgment Messages to the Lower Tester.

4.10.1 Send Segmented Messages Using the SAR Transmitter State Values - Unicast

- Test Purpose

  Verify that a node supporting the SAR Configuration Server model can perform message segmentation to a unicast address using the configured values of the SAR Transmitter state.

- Reference

  [3] 3.5.3, 4.2.23, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The Network Transmit state on the IUT is set to 0x00.

- Test Procedure

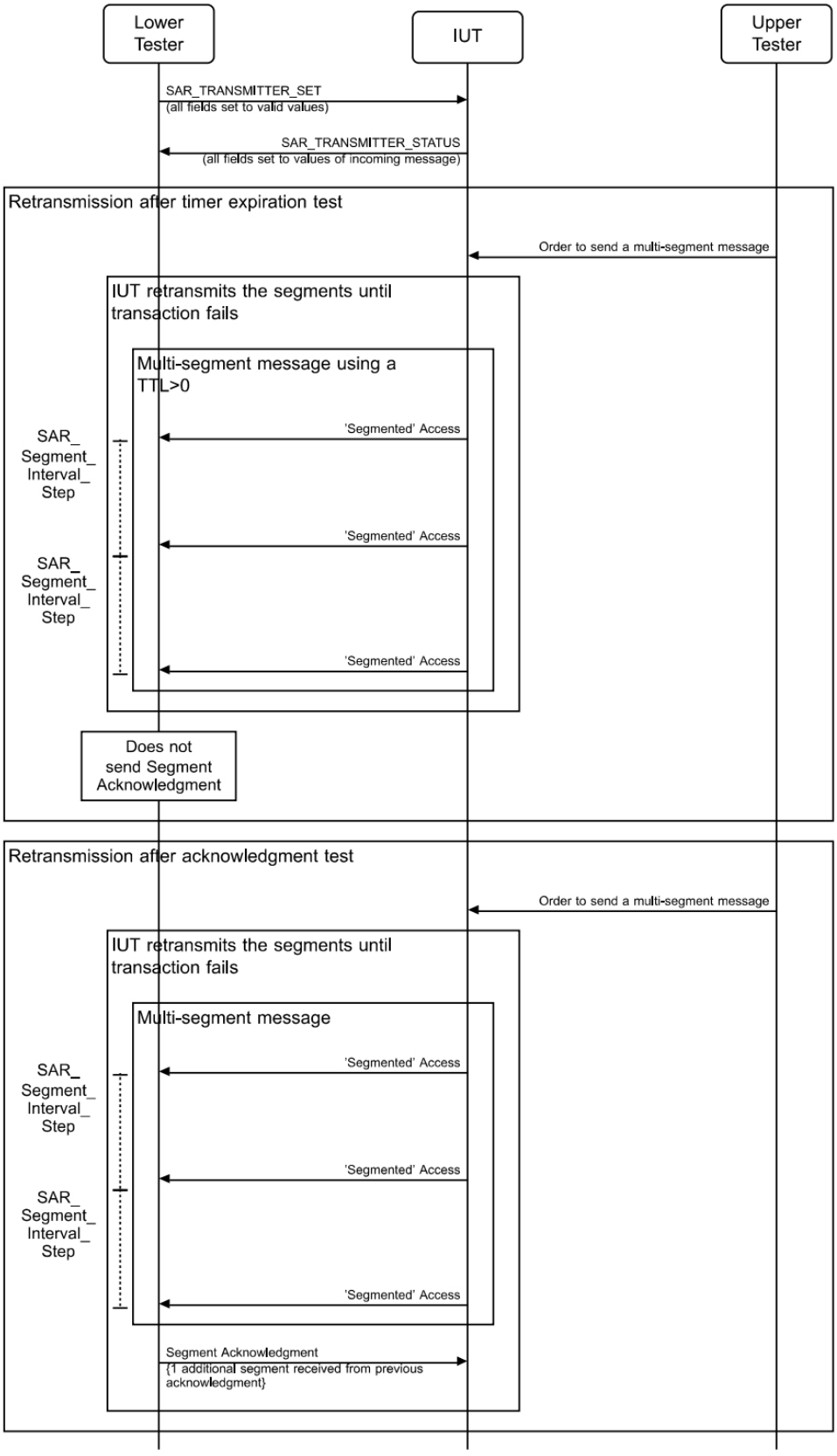


Figure 4.37: Test procedure for Send Segmented Messages using the SAR Transmitter state values - Unicast MSC - Page 1 of 2

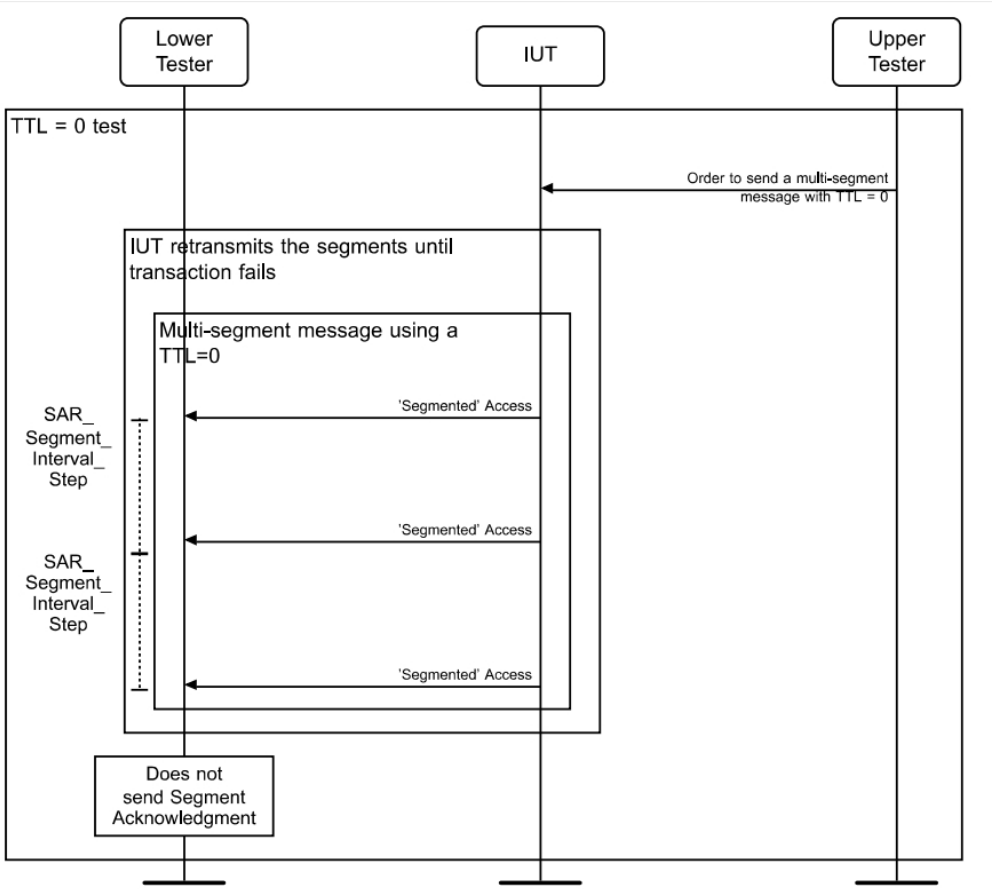


Figure 4.38: Test procedure for Send Segmented Messages using the SAR Transmitter state values - Unicast MSC - Page 2 of 2

1. The Lower Tester sends a SAR_TRANSMITTER_SET message to the IUT with all fields set to valid values, reasonably chosen to be able to verify the Pass verdicts.
2. The IUT responds with a SAR_TRANSMITTER_STATUS message with the values of the incoming message.
3. The Upper Tester orders the IUT to send a multi-segment message (at least three segments) to the unicast address of the Lower Tester, using the TTL value specified in Table 4.12. The IUT sends the segments with valid fields. The interval between two segments is equal to the value of the SAR_Segment_Interval_Step.
4. Execute the instructions listed in Table 4.12.

- Test Case Configuration

TCID	TTL (Step 3)	Step 4
MESH/NODE/TNPT/BV-14-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, Retransmission after Timer Expiration]	Greater than 1	The Lower Tester does not respond with a Segment Acknowledgment message. The IUT retransmits the segments a number of times indicated by the SAR_Unicast_Retransmissions_Without_Progress_Count state, before failing the transaction. The interval between each set of retransmissions is computed based on the values of the SAR_Unicast_Retransmissions_Interval_Step and SAR_Unicast_Retransmissions_Interval_Increment states, as well as on the value of the TTL.
MESH/NODE/TNPT/BV-15-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, Retransmission after Acknowledgment]	Greater than 1	The Lower Tester sends a Segment Acknowledgment after each set of retransmissions, each time acknowledging the receipt of only one additional segment compared to the previous acknowledgment. The IUT starts retransmitting the segments as soon as the Segment Acknowledgment message is received. The IUT fails the transaction after a number of retransmissions indicated by the SAR_Unicast_Retransmissions_Count state.
MESH/NODE/TNPT/BV-16-C [Send Segmented Messages Using the SAR Transmitter State Values – Unicast, TTL set to 0]	0	The Lower Tester does not respond with a Segment Acknowledgment message. The IUT retransmits the segments a number of times indicated by the SAR_Unicast_Retransmissions_ Without_Progress_Count state, before failing the transaction. The interval between each set of retransmissions is computed based on the values of the SAR_Unicast_Retransmissions_Interval_Step and SAR_Unicast_Retransmissions_Interval_Increment states, as well as on the value of the TTL.

Table 4.12: Send Segmented Messages Using the SAR Transmitter State Values - Unicast test cases

- Expected Outcome

  Pass verdict

  The IUT uses the values from the SAR Transmitter state when performing message segmentation: SAR_Segment_Interval_Step, SAR_Unicast_Retransmissions_Count, SAR_Unicast_Retransmissions_Without_Progress_Count, SAR_Unicast_Retransmissions_Interval_Increment.

  All measured time intervals are within 20% accuracy. Multiple measurements of the same interval should be

averaged before the comparison with the expected values.

- Notes

  This test case should be run in an isolated environment to avoid packet loss at the RF level. Multiple test case runs may be necessary to complete the procedure due to imperfect RF reception.

4.10.2 Receive Segmented Messages Using the SAR Receiver State Values - Unicast

- Test Purpose

  Verify that a node supporting the SAR Configuration Server model can perform message reassembly to a unicast address using the configured values of the SAR Receiver state.

- Reference

  [3] 3.5.3, 4.2.23, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The Network Transmit states on the Lower Tester and IUT are set to 0x00.
  - The SAR_Receiver_Segment_Interval_Step state on the IUT has been set to a known value, sufficiently large to allow measurement of time intervals with reasonable accuracy.

- Test Procedure

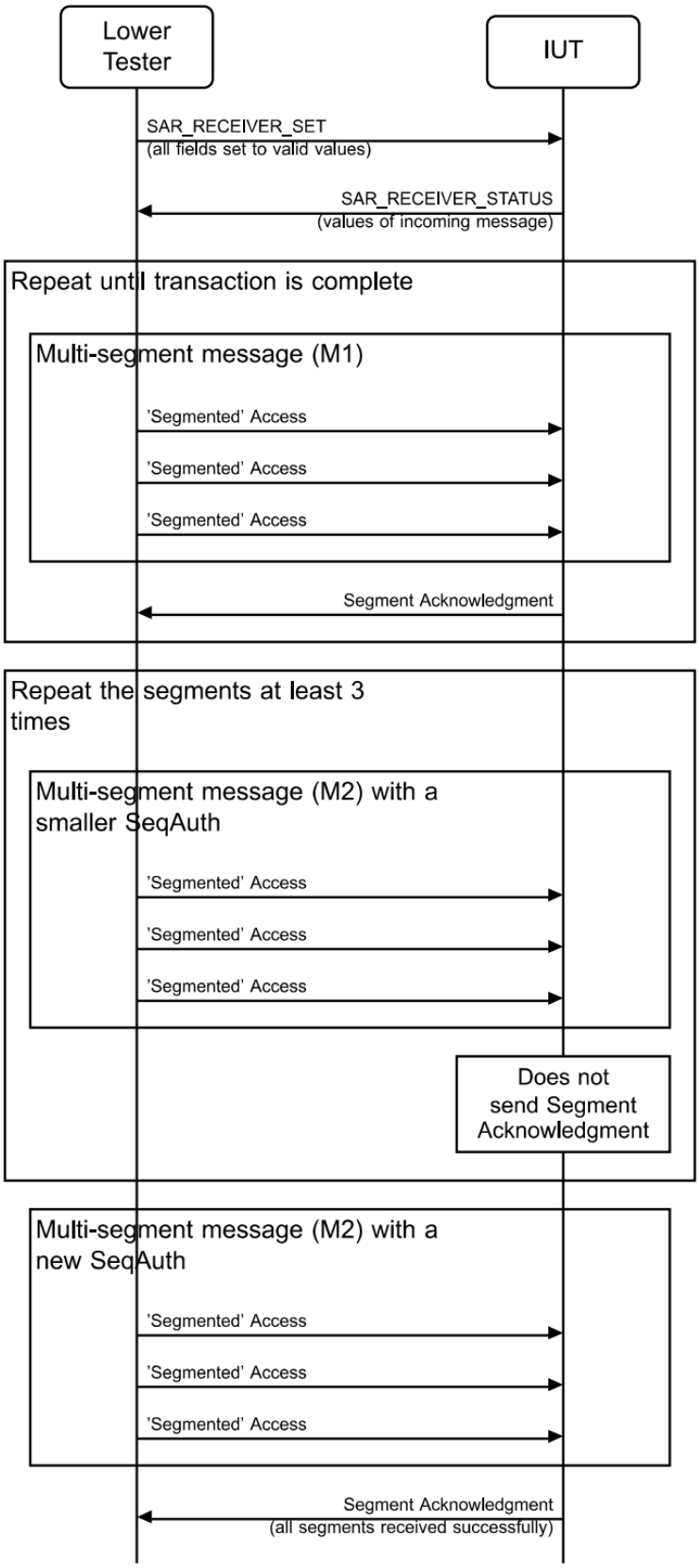


Figure 4.39: Test procedure for Receive Segmented Messages using the SAR Receiver state values - Unicast MSC - Page 1 of 2

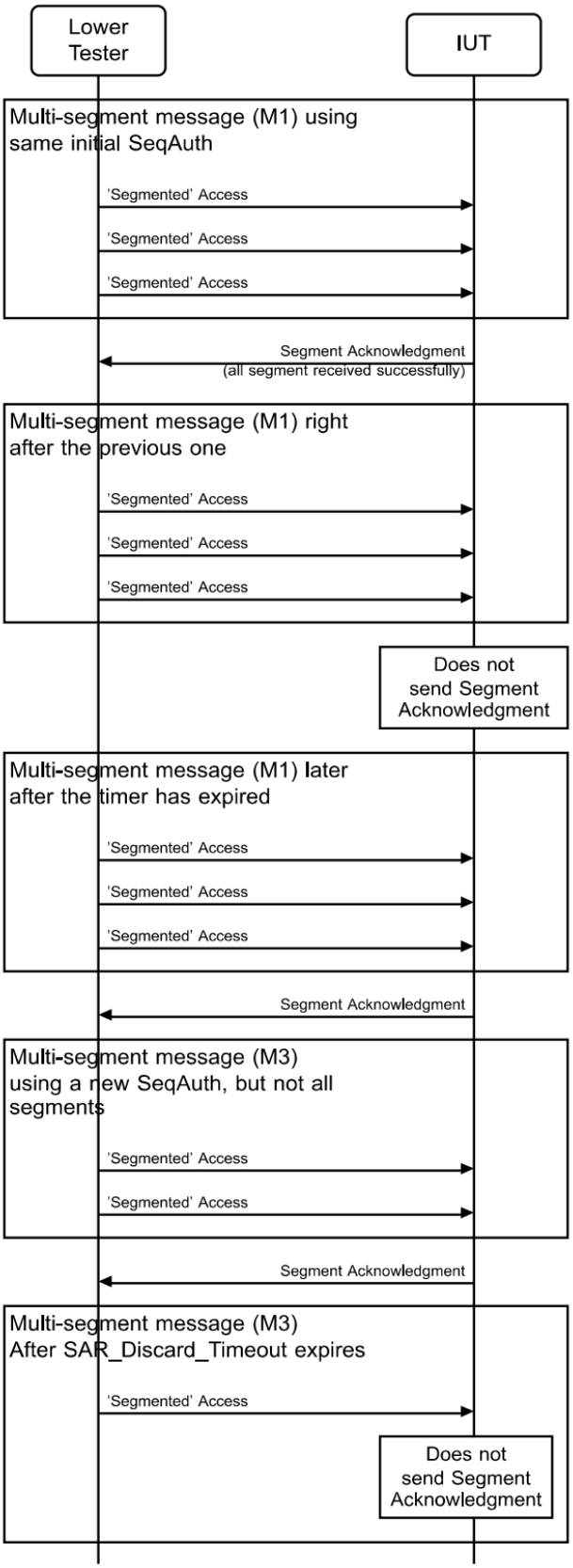


Figure 4.40: Test procedure for Receive Segmented Messages using the SAR Receiver state values - Unicast MSC - Page 2 of 2

1. The Lower Tester sends a SAR_RECEIVER_SET message with all fields set to valid values, reasonably chosen to be able to verify the Pass verdicts.
2. The IUT responds with a SAR_RECEIVER_STATUS message with parameters set as in Step 1.
3. Execute the steps listed in Table 4.13.

- Test Case Configuration

TCID	Additional Steps
MESH/NODE/TNPT/BV-17-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, Complete Receival]	3. The Lower Tester sends a multi-segment message to the unicast address of the IUT, repeating the segments as necessary until the transaction is complete (i.e., the IUT has acknowledged all segments). The number of segments is greater than the value of the SAR_Segments_Threshold state and greater than the value of the SAR Acknowledgment Delay Increment state, as set in Step 1. The IUT sends Segment Acknowledgment messages; each such message is sent after a time interval equal to SAR_Acknowledgment_Delay_Increment * SAR_Receiver_Segment_Interval_Step, measured since the last new segment was received by the IUT.

TCID	Additional Steps
MESH/NODE/TNPT/BV-18-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, SeqAuth Validation]	3. The Lower Tester sends a multi-segment message (M1) to the IUT, and the IUT successfully receives it. 4. The Lower Tester sends a new multi-segment message (M2) to the unicast address of the IUT, but with a SeqAuth value smaller than the one used in Step 3, and the IUT does not respond. Repeat the segments at least three times. The number of segments is smaller than the value of the SAR_Acknowledgment_Delay_Increment state. 5. The Lower Tester resends message M2 but with a new SeqAuth value (SeqAuth is greater than the previous SeqAuth plus 8192), and the IUT successfully receives the message. The IUT sends Segment Acknowledgment messages, initially after a time interval equal to (SegN + 0.5) * SAR_Receiver_Segment_Interval_Step, measured since the last new segment was received by the IUT, and subsequently after a time interval equal to SAR_Acknowledgment_Delay_Increment * SAR_Receiver_Segment_Interval_Step. 6. The Lower Tester resends message M2 (with the same SeqAuth as in Step 5), but within a total time interval smaller than (SAR_Acknowledgment_Retransmissions_Interval + SAR_Receiver_Segment_Interval_Step). The IUT does not send another Segment Acknowledgment message. 7. The Lower Tester resends message M2 (with the same SeqAuth as in Step 5), but after the specified time interval (SAR_Acknowledgment_Retransmissions_Interval + SAR_Receiver_Segment_Interval_Step) has passed. The IUT immediately sends another Segment Acknowledgment message.
MESH/NODE/TNPT/BV-19-C [Receive Segmented Messages Using the SAR Receiver State Values – Unicast, Discard Timeout]	3. The Lower Tester prepares to send a multi-segment message to the unicast address of the IUT, but it sends only some of the segments over the air, and the IUT acknowledges the receipt of those segments (or potentially fewer if radio packets are lost). The Lower Tester then waits for the SAR_Discard_Timeout interval to expire without sending any segments, and then it sends a new segment. The IUT does not send another Segment Acknowledgment message.

Table 4.13: Receive Segmented Messages Using the SAR Receiver State Values - Unicast test cases MSC

- Expected Outcome

Pass verdict

The IUT uses the values from the SAR Receiver state when performing message reassembly:

SAR_Segments_Threshold, SAR_Acknowledgment_Delay_Increment, SAR_Receiver_Segment_Interval_Step, SAR_Acknowledgment_Retransmissions_Interval, SAR_Discard_Timeout.

All measured time intervals are within 10% accuracy.

The Segment Acknowledgment message is always encrypted with the same network key as the first received segment of the segmented message and the DST field has the same value as the SRC field of the first received segment.

MESH/NODE/TNPT/BV-20-C [Send Segmented Messages Using the SAR Transmitter State Values – Group/Virtual Address]

- Test Purpose

  Verify that a node supporting the SAR Configuration Server model can perform message segmentation to a group address and a virtual address using the configured values of the SAR Transmitter state.

- Reference

  [3] 3.5.3, 4.2.23, 4.3.5, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - A known virtual address has been set as publication address for one of the IUT's supported models (e.g., Health Server).
  - The Network Transmit state on the IUT is set to 0x00.

- Test Procedure

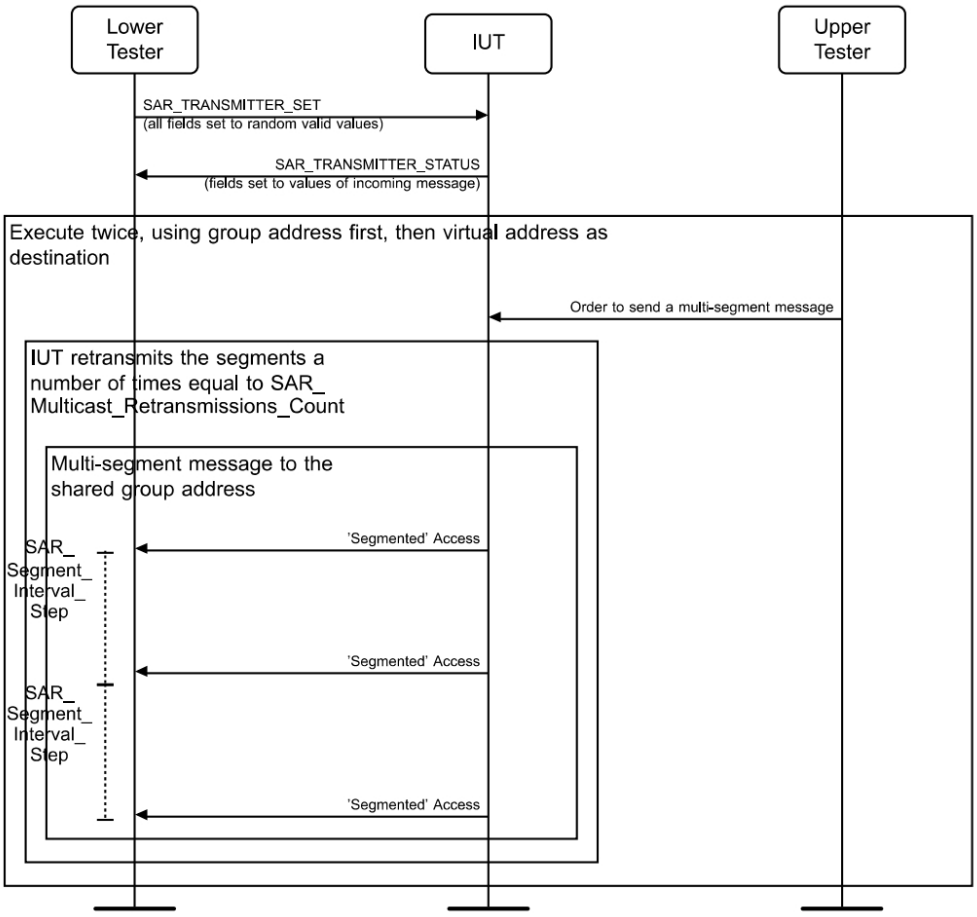


Figure 4.41: Test procedure for MESH/NODE/TNPT/BV-20-C MSC

1. The Lower Tester sends a SAR_TRANSMITTER_SET message to the IUT with all fields set to valid values, reasonably chosen to be able to verify the Pass verdicts.
2. The IUT responds with a SAR_TRANSMITTER_STATUS message with values from incoming message.
3. The Upper Tester orders the IUT to send a multi-segment message (at least three segments) to the shared group address. The IUT sends the segments with valid fields. The interval between two segments is equal to the value of the SAR_Segment_Interval_Step.
4. The IUT retransmits the segments a number of times indicated by the SAR_Multicast_Retransmissions_Count state. The interval between each set of retransmissions is computed based on the values of the SAR_Multicast_Retransmissions_Interval state.
5. Repeat Steps 3 and 4 using the known virtual address as a destination address.

- Expected Outcome

  Pass verdict

  The IUT uses the values from the SAR Transmitter state when performing message segmentation: SAR_Segment_Interval_Step, SAR_Multicast_Retransmissions_Count, SAR_Multicast_Retransmissions_Interval.

All measured time intervals are within 10% accuracy.

MESH/NODE/TNPT/BV-21-C [Receive Segmented Messages Using the SAR Receiver State Values – Group/Virtual Address]

- Test Purpose

  Verify that a node supporting the SAR Configuration Server model can perform message reassembly to a group address or a virtual address using the configured values of the SAR Receiver state.

- Reference

  [3] 3.5.3, 4.3.5, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is subscribed to a shared group address and to a shared virtual address on one of the supported models (e.g., Health Server).
  - The Network Transmit states on the Lower Tester and IUT are set to 0x00.

- Test Procedure

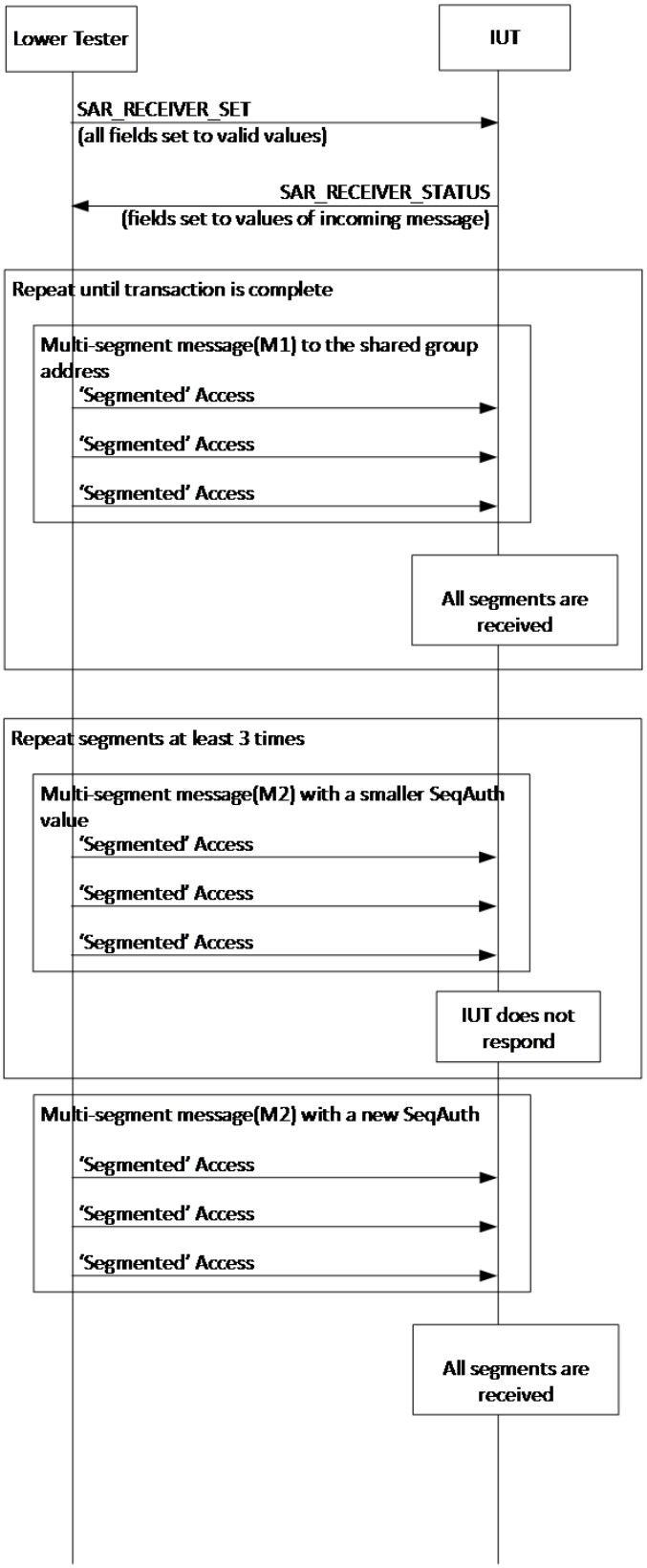


Figure 4.42: Test procedure for MESH/NODE/TNPT/BV-21-C MSC - Page 1 of 2

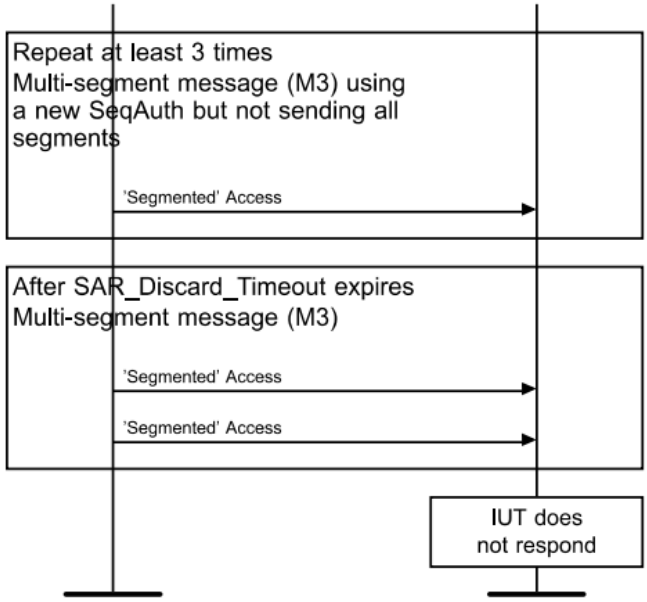


Figure 4.43: Test procedure for MESH/NODE/TNPT/BV-21-C MSC - Page 2 of 2

1. The Lower Tester sends a SAR_RECEIVER_SET message with all fields set to valid values.
2. The IUT responds with a SAR_RECEIVER_STATUS message with parameters set as in Step 1.
3. The Lower Tester sends a multi-segment acknowledged message (M1) to the shared group address, repeating the segments as necessary until the transaction is complete. The IUT responds with the correct status message, confirming the receipt of all segments.
4. The Lower Tester sends a new multi-segment message (M2) to the shared group address, but with a SeqAuth value smaller than the one used in Step 3, and the IUT does not respond. Repeat the segments at least three times.
5. The Lower Tester resends the message M2 but with a new SeqAuth value, and the IUT successfully receives the message.
6. The Lower Tester prepares to send a new multi-segment message (M3) with a new SeqAuth to the shared group address, but it sends only one of the segments over the air, at least three times. The Lower Tester then waits for the SAR_Discard_Timeout interval to expire without sending any segments, and then it sends all remaining segments. The IUT does not respond, confirming that it has cancelled the transaction after the timeout.
7. Repeat Steps 3–6 using the shared virtual address as the destination address.

- Expected Outcome

  Pass verdict

  The IUT uses the values from the SAR Receiver state when performing message reassembly.

MESH/NODE/TNPT/BI-03-C [Ignore Invalid Segment Acknowledgment Messages]

- Test Purpose

  Verify that a node ignores invalid Segment Acknowledgment messages when performing message

segmentation.

- Reference

  [3] 3.5.3, 4.2.23, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure

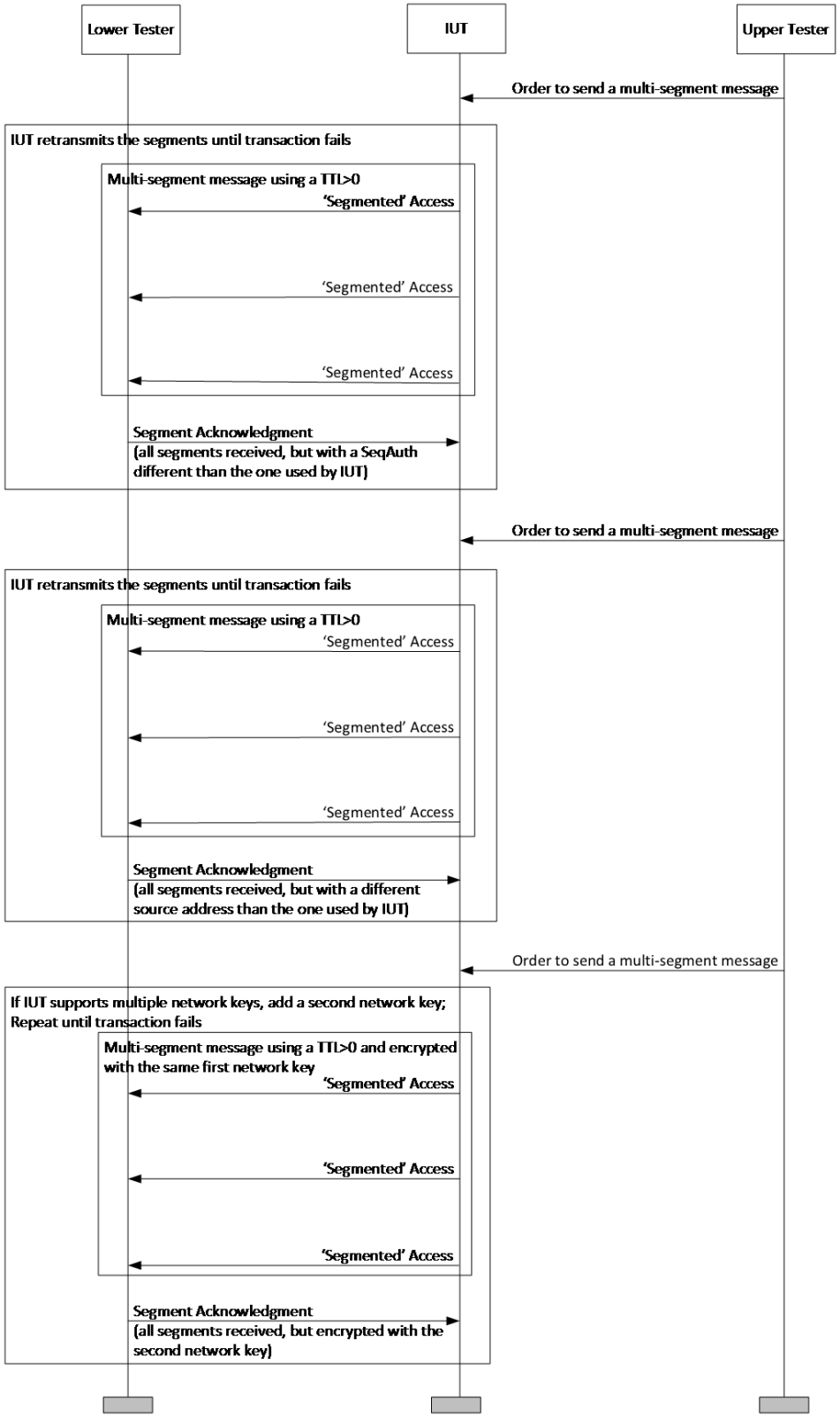


Figure 4.44: Test procedure for MESH/NODE/TNPT/BI-03-C MSC

1. The Upper Tester orders the IUT to send a multi-segment message (at least three segments) to the unicast address of the Lower Tester, using a TTL greater than 1. The IUT sends the segments with valid fields.
2. The Lower Tester sends a Segment Acknowledgment message acknowledging the receipt of all segments, but with a SeqAuth different than the one used by the IUT (SeqAuth is greater than the expected value plus 8192). The IUT continues to resend all segments until the transaction fails.
3. Repeat Step 1 with a new message, then the Lower Tester sends a Segment Acknowledgment message acknowledging the receipt of all segments, but with a different source address than the one used by the IUT as destination. The IUT continues to resend all segments until the transaction fails.
4. If the IUT supports multiple network keys, then add a second network key on the IUT and repeat Step 1 with a new message (encrypted with the same first network key), then the Lower Tester sends a Segment Acknowledgment message acknowledging the receipt of all segments, but encrypted with the second network key. The IUT continues to resend all segments until the transaction fails.

- Expected Outcome

  Pass verdict

  The IUT ignores the Segment Acknowledgment messages that do not meet the validation conditions.

4.11 IV Update

The test group objective is to verify the IV Update feature.

4.11.1 IV Update Initiated by Receiving Mesh Beacons

- Test Purpose

  Verify that a Node IUT can transition into the IV Update in Progress state upon receiving a Secure Network beacon or a Mesh Private beacon with the IV Update Flag set to 1 and IV Index set to current IV Index + 1.

- Reference

  [3] 3.8.2, 3.9.3.1, 3.10.4, 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update procedure is not already in progress.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.14.

- Test Procedure

  1. The Lower Tester generates a Mesh Beacon with Beacon Type as described by the Beacon Type column in Table 4.14; with the same network credentials and IV Index as the IUT (IV Index = n), the

Key Refresh flag and IV Update flag set to 0, and valid security depending on beacon type. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.

2. The Lower Tester stops advertising and waits for the IUT to generate a non-connectable Mesh Beacon with Beacon Type as described by the Beacon Type column in Table 4.14 with the same network credentials value and IV Index value (n) as the Lower Tester, the Key Refresh and IV Index flags set to 0, and valid security depending on beacon type. The IUT begins advertising this Mesh Beacon.
3. The Lower Tester observes the beacon advertisement(s) by the IUT.
4. The Lower Tester sends a mesh message with IV Index set to the previous IV Index (n - 1) to the IUT (e.g., Config Beacon Get) and verifies that the message is accepted and processed by the IUT.
5. The Lower Tester sends a mesh message with IV Index set to the current IV Index (n) to the IUT.
6. The Upper Tester observes the IUT to confirm it accepts and processes the mesh message.
7. The Lower Tester generates a new beacon with the IV Update Flag set to 1, IV Index is incremented by 1 (new IV Index m = n+1), and sends the updated beacon to the IUT.
8. The Upper Tester observes the IUT to confirm that it receives the updated beacon from the Lower Tester, enters IV Update in Progress state, and begins advertising beacons with the new IV Index (m).
9. The Lower Tester receives beacons from the IUT.
10. The Lower Tester sends a mesh message with IV Index set to the old IV Index (m-1) to the IUT and verifies that the message is accepted and processed by the IUT by checking the message returned. The message received from the IUT has the sequence number set to zero.
11. The Lower Tester sends a mesh message with IV Index set to the new IV Index (m) to the IUT and verifies that the message is accepted and processed by the IUT by checking the message returned.
12. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
13. The Lower Tester expects to receive a beacon from the IUT with the IV Index set to the new IV Index (m) and the IV Update flag set to 0.
14. Repeat Step 11.

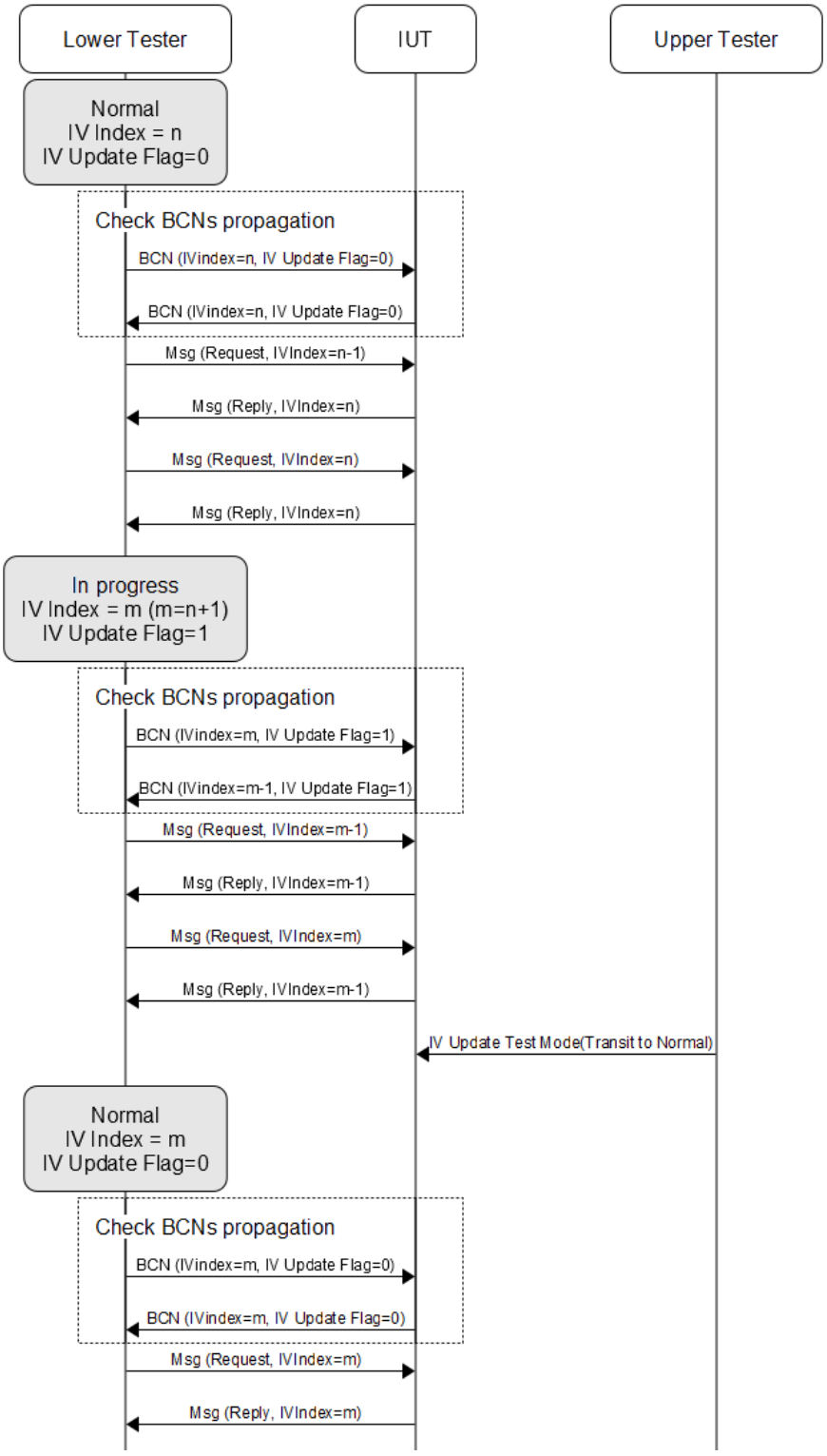


Figure 4.45: MESH/NODE/IVU/BV-01-C and MESH/NODE/IVU/BV-06-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BV-01-C [IV Update Initiated by Receiving Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BV-06-C [IV Update Initiated by Receiving Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.14: IV Update Initiated by Receiving Mesh Beacons test cases

- Expected Outcome

Pass verdict

When the IUT is in Normal Operation state, it transmits beacons using the current IV Index (n). Mesh messages are also transmitting using the current IV Index (n).

When the IUT is in Normal Operation state, it accepts and processes messages with the current IV Index (n) and with the previous IV Index (n-1).

The IUT observes the updated beacon from the Lower Tester with the new IV Index value (m, or n+1) and the IV Update Flag set to 1.

When the IUT transitions from the Normal Operation state to the IV Update in Progress state, the IV Index on the node is incremented by one.

When the IUT is in IV Update in Progress state, it transmits beacons using the new IV Index (m, or n+1).

When the IUT is in IV Update in Progress state, it accepts and processes messages with the old IV Index (m-1, or n) and with the new IV Index (m, or n+1).

When the IUT is in IV Update in Progress state, the IV Update Flag is set to 1.

When the IUT is in IV Update in Progress state, it transmits using the old IV Index (m-1, or n).

When the IUT returns to the Normal Operation state, the sequence number is reset to 0.

When the IUT returns to the Normal Operation state, the IV Update Flag is reset to 0.

When the IUT returns to the Normal Operation state, it transmits using the new IV Index (m, or n+1).

When the IUT transitions from IV Update in Progress state to Normal Operation state, it accepts and processes messages with the old IV Index (m-1, or n) and with the new IV Index (m, or n+1).

For beacons with the Mesh Private beacon type, when a new beacon is sent as a result of the IV Index being incremented or the IV Update Flag being flipped, the Random field and the advertising address both change.

4.11.2 IV Update Initiated by Sending Mesh Beacons

- Test Purpose

  Verify that an IUT Node can initiate the IV update Procedure.

  Note: This test covers the starting procedure up to returning to the Normal Operation state by injecting a Transit to Normal signal to IUT. The behaviors in the Normal Operation state before and after the IV Update are the same as being a receiving node.

- Reference

  [3] 3.9.3.1, 3.10.5

- Initial Condition

  - The IUT is a Node.
  - The IUT is provisioned to the primary Subnet.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.15.

- Test Procedure

  1. The Lower Tester generates a Mesh Beacon with Beacon Type as described by the Beacon Type column in Table 4.15 with the same network credentials and IV Index as the IUT (IV Index = n), the Key Refresh flag and IV Update flag set to 0, and valid security depending on beacon type. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.
  2. The Lower Tester stops advertising and waits for the IUT to generate and send a beacon using the same Beacon Type, network credentials, and IV Index as the Lower Tester (n).
  3. The Lower Tester receives the beacon from the IUT.
  4. The Upper Tester sends a Transit to IV Update in Progress signal to the IUT, triggering the IUT to transition to the IV Update in Progress state.
  5. The Lower Tester expects the IUT to advertise a beacon with the IV Update Flag set to 1 (IV Update in progress) and with the IV Index set to the new IV Index value (m, or n+1). The Lower Tester verifies the beacon based on the shared network security credentials.
  6. The Lower Tester generates a beacon using the same network credentials and IV Index (m, or n+1) as the IUT, and IV Update Flag set to 1. The Lower Tester begins advertising this beacon periodically while also scanning for neighboring beacons.
  7. The Lower Tester sends a mesh message with the old IV Index (m-1, or n) and verifies that it is accepted and processed by IUT by verifying the message returned.
  8. The Lower Tester sends a mesh message with the new IV Index (m, or n+1) and verifies that it is accepted and processed by IUT by verifying the message returned.
  9. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
  10. The Lower Tester stops advertising and receives a beacon from the IUT showing it is in Normal

Operation state with the IV Update flag set to 0 and the IV Index set to the new IV Index (m).

11. The Lower Tester sends a message to IUT with the old IV Index (m-1) and a different SRC address and verifies that the message is accepted and processed by the IUT by checking the message returned.
12. The Lower Tester sends a message to IUT with the new IV Index (m) and verifies that the message is accepted and processed by the IUT by checking the message returned.

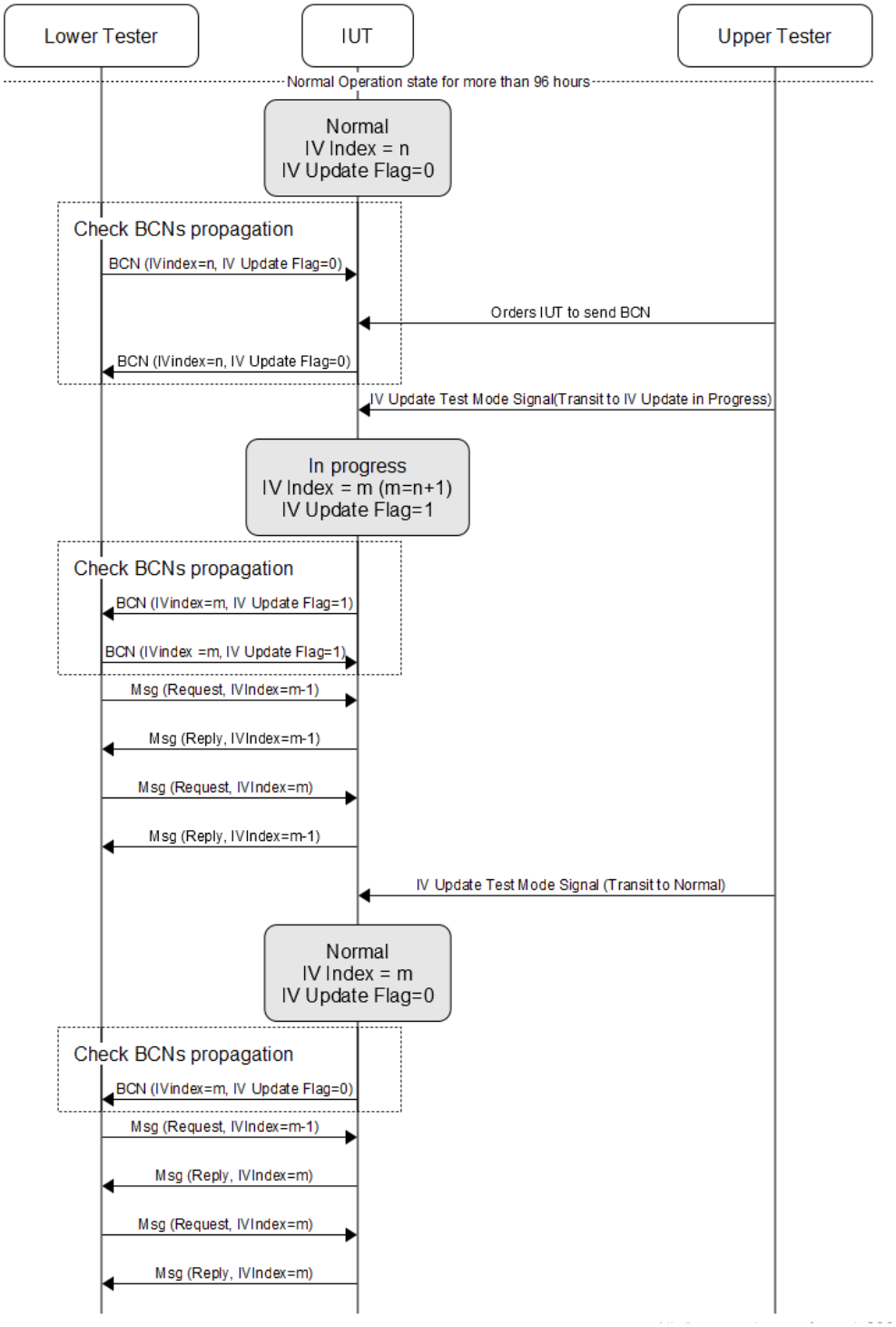


Figure 4.46: MESH/NODE/IVU/BV-02-C and MESH/NODE/IVU/BV-07-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BV-02-C [IV Update Initiated by Sending Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BV-07-C [IV Update Initiated by Sending Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.15: IV Update Initiated by Sending Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  The IUT correctly sets the SEQ field as the message originator.

  The IUT increments the SEQ field by 1 with every subsequent message sent to the Lower Tester.

  The IUT transitions to IV Update in Progress state upon receiving the Test Mode Transit to IV Update in Progress signal.

  When the IUT is in IV Update in Progress state, the IUT begins to advertise a new beacon with the IV Update Flag set to 1 and with the new IV Index (m).

  When the IUT is in IV Update in Progress state, it accepts and processes messages with new IV Index (m, or n+1) and with the previous IV Index (n, or m-1).

  The IUT continues advertising the new beacon after another node reacts to the IV Update until the IUT receives a Test Mode Transit to Normal signal and transitions to Normal Operation state.

  After the IUT returns to Normal Operation state, it advertises beacons with the IV Update flag set to 0 and the IV Index set to the new IV Index (m).

4.11.3 IV Index Recovery

- Test Purpose

  Verify that a Node IUT operating in the normal operation state will update its IV Index after receiving a Secure Network beacon or a Mesh Private beacon with an updated valid IV Index.

- Reference

  [3] 3.9.3.1, 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - IUT and the Lower Tester share previously established network security credentials.

- IV Update test mode is activated on the IUT.
- IUT is in IV Index recovery state and ready to accept any IV index update no greater than n + 42.
- Additional conditions are specified in Table 4.16.

• Test Procedure

1. The Lower Tester transmits beacons (with Beacon Type as specified by the Beacon Type column in Table 4.16) with the current IV Index and IV Update Flag set to 0.
2. The Lower Tester stops advertising and waits for the IUT to generate and send beacons with the IV Update Flag set to 0 (zero).
3. The Lower Tester generates and transmits a beacon with the IV Index set to the current value + 42 and the IV Update Flag set to 0.
4. The Upper Tester confirms that the IUT receives the beacon.
5. The IUT begins transmitting beacons with the IV Index set to the value received in the Lower Tester's beacons.

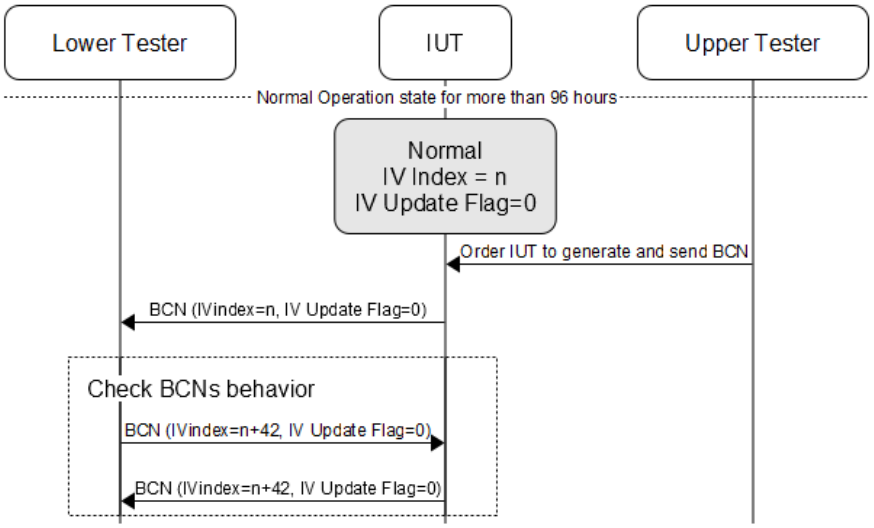


Figure 4.47: MESH/NODE/IVU/BV-03-C and MESH/NODE/IVU/BV-08-C MSC

• Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BV-03-C [IV Index Recovery using Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BV-08-C [IV Index Recovery using Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.16: IV Update when Operating in Normal Operation using Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  The IUT transmits beacons with the updated IV Index value after receiving an updated value in beacons from the Lower Tester.

4.11.4 IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment

- Test Purpose

  Verify that a Node IUT can remain in IV Update in Progress state while waiting for acknowledgment of a Transport PDU segmented message and ignore Secure Network beacons or Mesh Private beacons with the IV Update Flag set to 0.

- Reference

  [3] 3.9.3.1, 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.
  - The IUT sends Segmented Transport PDUs using the maximum value for TTL (0x7F).
  - Additional conditions are specified in Table 4.17.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.17.

- Test Procedure

  1. The Upper Tester orders the IUT to transmit Segmented Transport PDUs to the Lower Tester.
  2. The Lower Tester does not reply with the Segment Acknowledgment messages to the IUT; instead, it generates a beacon (with Beacon Type as specified by the Beacon Type column in Table 4.17) using the same network credentials and IV Update Flag set to 0 and sends this beacon to the IUT.
  3. The Lower Tester expects the IUT to perform segment retransmissions (if any) using the previous IV Index, and to receive updated beacons from the IUT after the IUT's segment transmission timer expires and the IUT has performed all segment retransmissions and considers the transaction failed.

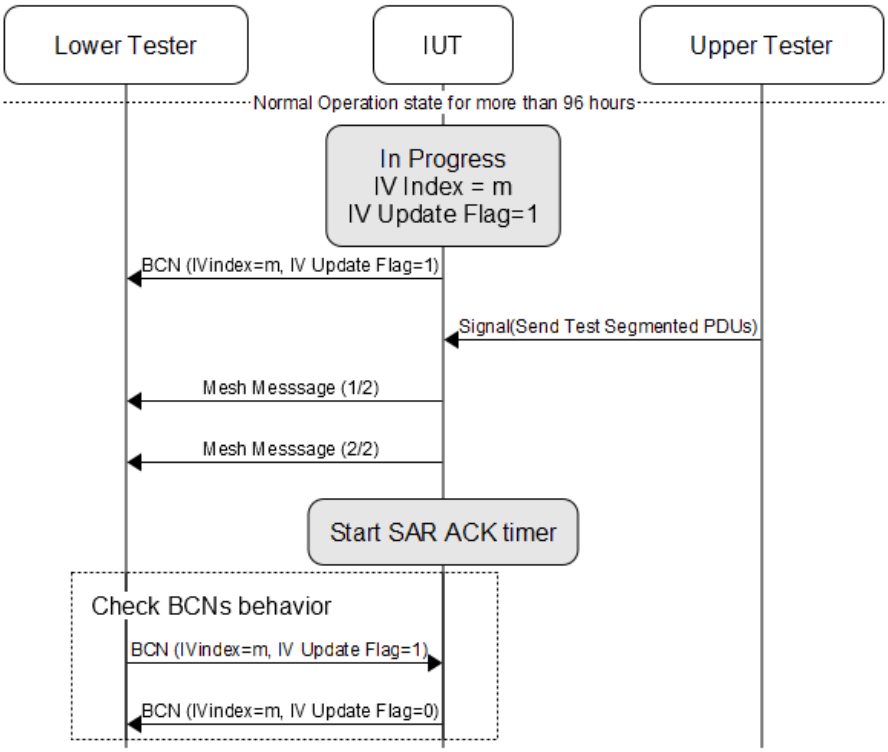


Figure 4.48: MESH/NODE/IVU/BV-04-C and MESH/NODE/IVU/BV-09-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BV-04-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment using Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01). The IUT is advertising Secure Network beacons with the IV Update Flag set to 1.	0x01 (Secure Network beacon)
MESH/NODE/IVU/BV-09-C [Stay in IV Update In Progress Operation When Waiting On Segmented Message Transport PDU Acknowledgment using Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01). The IUT is advertising Mesh Private beacons with the IV Update Flag set to 1.	0x02 (Mesh Private beacon)

Table 4.17: Stay in IV Update When Waiting On Segment Acknowledgment test cases

- Expected Outcome

  Pass verdict

  The IUT observes the beacon with the IV Update Flag set to 0.

The IUT continues to retransmit segments using the previous IV Index. After its segment transmission timer expires and the IUT has performed all segment retransmissions and considers the transaction failed, it sends beacons with the IV Update Flag set to 0 and the IV Index set to m.

4.11.5 IV Update for a Node Not on Primary Subnet

- Test Purpose

  Verify the IV Update procedure when the IUT is a node not on the primary subnet that receives a Secure Network beacon or a Mesh Private beacon with the IV Update announcement (IV Update Flag set to 1 and IV Index set to current IV Index + 1). Verify that the IUT can passively accept IV Index updates in the IV Update procedure.

- Reference

  [3] 3.9.3.1, 3.10.5

- Initial Condition

  - The IUT is a Node.
  - The IUT belongs to a subnet that is not the primary subnet (provisioned with Network Key not at NetKeyIndex set to 0).
  - The Lower Tester is a Node.
  - The Lower Tester belongs both to the primary subnet and to the same subnet as the IUT.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update procedure is not already in progress.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.18.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.18.

- Test Procedure

  1. The Lower Tester waits for the IUT to generate and transmit beacons with the IV Update flag set to 0 and the IV Index set to the current value (n).
  2. The Lower Tester generates a beacon using the same network credentials and IV Index as the IUT (n) and sends this beacon to the IUT.
  3. The Lower Tester observes the beacon advertisement(s) by the IUT.
  4. The Lower Tester sends a mesh message with the IV Index field set to the current IV Index - 1(n-1) to the IUT.
  5. The IUT accepts and processes the message from the Lower Tester.
  6. The Lower Tester sends a mesh message with the IV Index field set to current IV Index (n) to the IUT.
  7. The IUT accepts and processes the message from the Lower Tester.
  8. The Lower Tester generates a new beacon with the IV Update Flag set to 1, and the IV Index value incremented by 1 (m = n+1), and sends the updated beacon to the IUT.
  9. The IUT updates its IV Index to the new value (m = n+1) and begins sending beacons with the

updated IV Index (m).

10. The Lower Tester sends a mesh message with IV Index set to the updated IV Index (m) to the IUT.
11. The IUT accepts the message from the Lower Tester.
12. The Lower Tester sends a mesh message with IV Index set to updated IV Index -1 (m-1) to the IUT.
13. The IUT accepts and processes the message from the Lower Tester.
14. The Upper Tester sends a Transit to Normal signal to the IUT to transition to the Normal Operation state.
15. The Lower Tester expects to receive a beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the new value (m).

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BV-05-C [IV Update for a Node Not on Primary Subnet using Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BV-10-C [IV Update for a Node Not on Primary Subnet using Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.18: IV Update for a Node Not on Primary Subnet test cases

- Expected Outcome

  Pass verdict

  The IUT accepts and processes the message from the Lower Tester signaling the start of the IV Update process.

  The IUT transmits beacons with the updated IV Index (m) while the IUT is in IV Update in Progress state.

  In Step 15, the IUT sends a beacon with the IV Update Flag set to 0 and the IV Index set to the new IV Index value (m).

4.11.6 Ignore IV Update Flag When Operating In Normal Operation For Less Than the Minimum Time

- Test Purpose

  Verify that an IUT that has been in Normal Operation state for less than 96 hours after an IV Update procedure has completed can ignore Secure Network beacons or Mesh Private beacons with the IV Update Flag set to 1.

- Reference

  [3] 3.10.5

- Initial Condition

- The IUT is a Node.
- The Lower Tester is a Node.
- The IUT and the Lower Tester share previously established network security credentials.
- Additional conditions are specified in Table 4.19.
- The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.19.

• Test Procedure

1. The Lower Tester generates a beacon using the same network credentials and IV Index (n) as the IUT and begins advertising this beacon periodically, and scanning for neighboring beacons.
2. The Lower Tester stops advertising and waits for the IUT to generate and send a beacon with the same security as the Lower Tester.
3. The Lower Tester observes the beacon advertisement(s) by the IUT.
4. The Lower Tester generates a new beacon with the IV Update Flag set to 1 and IV Index incremented by 1 (n+1), and sends the updated beacon to the IUT.
5. The Upper Tester confirms that the IUT receives the beacon.
6. The Lower Tester expects to receive beacons from the IUT with the IV Update Flag set to 0 and IV Index set to the original IV Index value (n).

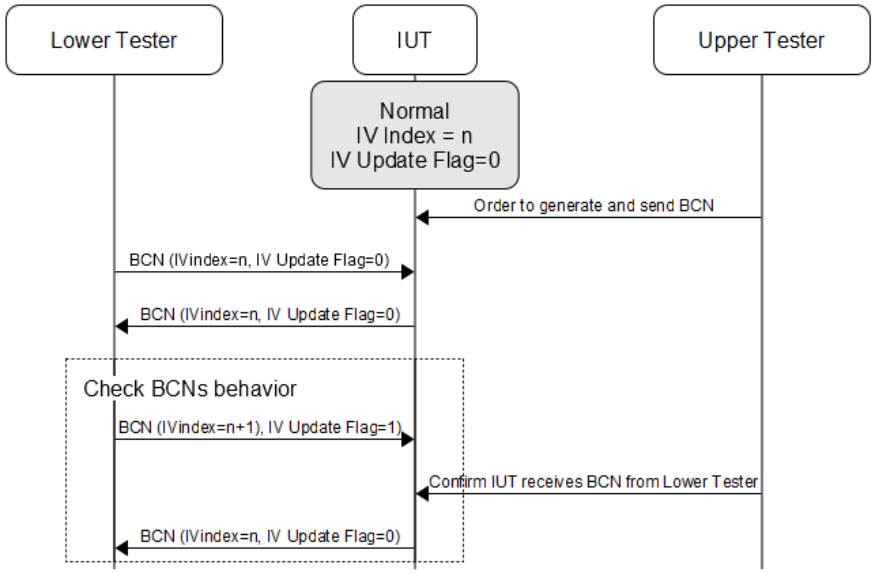


Figure 4.49: MESH/NODE/IVU/BI-01-C and MESH/NODE/IVU/BI-07-C MSC

• Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BI-01-C [Ignore IV Update Flag from Secure Network Beacons When Operating In Normal Operation For Less Than the Minimum Time]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01). The IUT has just updated its IV Index to the current value (n), e.g., by running the test case MESH/NODE/IVU/BV-01-C. The IV Update test mode is not active on the IUT.	0x01 (Secure Network beacon)
MESH/NODE/IVU/BI-07-C [Ignore IV Update Flag from Mesh Private Beacons When Operating In Normal Operation For Less Than the Minimum Time]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01). The IUT has just updated its IV Index to the current value (n), e.g., by running the test case MESH/NODE/IVU/BV-06-C. The IV Update test mode is not active on the IUT.	0x02 (Mesh Private beacon)

Table 4.19: Ignore IV Update Flag When Operating In Normal Operation test cases

- Expected Outcome

  Pass verdict

  In Step 6, the IUT sends a Secure Network beacon with the IV Update Flag set to 0 and the IV Index set to the original IV Index (n).

4.11.7 Ignore IV Update Flag When Operating In Normal Operation For an Abnormal IV Index

- Test Purpose

  Verify that a Node IUT that has been in Normal Operation state for more than 96 hours can ignore Secure Network beacons or Mesh Private beacons with the IV Update Flag set to 1 if the IV Index in the beacon is greater than the last known IV Index plus 42.

- Reference

  [3] 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.20.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.20.

- Test Procedure

  1. The Lower Tester waits for the IUT to generate and send beacons with the IV Update Flag set to 0 and the current IV Index value (n).
  2. The Lower Tester generates a new beacon with the IV Update Flag set to 1 and IV Index is greater than the last known IV Index plus 42 (n+43), and sends the updated beacon to the IUT.
  3. The Upper Tester confirms that the IUT receives the beacon.
  4. The Lower Tester expects to receive beacons from the IUT with the IV Update Flag set to 0 and the original IV Index (n).

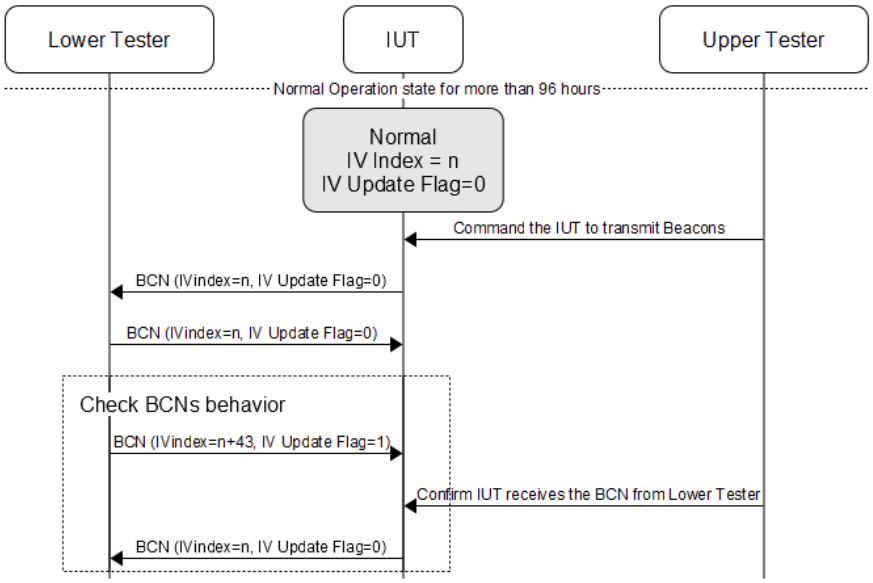


Figure 4.50: MESH/NODE/IVU/BI-02-C and MESH/NODE/IVU/BI-08-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BI-02-C [Ignore IV Update Flag in Secure Network Beacon When Operating In Normal Operation For an Abnormal IV Index]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BI-08-C [Ignore IV Update Flag in Mesh Private Beacon When Operating In Normal Operation For an Abnormal IV Index]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.20: Ignore IV Update Flag In Normal Operation for Abnormal IV Index test cases

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends beacons with IV Update Flag set to 0 and the original IV Index (n).

4.11.8 Ignore IV Update Flag When Operating In Normal Operation For Message From Other Subnets

- Test Purpose

  Verify that a Node IUT that is a member of a primary subnet and has been in the Normal Operation state will ignore Secure Network beacons or Mesh Private beacons with the IV Update Flag set to 1 and an IV Index greater than the last known IV Index from other subnets.

- Reference

  [3] 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The IUT is provisioned to the primary subnet.
  - The IUT has the Network Key of a Subnet that is not the primary subnet.
  - The Lower Tester is a Node.
  - The Lower Tester is provisioned to a subnet that is not the primary subnet.
  - The IUT and the Lower Tester share previously established network security credentials for the same non-primary subnet.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.21.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.21.

- Test Procedure

  1. The Lower Tester waits for the IUT to send beacons with the current IV Index value (n) and the IV Update Flag set to 0 (zero).
  2. The Lower Tester generates a new beacon with the IV Update Flag set to 1, and IV Index is greater than the last known IV Index (n+1), and sends the updated beacon on a subnet that is not the primary subnet.
  3. The Upper Tester confirms that the IUT receives the beacon.
  4. The Lower Tester expects to receive a beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the original value (n).

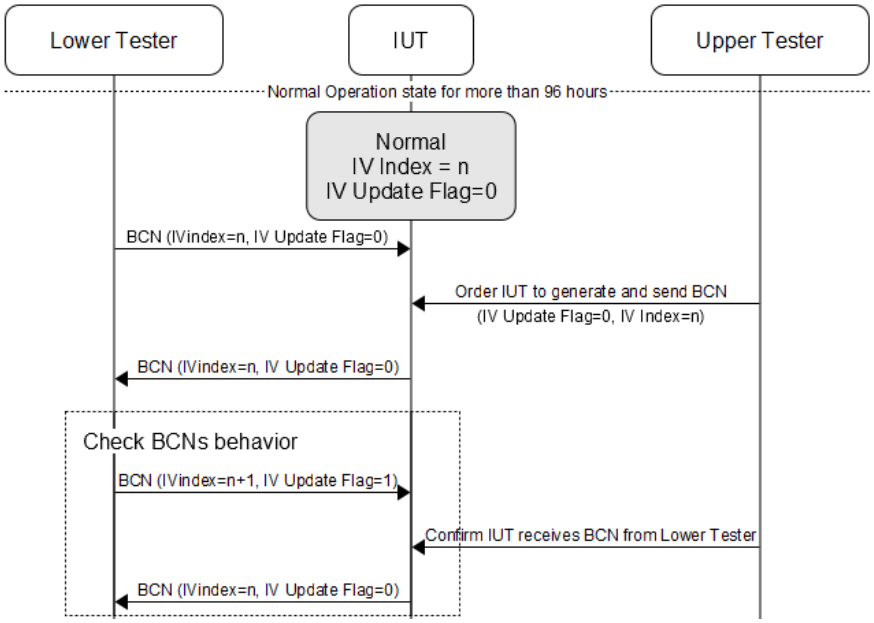


Figure 4.51: MESH/NODE/IVU/BI-04-C and MESH/NODE/IVU/BI-09-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BI-04-C [Ignore IV Update Flag in Secure Network Beacons When Operating In Normal Operation For Message From Other Subnets]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BI-09-C [Ignore IV Update Flag in Mesh Private Beacons When Operating In Normal Operation For Message From Other Subnets]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.21: Ignore IV Update Flag In Normal Operation For Message From Other Subnets test cases

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends beacons with IV Update Flag set to 0 and the original IV Index (n).

4.11.9 Ignore IV Update Flag When in Normal Operation within 96 Hours of Exiting IV Update In Progress State

- Test Purpose

  Verify that a Node IUT in Normal Operation state will ignore Secure Network beacons or Mesh Private beacons with the IV Update Flag set to 1 within 96 hours of completing an IV Update procedure.

- Reference

[3] 3.10.5

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.22.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.22.
- Test Procedure
  1. The IUT performs the IV Update procedure, e.g., by executing the procedure in MESH/NODE/IVU/BV-01-C [IV Update Initiated by Receiving Secure Network Beacons] when using Secure Network beacons or by executing the procedure in MESH/NODE/IVU/BV-06-C [IV Update Initiated by Receiving Mesh Private Beacons] when using Mesh Private beacons.
  2. After the IV Update procedure is complete, the IV Update test mode is deactivated on the IUT in order to set it to a state where it cannot accept a new IV Update procedure for at least 96 hours. The current IV Index value is n.
  3. The Lower Tester generates a new beacon with an IV Update Flag set to 1 and IV Index set to n+1, and sends the updated beacon to the IUT.
  4. The Upper Tester confirms that the IUT receives the new beacon from the Lower Tester.
  5. The Lower Tester expects to receive a beacon from the IUT with the IV Update Flag set to 0 and the IV Index set to the original value (n).

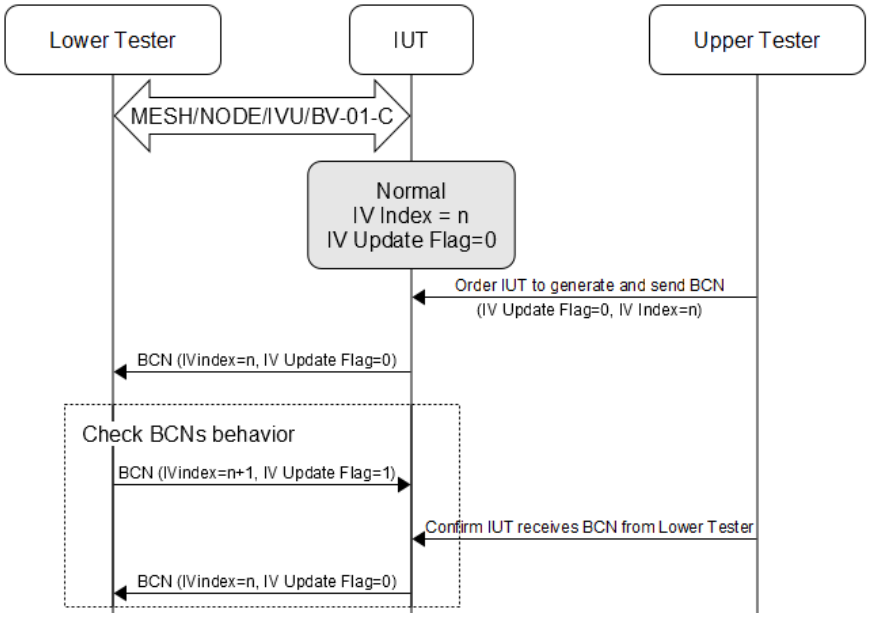


Figure 4.52: MESH/NODE/IVU/BI-05-C and MESH/NODE/IVU/BI-10-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BI-05-C [Ignore IV Update Flag in Secure Network Beacons When in Normal Operation State within 96 Hours of Exiting IV Update In Progress State]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BI-10-C [Ignore IV Update Flag in Mesh Private Beacons When in Normal Operation within 96 Hours of Exiting IV Update In Progress State]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.22: Ignore IV Update Flag in Normal Operation within 96 Hours of Exiting IV Update In Progress test cases

- Expected Outcome

  Pass verdict

  In Step 5, the IUT sends a beacon with IV Update Flag set to 0 and the original IV Index (n).

4.11.10 IV Update - Invalid IV Index Value

- Test Purpose

  Verify the IV Update procedure when the IUT is a node belonging to a primary Subnet and receives a Secure Network beacon or a Mesh Private beacon with the IV Index set to a value that cannot be accepted during an IV Update procedure.

- Reference

  [3] 3.10.5, 3.10.5.1

- Initial Condition
  - The IUT is a Node.
  - The IUT is provisioned to the primary subnet.
  - The Lower Tester is a Node.
  - The Lower Tester is provisioned to the primary subnet.
  - The IUT and the Lower Tester share previously established network security credentials.
  - IV Update test mode is activated on the IUT.
  - Additional conditions are specified in Table 4.23.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.23.
- Test Procedure

  1. The Lower Tester generates a beacon using the same network credentials and IV Index (n) as the IUT. The Lower Tester begins advertising this beacon periodically and is also scanning for neighboring beacons.
  2. The IUT detects the beacon using the correct IV Index value (n) from the Lower Tester.
  3. The Upper Tester sends a Transit to IV Update in Progress signal to the IUT to transition to the IV Update in Progress state.
  4. The Lower Tester expects the IUT to advertise a beacon with the IV Index value incremented by one (m = n+1), the IV Update Flag set to 1 (IV Update in Progress). The Lower Tester validates the beacon with the shared network security credentials.
  5. The Lower Tester generates and sends a beacon to the IUT using the same network credentials as the IUT and with the IV Index set to the value (k) greater than the new IV Index (greater than n+1).
  6. The Upper Tester confirms that the IUT receives the beacon.
  7. The Lower Tester expects the IUT to send the same beacon as received in Step 5.
  8. The Lower Tester generates and advertises a beacon using the same network credentials as the IUT and with the IV Index set to a value greater than the current IV Index (k) plus 42.
  9. The Upper Tester confirms that the IUT receives the beacon.
  10. The IUT continues to send the same beacon as received in Step 5.

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/IVU/BI-06-C [IV Update – Invalid IV Index value in Secure Network beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	0x01 (Secure Network beacon)
MESH/NODE/IVU/BI-11-C [IV Update – Invalid IV Index value in Mesh Private beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	0x02 (Mesh Private beacon)

Table 4.23: IV Update - Invalid IV Index value test cases

- Expected Outcome

  Pass verdict

  The IUT ignores beacons from the Lower Tester with an invalid IV Index value.

4.12 Key Refresh

The test group objective is to verify the Key Refresh procedure.

MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via Configuration Client Messages]

- Test Purpose

  Verify that a Node IUT supporting the Configuration Server Model can participate in the Key Refresh procedure initiated by a Configuration Client.

- Reference

  [3] 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
  - If a second subnet is supported, the IUT has at least one application key bound to the network key used in the second subnet.

- Test Procedure

  1. The Lower Tester sends a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in Step 1.
  2. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.

3. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
4. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.
5. The Lower Tester receives a Config Key Refresh Phase Status from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x02 (Second phase).
6. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
7. Repeat Steps 4–6 to verify that the Phase remains at the second phase.
8. The Lower Tester sends a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.
9. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.
10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT using the new keys.
11. If a second subnet is supported, then repeat Steps 1–10 using the second NetKeyIndex.
12. Repeat Steps 1–10, skipping Step 2, using only the updated NetKey in the Key Refresh procedure.
13. Repeat Steps 1–3 and Steps 8–10.

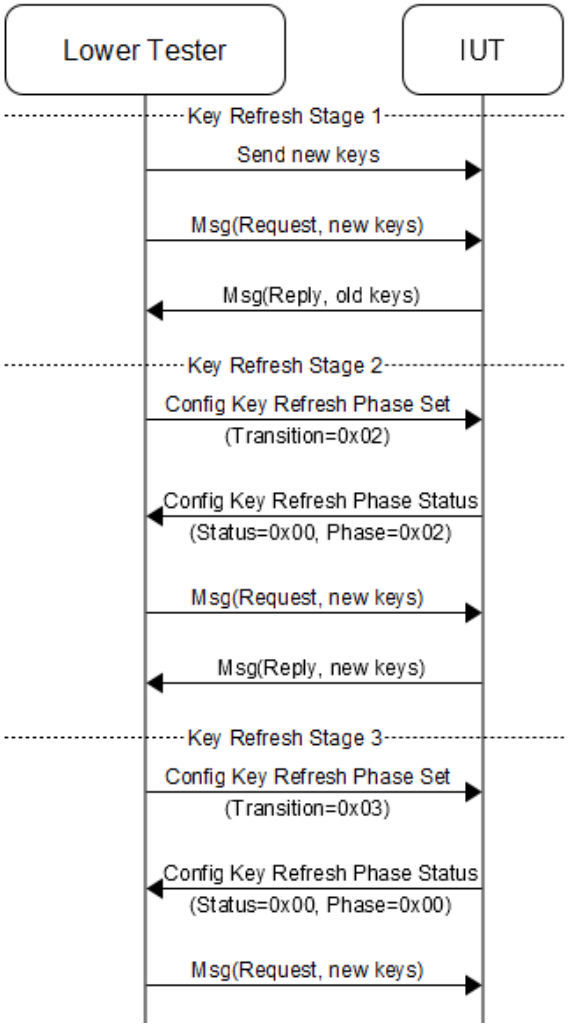


Figure 4.53: MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via Configuration Client Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not update any AppKey values when updating the NetKey value.

  In Step 5 and the repeated step from Step 7, the IUT sends a Config Key Refresh Phase Status message with the Status of 0x00, the NetKeyIndex of the new NetKey, and Phase set to 0x02.

  In Step 9 and the repeated step from Step 13, the IUT sends a Config Key Refresh Phase Status message to the Lower Tester with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.

  All messages are encrypted and authenticated with the DevKey of the IUT.

4.12.1 Key Refresh Procedure as Node via Mesh Beacons

- Test Purpose

  Verify that a Node IUT can participate in the Key Refresh procedure where the phases are initiated by the receipt of Secure Network beacons or Mesh Private beacons.

- Reference

  [3] 3.9.3.1, 3.10.4

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
  - If a second network key is supported, the IUT has at least one application key bound to it.
  - Additional conditions are specified in Table 4.24.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.24.

- Test Procedure
  1. The Lower Tester generates and advertises a beacon with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.24.
  2. The Lower Tester stops advertising and waits for the IUT to generate and send a beacon using the same values as the Lower Tester uses in Step 1.
  3. The Lower Tester sends a new NetKey to the IUT by sending a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same beacon as in Step 1.
  4. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key and continues advertising the same beacon as in Step 1.
  5. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  6. The Lower Tester generates and sends a new beacon to the IUT using the same network credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 1 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.24.
  7. The Lower Tester expects to receive a new beacon from the IUT with the following field values: the Key Refresh Flag (Flags bit 0) is set to 1 and the IV Update Flag (Flags bit 1) is set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.24. The AD Type is formatted correctly.
  8. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.

9. The Lower Tester generates and sends a new beacon to the IUT with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.24.
10. The Lower Tester expects to receive a new beacon from the IUT using the new credentials with the Key Refresh Flag set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.24.
11. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT using the new keys.
12. If a second subnet is supported, then repeat Steps 1–11 using the second NetKeyIndex.
13. Repeat Steps 1–11, skipping Step 4, using only the updated NetKey in the Key Refresh procedure.

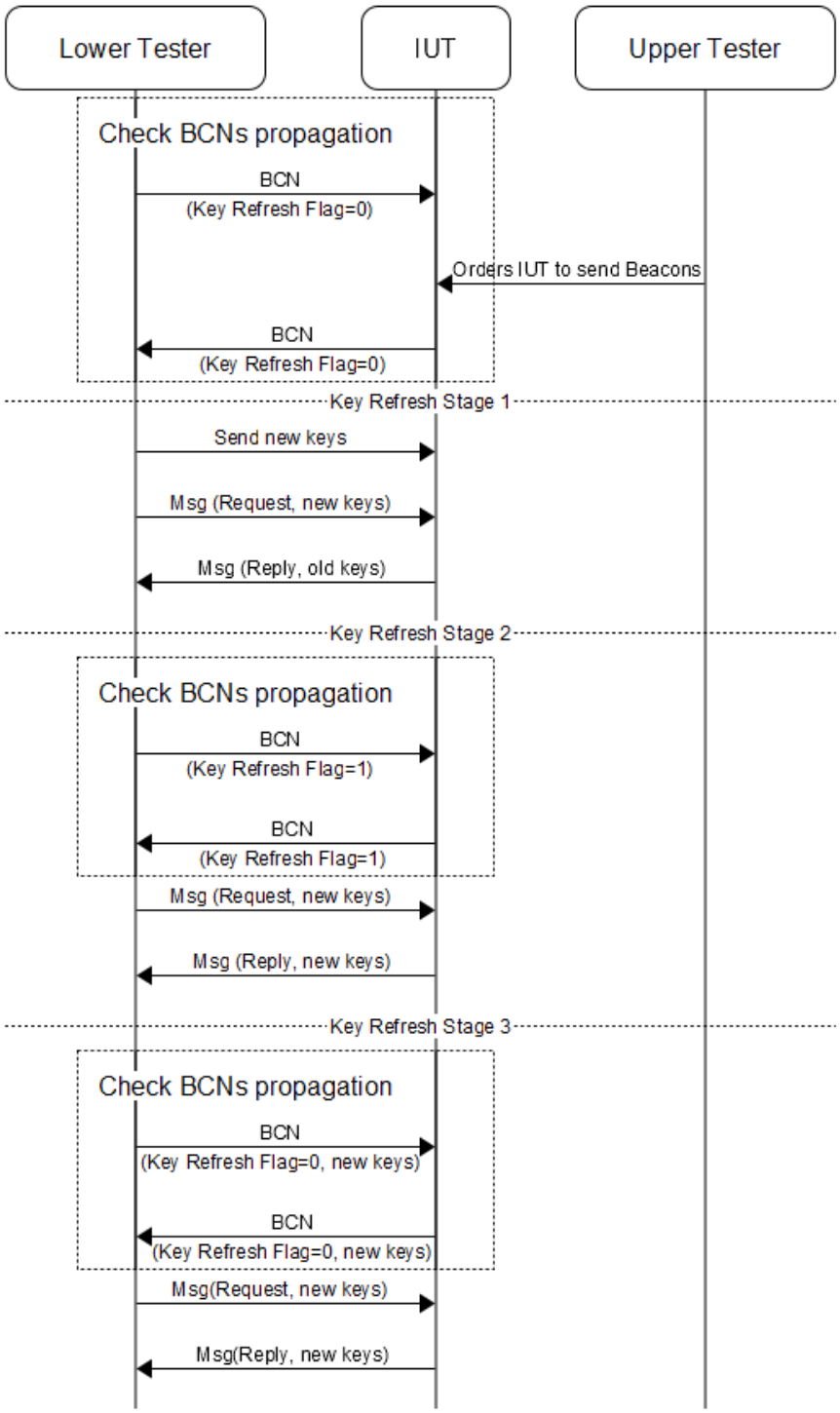


Figure 4.54: MESH/NODE/KR/BV-02-C and MESH/NODE/KR/BV-04-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/NODE/KR/BV-02-C [Key Refresh Procedure as Node via Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	The Network ID is set to the current network ID. The Authentication Value is calculated based on the Flags, Network ID, and IV index fields, using the value of the NetKey as follows: In Steps 1–5, the old key is used. In Steps 6–10, the new key is used.	0x01 (Secure Network beacon)
MESH/NODE/KR/BV-04-C [Key Refresh Procedure as Node via Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	The Private Beacon Flags and IV index fields are obfuscated and authenticated using the value of the Random field and the value of the NetKey as follows: In Steps 1–5, the old key is used. In Steps 6–10, the new key is used.	0x02 (Mesh Private beacon)

Table 4.24: Key Refresh Procedure as Node via Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  In Step 2, the IUT transmits a correctly formatted beacon with valid values.

  In Step 3, the IUT does not update any AppKey values when updating the NetKey value.

  In Step 7, the IUT sends beacons with the Key Refresh Flag enabled and secured with new keys.

  In Step 10, the IUT sends beacons using the new keys with the Key Refresh Flag set to 0 and other fields updated based on the new NetKey.

  All messages are encrypted and authenticated with the DevKey of the IUT.

  For beacons with the Mesh Private beacon type, when a new beacon is sent as a result of the Key Refresh Flag being flipped, the Random field and the advertising address both change.

4.12.2 Key Refresh Procedure Skipping Phase 2

- Test Purpose

  Verify that a Node IUT in Phase 1 of the Key Refresh procedure can skip Phase 2 when it receives a Secure Network beacon or a Mesh Private beacon with the Key Refresh flag set to 0 and the new NetKey.

- Reference

[3] 3.9.3.1, 3.10.4.1

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.25.
- Test Procedure
  1. Execute Steps 1–5 of MESH/NODE/KR/BV-01-C [Key Refresh Procedure as Node via Configuration Client Messages].
  2. The Lower Tester generates and sends a new beacon to the IUT using the same network credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.25.
  3. The Lower Tester expects to receive an updated beacon from the IUT with the new security credentials and the Key Refresh flag set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.25.

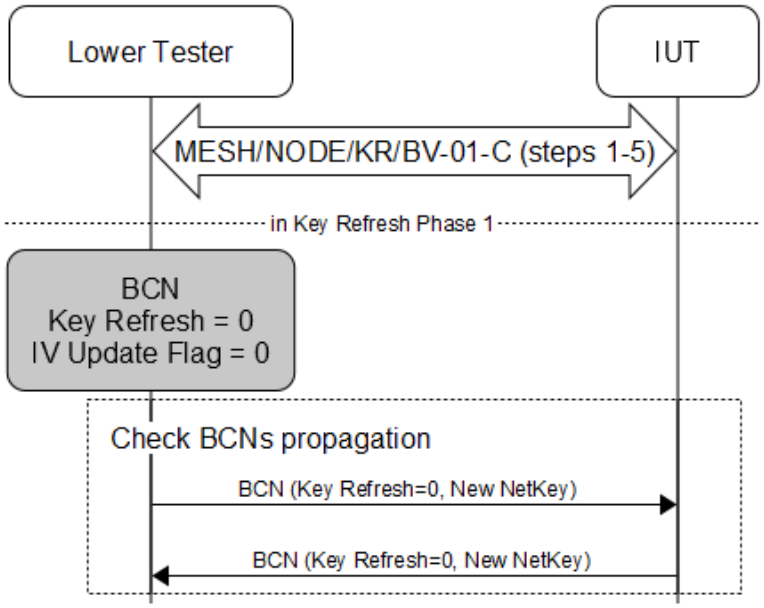


Figure 4.55: MESH/NODE/KR/BV-03-C and MESH/NODE/KR/BV-05-C MSC

- Test Case Configuration

Test Case	Additional Beacon Requirements	Beacon Type
MESH/NODE/KR/BV-03-C [Key Refresh Procedure Skipping Phase 2 via Secure Network Beacons]	The Network ID is set to the current network ID. The Authentication Value is calculated based on the Flags, Network ID and IV index fields, using the new key.	0x01 (Secure Network beacon)
MESH/NODE/KR/BV-05-C [Key Refresh Procedure Skipping Phase 2 via Mesh Private Beacons]	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the new key.	0x02 (Mesh Private beacon)

Table 4.25: Key Refresh Procedure Skipping Phase 2 test cases

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a beacon with the same network credentials as the Lower Tester with the Key Refresh flag set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.25.

MESH/CFGCL/KR/BV-01-C [Key Refresh as Configuration Client via Configuration Messages]

- Test Purpose

  Verify that an IUT supporting the Configuration Client Model can perform the Key Refresh procedure with a Node supporting the Configuration Server Model. The test procedure is executed with the Relay feature disabled and again with the Relay feature enabled if supported.

- Reference

  [3] 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14

- Initial Condition
  - The IUT is a Configuration Client.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure
  1. The Upper Tester orders the IUT to transmit a Config Key Refresh Phase Get message to the Lower Tester with the NetKeyIndex set to the current Network Key.
  2. The Upper Tester confirms that the IUT receives a Config Key Refresh Phase Status message in response from the Lower Tester using the correct keys with the Status set to 0x00 and the Phase set to 0x00 (Normal operation).
  3. The Upper Tester orders the IUT to send a new NetKey to the Lower Tester using the procedure in

MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]. The Lower Tester stores the new key and continues advertising the same Secure Network beacon as in Step 1.

4. The Upper Tester orders the IUT to send a new AppKey to the Lower Tester using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages].
5. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
6. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.
7. The Lower Tester sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x02.
8. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.
9. The Lower Tester transmits a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.
10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.

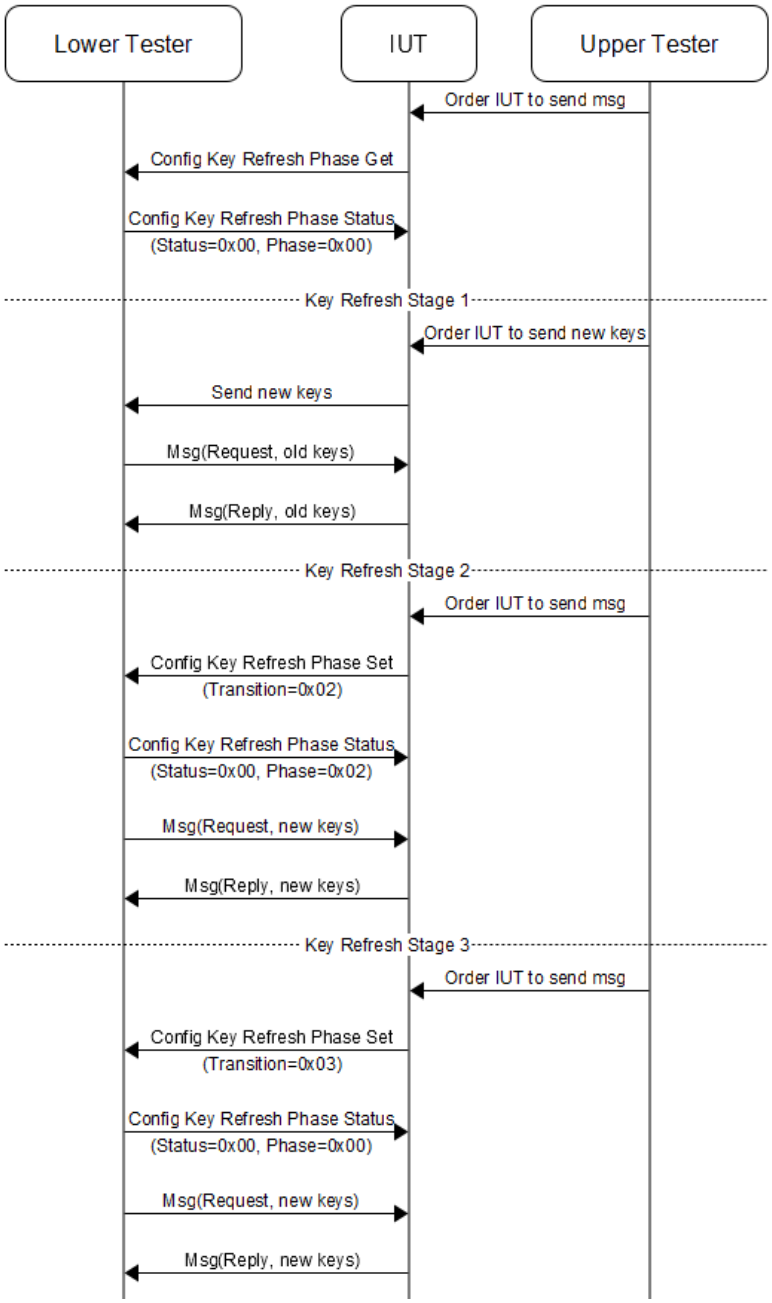


Figure 4.56: MESH/CFGCL/KR/BV-01-C [Key Refresh as Configuration Client via Configuration Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT transmits new keys to the Lower Tester. These keys are not the same as the ones defined in the sample data.

  In Step 6, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and the Transition set to 0x02. The IUT receives a Config Key Refresh Phase Status message in

response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x02.

In Step 8, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x02 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the NetKeyIndex for the new key, and Phase set to 0x00.

All messages are encrypted and authenticated with the DevKey of the Lower Tester.

4.12.3 Key Refresh as Configuration Client via Mesh Beacons

- Test Purpose

  Verify that an IUT supporting the Configuration Client Model can perform the Key Refresh procedure with a Node supporting the Configuration Server Model. The test procedure is executed with the Relay feature enabled (if supported) and with the Relay feature disabled.

- Reference

  [3] 3.9.3.1, 3.10.4

- Initial Condition

  - The IUT is a Configuration Client.
  - The Lower Tester is a Node.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Additional conditions are specified in Table 4.26.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.26.

- Test Procedure

  1. The IUT is induced to send a new NetKey to the Lower Tester, using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]. The Lower Tester stores the new key and is advertising beacons.
  2. The IUT is induced to send a new AppKey to the Lower Tester, using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages]. The Lower Tester stores the new key and continues advertising the same beacon as in Step 1.
  3. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
  4. The Upper Tester orders the IUT to generate and advertise a new beacon with the Key Refresh Flag (Flags bit 0) set to 1 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.26.
  5. The Lower Tester receives the new beacon.
  6. The Lower Tester generates and sends a new beacon with the Key Refresh Flag set to 1 and authenticated or secured (based on Beacon Type) with the new NetKey.
  7. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
  8. The Upper Tester orders the IUT to generate and advertise a new beacon with the Key Refresh Flag (Flags bit 0) set to 0, the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are

described by the Additional Beacon Requirements column in Table 4.26.

9. The Lower Tester expects to receive the new beacon from the IUT. Upon receipt, it discards the old credentials and begins transmitting beacons using the new credentials with the Key Refresh bit cleared; other requirements for the beacons are described by the Additional Beacon Requirements column in Table 4.26.
10. The Lower Tester sends a mesh message with the new keys to the IUT and receives a response from the IUT with a mesh message using the new keys.

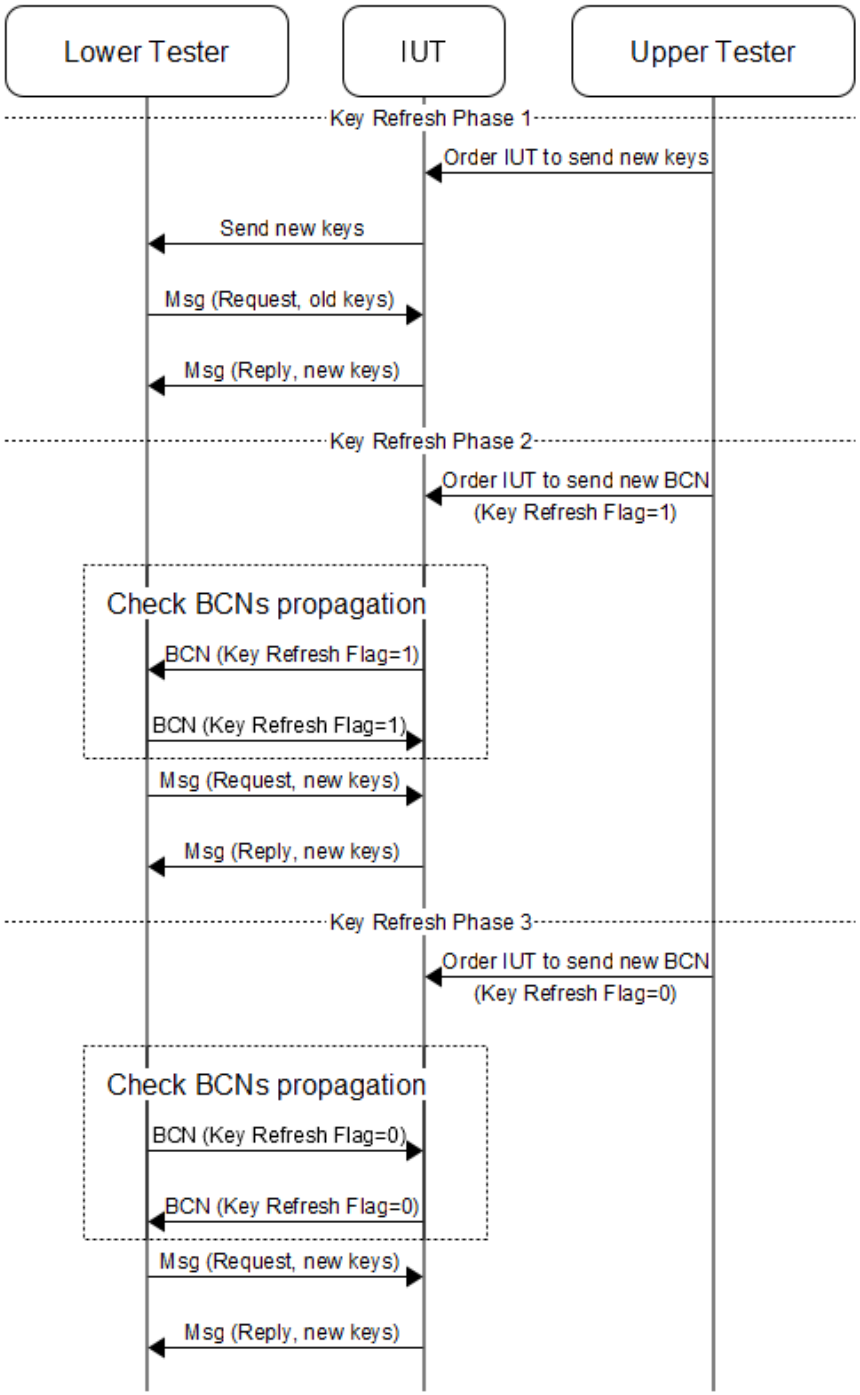


Figure 4.57: MESH/CFGCL/KR/BV-02-C and MESH/CFGCL/KR/BV-05-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/CFGCL/KR/BV-02-C [Key Refresh as Configuration Client via Secure Network Beacons]	The IUT and the Lower Tester can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is calculated based on the Flags, Network ID, and IV index fields, using the new key.	0x01 (Secure Network beacon)
MESH/CFGCL/KR/BV-05-C [Key Refresh as Configuration Client via Mesh Private Beacons]	The IUT and the Lower Tester can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the new key.	0x02 (Mesh Private beacon)

Table 4.26: Key Refresh as Configuration Client via Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  The IUT receives a beacon from the Lower Tester that is formatted correctly and with valid values.

  The IUT sends a new NetKey to the Lower Tester.

  The IUT sends a new AppKey to the Lower Tester.

  These keys are not the same as the ones defined in the sample data.

  The IUT transmits a beacon to the Lower Tester with the Key Refresh Flag enabled and begins advertising beacons with the Key Refresh Flag enabled and secured with the new key.

  The IUT transmits a new beacon with the expected values. The IUT revokes the old keys.

  All messages are encrypted and authenticated with the DevKey of the Lower Tester.

MESH/CFGCL/KR/BV-03-C [Key Refresh with Reject Listing via Configuration Messages]

- Test Purpose

  Verify that an IUT acting as Configuration Client can perform the Key Refresh procedure using configuration messages while excluding nodes that are reject listed by the IUT.

- Reference

  [3] 3.10.4

- Initial Condition
  - The IUT is a Configuration Client.
  - Lower Tester 1 is a Node.
  - Lower Tester 2 is a Node.
  - The Relay feature is not enabled on Lower Tester 1 (if Relay Mode is supported, the Relay state is set to 0x00, otherwise it is set to 0x02).
  - The IUT and both Lower Testers share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to send a new NetKey to the IUT using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages].
  2. The Upper Tester orders the IUT to send a new AppKey to Lower Tester 1 using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages].
  3. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to Lower Tester 1 with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x02.
  4. Lower Tester 1 sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, the NetKeyIndex set to the index of the new NetKey, and the Phase set to 0x02.
  5. The Upper Tester orders the IUT to send a Config Key Refresh Phase Set message to Lower Tester 1 with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x03.
  6. Lower Tester 1 sends a Config Key Refresh Phase Status message to the IUT with the Status set to 0x00, and the Phase set to 0x00.
  7. Lower Tester 2 sends a mesh message to the IUT. The Upper Tester confirms that the IUT ignores the message.
  8. If Relay feature is supported by Lower Tester 1, set its Relay state to 0x01 and repeat Steps 1–7 with Lower Tester 1 acting as a Relay node.

- Expected Outcome

  Pass verdict

  The IUT sends new keys to Lower Tester 1. These keys are not the same as the ones defined in the sample data.

  In Step 3, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x02 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x02.

  In Step 5, the IUT transmits a Config Key Refresh Phase Set message with the NetKeyIndex for the new NetKey, and Transition set to 0x03 and receives a Config Key Refresh Phase Status message in response from the Lower Tester with Status of 0x00, the new key, and Phase set to 0x00.

In Step 7, the IUT ignores the mesh message.

All messages are encrypted and authenticated with the DevKey of the Lower Tester.

4.12.4 Key Refresh with Reject Listing via Mesh Beacons

- Test Purpose

  Verify that an IUT acting as Configuration Client can perform the Key Refresh procedure using Secure Network beacons or Mesh Private beacons while excluding nodes that are reject listed by the IUT.

- Reference

  [3] 3.9.3.1, 3.10.4

- Initial Condition
  - The IUT is a Configuration Client.
  - Lower Tester 1 is a Node.
  - Lower Tester 2 is a Node that has been reject listed.
  - The Relay feature is not enabled on Lower Tester 1 (if Relay Mode is supported, the Relay state is set to 0x00, otherwise it is set to 0x02).
  - The IUT and both Lower Testers share previously established network security credentials.
  - Additional conditions are specified in Table 4.27.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.27.

- Test Procedure
  1. Lower Tester 1 advertises beacons with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacons are described by the Additional Beacon Requirements column in Table 4.27.
  2. The IUT is induced to send a new AppKey to Lower Tester 1, using the procedure in MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages]. The IUT sends a new NetKey to the IUT using the procedure in MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]. Lower Tester 1 stores the new keys and continues advertising the same beacon as in Step 1. Lower Tester 2 does not receive new keys.
  3. The Upper Tester orders the IUT to generate and send a new beacon with the Key Refresh Flag set to 1, secured with the new NetKey.
  4. Lower Tester 1 receives the new beacon from the IUT. Lower Tester 1 begins advertising a new beacon with the Key Refresh Flag set to 1, secured with the new NetKey.
  5. Lower Tester 2 does not expect to receive the new beacon from the IUT.
  6. The Upper Tester orders the IUT to generate and send a new beacon with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.27.
  7. Lower Tester 1 expects to receive the new beacon from the IUT. Upon receipt, it discards the old credentials and begins transmitting beacons using the new credentials with the Key Refresh bit cleared; other requirements for the beacons are described by the Additional Beacon Requirements

column in Table 4.27.

8. Lower Tester 2 does not expect to receive the new beacon.
9. If Relay feature is supported by Lower Tester 1, set its Relay state to 0x01 and repeat Steps 1–8 with Lower Tester 1 acting as a Relay node.

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/CFGCL/KR/BV-04-C [Key Refresh with Reject Listing via Secure Network Beacons]	The IUT and Lower Tester 1 can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is calculated based on the Flags, Network ID, and IV index fields, using the new key.	0x01 (Secure Network beacon)
MESH/CFGCL/KR/BV-06-C [Key Refresh with Reject Listing via Mesh Private Beacons]	The IUT and Lower Tester 1 can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the new key.	0x02 (Mesh Private beacon)

Table 4.27: Key Refresh with Reject Listing via Mesh Beacons test cases

- Expected Outcome

  Pass verdict

  The IUT sends new keys to Lower Tester 1. These keys are not the same as the ones defined in the sample data.

  In Step 3, the IUT sends a new beacon with the Key Refresh Flag enabled and secured using the new key.

  In Step 6, the IUT sends a new beacon with the Key Refresh Flag set to 0 and secured using the new key.

MESH/NODE/KR/BI-01-C [Receive Config Key Refresh Phase Set Messages with Valid and Invalid Transition Values]

- Test Purpose

  Verify that a Node IUT participating in the Key Refresh procedure responds appropriately to Config Key Refresh Phase Set messages with valid and invalid Transition field values during each Key Refresh Phase.

- Reference

  [3] 3.1.2, 3.10.4, 4.3.2.59, 4.3.2.60

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The Lower Tester has previously discovered the ElementAddress and Model Identifier of the Mesh Configuration Model and using that ElementAddress for all communication.
  - Key Refresh procedure is not in progress (Key Refresh Phase=0x00) on the IUT.
- Test Procedure
  1. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x01.
  2. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  3. Repeat Step 1 substituting a Transition field value of 0x02.
  4. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  5. Repeat Step 1 substituting a Transition field value of 0x03.
  6. The Lower Tester receives a Config Key Refresh Phase Status message with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase field set to 0x00 (Normal operation).
  7. To begin Phase 1 of Key Refresh procedure, the Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT stores the new key.
  8. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
  9. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x00.
  10. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  11. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x02.
  12. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00 (Success), the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).
  13. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x01.
  14. The IUT ignores the Config Key Refresh Phase Set message from the Lower Tester and the Lower Tester expects no response.
  15. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition field set to 0x02.

16. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).

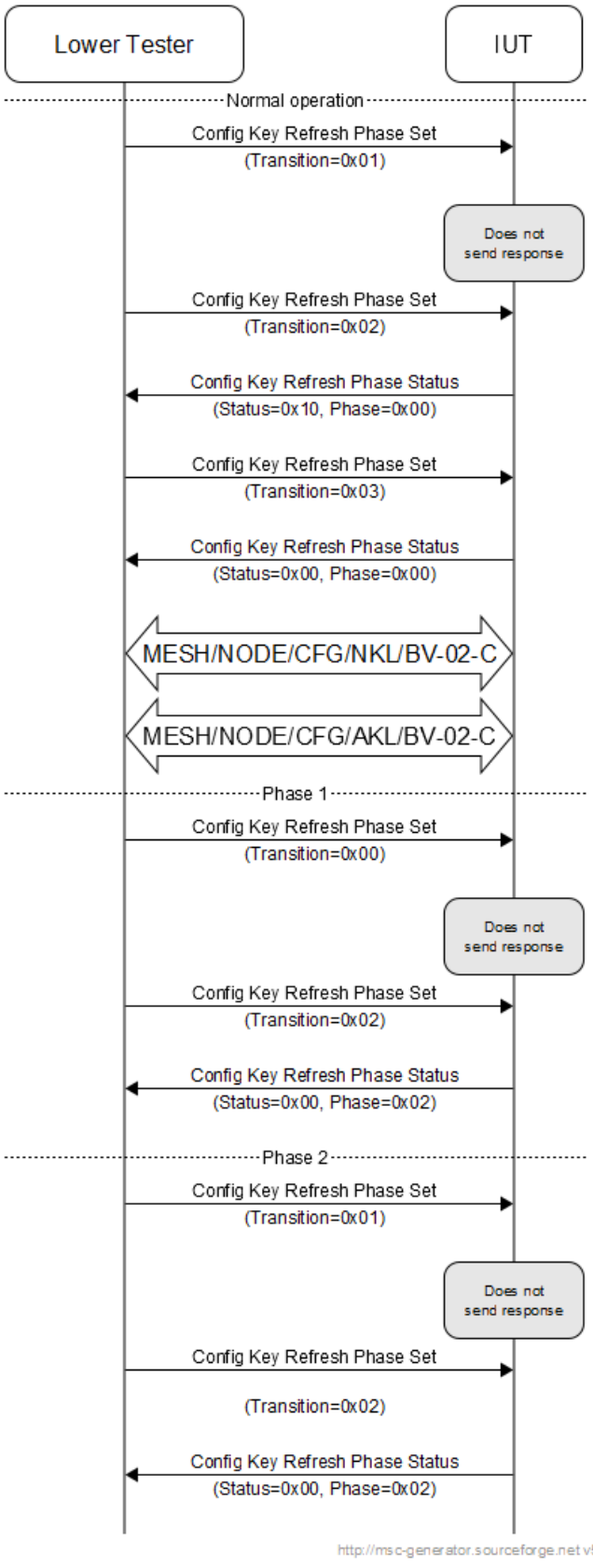


Figure 4.58: MESH/NODE/KR/BI-01-C [Receive Config Key Refresh Phase Set Messages with Valid and Invalid Transition Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not transmit a Config Key Refresh Phase Status message.

  In Step 4, the IUT does not transmit a Config Key Refresh Phase Status message.

  In Step 6, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x00.

  In Step 10, the IUT does not transmit a Config Key Refresh Phase Status message.

  In Step 12, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x02.

  In Step 14, the IUT does not transmit a Config Key Refresh Phase Status message.

  In Step 16, the IUT transmits a Config Key Refresh Phase Status message with the Status set to 0x00 and the Phase set to 0x02.

MESH/NODE/KR/BI-02-C [Ignore New Keys During Key Refresh Procedure]

- Test Purpose

  Verify that a Node IUT supporting the Configuration Server Model and participating in the Key Refresh procedure will ignore Config NetKey Update and Config AppKey Update messages after Phase 1 of Key Refresh Procedure is complete.

- Reference

  [3] 3.10.4, 4.3.2.32, 4.3.2.34, 4.3.2.38, 4.3.2.40, 4.3.2.59, 4.3.2.60

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Key Refresh procedure is not in progress (Key Refresh Phase=0x00) on the IUT.
- Test Procedure
  1. The Lower Tester attempts to send a new AppKey to the IUT using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT does not store the new AppKey value.
  2. The Lower Tester receives a Config AppKey Status from the IUT with the Status field set to 0x0B (Cannot Update).
  3. To begin Phase 1 of Key Refresh procedure, the Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT stores the new key.

4. The Lower Tester receives a Config NetKey Status message with the Status field set to 0x00.
5. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
6. The Lower Tester receives a Config AppKey Status message with the Status field set to 0x00.
7. The Lower Tester attempts to send a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message].
8. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x0B (Cannot Update).
9. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the current NetKey, and the Transition set to 0x02.
10. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the value sent in the Config Key Refresh Phase Set message and the Phase set to 0x02 (Second phase).
11. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The Lower Tester sends a new NetKey to the IUT using the procedure in MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]. The IUT does not store the new keys.
12. The Lower Tester receives from the IUT a Config AppKey Status message with the Status field set to 0x0B and a Config NetKey Status message with the Status field set to 0x0B.

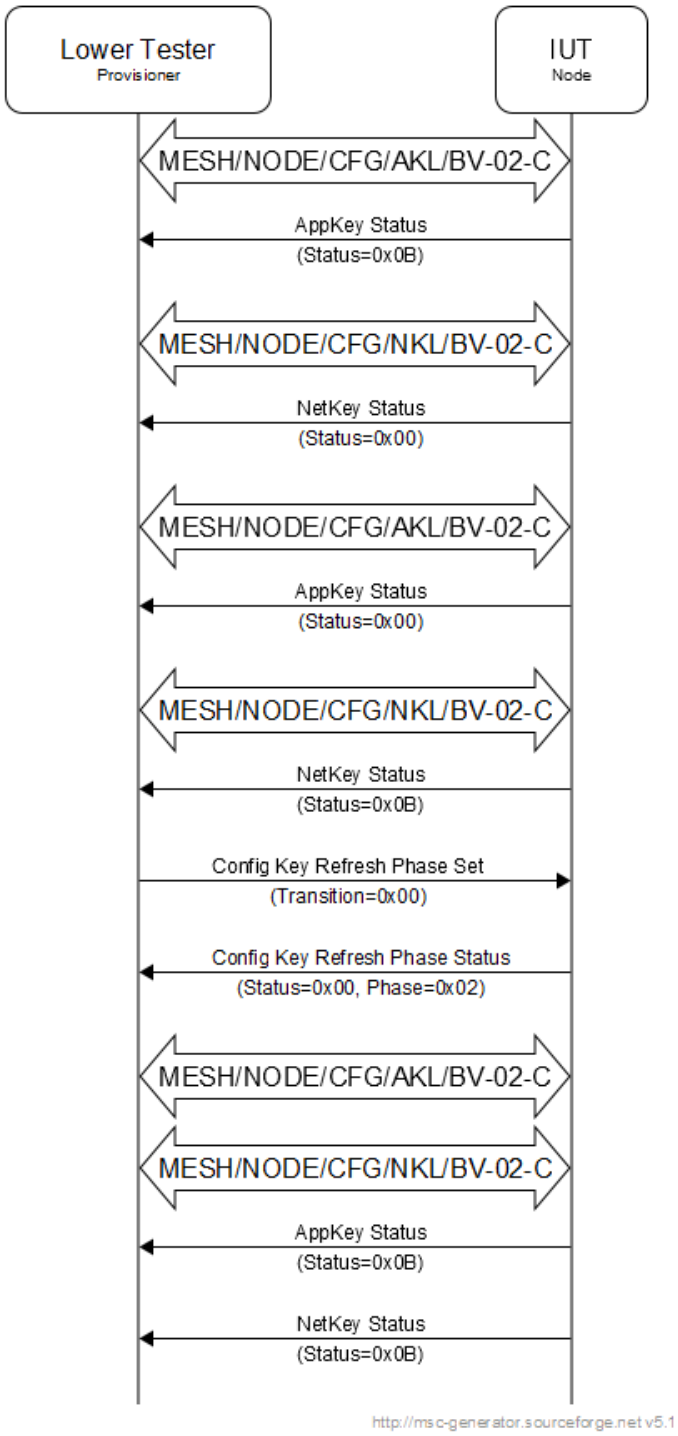


Figure 4.59: MESH/NODE/KR/BI-02-C [Ignore New Keys During Key Refresh Procedure] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config AppKey Status message with the Status field set to 0x0B.

  In Step 4, the IUT sends a Config NetKey Status message with the Status field set to 0x00.

In Step 6, the IUT sends a Config AppKey Status message with the Status field set to 0x00.

In Step 8, the IUT sends a Config NetKey Status message with the Status field set to 0x0B.

In Step 10, the IUT sends a Config Key Refresh Phase Status message.

In Step 12, the IUT sends a Config AppKey Status with the Status field set to 0x0B and a Config NetKey Status message with the Status field set to 0x0B.

MESH/NODE/KR/BI-03-C [Receive Config Key Refresh Phase Get with Invalid NetKeyIndex]

- Test Purpose

  Verify that a Node IUT can respond to a Config Key Refresh Phase Get message with an invalid NetKeyIndex value with a Config Key Refresh Phase Status message containing the correct Status code.

- Reference

  [3] 3.9.3.1, 3.10.4, 4.4.1.2.14

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.

- Test Procedure
  1. The Lower Tester sends a Config Key Refresh Phase Get message to the IUT with the NetKeyIndex set to a value higher than the largest NetKey Index on the IUT.
  2. The Lower Tester expects a Config Key Refresh Phase Status message in response from the IUT with the Status set to 0x04.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Key Refresh Phase Status message with the Status set to 0x04.

MESH/NODE/KR/BI-04-C [Receive Messages with Old Keys During Configuration Client Key Refresh]

- Test Purpose

  Verify that a Node IUT supporting the Configuration Server Model can participate in the Key Refresh procedure initiated by a Configuration Client.

- Reference

[3] 3.10.4, 4.2.14, 4.3.2.58, 4.3.2.59, 4.3.2.60, 4.4.2.2.14

- Initial Condition
  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
- Test Procedure
  1. The Lower Tester transmits a Config Key Refresh Phase Get message to the IUT with the NetKeyIndex set to the current Network Key and expects a Config Key Refresh Phase Status message in response from the IUT with the Phase set to 0x00 (Normal operation).
  2. The Lower Tester sends a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same Secure Network beacon as in Step 1.
  3. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key.
  4. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  5. The Lower Tester transmits a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x02.
  6. The Lower Tester receives a Config Key Refresh Phase Status from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x02 (Second phase).
  7. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
  8. The Lower Tester sends a Config Key Refresh Phase Set message to the Lower Tester with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition set to 0x03.
  9. The Lower Tester receives a Config Key Refresh Phase Status message from the IUT with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the new NetKey, and the Phase set to 0x00.
  10. The Lower Tester sends a mesh message with the old keys to the IUT and receives no response from the IUT.
- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends a reply to the Lower Tester using the old keys.

  In Step 7, the IUT sends a reply to the Lower Tester using the new keys.

  In Step 10, the IUT does not send a reply to the Lower Tester.

  All messages are encrypted and authenticated with the DevKey of the IUT.

4.12.5 Receive Messages with Old Keys During Mesh Beacon Key Refresh

- Test Purpose

  Verify that a Node IUT can participate in the Key Refresh procedure where the phases are initiated by the receipt of Secure Network beacons or Mesh Private beacons.

- Reference

  [3] 3.9.3.1, 3.10.4

- Initial Condition

  - The IUT is a Node.
  - The Lower Tester is a Configuration Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT is in normal operation. Key Refresh procedure is not in progress.
  - Additional conditions are specified in Table 4.28.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.28.

- Test Procedure

  1. The Lower Tester generates and advertises a beacon with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.28.
  2. The Lower Tester stops advertising and waits for the IUT to generate and send a beacon using the same values as the Lower Tester uses in Step 1.
  3. The Lower Tester sends a new NetKey to the IUT by sending a Config NetKey Update message to the IUT with a NetKeyIndex value of 0 and a new NetKey. The IUT stores the new key and continues advertising the same beacon as in Step 1.
  4. The Lower Tester sends a new AppKey to the IUT, using the procedure in MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]. The IUT stores the new key and continues advertising the same beacon as in Step 1.
  5. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the old keys.
  6. The Lower Tester generates and sends a new beacon to the IUT using the same network credentials as the IUT with the Key Refresh Flag (Flags bit 0) set to 1 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.28.
  7. The Lower Tester expects to receive a new beacon from the IUT with the following field values: the Key Refresh Flag (Flags bit 0) is set to 1 and the IV Update Flag (Flags bit 1) is set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.28. The AD Type is formatted correctly.
  8. The Lower Tester sends a mesh message with the old keys to the IUT and receives a response from the IUT with a mesh message using the new keys.
  9. The Lower Tester generates and sends a new beacon to the IUT with the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0; other requirements for the beacon are

described by the Additional Beacon Requirements column in Table 4.28.

10. The Lower Tester expects to receive a new beacon from the IUT using the new credentials with the Key Refresh Flag set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.28.
11. The Lower Tester sends a mesh message with the old keys to the IUT and receives no response.

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/NODE/KR/BI-05-C [Receive Messages with Old Keys During Secure Network Beacon Key Refresh]	The IUT and Lower Tester 1 can broadcast Secure Network beacons (Secure Network Beacon state set to 0x01).	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is calculated based on the Flags, Network ID, and IV index fields, using the value of the NetKey as follows: In Step 1, the old key. In Steps 6, 7, and 9, the new key.	0x01 (Secure Network beacon)
MESH/NODE/KR/BI-06-C [Receive Messages with Old Keys During Mesh Private Beacon Key Refresh]	The IUT and Lower Tester 1 can broadcast Mesh Private beacons (Private Beacon state set to 0x01).	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field and using the value of the NetKey as follows: In Step 1, the old key. In Steps 6, 7, and 9, the new key.	0x02 (Mesh Private beacon)

Table 4.28: Receive Messages with Old Keys During Mesh Beacon Key Refresh test cases

- Expected Outcome

  Pass verdict

  In Step 5, the IUT sends a message to the Lower Tester authenticated using the old keys.

  In Step 8, the IUT sends a message to the Lower Tester authenticated using the new keys.

  In Step 11, the IUT does not respond to the Lower Tester.

  All messages are encrypted and authenticated with the DevKey of the IUT.

4.13 Friendship

The test group objective is to verify functionality related to the Friend feature, including Friend Establishment, Friend Messaging, and Friend Management.

4.13.1 Friend Node

This section contains tests related to behavior from the perspective of a node participating in Friendship procedures as a Friend node.

MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]

- Test Purposes

  Verify that an IUT node supporting the Friend feature can perform the Friendship Establishment procedure with a Low Power Node.

- Reference

  [3] 3.6.6, 3.6.5.1, 3.6.5.2, 3.6.5.3, 3.6.5.4, 3.6.6.3.1, 4.2.2.4

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- Test Procedure

  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL field value of 0x00, Criteria field value of 0x21, ReceiveDelay value of 0xA0, PollTimeout value of 0x00012C, a LPNCounter field set to 0x00, NumElements field value set to 0x01, and a PreviousAddress field value set to 0x0000. The Lower Tester then waits 100 ms.
  2. The Lower Tester expects to receive a Friend Offer message. The Friend Offer message should be sent after the Friend Offer Delay which is calculated by using the RSSI factor and Received Window Factor.
  3. The Lower Tester sends a Friend Poll message addressed to the IUT within 1 second of receiving the Friend Offer message with an FSN field value set to 0. The Friend Poll message is secured using the friendship security credentials found in [3] Section 3.8.5.3.1.
  4. The Lower Tester expects to receive a Friend Update message from the IUT.

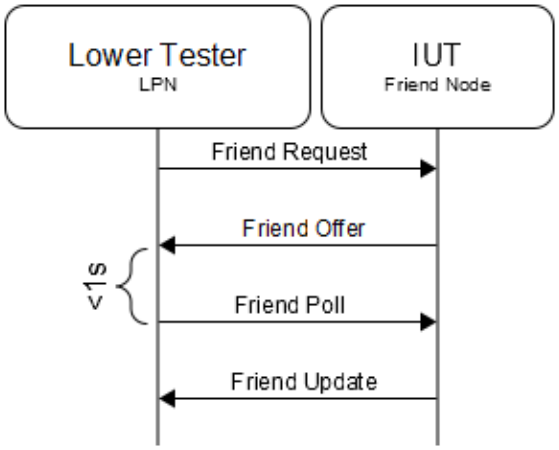


Figure 4.60: MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Offer message to the Lower Tester's unicast address, with a TTL value of 0, a TCM Message Opcode value of 0x04, a ReceiveWindow field value between 0x01 - 0xFF, RSSI value between 0x00 - 0xFF, and a FriendshipCounter field with valid 2-byte value starting at 1 for first Friend Offer and incrementing by 1 for each additional.

  The IUT sends the Friend Offer message after the calculated Friend Offer Delay.

  The IUT sends a Friend Update message within the [ReceiveDelay, ReceiveDelay + ReceiveWindow] time interval from the Friend Poll message from the Lower Tester.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-02-C [Friendship Messaging – Friend Node]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can cache Unsegmented Access messages and report them to an LPN.

- Reference

  [3] 3.1.2, 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition

  - The IUT is a node that supports the Friend Feature and the feature is enabled.

- Lower Tester 1 is a node that supports the Low Power Feature.
- Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time after the friendship establishment in Step 1, the Transport Control Message Opcode field is set to 0x01, the RFU bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.
3. Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.
4. Lower Tester 2 sends an Unsegmented Access message to Lower Tester 1's address emulating the LPN and expects the IUT to cache this message.
5. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
6. Lower Tester 1 expects to start receiving the cached message from Lower Tester 2 sent in Step 4. Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 4 to Lower Tester 1. Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.
7. Lower Testers 1 and 2 repeat Steps 2–6 for 3 more cycles, alternating between one and N messages, where N can be any value from 2 to 10 and alternating the FSN bit for each Friend Poll message.
8. Lower Tester 2 sends a message to the IUT with a different destination address as Lower Tester 1 and the IUT is expected to ignore and not pass this message to Lower Tester 1 acting as the LPN.

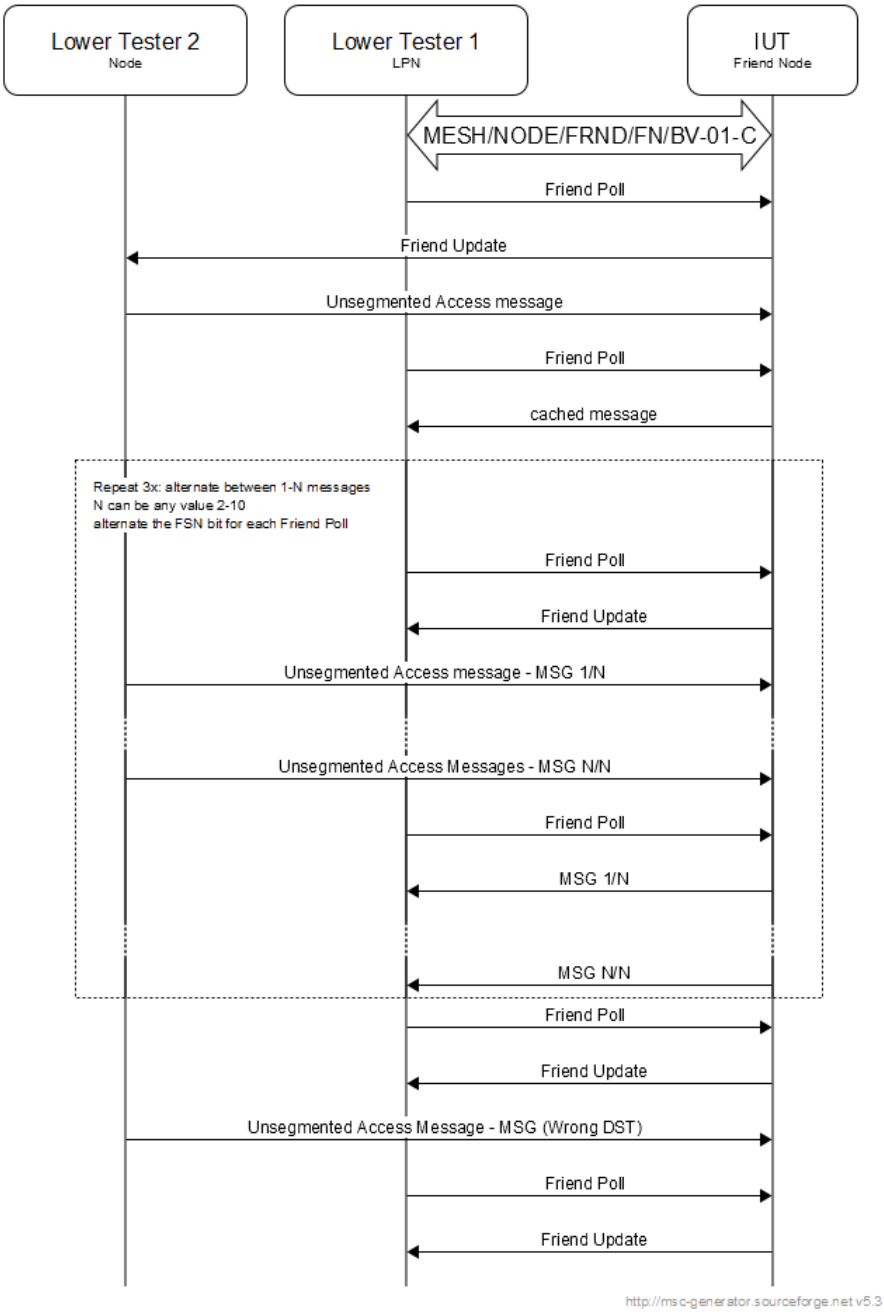


Figure 4.61: MESH/NODE/FRND/FN/BV-02-C [Friendship Messaging – Friend Node] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Friend Update message within the interval [ReceiveDelay, ReceiveDelay+ReceiveWindow] from the Friend Poll , with the MD field set to 0x00, the TransOpCode field set to 0x02, and the Key Refresh and IV Index flags set to 0.

  The IUT successfully transmits all cached messages to Lower Tester 1.

The IUT does not cache the message sent by Lower Tester 2 in Step 8 and does not relay the message to Lower Tester 1.

The IUT sends each cached message within the [ReceiveDelay, ReceiveDelay + ReceiveWindow] time interval from the Friend Poll message. When there are no more cached messages, the IUT sends a Friend Update message with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-03-C [Segment Acknowledgment – Friend Node]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can respond to segmented messages destined to an LPN, cache and deliver the Segmented Access message to the LPN.

- Reference

  [3] 3.1.2, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1, 3.6.6.5, 3.6.6.5.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends a message as in test MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] except the unicast destination address is that of Lower Tester 1.
  3. Lower Tester 2 expects the IUT to perform the SAR operation on the message on behalf of Lower Tester 1, for the entire test case.
  4. Lower Tester 2 expects the OBO bit of all Segment Acknowledgment messages to be set to 1 indicating it is acknowledging this message on behalf of an LPN.

5. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time after the friendship establishment in Step 1, the Transport Control Message Opcode field set to 0x01, RFU bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.
6. Lower Tester 1 expects to start receiving the cached fragments of a Segmented Access Message from Lower Tester 2 sent in Step 2. Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the all fragments of a Segmented Access message sent in Step 2 to Lower Tester 1 with the expected contents. Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.

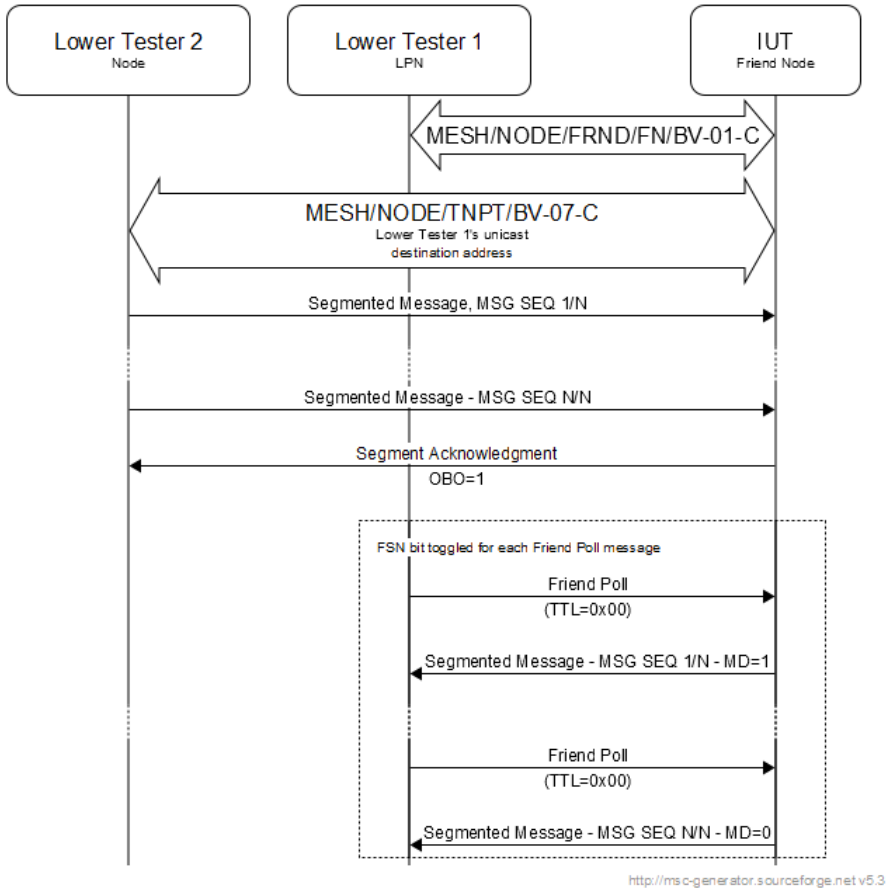


Figure 4.62: MESH/NODE/FRND/FN/BV-03-C [Segment Acknowledgment – Friend Node] MSC

- Expected Outcome

  Pass verdict

  The IUT receives the segments from Lower Tester 2 and preserves them in the order they were received.

  The IUT sends all Segment Acknowledgment messages with the OBO bit set to 1 and SRC set to the address of Lower Tester 1 to Lower Tester 2.

  The IUT successfully transmits all cached segments of the Segmented Access Message to Lower Tester 1 and then it sends a Friend Update message with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.13.1.1 Friend IV Update - Friend Node

- Test Purpose

  Verify that an IUT node acting as a Friend Node can properly handle an IV update procedure.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.8.3, 3.10.5

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - IV Index update is not already in progress.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
  - The IV Update Test Mode is activated on the IUT.
  - Additional conditions are specified in Table 4.29.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.29.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll has a TTL field set to 0x00.
  3. Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and indicating the current IV Index is still in use.
  4. The Upper Tester induces the IUT to ignore IV Update procedure time constrains.
  5. Lower Tester 2 begins to broadcast a beacon, emulating a different device on the same network, with the IV Index incremented by one and the IV Update Flag set to 1.
  6. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout and with the new IV Index minus 1. The Friend Poll message will have a TTL of zero and be addressed to the IUT.

7. Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index has been incremented.
8. Lower Tester 2 begins to broadcast a beacon, emulating a different device on the same network, with the IV Index and the IV Update Flag set to 0.
9. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout and with the new IV Index - 1. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
10. Lower Tester 1 expects a Friend Update message from the IUT.

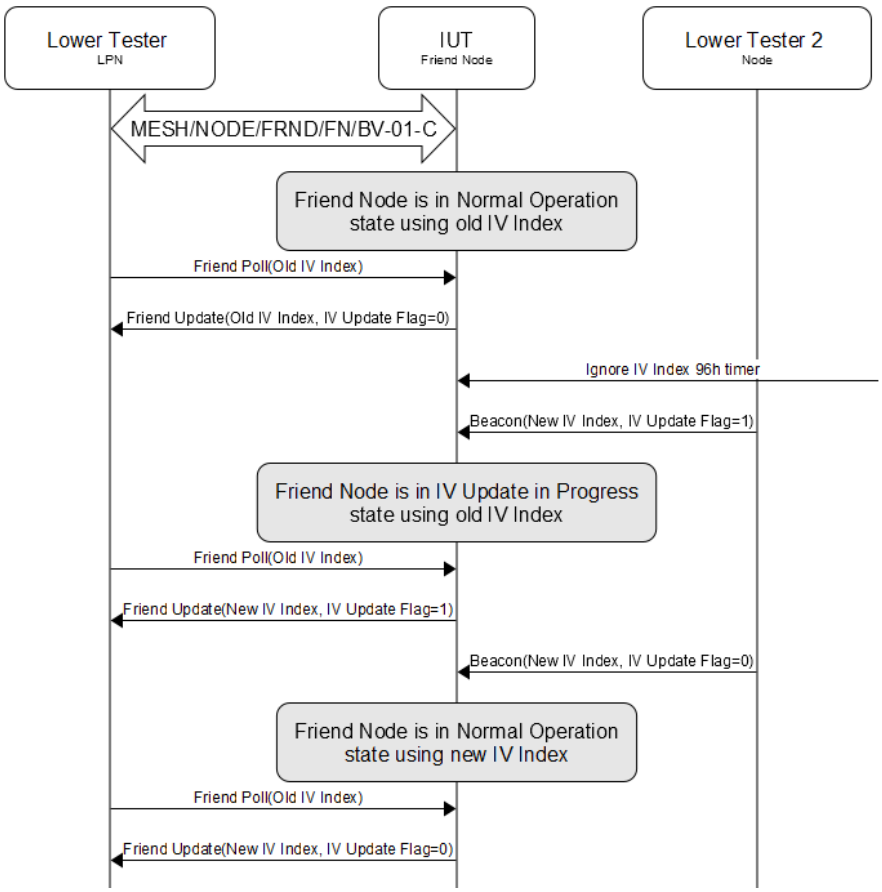


Figure 4.63: MESH/NODE/FRND/FN/BV-04-C and MESH/NODE/FRND/FN/BV-24-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/FRND/FN/BV-04-C [Friend IV Update due to Secure Network Beacon – Friend Node]	Lower Tester 2 emulates another (3rd) node sending Secure Network beacons.	0x01 (Secure Network beacon)
MESH/NODE/FRND/FN/BV-24-C [Friend IV Update due to Mesh Private Beacon – Friend Node]	Lower Tester 2 emulates another (3rd) node sending Mesh Private beacons.	0x02 (Mesh Private beacon)

Table 4.29: Friend IV Update - Friend Node test cases

- Expected Outcome

  Pass verdict

  The IUT communicates the updated IV Index to the Lower Tester with the MD bit set to 0 and, the IV Update Flag set to 1, and the Key Refresh flag set to 0 in the Friend Update message after receiving the updated IV Index via the beacon in Step 5. The Friend Update message is using old IV Index value indicated by IVI bit.

  The IUT communicates the updated IV Index to the Lower Tester with the MD bit set to 0 and, the IV Update Flag set to 0, and the Key Refresh flag set to 0 in the Friend Update message in Step 10 after receiving the updated IV Index via the beacon in Step 8. The Friend Update message is using new IV Index value indicated by IVI bit.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message]

- Test Purpose

  Verify that an IUT acting as a Friend Node can respond to and act conformingly upon reception of the Friend Subscription List Add message.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.9, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The value of the Network Transmit Count state on the IUT is greater than 0.
  - Lower Tester 1 is a node that supports the Low Power Feature.

- Lower Tester 2 is acting as another (3rd) node sending messages to the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Lower Tester and the IUT have previously established a friend relationship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. Lower Tester 2 sends an Unsegmented Access Message to one group address from the Group Address(es) and expects the IUT to ignore this message.
2. Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
3. Lower Tester 1 expects a Friend Update message from the IUT indicating it has no message for Lower Tester 1 in cache.
4. Lower Tester 1 sends a Friend Subscription List Add message to the IUT with the AddressList field value containing a valid Group Address(es) value(s) and a random TransactionNumber field value.
5. Lower Tester 1 expects to receive a Friend Subscription List Confirm message with a TransactionNumber field equal to that of the value it sent in Step 4 from the IUT.
6. Lower Tester 2 sends an Unsegmented Access Message (encrypted with the known AppKey) to one group address from the Group Address(es) and expects the IUT to cache this message.
7. Lower Tester 1 sends a Friend Poll message to Lower Tester 1 within the "PollTimeout" time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
8. Lower Tester 1 expects a message from the IUT and confirms that it retransmits the message sent in Step 6 to Lower Tester 1 with the expected contents.

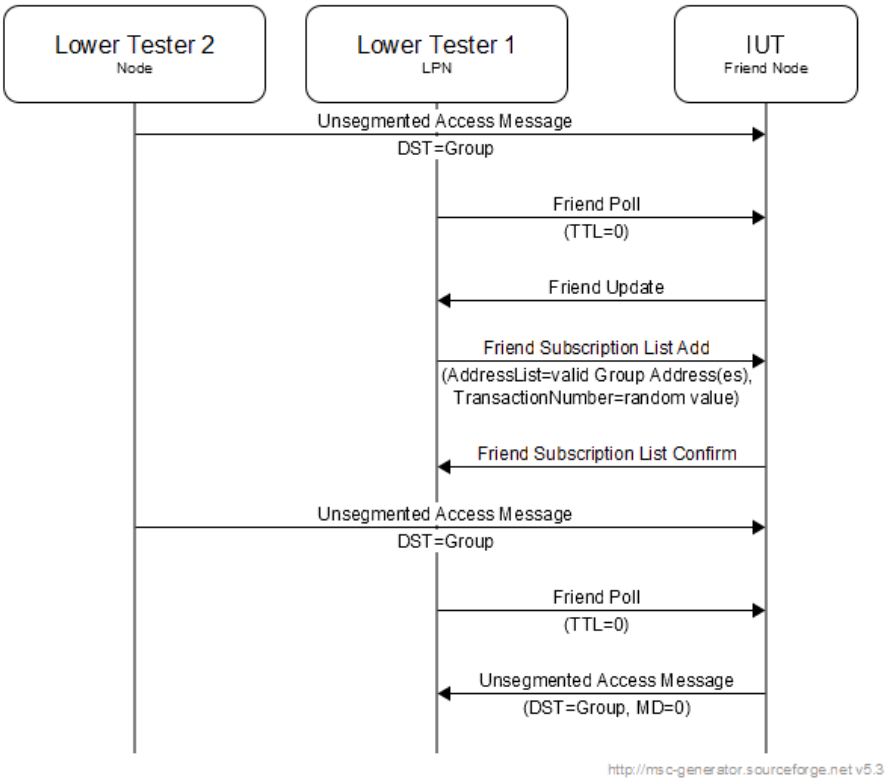


Figure 4.64: MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Subscription List Confirm message within the interval [ReceiveDelay, ReceiveDelay+ReceiveWindow] and with a TransactionNumber field equal to that sent in Step 2.

  The IUT caches messages for the Lower Tester when they are addressed to the Lower Tester's Group Address and relays those messages to the Lower Tester when requested.

  All Friend Poll, Friend Update, Friend Subscription List Add, and Friend Subscription List Confirm messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-06-C [Friendship – Respond to Friend Subscription List Remove Message]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can respond to and act conformingly upon reception of the Friend Subscription List Remove message.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.8, 3.6.5.9, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is another node on the network.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and the IUT have executed test MESH/NODE/FRND/FN/BV-05-C [Friendship – Respond to Friend Subscription List Add Message].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. Lower Tester 1 sends a Friend Subscription List Remove message to the IUT with the AddressList field containing the same values in the Friend Subscription List Add Message and the TransactionNumber field equal to 2.
  2. Lower Tester 1 expects to receive a Friend Subscription List Confirm message from the IUT with a TransactionNumber field equal to 2.
  3. Lower Tester 2 sends an Unsegmented Access Message to one group address from Group Address(es) and expects the IUT not to cache this message.
  4. Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  5. Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.

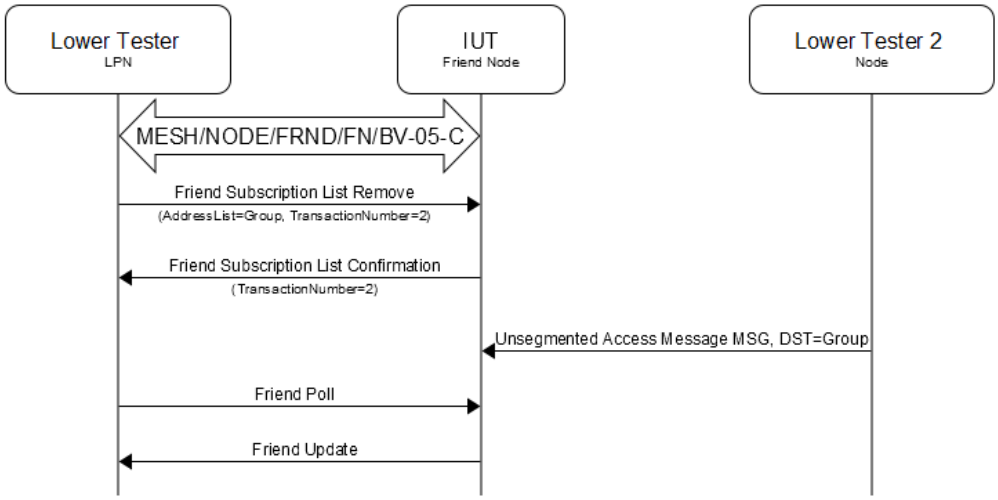


Figure 4.65: MESH/NODE/FRND/FN/BV-06-C [Friendship – Respond to Friend Subscription List Remove Message] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Subscription List Confirm message within the ReceiveWindow time window of the Lower Tester and with a TransactionNumber field equal to 2.

  The IUT does not relay the message sent to the Lower Tester's Group Address.

  All Friend Poll, Friend Update, Friend Subscription List Add, and Friend Subscription List Confirm messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-07-C [Friendship – Friend Node Friendship Not Established on Missed Friend Poll]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can clear all data associated with an LPN when it does not receive the Friend Poll from it in time.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.3

- Initial Condition

- The IUT is a node that supports the Friend Feature and the feature is enabled.
- The Lower Tester is a node that supports the Low Power Feature.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- Test Procedure

1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment procedure with a Transport Control Message Opcode field set to 0x03, the DST field set to the all-friends address, a TTL field set to 0x00, Criteria field value set to 0x21, ReceiveDelay value set to 0x0A, PollTimeout value of 0x000064, a LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.
2. The Lower Tester expects to receive a Friend Offer message from the IUT.
3. The Lower Tester waits for 2 seconds after the offer window and then sends a Friend Poll Message.
4. The Lower Tester expects to receive no response from the Friend Poll message as the friendship procedure was not established.

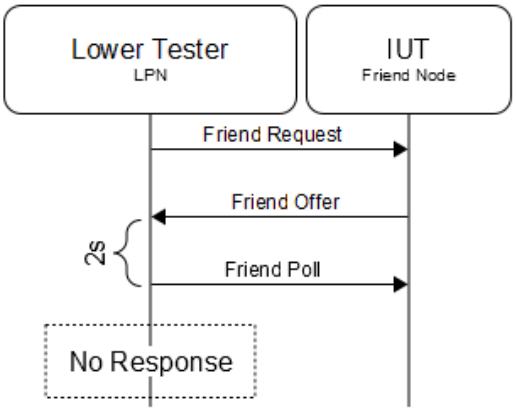


Figure 4.66: MESH/NODE/FRND/FN/BV-07-C [Friendship – Friend Node Friendship Not Established on Missed Friend Poll] MSC

- Expected Outcome

Pass verdict

The IUT sends Friend Offer message to the Lower Tester's unicast address, with a TTL value of 0, a Transport Control Message Opcode value of 0x04, and a ReceiveWindow field value between 0x01 - 0xFF.

The IUT does not respond to the Friend Poll sent in Step 3.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-08-C [Friendship – Friend Node Circular Queue]

- Test Purpose

  Verify that an IUT node acting as a Friend Node implements its cache as a circular queue and messages are only cleared from the queue once delivery is confirmed and older messages are discarded when the queue is full.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.2, 3.6.6.3.3

- Initial Condition

  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 uses the cache size information of the IUT retained in Step 1 and sends messages-N to Lower Tester 1 where N is equal or smaller than the size of the cache of the IUT, and expects the IUT to cache these messages. All messages are sent within the PollTimeout time.
  3. The Upper Tester expects to receive positive acknowledgment of each message sent by Lower Tester 2 in Step 2. If there are messages that are not acknowledged Lower Tester 2 will re-send until the number of cached message is known.
  4. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  5. Lower Tester 1 expects a message from the IUT and confirms that it retransmits the message sent in Step 2 to Lower Tester 1 with the expected contents and in the order of reception.
  6. Lower Tester 1 will continue sending the Friend Poll message until it receives a Friend Update message with the MD field set to 0x00.
  7. Lower Tester 2 uses the cache size information of the IUT retained in Step 1 and sends messages-n to Lower Tester 1 until n is greater than the size of the cache of the IUT and expects the IUT to cache these messages. All messages are sent within the PollTimeout time.
  8. Lower Tester 1 sends a Friend Poll Message to the IUT within the PollTimeout time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  9. Lower Tester 1 expects a message from the IUT and confirms that it does not retransmit Message-1 sent in Step 6 as this message should have been discarded due to cache size. Lower Tester 1 confirms that messages are retransmitted by IUT with expected contents and in the order of reception.

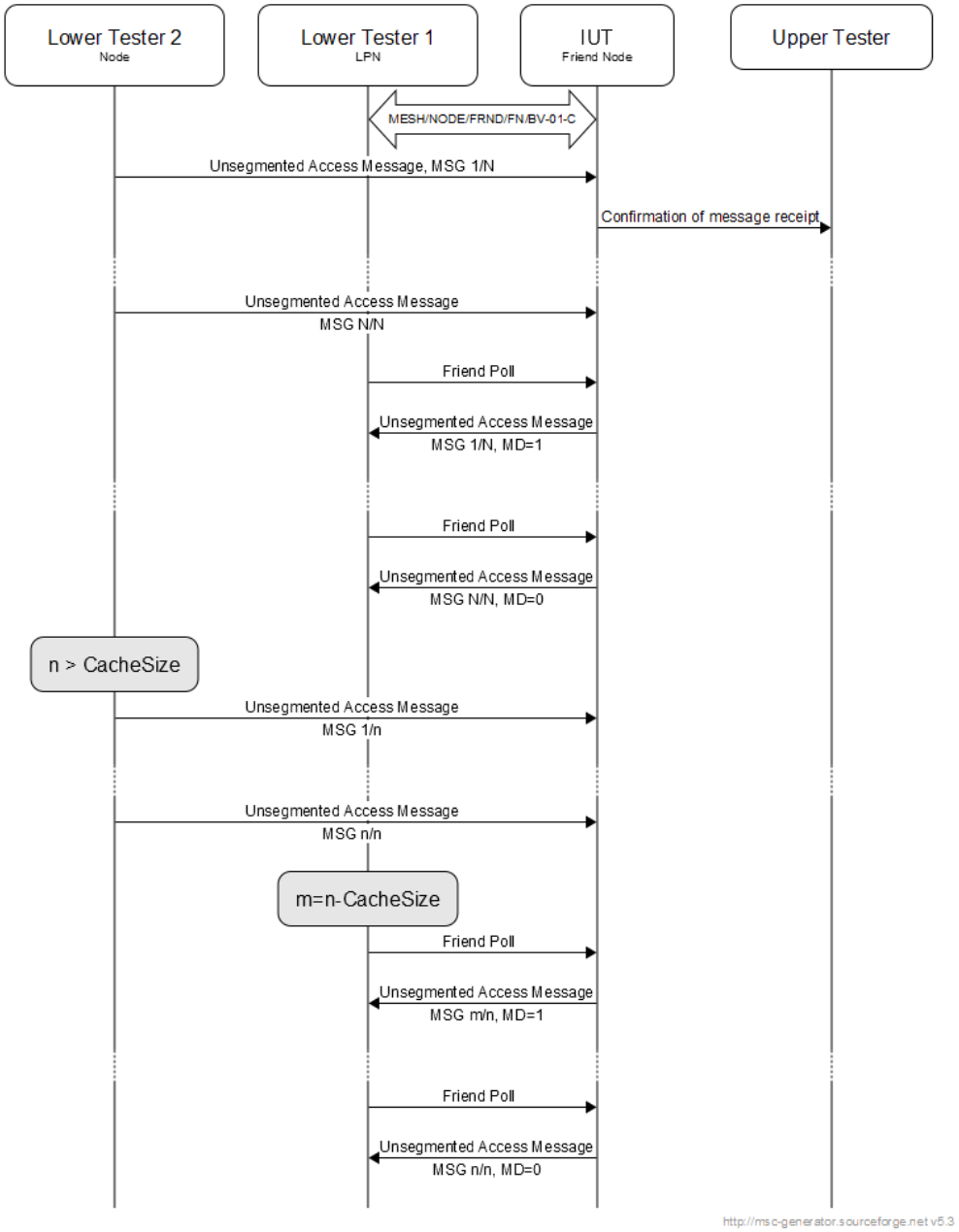


Figure 4.67: MESH/NODE/FRND/FN/BV-08-C [Friendship – Friend Node Circular Queue] MSC

- Expected Outcome

  Pass verdict

  The IUT successfully transmits all cached messages to Lower Tester 1.

  The IUT sends a Friend Update message with the MD field set to 0x00 when there are no more cached messages.

  All Cached Messages are received at the Lower Tester in the proper order.

  All oldest messages sent that do not fit in the IUT's cache are discarded in Step 7 and not sent.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-09-C [Friendship – Friend Node Cached Cleared on Poll Timeout]

- Test Purpose

  Verify that an IUT node acting as a Friend Node discards messages in its cache when friendship is lost because of a PollTimeout.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is a node supporting Configuration Client Model.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The IUT and Lower Tester 1 have run through the Friendship Establishment Procedure in test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 2 sends a message to the IUT, emulating a different device attempting to send a message to the Lower Tester's address emulating the LPN and expects the IUT to cache this message.
  2. Lower Tester 1 does not send the Friend Poll message to the IUT for twice the PollTimeout time.
  3. The IUT should remove Lower Tester 1 as a Friend.
  4. The Lower Tester sends Friend Poll messages and expects no response from IUT indicating that the Friendship between Lower Tester 1 and the IUT has been removed.
  5. Lower Tester 1 runs through the Friendship Establishment Procedure again in test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].
  6. Lower Tester 1 sends a Friend Poll message to the IUT within the ReceiveDelay time. The Friend Poll Message must have a TTL of zero and be addressed to the IUT.
  7. Lower Tester 1 expects a Friend Update message from the IUT with the MD field set to 0x00.

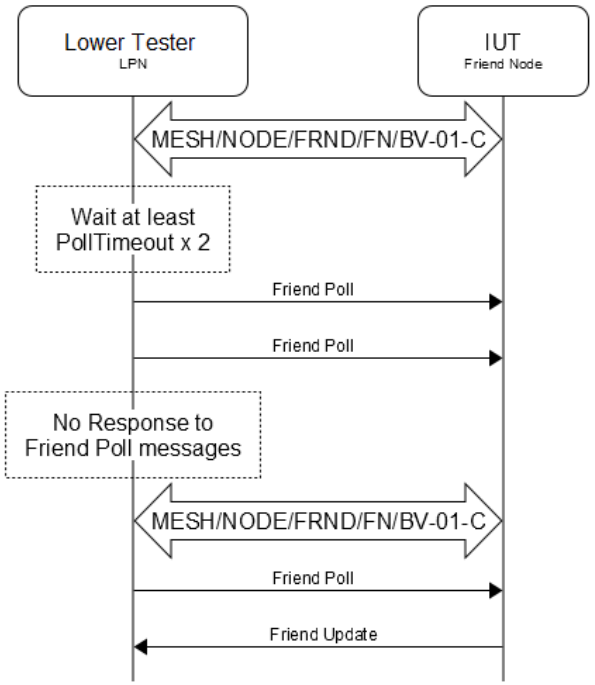


Figure 4.68: MESH/NODE/FRND/FN/BV-09-C [Friendship – Friend Node Cached Cleared on Poll Timeout] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT does not respond to Friend Poll messages indicating that Friendship between the Lower Tester and the IUT was removed.

  In Step 7, the IUT sends a Friend Update message to the Lower Tester indicating there are no cached messages for the Lower Tester and new Friendship was established with empty cache.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-10-C [Friendship – Friend Node Config Friend Set Received]

- Test Purpose

  Verify that an IUT node acting as a Friend Node clears all associated Friendship Relationships and all data associated with them when it receives the Config Friend Set message with "State" field set appropriately.

- Reference

  [3] 3.6.6, 4.3.2.56, 4.4.1.2.5

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is a Configuration Client.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends a message to the IUT, emulating a different device attempting to send a message to the Lower Tester’s address emulating the LPN and expects the IUT to cache this message.
  3. Lower Tester 2 sends the Config Friend Set message with the "State" parameter value equal to 0 to the IUT, the message is encrypted with the Device Key of the IUT.
  4. Lower Tester 1 sends three Friend Poll messages to the IUT within the PollTimeout time. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  5. Lower Tester 1 expects no response from IUT indicating that the Friendship between Lower Tester 1 and the IUT has been removed.

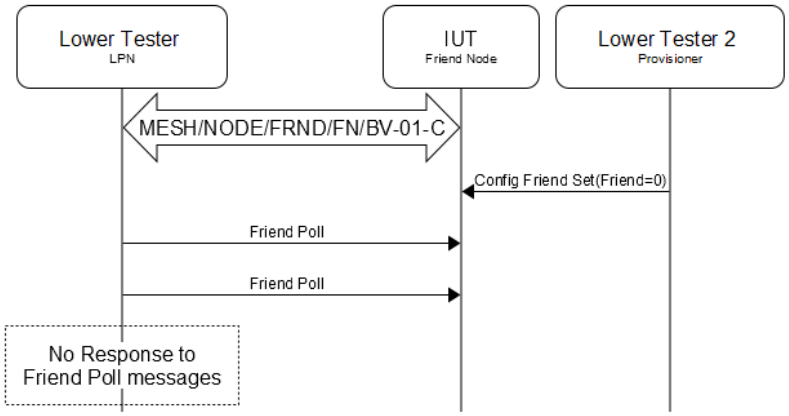


Figure 4.69: MESH/NODE/FRND/FN/BV-10-C [Friendship – Friend Node Config Friend Set Received] MSC

- Expected Outcome

  Pass verdict

  The IUT does not respond to Friend Poll messages sent in Step 3 indicating that Friendship between Lower Tester 1 and the IUT was removed.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-11-C [Friendship – Friend Node Friend Clear Send and Retry]

- Test Purpose

  Verify that an IUT node acting as a Friend Node sends the Friend Clear message to the previous Friend Node after the Friend Establishment Procedure to inform it of the new establishment and will retry sending if Friend Clear Confirmation is not received.

- Reference

  [3] 3.6.5.5, 3.6.5.6, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.

- Lower Tester 2 is a Friend Node and supports the Friend Feature.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- Test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] has been executed.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

• Test Procedure

1. Lower Tester 1 completes the test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] except the PreviousAddress value in Step 1 is set to Lower Tester 2's address.
2. Lower Tester 2 expects the Friend Clear message from the IUT.
3. Lower Tester 2 waits for greater than the 1 second.
4. Lower Tester 2 expects the IUT to send the Friend Clear message again as in Step 2.
5. Lower Tester 2 then sends the Friend Clear Confirm message to the IUT with the IUT's address as the LPNAddress field.
6. The Upper Tester expects to receive positive acknowledgment from the IUT that it has received the Friend Clear Confirm message. If this is not received, then Lower Tester 2 retries sending the Friend Clear Confirm message.

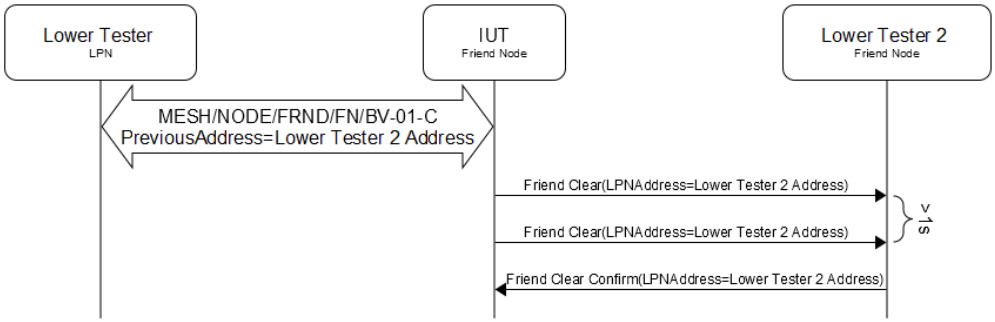


Figure 4.70: MESH/NODE/FRND/FN/BV-11-C [Friendship – Friend Node Friend Clear Send and Retry] MSC

• Expected Outcome

Pass verdict

The IUT sends the Friend Clear message to Lower Tester 2 with the LPNAddress value of Lower Tester 1 address and a TTL value set to the maximum valid value in Step 2.

The IUT sends a Friend Clear message again in Step 4.

After Step 6 is complete the IUT does not send any more Friend Clear Messages.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Friend Clear and Friend Clear Confirm messages are secured using managed flooding security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-12-C [Friend Clear and Friend Request Friendship Termination Scenarios]

- Test Purpose

  Verify that an IUT node acting as a Friend node terminates the friendship on receiving a Friend Clear message, or a Friend Request message from the Low Power node.

- Reference

  [3] 3.6.5.1, 3.6.5.3, 3.6.5.5, 3.6.5.6, 3.6.6, 3.6.6.3.3, 3.6.6.3.1, 3.6.5.5, 3.6.6.4.2

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is a Friend Node and supports the Friend Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 completes the test case MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node] with the IUT.
  2. Lower Tester 2 sends a Friend Clear message to the IUT with the LPNAddress field set to the unicast address of Lower Tester 1.
  3. Lower Tester 2 expects a Friend Clear Confirm message in response from the IUT.
  4. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1 expects no response from the IUT, and repeats the Friend Poll message two more times.
  5. Lower Tester 1 repeats Step 1.
  6. Lower Tester 1 sends a Friend Request message using the same unicast address from the friendship establishment. The Lower Tester ignores any Friend Offer message from the IUT, if any.
  7. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1 expects no response from the IUT, and repeats the Friend Poll message two more times.
  8. Lower Tester 1 repeats Step 1.
  9. Lower Tester 1 sends a Friend Clear message to the IUT, with the LPNAddress field set to its own unicast address.
  10. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time. Lower Tester 1

expects no response from the IUT, and repeats the Friend Poll message two more times.

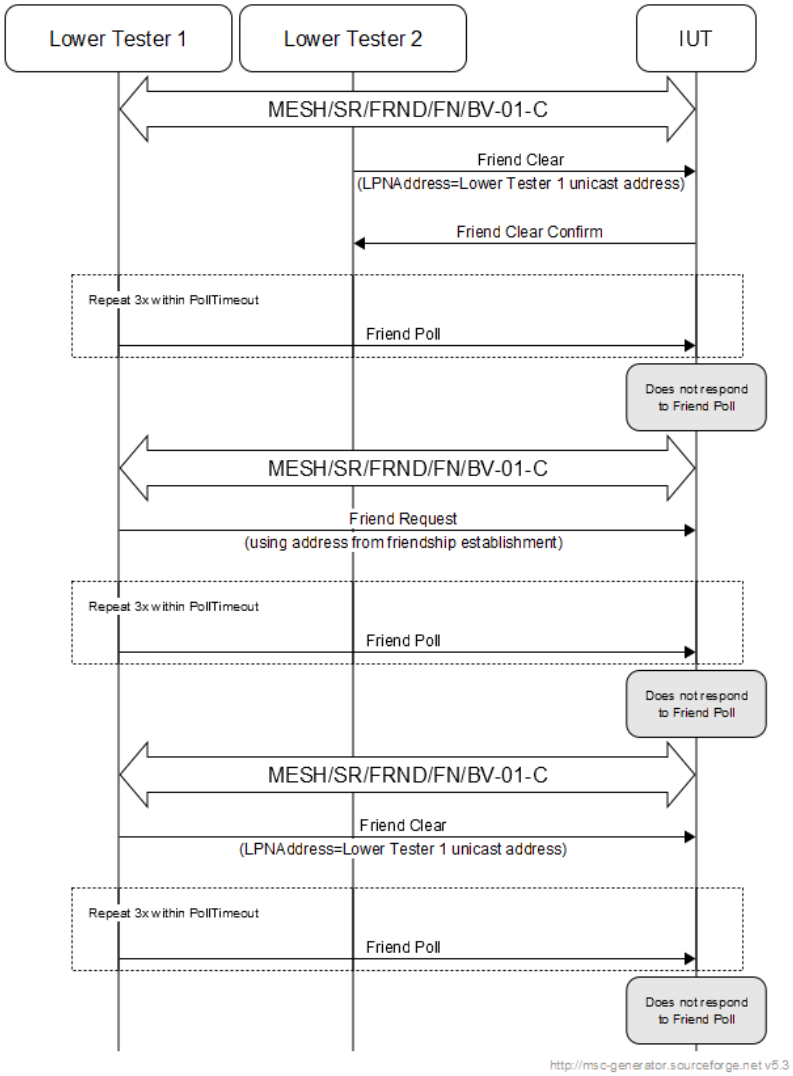


Figure 4.71: MESH/NODE/FRND/FN/BV-12-C [Friend Clear and Friend Request Friendship Termination Scenarios] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT responds with a Friend Clear Confirm message with valid parameters.

  In Steps 4, 7, and 10, the IUT does not respond to the Friend Poll message, confirming it has terminated the friendship.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Friend Clear and Friend Clear Confirm messages are secured using the managed flooding security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the

unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-13-C [Friendship – Friend Node – Friend Offer Delay]

- Test Purpose

  Verify that an IUT node acting as a Friend Node calculates the correct Friend Offer Delay.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature. The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode field set to 0x03, the DST field set to the all-friends address, a TTL field set to 0x00, Criteria field set to a value between 0x39 and 0x3F, ReceiveDelay field set to 0x0A, PollTimeout field set to 0x000064, a LPNCounter set to 0, NumElements field set to 0x01, and a PreviousAddress field set to 0x0000. The Lower Tester then waits 100 ms.
  2. The Lower Tester starts a timer after sending the Friend Request.
  3. The Lower Tester then expects to receive a Friend Offer message. The Friend Offer message is expected to be received after the Friend Offer Delay which is calculated by using the RSSIFactor and ReceiveWindowFactor values.
  4. When the Lower Tester receives the Friend Offer message from the IUT it checks that the value of the timer started in Step 2 is equal to or greater than the calculated Friend Offer Delay.

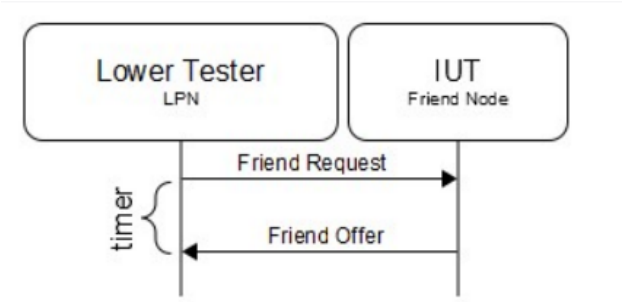


Figure 4.72: MESH/NODE/FRND/FN/BV-13-C [Friendship – Friend Node – Friend Offer Delay] MSC

- Expected Outcome

  Pass verdict

  The IUT sends the Friend Offer messages after the appropriate Friend Offer Delay. For the Lower Tester to calculate the Friend Offer Delay it will use the following formula.

  Local Delay is computed using the formula found in [3] Section 3.6.6.3.1. If the IUT sent the Friend Offer after this appropriate Friend Offer Delay, then test passes.

  Each Friend Request and Friend Offer messages are secured using the managed flooding security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.13.1.2 Friend Key Refresh via Mesh Beacons - Friend Node

- Test Purpose

  Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure using Secure Network beacons or Mesh Private beacons.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- Lower Tester 2 and the IUT share device key of the IUT.
- Key Refresh procedure is not already in progress.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Additional conditions are specified in Table 4.30.
- The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.30.

- Test Procedure

1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
3. Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and using the current IV Index, with the IV Update and Key Refresh flags set to 0 and the MD field set to 0x00.
4. Lower Tester 2 sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.
5. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
6. Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester.
7. Lower Tester 2 sends a beacon to the IUT secured with the new network key and KR flag set to 1.
8. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
9. Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.
10. Lower Tester 2 sends a beacon to the IUT with the secured with new NetKey and KR flag set to 0.
11. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll Message will have a TTL of zero, be addressed to the IUT and secured with new NetKey.
12. Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index, the MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The Friend Update messages is secured using new NetKey.

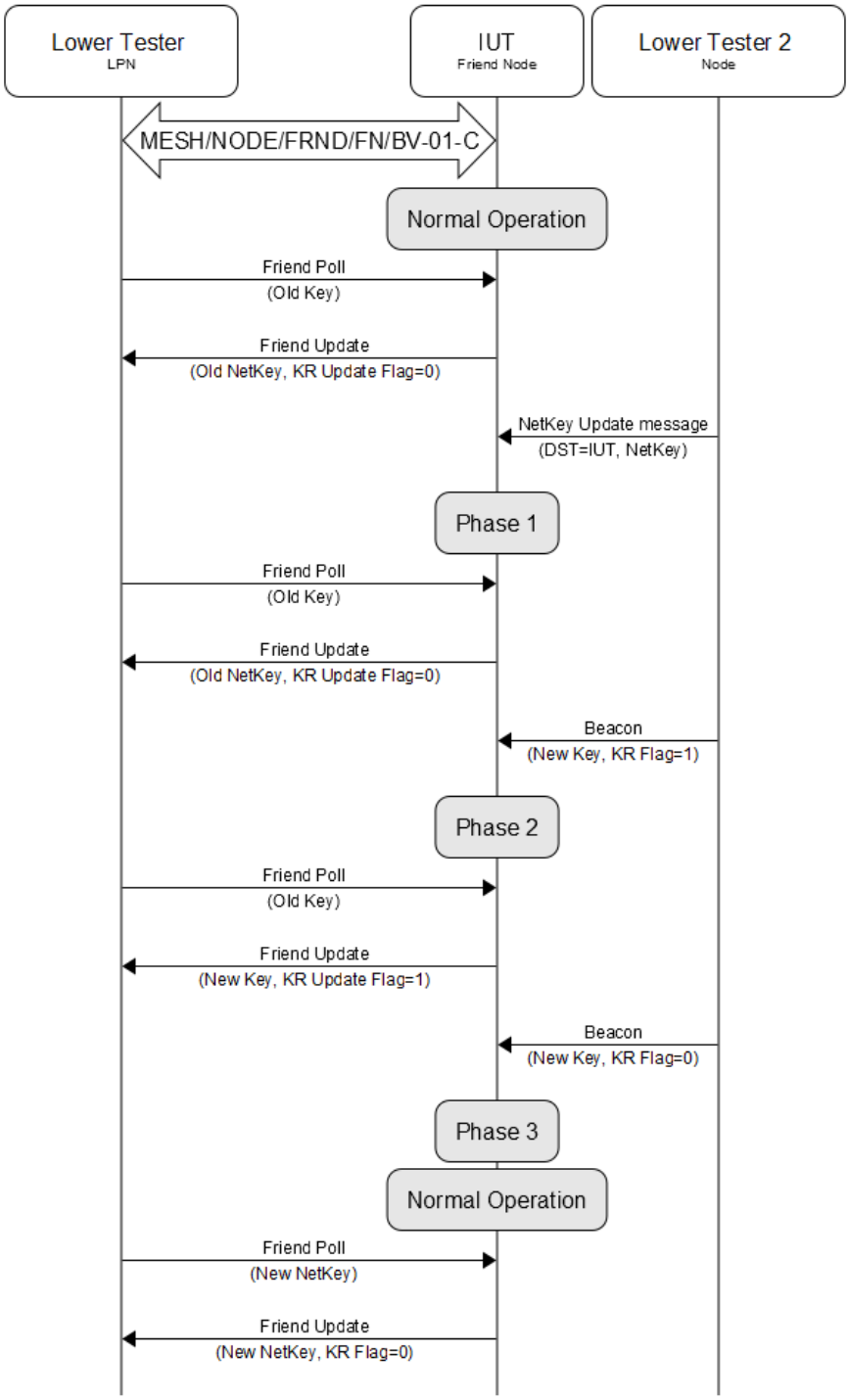


Figure 4.73: MESH/NODE/FRND/FN/BV-14-C and MESH/NODE/FRND/FN/BV-25-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/FRND/FN/BV-14-C [Friend Key Refresh via Secure Network Beacons – Friend Node]	Lower Tester 2 emulates another (3rd) node sending Secure Network beacons.	0x01 (Secure Network beacon)
MESH/NODE/FRND/FN/BV-25-C [Friend Key Refresh via Mesh Private Beacons – Friend Node]	The IUT—and the Lower Tester—can broadcast Mesh Private beacons (Private Beacon state = 0x01). Lower Tester 2 emulates another (3rd) node sending Mesh Private beacons.	0x02 (Mesh Private beacon)

Table 4.30: Friend Key Refresh via Mesh Beacons - Friend Node test cases

- Expected Outcome

  Pass verdict

  The IUT communicates the updated key to the Lower Tester with the MD field set to 0x00, the IV Update Flag set to 0, and the Key Refresh flag set to 1 in the Friend Update Message in Step 9 after receiving the beacon with KR Bit Set to 1 in Step 7. The IUT sends a Friend update message in Step 12 with the IV Index, MD field being set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0 secured with the new network key.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-15-C [Friend Key Refresh via Configuration Client Messages – Friend Node]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure using Configuration Client messages.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is a Configuration Client.

- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- Lower Tester 2 and the IUT share device key of the IUT.
- Key Refresh procedure is not already in progress.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
3. Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester and indicating the current IV Index, the IV Update and Key Refresh flags set to 0 and the MD field set to 0x00.
4. Lower Tester 2 sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.
5. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
6. Lower Tester 1 expects a Friend Update from the IUT indicating it has zero messages for the Lower Tester.
7. Lower Tester 2 sends a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x02 and expects a Config Key Refresh Phase Status message from the IUT in response with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x02.
8. Lower Tester 1 sends a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
9. Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.
10. Lower Tester 2 sends a Config Key Refresh Phase Set message to the IUT with the NetKeyIndex set to the global NetKey Index of the new NetKey and the Transition field set to 0x03 and expects a Config Key Refresh Phase Status message from the IUT in response with the Status set to 0x00, the NetKeyIndex set to the global NetKey Index of the NetKey, and the Phase set to 0x00.
11. Lower Tester 1 sends a Friend Poll Message to the IUT within the negotiated PollTimeout. The Friend Poll Message will have a TTL of zero, be addressed to the IUT and secured with new NetKey.
12. Lower Tester 1 expects a Friend Update message from the IUT indicating the IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The Friend Update messages is secured using new NetKey.

- Expected Outcome

Pass verdict

The IUT communicates the updated key to the Lower Tester with the MD field set to 0x00 and, the IV Update Flag set to 0, and the Key Refresh flag set to 1 in the Friend Update Message in Step 9 after receiving the Secure Network Beacon with KR Bit Set to 1 in Step 7.

The IUT sends a Friend update message in Step 12 with the IV Index, the MD field set to 0x00, the IV Update Flag equal to 0, and Key Refresh flag set to 0 secured with the new network key.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-16-C [Friendship – Friend Node - FSN Field Not Changed]

- Test Purpose

  Verify that an IUT node acting as a Friend Node handles a Friend Poll message properly if the FSN field is not toggled.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, RFU bits set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  3. Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.
  4. Lower Tester 2 sends an Unsegmented Access Message to Lower Tester 1 and expects the IUT to cache this message.

5. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
6. Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in Step 4. Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 4 by Lower Tester 2 with the expected contents.
7. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit not toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
8. Lower Tester 1 expects to receive the same message from the IUT as it did in Step 6.
9. Lower Tester 1 repeats Steps 7 and 8 two times to ensure the same message is always delivered from the IUT.
10. Lower Tester 2 then sends an Unsegmented Access Message to Lower Tester 1 and expects the IUT to cache this message.
11. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x00, RFU set to 0, and FSN bit toggled. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
12. Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in Step 4. Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 4 by Lower Tester 2 with the expected contents.
13. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
14. Lower Tester 1 expects to receive the cached message from Lower Tester 2 sent in Step 10. Lower Tester 1 listens for the message from the IUT and confirms that it retransmits the message sent in Step 10 by Lower Tester 2 with the expected contents.
15. Lower Tester 1 sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01, RFU set to 0, and FSN bit toggled. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
16. Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.

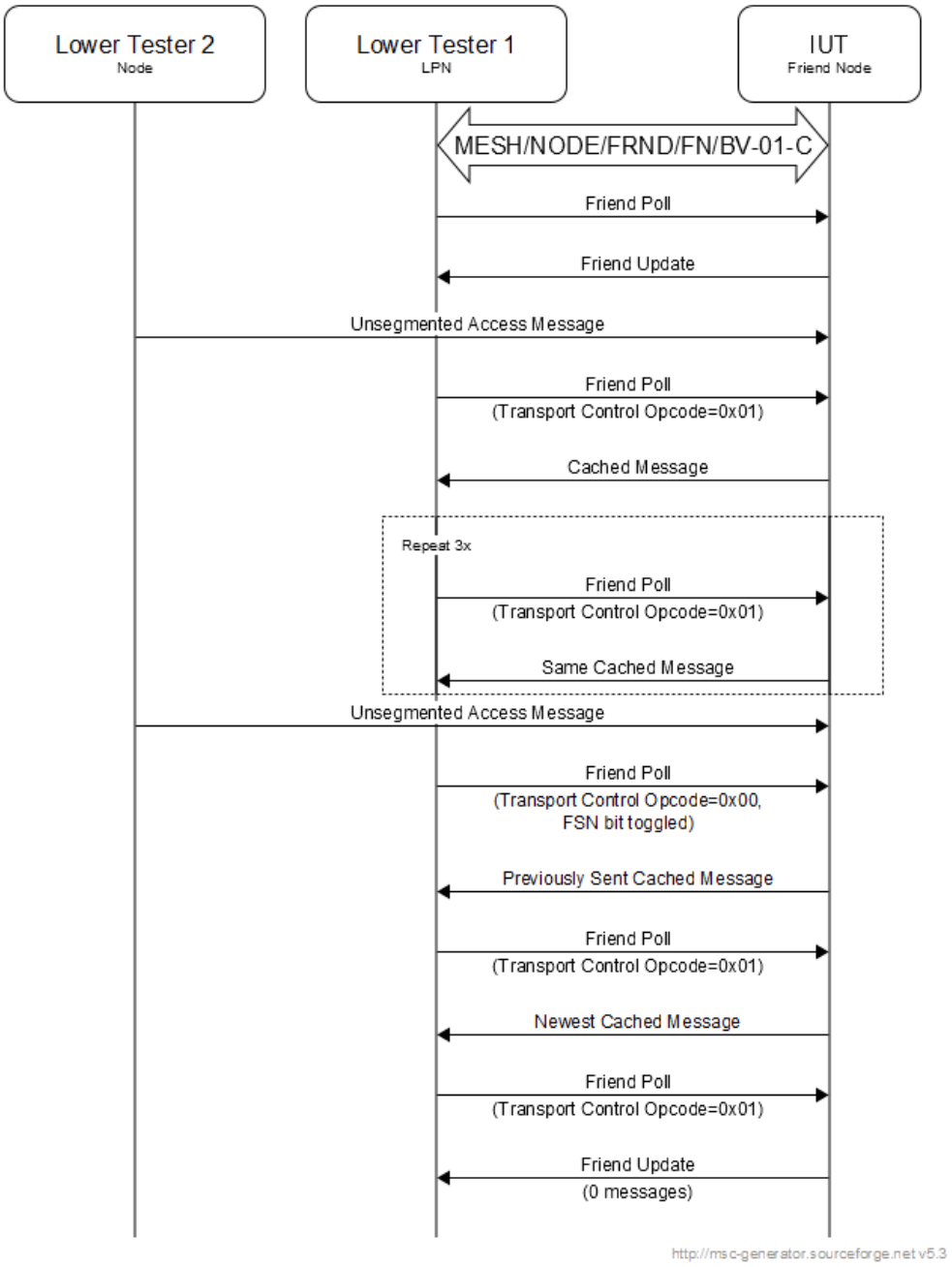


Figure 4.74: MESH/NODE/FRND/FN/BV-16-C [Friendship – Friend Node - FSN Field Not Changed] MSC

- Expected Outcome

  Pass verdict

  The IUT sends the same message in Step 8 as it did in Step 6. The FSN bit must be 1 every time this same message is sent.
  The IUT sends the same message two more times in Step 9.

  The IUT sends the cached message from Step 4 in Step 12.

The IUT sends the cached message from Step 10 in Step 14. The IUT sends the Friend Update Message to Lower Tester 1 indicating it has zero messages for it in Step 16.

The IUT sends each response within the [ReceiveDelay, ReceiveDelay + ReceiveWindow] time interval from the Friend Poll Message. The IUT sends a Friend Update message with the MD field set to 0x00 when there are no more cached messages.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-17-C [Friendship – Friend Node – SAR Interruption]

- Test Purpose

  Verify that an IUT node acting as a Friend Node handles a SAR transaction properly when the segment acknowledgment is duplicated in the Friend Node cache.

- Reference

  [3] 3.5.3, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends a Segment Acknowledgment message with Lower Tester 1's address and BlockAck field set to 0x00000001. The IUT is expected to cache this message.
  3. Lower Tester 1 does not send a Friend Poll yet and Lower Tester 2 sends another Segment Acknowledgment message to Lower Tester 1's address with the same SeqZero value and BlockAck field set to 0x00000003. The IUT is expected to cache this message and discard the older Segment Acknowledgment message.

4. Lower Tester 1 then sends the Friend Poll.
5. The IUT sends the cached message to Lower Tester 1.
6. Lower Tester 2 sends a message as in test MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] except the unicast destination address is that of Lower Tester 1 and in Step 1 Lower Tester 2 does not complete the entire SAR transaction by not sending all segments of the message.
7. Lower Tester 1 then sends a Friend Poll message to the IUT within the PollTimeout time, the Transport Control Message Opcode field is set to 0x01, the RFU bits set to 0, TTL field set to 0x00, and FSN bit toggled for each new Friend Poll message if the previous Friend Poll message has triggered a response.
8. Lower Tester 1 expects a Friend Update message from the IUT indicating it has zero messages for the Lower Tester.

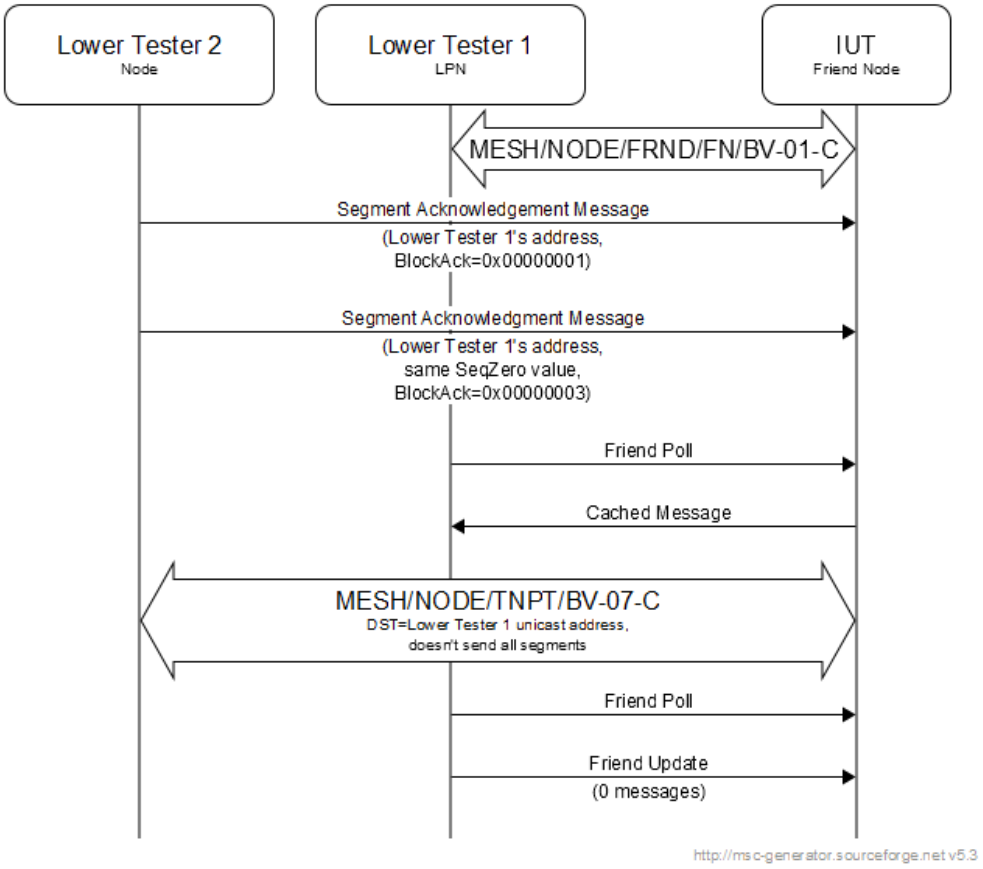


Figure 4.75: MESH/NODE/FRND/FN/BV-17-C [Friendship – Friend Node – SAR Interruption] MSC

- Expected Outcome

  Pass verdict

  The IUT sends the second Segment Acknowledgment Message to Lower Tester 1 during Step 6 and does not receive two Segment Acknowledgment messages from the IUT.

  The IUT sends the Friend Update message in Step 8 with the MD field set to 0x00.

All Friend Poll and Friend Update messages are secured using the friendship security credentials.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-18-C [Message Caching based on TTL and SRC – Friend Node]

- Test Purpose

  Verify that an IUT acting as a Friend Node only caches messages with a TTL greater than or equal to 2 and with an SRC different from the Low Power node's unicast address.

- Reference

  [3] 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1
- Test Procedure
  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends a message to the IUT with the DST field set to the unicast address of Lower Tester 1 and with the TTL field set to a random valid value, greater than or equal to 2, expecting the IUT to cache the message.
  3. Lower Tester 1 sends a Friend Poll message to the IUT.
  4. Lower Tester 1 expects the IUT to respond with the message cached in Step 2.
  5. Lower Tester 2 sends another message to the IUT with the DST field set to the unicast address of Lower Tester 1 and with the TTL field set to 1.
  6. Lower Tester 1 sends a Friend Poll message to the IUT.
  7. Lower Tester 1 expects the IUT to respond with a Friend Update message indicating it has not cached the message sent in Step 5.
  8. Repeat Steps 5–7 using a TTL field value of 0 in Step 5.
  9. Add a random group address to the Friend Subscription List.
  10. Repeat Steps 5–7 using a TTL field value greater than 1 but setting the SRC field to the unicast address of Lower Tester 1 and the DST field to the group addresss added in Step 9.

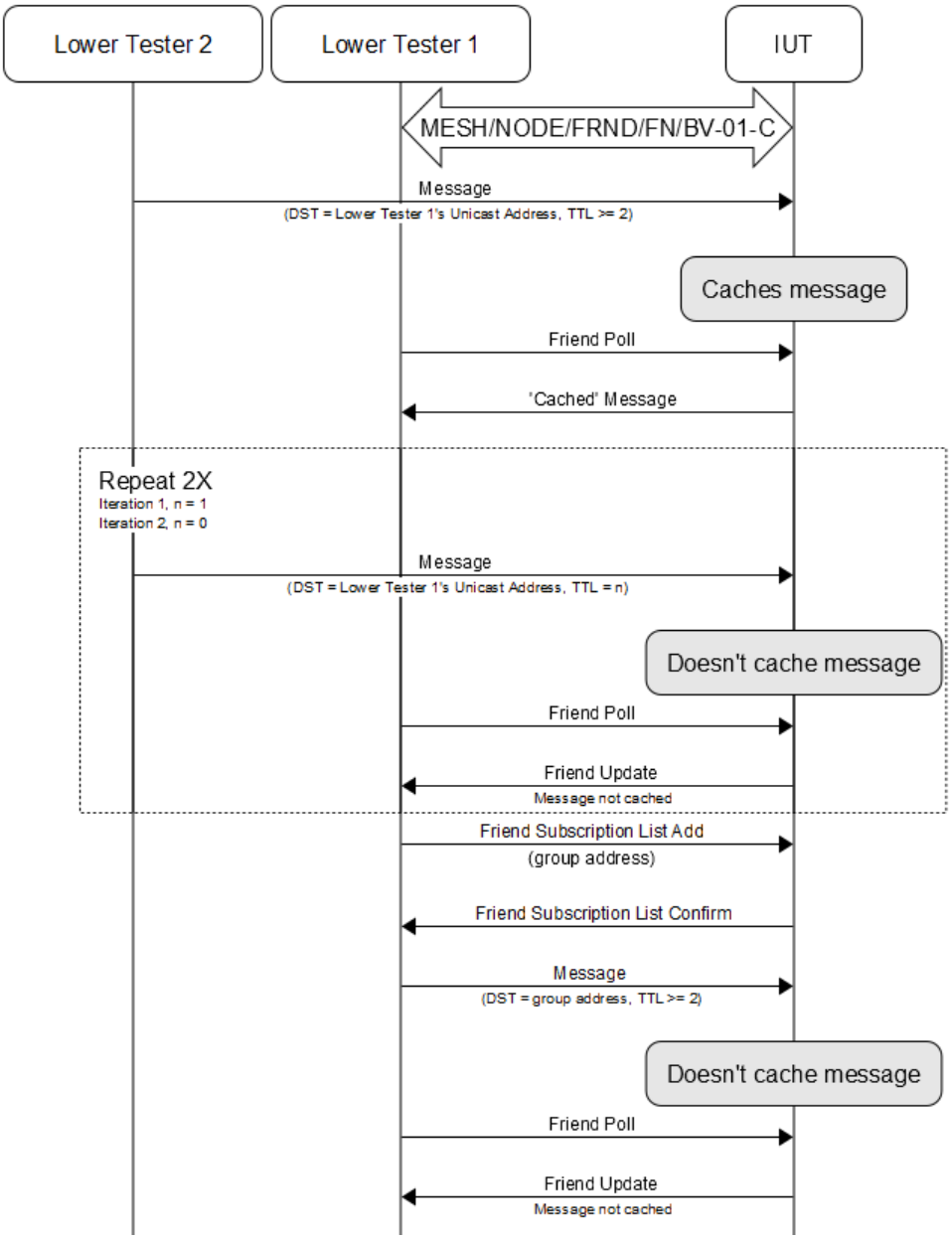


Figure 4.76: MESH/NODE/FRND/FN/BV-18-C [Message Caching based on TTL and SRC – Friend Node] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends the message received in Step 2, with the MD bit set to 0 and the TTL field value decremented by 1. All other fields are unchanged.

  In each iteration through Step 6, the IUT sends a Friend Update message, with the MD field set to 0x00.

  The Friend Poll and Friend Update messages and the retransmitted message are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-19-C [Segment Caching – Friend Node]

- Test Purpose

  Verify that an IUT acting as a Friend Node only places message segments in the Friend Cache when the complete segmented message has been received.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1, 3.6.6.5, 3.6.6.5.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 executes only Steps 1–2 of the procedure described in MESH/NODE/TNPT/BV-07-C [Receive a Segmented Message – Unicast] setting the destination address to the unicast address of Lower Tester 1.
  3. Lower Tester 1 sends a Friend Poll message to the IUT.
  4. Lower Tester 1 expects the IUT to respond with a Friend Update message, indicating that it has not cached any of the segments received in Step 2.
  5. Lower Tester 2 executes Step 3 of the procedure started in Step 2.
  6. Lower Tester 1 sends a Friend Poll message to the IUT.
  7. Lower Tester 1 expects the IUT to respond with a cached message, containing a segment of the segmented message sent by Lower Tester 2.
  8. Lower Tester 1 repeats Steps 6–7 until the IUT responds with a Friend Update message and reassembles the complete message.

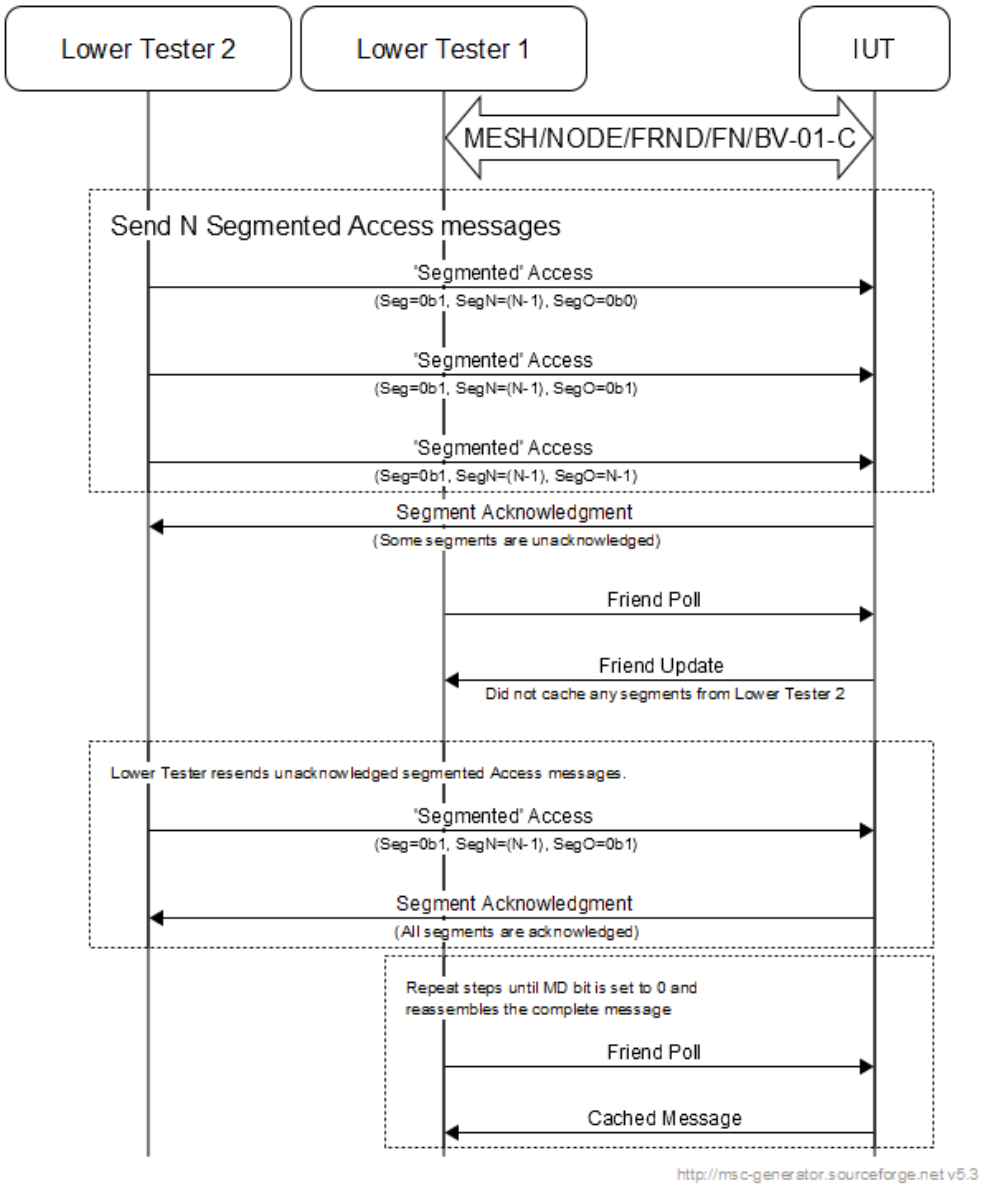


Figure 4.77: MESH/NODE/FRND/FN/BV-19-C [Segment Caching – Friend Node] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends a Friend Update message.

  In each but the last iteration through Step 7, the IUT sends a message containing a segment of the same segmented message. In the last iteration through Step 7, the IUT sends a Friend Update message with the MD field set to 0x00.

  The complete message reassembled by Lower Tester 1 is the same as the message sent by Lower Tester 2.

  The Friend Poll and Friend Update messages and the retransmitted messages are secured using the

friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

4.13.1.3 Adding another Friend Update - Friend Node

- Test Purpose

  Verify that an IUT acting as a Friend node adds two Friend Update messages in the Friend Queue.

- Reference

  [3] 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 2 supports the Configuration Client Model and has access to the device key of the IUT.
  - Lower Tester 1 is a node that supports the Low Power feature.
  - The IUT and the Lower Testers share previously established network security credentials. The IUT has been in Normal Operation state of the IV Update procedure for at least 96 hours. There is no Key Refresh procedure in progress on the IUT.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
  - Additional conditions are specified in Table 4.31.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.31.

- Test Procedure
  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 updates the network key on the IUT by sending a Config NetKey Update message, containing a new value for the network key. The IUT responds with a Config NetKey Status message, confirming the reception of the new key value.
  3. Lower Tester 2 sends a beacon to the IUT, containing the new IV Index value, setting the IV Update Flag to 1 and setting the Key Refresh flag to 0.
  4. Lower Tester 2 sends another beacon to the IUT, containing the new IV Index value, setting the IV Update Flag to 1 and setting the Key Refresh flag to 1.
  5. Lower Tester 1 sends a Friend Poll message to the IUT.
  6. Lower Tester 1 expects the IUT to respond with a Friend Update message containing the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 0 and the MD field set to 0x01.

7. Lower Tester 1 sends another Friend Poll message to the IUT, with a toggled FSN field value.
8. Lower Tester 1 expects the IUT to respond with a Friend Update message containing the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 1 and the MD field set to 0x00.

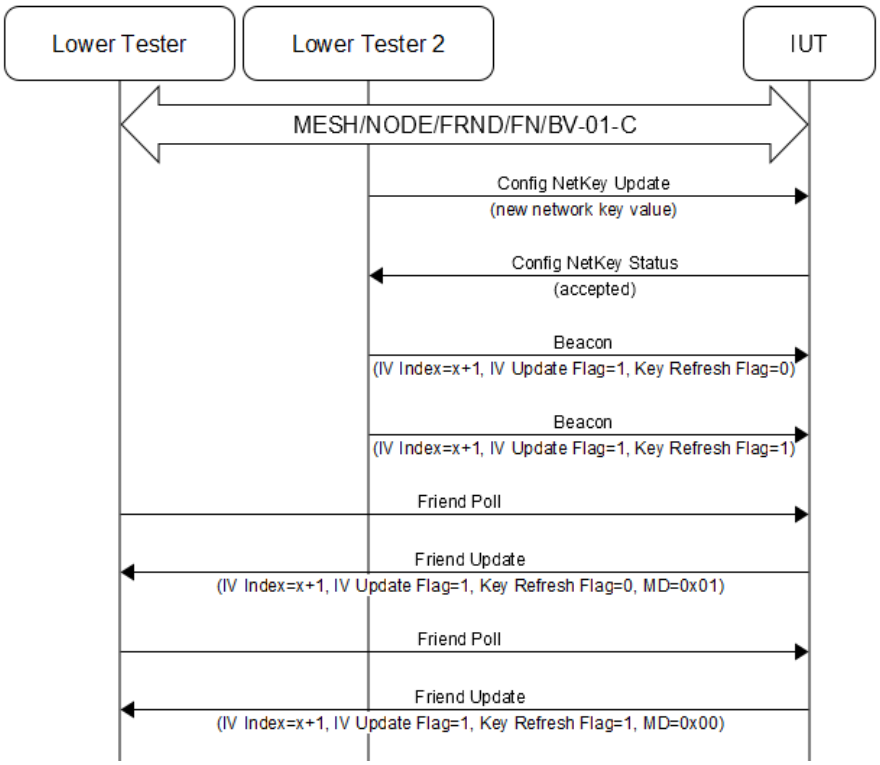


Figure 4.78: MESH/NODE/FRND/FN/BV-20-C and MESH/NODE/FRND/FN/BV-26-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/NODE/FRND/FN/BV-20-C [Adding another Friend Update upon reception of Secure Network Beacons – Friend Node]	Lower Tester 2 can broadcast Secure Network beacons.	0x01 (Secure Network beacon)
MESH/NODE/FRND/FN/BV-26-C [Adding another Friend Update upon reception of Mesh Private Beacons – Friend Node]	Lower Tester 2 can broadcast Mesh Private beacons.	0x02 (Mesh Private beacon)

Table 4.31: Adding another Friend Update - Friend Node test cases

- Expected Outcome

Pass verdict

In Step 6, the IUT sends a Friend Update message with the new IV Index value, the IV Update flag set to 1, the Key Refresh flag set to 0 and the MD field set to 0x01.

In Step 8, the IUT sends a Friend Update message with the new IV Index value, the IV Update flag set to 1,

the Key Refresh flag set to 1 and the MD field set to 0x00.

The Friend Poll and Friend Update messages and the retransmitted message are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-21-C [Friend Node Respond to Config Low Power Node PollTimeout Get]

- Test Purpose

  Verify that an IUT node acting as a Friend Node responds to the Config Low Power Node PollTimeout Get message with the correct PollTimeout for each LPN it has a Friend relationship with.

- Reference

  [3] 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 is a Configuration Client. It has access to the device key of the IUT defined in the IXIT file.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends the Config Low Power Node PollTimeout Get message with the LPNAddress field value equal to the address of Lower Tester 1.
  3. Lower Tester 2 expects to receive the Config Low Power Node PollTimeout Status message from the IUT with the LPNAddress field equal to that of Lower Tester 1 address and the PollTimeout field equal to the PollTimeout of Lower Tester 1 that was stored during Step 1.
  4. Lower Tester 2 sends the Config Low Power Node PollTimeout Get message with the LPNAddress parameter value equal to any address that isn’t the address of Lower Tester 1.
  5. Lower Tester 2 expects to receive the Config Low Power Node PollTimeout Status message from the

IUT with the LPNAddress field equal to that of the address sent by Lower Tester 2 in Step 4 and the PollTimeout field equal to the 0x000000.

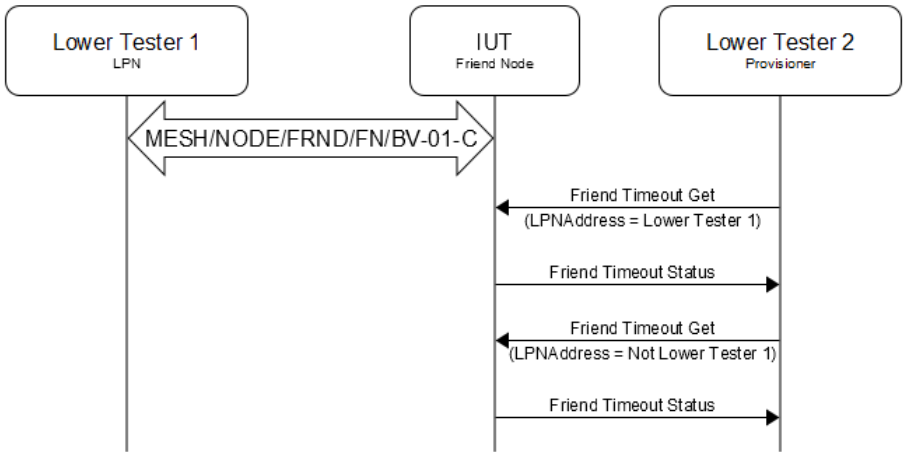


Figure 4.79: MESH/NODE/FRND/FN/BV-21-C [Friend Node Respond to Config Low Power Node PollTimeout Get] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Config Low Power Node PollTimeout Status message to Lower Tester 2 with the LPNAddress field equal to that of Lower Tester 1 address and the PollTimeout field equal to the PollTimeout of Lower Tester 1 that was stored during Step 1.

  In Step 5, the IUT sends a Config Low Power Node PollTimeout Status message to Lower Tester 2 with the LPNAddress field equal to the address sent in Step 4 from Lower Tester 2 and the PollTimeout field equal to 0x000000. All Friend Poll and Friend Update messages are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  The Config Low Power Node PollTimeout Get and Config Low Power Node PollTimeout Status messages are encrypted using the device key of the IUT.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Low Power node role.

MESH/NODE/FRND/FN/BV-22-C [Filter Duplicates in the Friend Queue]

- Test Purpose

  Verify that an IUT acting as a Friend Node does not cache two messages with the same SRC and SEQ values.

- Reference

  [3] 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature, and the feature is enabled.
  - Lower Tester 1 is a node that supports the Low Power Feature.
  - Lower Tester 2 emulates another (3rd) Node sending messages to the LPN.
  - The maximum number of entries supported by the network message cache on the IUT has been supplied in the IXIT [6]. This number is denoted as N and is greater than or equal to 2.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in the Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. Lower Tester 2 sends a message to the IUT with the DST field set to the unicast address of Lower Tester 1 and with the TTL field set to a random valid value, greater than or equal to 3, expecting the IUT to cache the message.
  3. Lower Tester 2 then sends N messages to the IUT, with the DST field set to the unicast address of the IUT, and the SRC field set to random, different values for each message. After this step is executed, the IUT has discarded the message received in Step 2 from its network message cache.
  4. Lower Tester 2 resends the message in Step 2.
  5. Lower Tester 1 sends a valid Friend Poll message, and the IUT replies with the message cached in Step 2.
  6. Lower Tester 1 sends another valid Friend Poll message (FSN field toggled), and the IUT replies with a Friend Update message, with the MD field set to 0x00.
  7. Repeat Steps 2–6, but in Step 4 use a TTL decremented by 1.

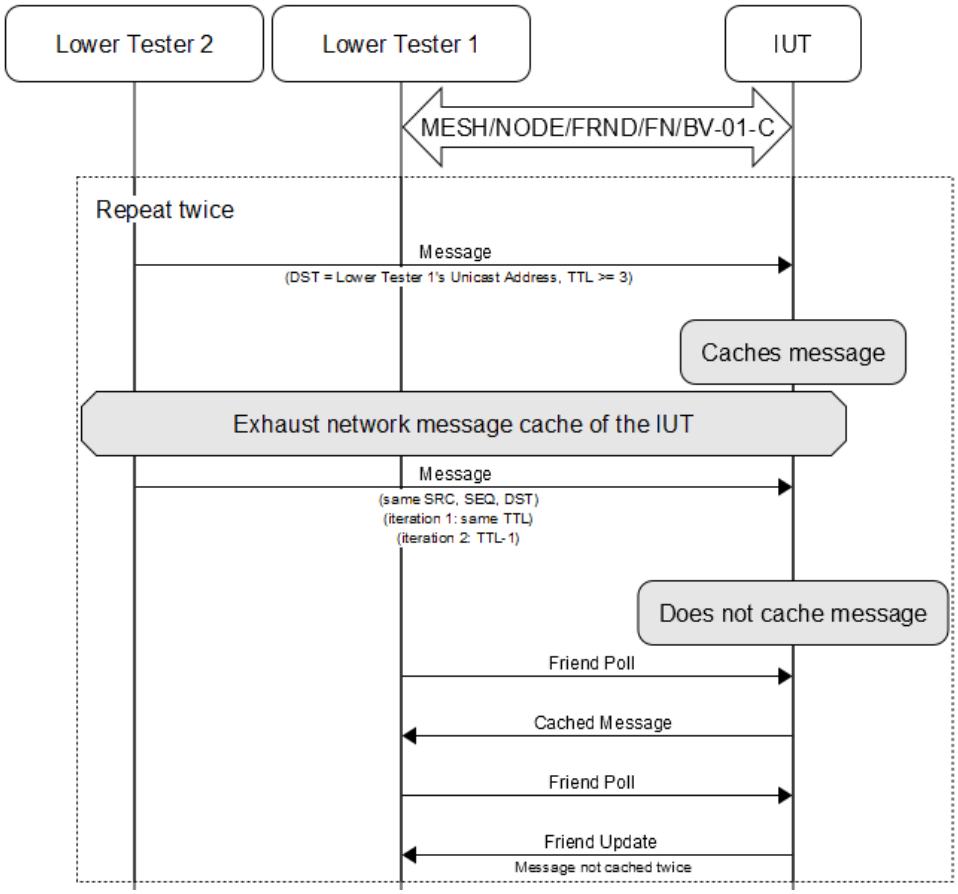


Figure 4.80: MESH/NODE/FRND/FN/BV-22-C [Filter Duplicates in the Friend Queue] MSC

- Expected Outcome

  Pass verdict

  In both iterations through Step 6, the IUT sends a Friend Update message, confirming that it has not cached the same message twice (from Steps 2 and 4).

MESH/NODE/FRND/FN/BV-23-C [Retransmit Message from LPN]

- Test Purpose

  Verify that an IUT acting as a Friend Node correctly retransmits a message received from the LPN, when the LPN uses the friendship security credentials but the Friend Node is not the final destination of the message.

- Reference

  [3] 3.5.2.1, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition

  - The IUT is a node that supports the Friend Feature, and the feature is enabled.

- If the IUT supports the Relay feature, then it is disabled.
- Lower Tester 1 is a node that supports the Low Power Feature.
- Lower Tester 2 emulates another (3rd) Node receiving messages from the LPN.
- The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in the Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

1. Lower Tester 1 elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
2. Lower Tester 1 sends a message to the IUT with the DST field set to the unicast address of Lower Tester 2 and with the TTL field set to a random valid value, greater than or equal to 2, using the friendship security credentials.
3. The IUT forwards the message to Lower Tester 2, decrementing the TTL by 1 and re-encrypting the message using the managed flooding security credentials.

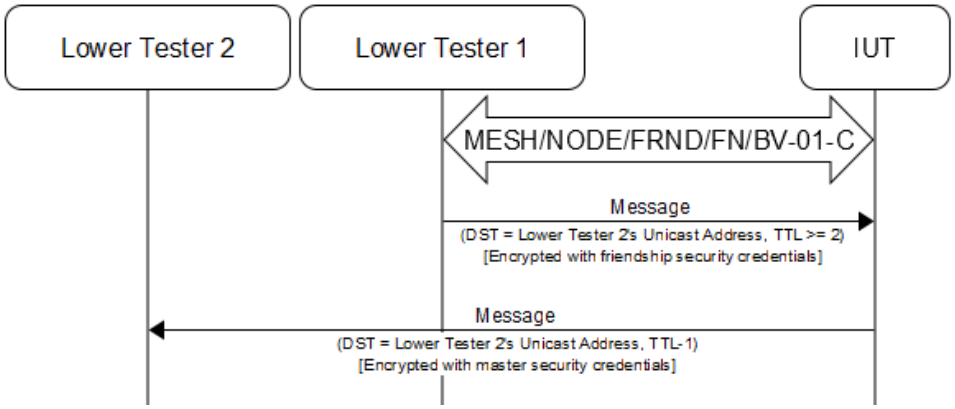


Figure 4.81: MESH/NODE/FRND/FN/BV-23-C [Retransmit Message from LPN] MSC

- Expected Outcome

Pass verdict

The IUT decrements the TTL and changes the security credentials when retransmitting the message.

MESH/NODE/FRND/FN/BI-01-C [Friendship – Friend Node – Invalid Transport Control OpCode]

- Test Purpose

Verify that an IUT node acting as a Friend Node ignores a Control message with an invalid Transport Control Opcode.

- Reference

[3] 3.6.5, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. The Lower Tester elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester sends a misformed Friend Poll message to the IUT within the PollTimeout time, with the Transport Control Opcode set to 0x0C, RFU set to 0, and FSN bit is toggled for every Friend Update Message received. The Friend Poll Message will have a TTL of zero and be addressed to the IUT.
  3. The Lower Tester expects no Friend Update message from the IUT and no response to this invalid message.

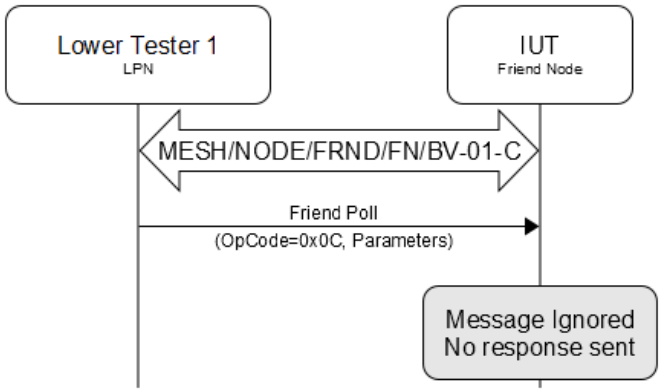


Figure 4.82: MESH/NODE/FRND/FN/BI-01-C [Friendship – Friend Node – Invalid Transport Control OpCode] MSC

- Expected Outcome

  Pass verdict

  The IUT ignores the misformed Friend Poll message with an invalid Transport Control OpCode and does not send any Friend Update or any other messages in return.

  All Friend Poll and Friend Update messages and the invalid message with Opcode equal to 0x0C are secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/FN/BI-02-C [Friendship – Friend Node – Friend Poll Invalid Prohibited Bits]

- Test Purpose

  Verify that an IUT node acting as a Friend node handles a Friend Poll message with incorrectly set Prohibited field.

- Reference

  [3] 3.1.2, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.1

- Initial Condition
  - The IUT is a node that supports the Friend feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Lower Tester elects the IUT as the Friend Node using the procedure defined in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  2. The Lower Tester sends a Friend Poll message to the IUT within the PollTimeout time, with Transport Control Opcode set to 0x01 and the Prohibited field set to 0x7F. The Friend Poll message will have a TTL of zero and be addressed to the IUT.
  3. The Lower Tester expects no response from the IUT.

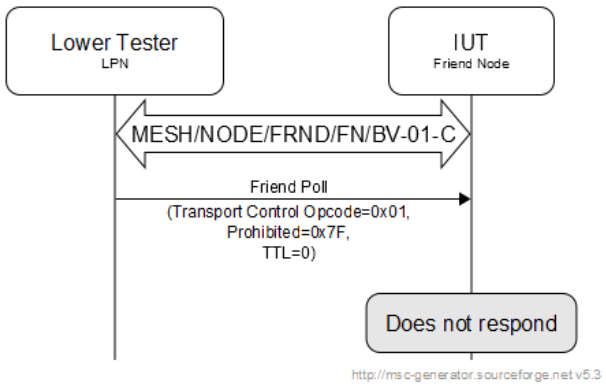


Figure 4.83: MESH/NODE/FRND/FN/BI-02-C [Friendship – Friend Node – Friend Poll Invalid Prohibited Bits] MSC

- Expected Outcome

  Pass verdict

  The IUT sends no response to the Friend Poll message with an invalid Prohibited field value.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/FN/BI-03-C [Friendship – Friend Node – Friend Request Invalid RFU Bits]

- Test Purpose

  Verify that an IUT node with Friendship feature handles a Friend Request message with incorrect set RFU bits.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.3.1

- Initial Condition
  - The IUT is a node that supports the Friend Feature and the feature is enabled.
  - The Lower Tester is a node that supports the Low Power Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- Test Procedure

  1. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, MinCacheSizeLog value of 0b000 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of 0x000064, LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.
  2. The Lower Tester expects no response from the IUT.
  3. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, MinCacheSizeLog value of 0b001 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay set to any random value between 0x00 - 0x09, PollTimeout value of 0x000064, LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.
  4. The Lower Tester expects no response from the IUT.
  5. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, MinCacheSizeLog value of 0b001 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of any random value between 0x000000 - 0x000009, LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.
  6. The Lower Tester expects no response from the IUT.
  7. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the

all-friends address, a TTL of zero, MinCacheSizeLog value of 0b001 in the Criteria field, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of 0x000064, a LPNCounter set to 0, NumElements value of 0x00, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.

8. The Lower Tester expects no response from the IUT.
9. The Lower Tester sends a Friend Request message to the IUT requesting to begin the Friendship Establishment Phase with a Transport Control Message Opcode value of 0x03, the DST field set to the all-friends address, a TTL of zero, Criteria field RFU value of 0b1, RSSIFactor value of 0b00, ReceiveWindowFactor value of 0b00, ReceiveDelay value of 0x0A, PollTimeout value of 0x000064, a LPNCounter set to 0, NumElements value of 0x01, and a PreviousAddress value of 0x0000. The Lower Tester then waits 100 ms.
10. The Lower Tester then expects to receive a Friend Offer message. The Friend Offer Message should be sent after the Friend Offer Delay which is calculated by using the RSSI factor and Received Window Factor.

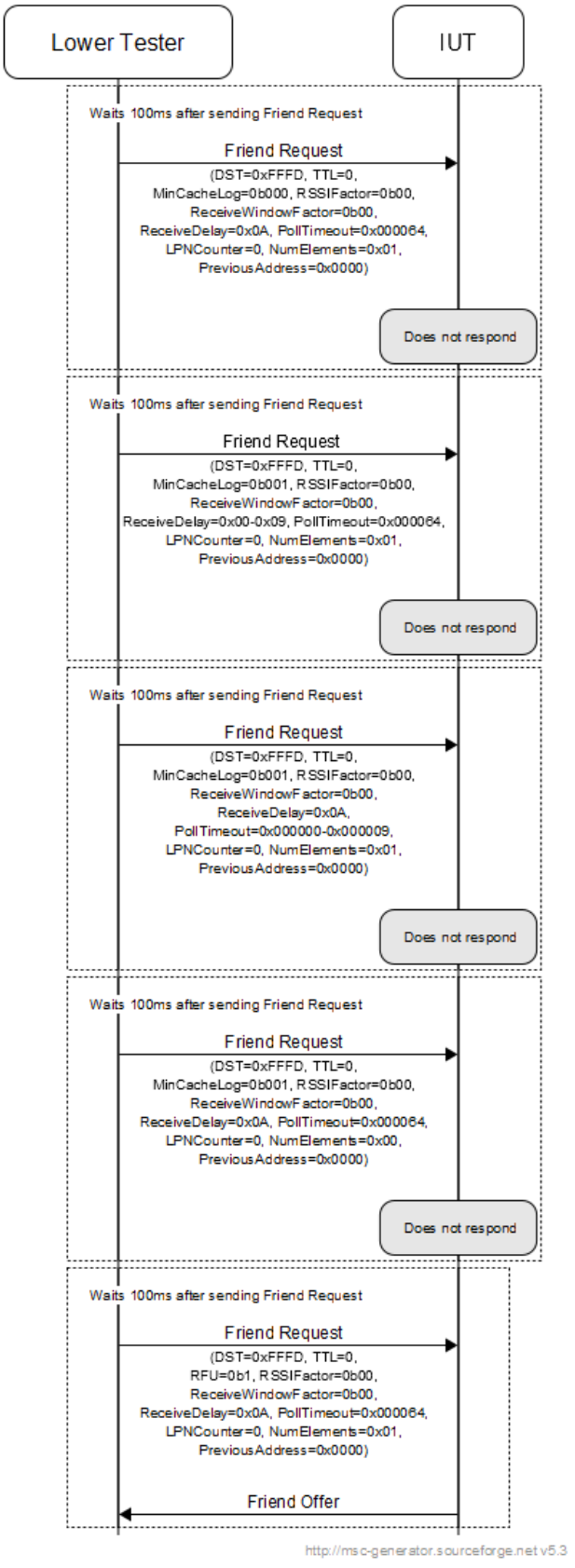


Figure 4.84: MESH/NODE/FRND/FN/BI-03-C [Friendship – Friend Node – Friend Request Invalid RFU Bits] MSC

- Expected Outcome

  Pass verdict

  The IUT sends Friend Offer message to the Lower Tester's unicast address ignoring RFU bit in Criteria field of the Friend Request message, with a TTL value of 0, and a Transport Control Message Opcode value of 0x04.

  The IUT sends the Friend Offer messages after the calculated Friend Offer Delay.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.13.2 Low Power Node

This section contains tests related to behavior from the perspective of a node participating in Friendship procedures as a Low Power node.

MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can perform the Friendship Establishment procedure.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 4.2.2.4

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The value of the Network Transmit Count state on the IUT is greater than 0.
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester is a node that supports the Friend Feature.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The IUT exposes some mechanism to initiate the Low Power Establishment operation.
  - The IUT has not ran the Low Power Establishment operation with other nodes yet.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
  2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.

3. The Lower Tester should wait the appropriate Friend Offer Delay computed using the formula found in [3] Section 3.6.6.3.1 from receiving the Friend Request message and then send Friend Offer message to the IUT's unicast address, with a TTL value of 0, a Transport Control Message Opcode value of 0x04, a ReceiveWindow field value equal to 0xFF, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sent a Friend Offer message.
4. The Lower Tester starts a timer once the Friend Offer has been sent.
5. The Lower Tester expects to receive a Friend Poll from the IUT within 1 second of sending the Friend Offer.
6. Once the Lower Tester receives the Friend Poll from the IUT it responds with a Friend Update message with the Key Refresh flag field set to 0, the IV Update flag field set to 0 and the MD field set to 0x00.

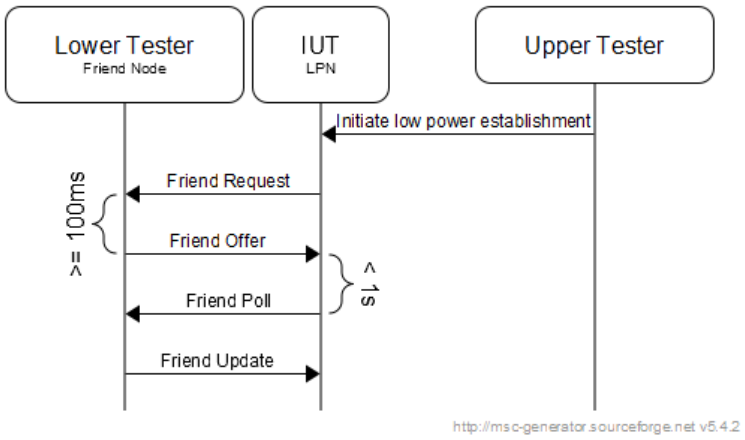


Figure 4.85: MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Request to the Lower Tester with the DST field set to the all-friends address, a TTL of 0, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, PollTimeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address 0x0000. The Transport Control Opcode field is set to 0x03.

  The IUT sends a Friend Poll message to the Lower Tester no more than 1 seconds after the Friend Offer was sent by the Lower Tester. The Transport Control Opcode field is set to 0x01, RFU field set to 0x00, FSN field set to 0, TTL field set to 0 and destination address set to the Lower Tester unicast address.

  The Friend Poll message is secured using the friendship security credentials.

  Each Friend Request message is secured using the managed flooding security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the

unicast address of the primary Element of the sender.

Fail verdict

The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Friend node role.

MESH/NODE/FRND/LPN/BV-02-C [Friendship Messaging – Low Power Node]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can request and retrieve cached messages from a Friend Node and can handle a Friendship Credentials Flag change.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

- Initial Condition

  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature and the Configuration Client Model.
  - The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester has access to the device key of the IUT.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  3. The following Steps 4–7 is repeated for N messages where N =1, 2, 3, and 4.
  4. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll Message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll Messages to ensure each Friend Poll has an appropriate FSN field.
  5. The Lower Tester sends a Friend Update Message to the IUT indicating it has zero messages for the IUT and expects the IUT to return to sleep.
  6. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll will have a TTL of zero and be addressed to the Lower Tester.
  7. The Lower Tester sends N message(s) to the IUT followed by a Friend Update message with the MD field set to 0x00. The Lower Tester expects an appropriate Friend Poll message from the IUT after sending each cached message, and no Friend Poll message after sending the Friend Update message.

8. The Upper Tester orders the IUT to publish a Health Status message.
9. The Lower Tester expects the IUT to use the managed flooding security credentials when sending the Health Status message.
10. The Lower Tester caches a Config Model Publication Set message with the CredentialFlag field set to 1.
11. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester
12. The Lower Tester sends the cached Config Model Publication Set message to the IUT and after another Friend Poll message from the IUT it sends a Friend Update message with the MD field set to 0x00.
13. The Upper Tester orders the IUT to publish a Health Status message.
14. The Lower Tester expects the IUT to use the friendship security credentials when sending the Health Status message.

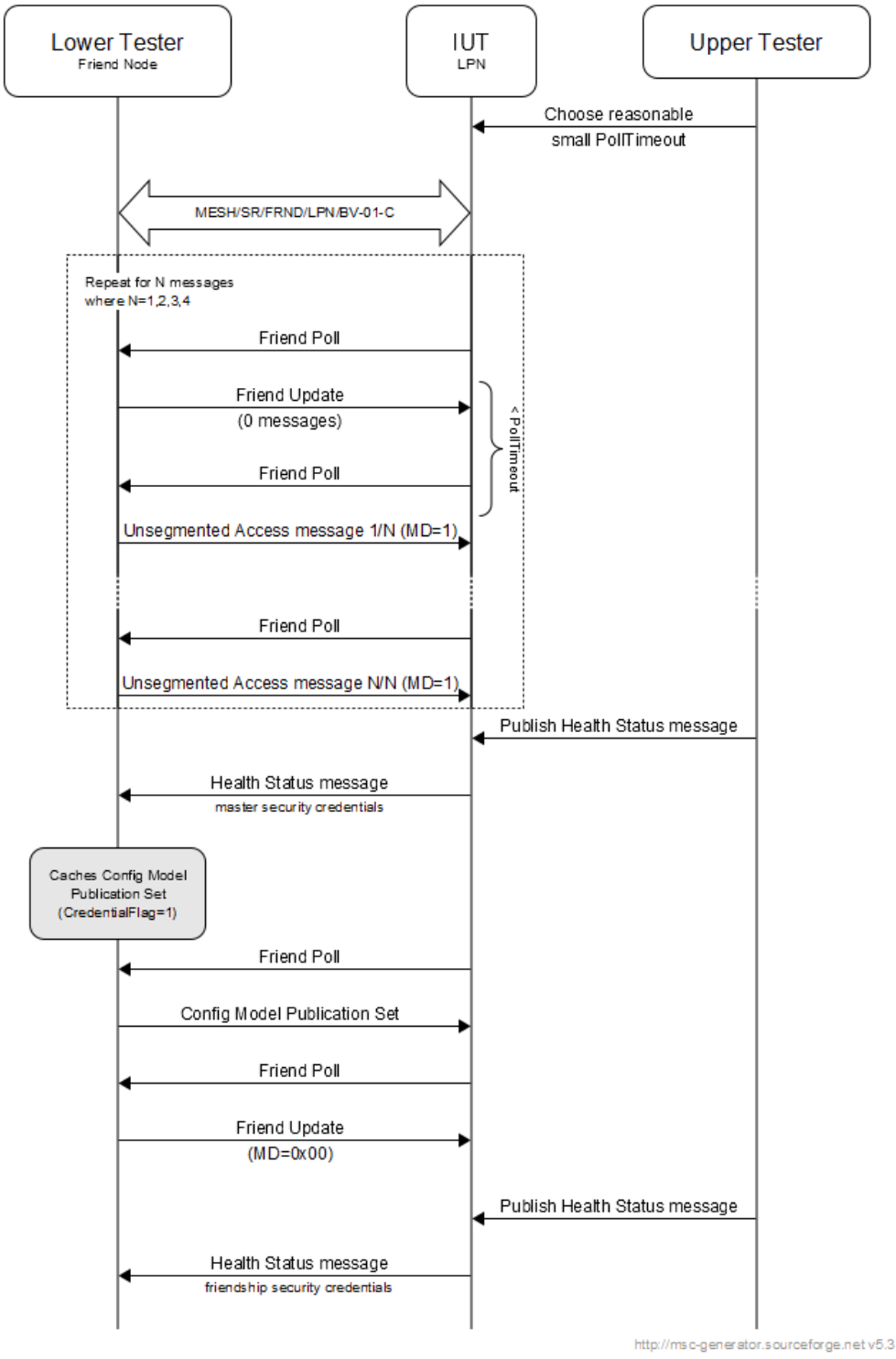


Figure 4.86: MESH/NODE/FRND/LPN/BV-02-C [Friendship Messaging – Low Power Node] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Poll Message to the Lower Tester within the PollTimeout in each cycle of the test

procedure.
Each Friend Poll Message has an appropriate FSN value, RFU field set to 0, TTL value set to 0, and the Transport Control message, Opcode set to 0x01 and destination address set to the Lower Tester unicast address.

The IUT successfully receives all cached messages and sends all expected Friend Poll messages. The IUT stops sending Friend Poll messages when it receives a Friend Update message with the MD field set to 0x00.

The IUT sends the Health Status message using the friendship security credentials in Step 14.

Each Friend Poll message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BV-03-C [Friend IV Update – Low Power Node]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can receive and react to IV Index updates from the Friend Node.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.8.3

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure
  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use. The Lower Tester will send all Friend Update messages in this test procedure with the MD field set to 0x00.
  3. The Upper Tester orders the IUT to ignore IV Update procedure time constrains.
  4. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll Message will utilize the old IV Index and have a TTL of zero and be addressed to the Lower Tester.
  5. The Lower Tester sends a Friend Update Message with the new IV Index and the IV Index flag set to 1, indicating that the IV Update procedure is in IV Update in Progress state.

6. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll Message will utilize the old IV Index and have a TTL of zero and be addressed to the Lower Tester.
7. The Lower Tester sends a Friend Update message with the new IV Index and the IV Index flag set to 0, indicating that the IV Update procedure is completed.
8. The Lower Tester expects the IUT to send a Friend Poll Message within the PollTimeout. The Friend Poll message will utilize the new IV Index and have a TTL of zero and be addressed to the Lower Tester.
9. In response to the Friend Poll message, the Lower Tester sends a Friend Update with the new IV Index and IV Index flag set to 0.

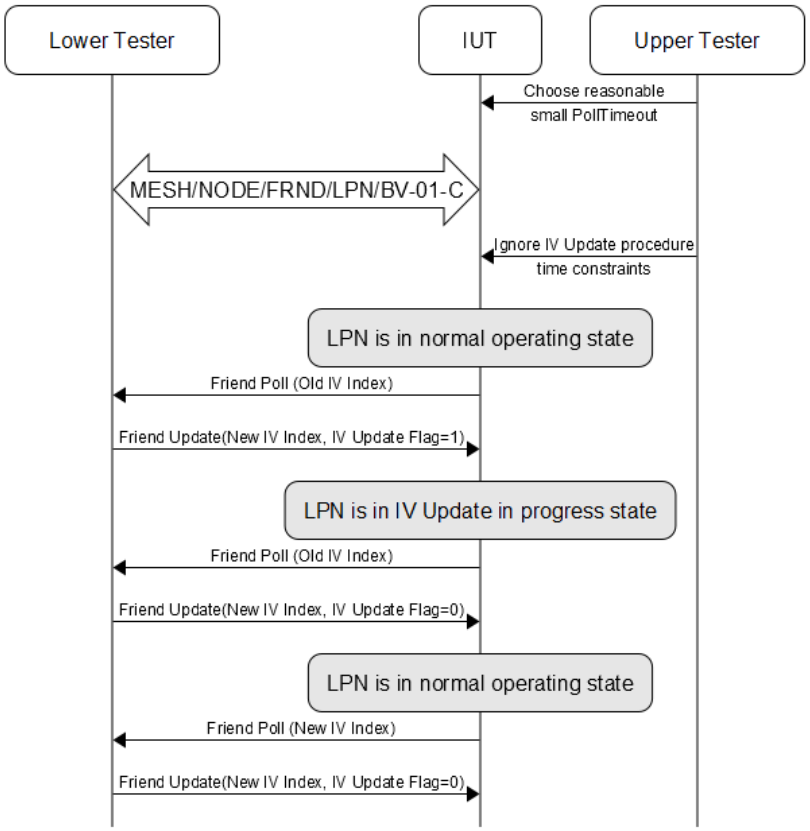


Figure 4.87: MESH/NODE/FRND/LPN/BV-03-C [Friend IV Update – Low Power Node] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Poll message to the Lower Tester within the PollTimeout in each cycle of the test procedure.

  Each Friend Poll message has an appropriate FSN value from the IUT, RFU field set to 0, TTL value set to 0, and the Transport Control message, Opcode set to 0x01 and destination address set to the Lower Tester unicast address.

  The IUT begins to utilize the new IV Index once it receives the Friend Update message with new IV Index

and IV Update flag set to 0 from the Lower Tester.

Each Friend Poll message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can send the Friend Subscription List Add message to a relay.

- Reference

  [3] 3.6.6, 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.9, 3.6.6.3.3, 3.6.6.4.3

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The value of the Network Transmit Count state on the IUT is greater than 0.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and the IUT have previously established a friend relationship through test MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Friend Subscription List Add message to the Lower Tester with the "AddressList" field containing values supplied in the IXIT [6] and the "TransactionNumber" field containing a valid value.
  2. The Lower Tester expects to receive a Friend Subscription List Add message.

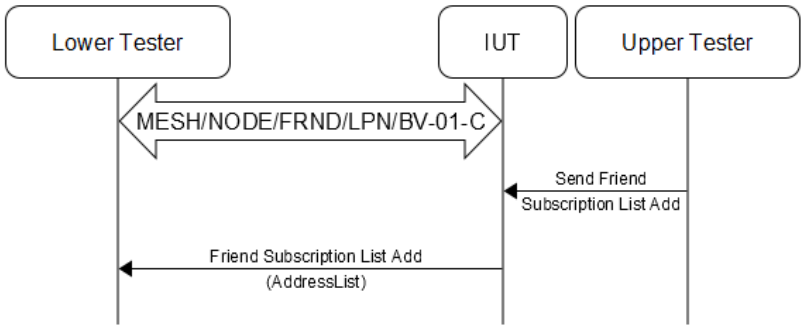


Figure 4.88: MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Subscription List Add message with values supplied in the IXIT [6].

  The TransactionNumber in Step 1 is the value 0x00.

  The Friend Subscription List Add message is secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

  Fail verdict

  The IUT transmits more than one Network PDU when sending any message to the Lower Tester in the Friend node role.

MESH/NODE/FRND/LPN/BV-05-C [Friendship – Send Friend Subscription List Remove Message]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can send the Friend Subscription List Remove message to a relay.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.5.7, 3.6.5.8, 3.6.5.9, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- The Lower Tester and the IUT have previously established a friend relationship and have executed the test procedure described in MESH/NODE/FRND/LPN/BV-04-C [Friendship – Send Friend Subscription List Add Message].
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

1. The Upper Tester orders the IUT to send a Friend Subscription List Remove message to the Lower Tester with the "AddressList" field containing values supplied in the IXIT [6] and the "TransactionNumber" field containing a valid value.
2. The Lower Tester expects to receive a Friend Subscription List Remove message.

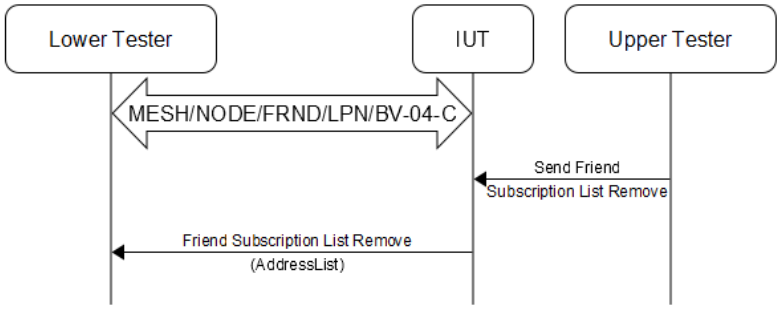


Figure 4.89: MESH/NODE/FRND/LPN/BV-05-C [Friendship – Send Friend Subscription List Remove Message] MSC

- Expected Outcome

Pass verdict

The IUT sends a Friend Subscription List Remove message with values supplied in the IXIT [6].

The Friend Subscription List Remove message is secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BV-06-C [Friendship Updates Failed Friend Update Message – Low Power Node]

- Test Purpose

Verify that an IUT node acting as a Low Power Node can handle a missed Friend update message.

- Reference

[3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.

- The Lower Tester is a node that supports the Friend Feature.
- The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
3. The Lower Tester expects the IUT to send a Friend Poll message within the PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester.
4. The Lower Tester receives the Friend Poll message, waits and does not send the Friend Update message.
5. The Lower Tester waits for the Friend Poll message to be sent again and checks to make sure the FSN field is the same as before since there was no acknowledgment of the Friend Poll.

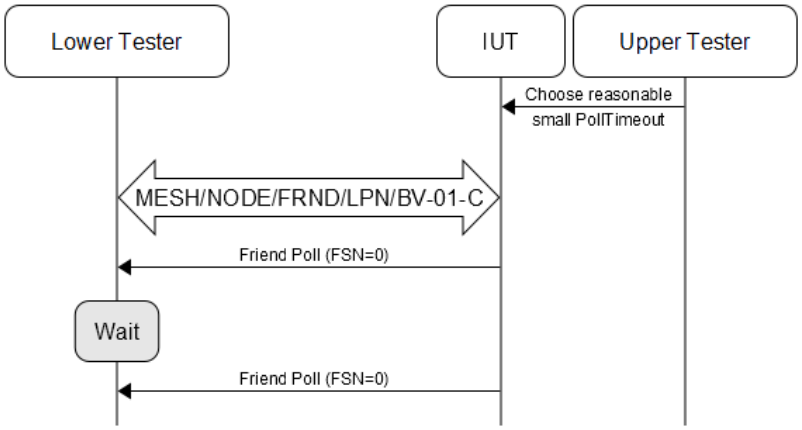


Figure 4.90: MESH/NODE/FRND/LPN/BV-06-C [Friendship Updates Failed Friend Update Message – Low Power Node] MSC

- Expected Outcome

Pass verdict

The IUT sends a Friend Poll messages to the Lower Tester within the PollTimeout in each cycle of the test procedure.

The IUT sends another Friend Poll message with the same FSN field value as the first in Step 4.

Each Friend Poll messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BV-07-C [Friend Key Refresh – Low Power Node]

- Test Purpose

  Verify that an IUT node acting as a Friend Node can properly handle a Key Refresh procedure.

- Reference

  [3] 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3, 3.6.6.4.1, 3.10.4

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester(s) share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - Key Refresh procedure is not already in progress.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
  - IV Update test mode is activated on the IUT.

- Test Procedure

  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use. The Lower Tester will send all Friend Update messages in this test procedure with the MD field set to 0x00.
  3. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll Message to the Lower Tester. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll message to ensure each Friend Poll has an appropriate FSN field.
  4. The Lower Tester sends Config NetKey Update message to IUT with new value of the NetKey thus starting Key Refresh procedure on IUT.
  5. The Lower Tester expects a Friend Poll message from the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT and is secured with the old NetKey.
  6. The Lower Tester sends a Friend Update message to the IUT indicating it has zero messages for the Lower Tester, indicating the current IV Index, the IV Update Flag equal to 0, and Key Refresh flag set to 1. The Friend Update messages are secured using updated Friendship Security Credentials with the new NetKey.
  7. The Lower Tester expects the IUT to send a Friend Poll message to the IUT within the negotiated PollTimeout. The Friend Poll message will have a TTL of zero and be addressed to the IUT and secured with the new NetKey.
  8. The Lower Tester sends a Friend Update message to the IUT indicating it has zero messages for the IUT, indicating the current IV Index, the IV Update Flag equal to 0, and Key Refresh flag set to 0. The

Friend Update messages is secured using updated Friendship Security Credentials with the new NetKey.

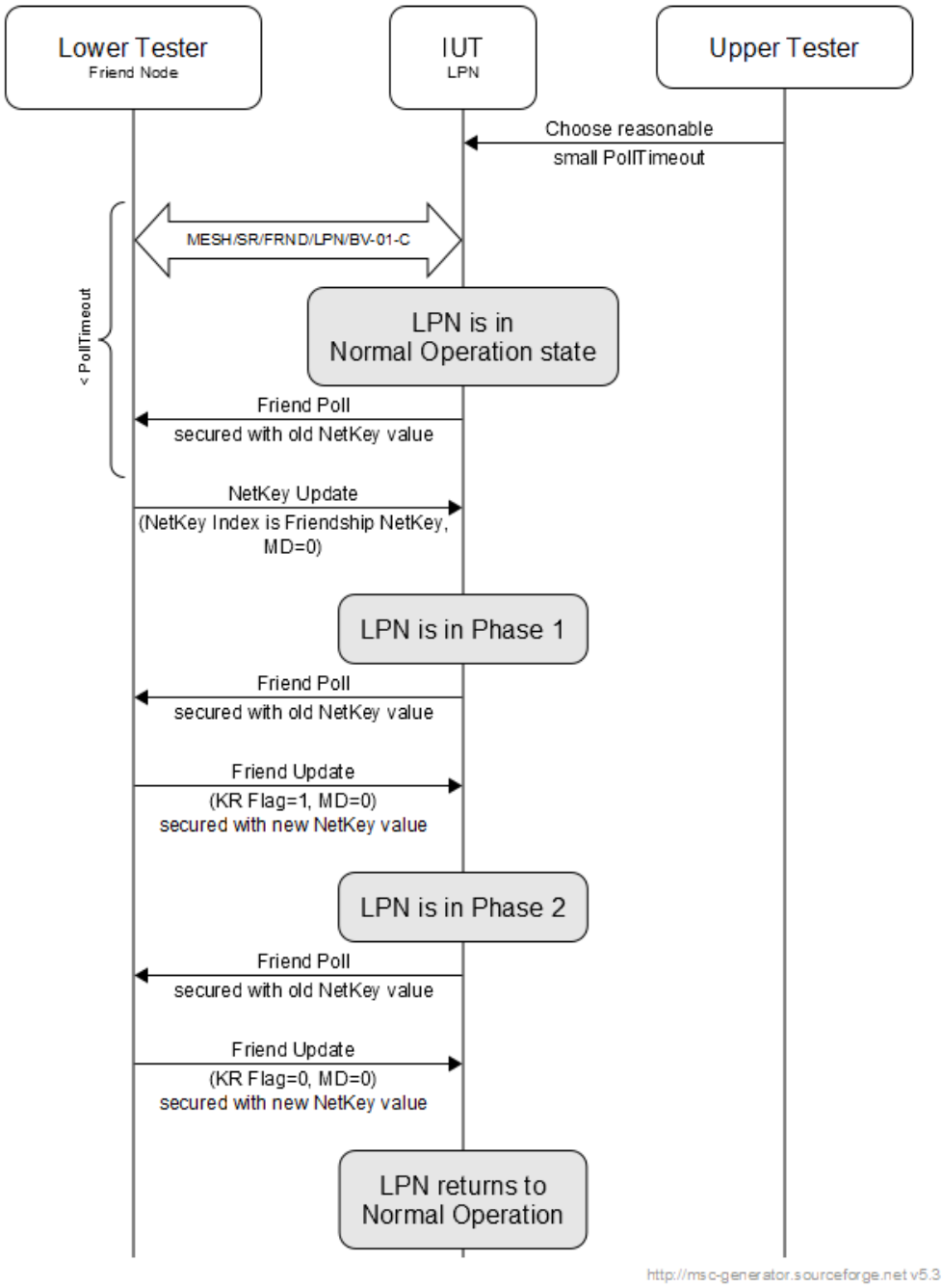


Figure 4.91: MESH/NODE/FRND/LPN/BV-07-C [Friend Key Refresh – Low Power Node] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Friend Poll message in Step 7 secured with the new network key.

  All Friend Poll and Friend Update messages are secured using the friendship security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BV-08-C [Low Power Node Send Friend Clear Message]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can send the Friend Clear message.

- Reference

  [3] 3.6.5.5, 3.6.6.4.2

- Initial Condition
  - The IUT is a node that supports the Low Power feature.
  - The Lower Tester acts as a Friend node.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Lower Tester and the IUT have previously established a friend relationship through test MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

  1. The Upper Tester orders the IUT to terminate the friendship using a Friend Clear message.
  2. The Lower Tester expects to receive a Friend Clear message from the IUT.

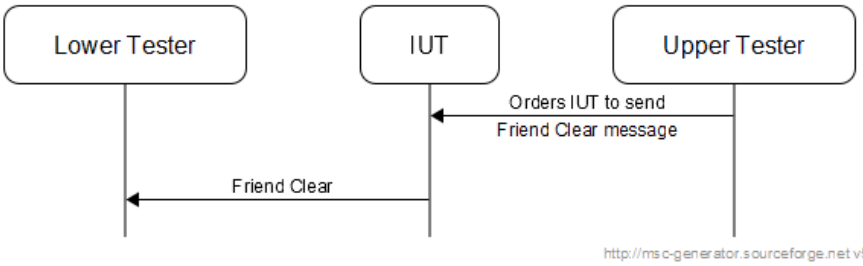


Figure 4.92: MESH/NODE/FRND/LPN/BV-08-C [Low Power Node Send Friend Clear Message] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Friend Clear message to the Lower Tester. The LPNAddress field is set to the unicast address of the IUT.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BI-01-C [Friendship Establishment Mode – Low Power Node Friend Poll Retries]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node will retry Friend Polls during the Friendship Establishment period until the maximum number of retries.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1, 3.6.6.4.2

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The IUT exposes some mechanism to initiate the Low Power Establishment operation.
  - The IUT has not yet run the Low Power Establishment operation with other nodes.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.
- Test Procedure
  1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
  2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.
  3. The Lower Tester should wait the calculated Friend Offer Delay from receiving the Friend Request

message and then send a Friend Offer message to the IUT's unicast address, with a TTL value of 0, a Transport Control message, Opcode value of 0x04, a ReceiveWindow field value equal to 0xFF, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sends a Friend Offer message.

4. The Lower Tester expects to receive the Friend Poll message no more than 1 second after sending the Friend Offer.
5. The Lower Tester waits and does not send a Friend Update message.
6. The Lower Tester expects the IUT to resend the Friend Poll message.
7. The Lower Tester responds with a Friend Update message with the MD field set to an invalid value in the interval 0x02-0xFF.
8. The Lower Tester expects the IUT to resend the Friend Poll message.

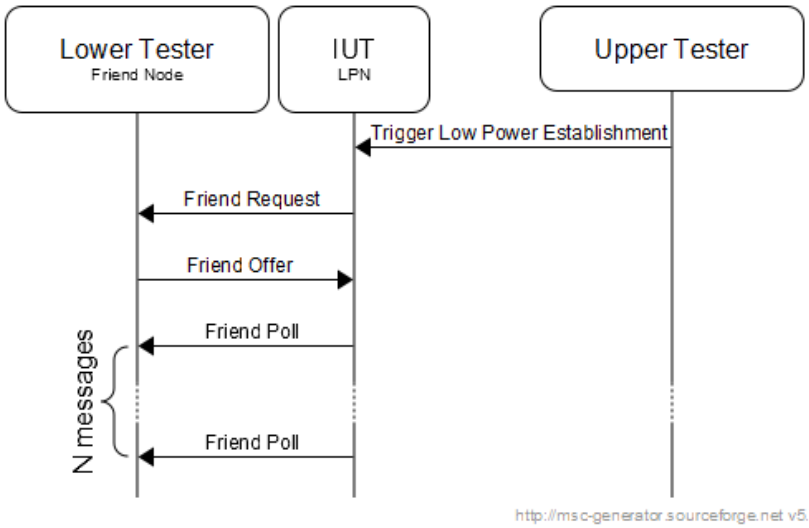


Figure 4.93: MESH/NODE/FRND/LPN/BI-01-C [Friendship Establishment Mode – Low Power Node Friend Poll Retries] MSC

- Expected Outcome

Pass verdict

The IUT sends a Friend Request message in Step 2 with the DST field set to the all-friends address, a TTL of 0, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, PollTimeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address.

The IUT sends a Friend Poll message in Steps 4, 6, and 8.

All Friend Poll messages are secured using the friendship security credentials.

All Friend Request messages are greater than 1.1 seconds apart from each other.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BI-02-C [Invalid Transport Control OpCode – Low Power Node]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can handle a message properly that contains an RFU Transport Control Opcode.

- Reference

  [3] 3.6.5, 3.6.5.1, 3.6.5.2, 3.6.6, 3.6.6.3.3

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.
  - The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure
  1. The Upper Tester orders the IUT to choose a reasonably small PollTimeout to ensure that this test can be completed within minutes after start.
  2. The Lower Tester is elected as the Friend Node for the IUT using the procedure defined in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node]. The ReceiveWindow, ReceiveDelay, and PollTimeout are retained for later use.
  3. The Lower Tester waits for up to the PollTimeout for the IUT to send a Friend Poll message to the Lower Tester. The Friend Poll message will have a TTL of zero and be addressed to the Lower Tester. The Lower Tester records the FSN field value of each Friend Poll message.
  4. The Lower Tester sends a message to the IUT with the Transport Control Opcode field set to an RFU value, setting the SRC field to its own unicast address (the Lower Tester’s unicast address used during friendship establishment).
  5. The Lower Tester expects to receive the next Friend Poll message with the FSN field value toggled.

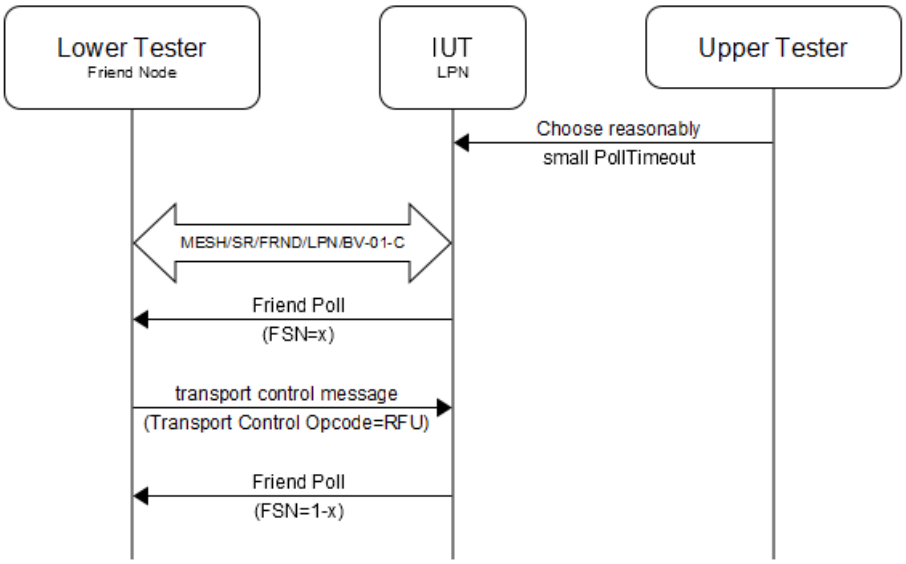


Figure 4.94: MESH/NODE/FRND/LPN/BI-02-C [Invalid Transport Control OpCode – Low Power Node] MSC

- Expected Outcome

  Pass verdict

  The IUT ignores the message with an RFU Transport Control Opcode value but sends the Friend Poll message in Step 5 with an alternating FSN value.

  Each Friend Poll message is secured using the friendship security credentials.

  All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

MESH/NODE/FRND/LPN/BI-03-C [Friendship Low Power Node – Friend Offer Invalid RFU Bits]

- Test Purpose

  Verify that an IUT node acting as a Low Power Node can handle a Friend Offer message with invalid set RFU bits.

- Reference

  [3] 3.6.5.3, 3.6.5.4, 3.6.6, 3.6.6.4, 3.6.6.4.1

- Initial Condition
  - The IUT is a node that supports the Low Power Feature.
  - The Lower Tester is a node that supports the Friend Feature.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT is in Normal Operation state of the IV Update and the Key Refresh procedures for all its network keys.

- The IUT exposes some mechanism to initiate the Low Power Establishment operation.
- The IUT has not ran the Low Power Establishment operation with other nodes yet.
- The Friendship Security Credentials are calculated as found in [3] Section 3.8.5.3.1.

- Test Procedure

1. The Upper Tester orders the IUT to initiate the Low Power Establishment operation.
2. The Lower Tester expects to receive a Friend Request message from the IUT requesting to begin the Friendship Establishment operation.
3. The Lower Tester should wait the appropriate Friend Offer Delay computed using the formula found in [3] Section 3.6.6.3.1 from receiving the Friend Request message and then send a Friend Offer messages to the IUT's unicast address, with a TTL value of 0, a Transport Control message, Opcode value of 0x04, a ReceiveWindow field value equal to 0x00, RSSI value of the measured RSSI of the IUT, and a FriendCounter field with value equal to the number of times the Lower Tester sends a Friend Offer message.
4. The Lower Tester starts a timer once the Friend Offer has been sent.
5. The Lower Tester does not expect the IUT to send a Friend Poll message and the Friendship should not be established.

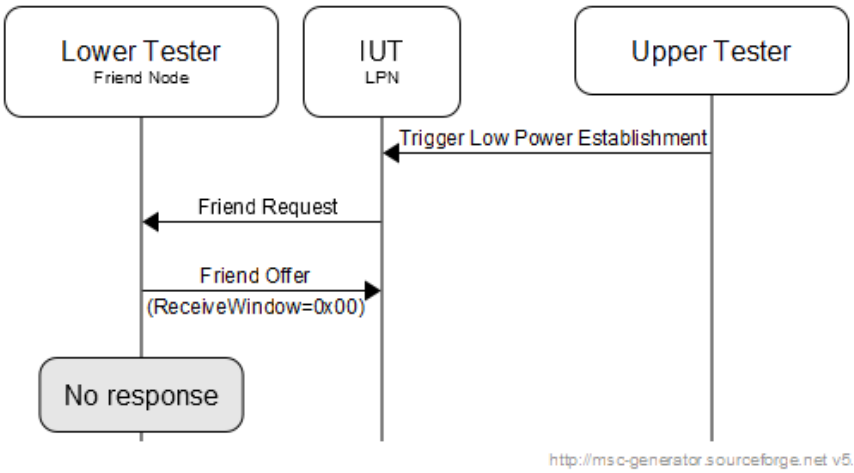


Figure 4.95: MESH/NODE/FRND/LPN/BI-03-C [Friendship Low Power Node – Friend Offer Invalid RFU Bits] MSC

- Expected Outcome

Pass verdict

The IUT sends a Friend Request to the Lower Tester with the DST field set to the all-friends address, a TTL of zero, Criteria field value between 0x00 and 0x7F excluding values when MinCacheSizeLog is equal to 0b000, ReceiveDelay value between 0x0A and 0xFF, Poll7imeout value between 0x00000A and 0x34BBFF, a LPNCounter set to 0, a NumElements field equal to that of the total number of elements on the device, and the PreviousAddress field value set to Unassigned Address. The Transport Control Opcode field should be set to 0x03.

The IUT does not send a Friend Poll message in response to the Friend Offer.

The Friend Request message is secured using the managed flooding security credentials.

All Transport Control messages are sent as Unsegmented Control Messages with the SRC field set to the unicast address of the primary Element of the sender.

4.14 Proxy Protocol

The test objective is to verify functionality of the proxy protocol which can be used over several connection oriented channels.

MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can receive the Proxy Filter for setup for a accept list filter.

- Reference

  [3] 6.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - The Lower Tester is a Proxy Client.
- Test Procedure
  1. The Lower Tester sends a Proxy PDU to the IUT with the SAR field set to b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value set to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (Accept List Filter).
  2. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" back to the Lower Tester with a FilterType field set to 0x00 (Accept List) and a ListSize field set to [0x00].
  3. The Lower Tester sends a Proxy PDU to the IUT with the SAR field set to b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value set to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to the of the Lower Tester.
  4. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" back to the Lower Tester with a FilterType value equal to 0x00 (Accept List) and a ListSize equal to [1].

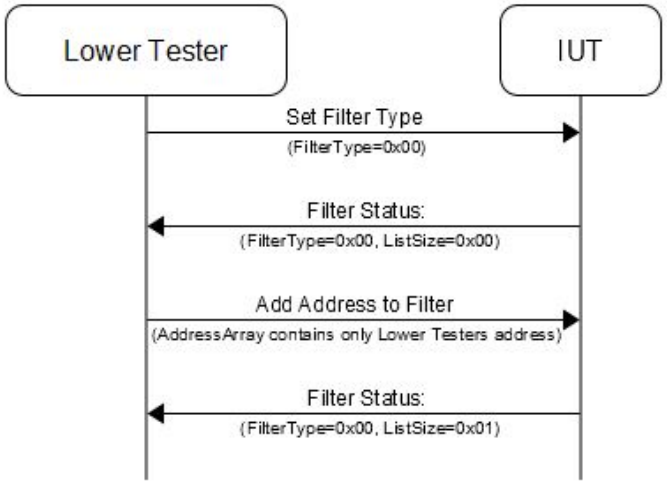


Figure 4.96: MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a single Proxy Configuration Message with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (Accept List), and a ListSize field value of [0] in Step 2.

  The IUT sends a single Proxy Configuration Message with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (Accept List), and a ListSize field value of [1] in Step 4.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address 0x0000.

4.14.1 Complete Message Network PDU Relay to Proxy Client

- Test Purpose

  Verify that an IUT that supports either the Proxy Protocol feature or the Private Proxy feature, and the Proxy Server role can send a Network PDU to the Proxy Client where one complete message fits in the PDU.

- Reference

  [3] 3.4.6.3, 6.3, 6.5, 6.6

- Initial Condition
  - The IUT has just performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT.
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.

- The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].
- Additional conditions are specified in Table 4.32.

- Test Procedure

1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1. The Proxy PDU does not require reassembly at the IUT.
2. Lower Tester 1 expects the IUT to send a Proxy PDU with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and the complete mesh message to Lower Tester 1.

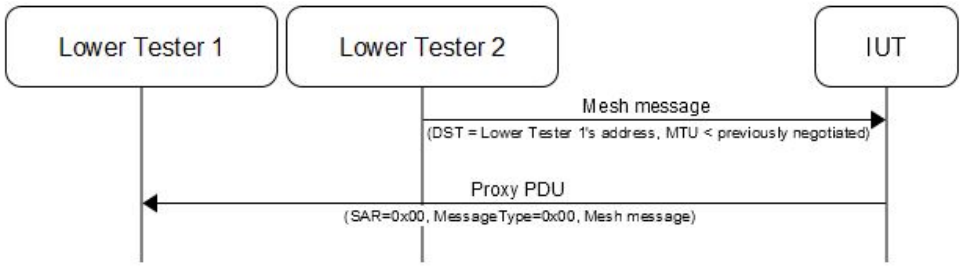


Figure 4.97: MESH/SR/PROX/BV-02-C and MESH/SR/PROX/BV-16-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition
MESH/SR/PROX/BV-02-C [Complete Message Network PDU Relay to Proxy Client – GATT Proxy Enabled]	GATT Proxy state is set to 0x01.
MESH/SR/PROX/BV-16-C [Complete Message Network PDU Relay to Proxy Client – Private GATT Proxy Enabled]	GATT Proxy state is set to 0x00. Private GATT Proxy state is set to 0x01.

Table 4.32: Complete Message Network PDU Relay to Proxy Client test cases

- Expected Outcome

Pass verdict

The IUT sends a single Proxy PDU to Lower Tester 1 with the same Access message data as that sent from Lower Tester 2 to the IUT. The TTL value should be decremented by 1 for the Proxy hop.

MESH/SR/PROX/BV-03-C [Segmented Message Network PDU Relay to Proxy Client]

- Test Purpose

Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU to the Proxy Client where one complete message does not fit into the PDU.

- Reference

[3] 6.3, 6.5, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].
- Test Procedure
  1. Lower Tester 2 sends a mesh message to the IUT larger than the negotiated MTU for that connection and with the destination address of Lower Tester 1.
  2. Lower Tester 1 expects the IUT to send a Proxy PDU to Lower Tester 1 with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field containing the first segment of the mesh message to Lower Tester 1.
  3. If the message size in Step 2 was greater than 2 times the negotiated MTU for the Proxy connection then IUT then sends a Proxy PDU to Lower Tester 1 with SAR field set to b10 ("Data field contains a continuation segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field containing the next complete segment of the mesh message to Lower Tester 1.
  4. Step 3 is repeated for as many times until the last segment is to be sent.
  5. If the message size in Step 2 was not greater than 2 times the negotiated MTU for that Proxy connection then Steps 3 and 4 will not occur and the IUT then sends a Proxy PDU to Lower Tester 1 with SAR field set to b11 ("Data field contains the last segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field containing the last segment of the mesh message to Lower Tester 1.
  6. Lower Tester 1 receives all Proxy PDUs from the IUT with the exact data contained in the entire message sent from Lower Tester 2 to the IUT.

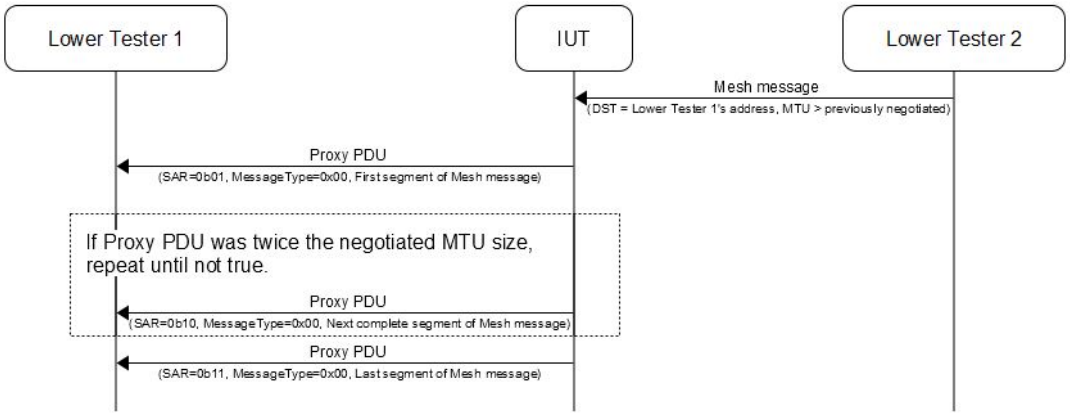


Figure 4.98: MESH/SR/PROX/BV-03-C [Segmented Message Network PDU Relay to Proxy Client] MSC

- Expected Outcome

  Pass verdict

  The IUT sends all the correct Proxy PDUs in Steps 2–5 with the SAR field set correctly for each associated step, Message Type equal to 0x00.

  Lower Tester 1 receives the entire mesh message data sent by Lower Tester 2.

4.14.2 Send a Complete Message Network PDU to Node from Proxy Client

- Test Purpose

  Verify that an IUT that supports either the Proxy Protocol feature or the Private Proxy feature, and the Proxy Server role can send a Network PDU from the Proxy Client to the Mesh Node where one complete message does fit into the PDU.

- Reference

  [3] 3.4.6.3, 6.3, 6.5, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT.
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].
  - Additional conditions are specified in Table 4.33.

- Test Procedure

  1. Lower Tester 1 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and with the DST address of Lower Tester 2. The Proxy PDU does not require reassembly at the IUT.
  2. Lower Tester 2 receives a mesh message from the IUT with the exact data contained in the Proxy PDU sent from Lower Tester 1 to the IUT.

- Test Case Configuration

Test Case	Additional Initial Condition
MESH/SR/PROX/BV-04-C [Send a Complete Message Network PDU to Node from Proxy Client – GATT Proxy Enabled]	GATT Proxy state is set to 0x01.
MESH/SR/PROX/BV-17-C [Send a Complete Message Network PDU to Node from Proxy Client – Private GATT Proxy Enabled]	GATT Proxy state is set to 0x00. Private GATT Proxy state is set to 0x01.

Table 4.33: Send a Complete Message Network PDU to Node from Proxy Client test cases

- Expected Outcome

  Pass verdict

  The IUT sends a single mesh message to Lower Tester 2 with the same data as in the Proxy PDU sent from Lower Tester 1 to the IUT.

MESH/SR/PROX/BV-05-C [Send a Segmented Message Network PDU to Node from Proxy Client]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can send a Network PDU from the Proxy Client to the Mesh Node where one complete message does not fit into the PDU.

- Reference

  [3] 6.3, 6.5, 6.6

- Initial Condition

  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].

- Test Procedure

  1. Lower Tester 1 sends a mesh message to the IUT larger than the negotiated MTU for that connection and with the destination address of Lower Tester 2.
  2. Lower Tester 1 first sends a Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the first portion of segmented mesh data to the IUT.
  3. If the message size in Step 1 was greater than 2 times the negotiated MTU for the Proxy connection Lower Tester 1 then sends a Proxy PDU to the IUT with SAR field set to b10 ("Data field contains a continuation segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field containing the next segment of mesh data to the IUT.
  4. Step 2 is repeated for as many times until the last mesh data segment is to be sent.
  5. If the message size in Step 1 was not greater than 2 times the negotiated MTU for that Proxy connection then Steps 2 and 3 will not occur and Lower Tester 1 then sends a Proxy PDU to the IUT with SAR field set to b11 ("Data field contains the last segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the last segment of mesh data.
  6. Lower Tester 2 expects the IUT to send a mesh message containing the reassembled message sent to the IUT in Step 1, the IUT performs segmentation as defined in test MESH/NODE/TNPT/BV-04-C [Send a Segmented Access Message – Unicast] if needed.
  7. Lower Tester 1 then sends another Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the first portion of segmented mesh data to the IUT.
  8. Lower Tester 1 waits for more than 20 seconds.
  9. Lower Tester 1 expects the IUT to terminate the connection.

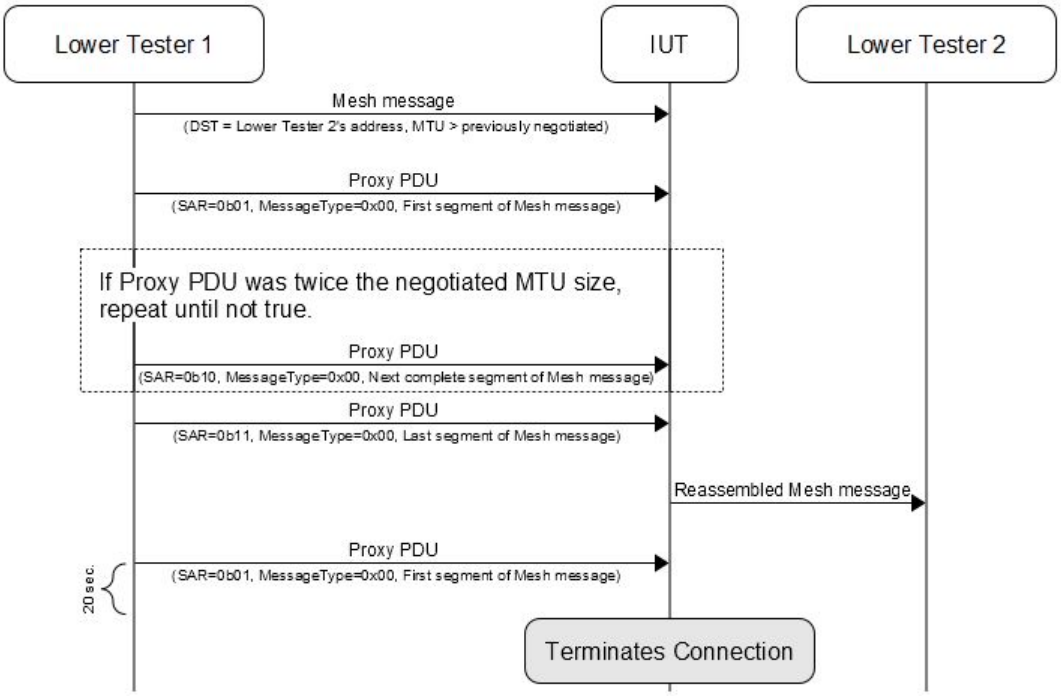


Figure 4.99: MESH/SR/PROX/BV-05-C [Send a Segmented Message Network PDU to Node from Proxy Client] MSC

- Expected Outcome

  Pass verdict

  The IUT sends the entire mesh message that Lower Tester 1 sends through Steps 1–4 to Lower Tester 2 in Step 6. In Step 9, the IUT terminates the connection.

MESH/SR/PROX/BV-06-C [Filter Accept List Addresses]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly filter messages for the appropriate accept list set.

- Reference

  [3] 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.

- The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].

- Test Procedure

  1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address equal to Lower Tester 1. The Proxy PDU does not require reassembly at the IUT.
  2. The IUT sends a Proxy PDU with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and Data field containing the complete mesh data to Lower Tester 1.
  3. Lower Tester 1 receives a mesh message from the IUT with the exact mesh data contained in the Proxy PDU sent from Lower Tester 2 to the IUT.
  4. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address that is not the address of the IUT and also not the address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  5. The IUT does not send Proxy PDUs with the mesh message from the previous step.

- Expected Outcome

  Pass verdict

  The IUT sends a single Proxy PDU to Lower Tester 1 with the SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and Data field containing the complete mesh data sent from Lower Tester 2 to the IUT in Step 2.

  The IUT does not send Proxy PDUs with MessageType equal to 0x00 to Lower Tester 1 in Step 5.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address 0x0000.

MESH/SR/PROX/BV-07-C [Receive Reject List Filter Setup]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can receive the Proxy Filter for setup of a reject list filter.

- Reference

  [3] 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - The Lower Tester is a Proxy Client.

- Test Procedure

1. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x01 (Reject List Filter).
2. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to [0].
3. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the Lower Tester.
4. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to [1].
5. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the Lower Tester.
6. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to [1].
7. The Lower Tester sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray field set with 1 address equal to that of the unassigned address.
8. The Lower Tester expects the IUT to send a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to [1].

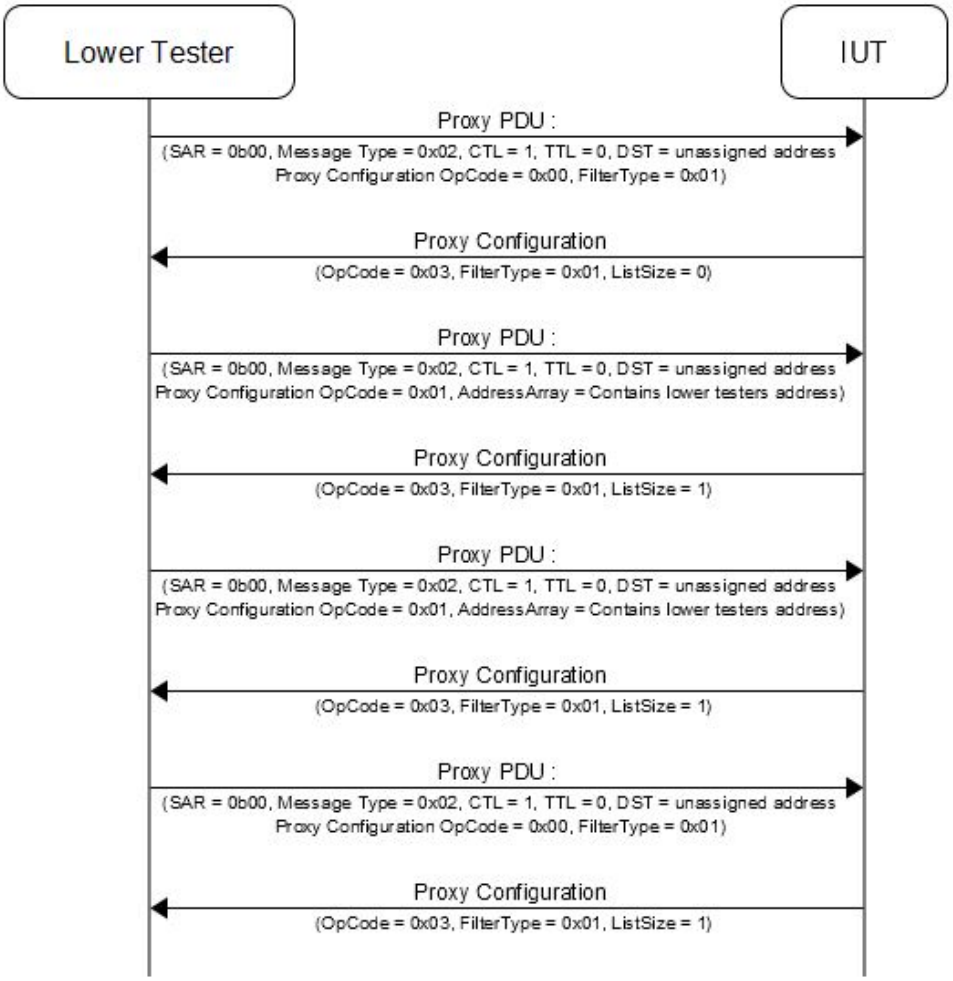


Figure 4.100: MESH/SR/PROX/BV-07-C [Receive Reject List Filter Setup] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with OpCode 0x03 "Filter Status", a FilterType field value of 0x01 (Reject List), and a ListSize field value of [0] in Step 2.

  The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Reject List), and a ListSize field value of [1] in Step 4.

  The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Reject List), and a ListSize field value of [1] in Step 6.

  The IUT sends a Proxy PDU with MessageType field equal to 0x02 "Proxy Configuration Message" with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Reject List), and a ListSize field value of [1] in Step 8.

Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

MESH/SR/PROX/BV-08-C [Filter Reject List Addresses]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly filter messages for the appropriate reject list set.

- Reference

  [3] 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition
  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - Test case MESH/SR/PROX/BV-07-C [Receive Reject List Filter Setup] has been executed.

- Test Procedure
  1. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  2. The IUT does not send Proxy PDUs with the mesh message from the previous step.
  3. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address that is not the address of the IUT but a valid mesh address. The Proxy PDU does not require reassembly at the IUT.
  4. The IUT sends a Proxy PDU with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x0 ("Network PDU"), and the complete mesh data to Lower Tester 1.
  5. Lower Tester 1 expects a Proxy PDU from the IUT with the exact data contained in the Proxy PDU sent from Lower Tester 2 to the IUT.

- Expected Outcome

  Pass verdict

  The IUT does not send Proxy PDUs with MessageType equal to 0x00 to Lower Tester 1 in Step 2.

  The IUT sends a single Proxy PDU to Lower Tester 1 with the exact data contained in the mesh message sent from Lower Tester 2 to the IUT in Step 3, except TTL will be decremented by 1.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the

unassigned address.

MESH/SR/PROX/BV-09-C [Remove Address from Accept List]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly remove addresses from the accept list address list.

- Reference

  [3] 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition

  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - The address of Lower Tester 1 has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].

- Test Procedure

  1. Lower Tester 1 sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses from Filter) and the AddressArray field set equal to Lower Tester 1.
  2. The IUT sends a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x00 (Accept List) and a ListSize equal to [0].
  3. Lower Tester 2 sends a single mesh message to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1. The mesh message does not require reassembly at the IUT.
  4. The IUT does not send Proxy PDUs with the mesh message from the previous step.

- Expected Outcome

  Pass verdict

  The IUT sends a single Proxy Configuration Message with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x00 (Accept List), and a ListSize field value of [0] in Step 2.

  The IUT does not send Proxy PDUs with MessageType equal to 0x00 to Lower Tester 1 in Step 4.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the unassigned address.

MESH/SR/PROX/BV-10-C [Remove Address from Reject List]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly remove addresses from the reject list address list.

- Reference

  [3] 6.2.1, 6.4, 6.4.1, 6.5, 6.6

- Initial Condition

  - The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network. Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
  - Test case MESH/SR/PROX/BV-07-C [Receive Reject List Filter Setup] has been executed right before this test.

- Test Procedure

  1. Lower Tester 1 sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses from Filter) and the AddressArray field set to the address of Lower Tester 1.
  2. Lower Tester 1 expects the IUT to send a Proxy Configuration Message of "Filter Status" with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to 0.
  3. Lower Tester 2 sends a mesh message to the IUT that will fit into one single Proxy PDU with the DST address set to Lower Tester 1’s address.
  4. Lower Tester 1 expects the IUT to send a Proxy PDU with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and the complete mesh data to Lower Tester 1.

- Expected Outcome

  Pass verdict

  The IUT sends a single Proxy Configuration Message with 0x03 "Filter Status" as the Proxy Configuration Status OpCode, a FilterType field value of 0x01 (Reject List), and a ListSize field value of 0 in Step 2.

  The IUT sends the entire mesh message from Lower Tester 2 to Lower Tester 1 in the Proxy PDU in Step 4, with TTL value decremented by 1.

  Each Proxy Configuration Messages from the IUT has a CTL field value set to 1, a TTL field value set to 0, SRC address equal to the unicast address of its primary element, and a DST field value equal to the

unassigned address.

4.14.3 Send Mesh Beacon When Proxy Connection Established

- Test Purpose

  Verify that the IUT can send Secure Network beacons or Mesh Private beacons when a proxy connection is established.

- Reference

  [3] 3.3.2, 6.2, 6.6, 6.7, 7.2.2

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Server.
  - The Lower Tester is a Proxy Client.
  - Additional conditions are specified in Table 4.34.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.34.

- Test Procedure

  1. The Lower Tester establishes a connection with the IUT.
  2. The IUT sends a beacon to the Lower Tester. The beacon has the Key Refresh Flag and IV Update Flag in the Flags field set to 0; other requirements for the beacon are described by the Additional Beacon Requirements column in Table 4.34.

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/SR/PROX/BV-11-C [Send Secure Network Beacon When Proxy Connection Established]	GATT Proxy state is 0x01. Secure Network Beacon state is 0x00.	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID and IV index fields.	0x01 (Secure Network beacon)

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/SR/PROX/BV-18-C [Send Mesh Private Beacon When Proxy Connection Established – using Private Network Identity]	The Proxy Privacy parameter is Enabled (0x01) as follows: GATT Proxy state is 0x00. Node Identity state is set to 0x00. Private Node Identity is 0x00. Private GATT Proxy state is 0x01. Additionally, the Secure Network Beacon state is set to 0x01, and the Private Beacon state is set to 0x00.	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)
MESH/SR/PROX/BV-19-C [Send Mesh Private Beacon When Proxy Connection Established – using Private Node Identity]	The Proxy Privacy parameter is Enabled (0x01) as follows: GATT proxy state is 0x00. Node Identity state is 0x00. Private GATT Proxy state is 0x00. Private Node Identity state is 0x01. Additionally, the Secure Network Beacon state is set to 0x01, and the Private Beacon state is set to 0x00.	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)

Table 4.34: Send Mesh Beacon When Proxy Connection Established test cases

- Expected Outcome

  Pass verdict

  The IUT sends a beacon to the Lower Tester after connecting.

  The Lower Tester receives valid message data from the IUT.

4.14.4 Send Mesh Beacon When IV Index Updated

- Test Purpose

  Verify that the IUT can send Secure Network beacons or Mesh Private beacons when the IV Index is updated.

- Reference

  [3] 3.3.2, 6.2, 6.6, 6.7, 7.2.2

- Initial Condition
  - The IUT is in a proxy connection with Lower Tester 1.
  - The IUT is the Proxy Server.
  - Lower Tester 1 is the Proxy Client.
  - Lower Tester 2 is another node in the mesh network.
  - The IV Update Test Mode is activated on the IUT.
  - Additional conditions are specified in Table 4.35.
  - The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.35.
- Test Procedure
  1. The Upper Tester orders the IUT to send a beacon to Lower Tester 1. The beacon has the Key Refresh Flag (Flags bit 0) and IV Update Flag (Flags bit 1) set to 0; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.35.
  2. Lower Tester 2 sends a beacon to the IUT, signaling an IV Index update. The beacon has the IV Update Flag (Flags bit 1) set to 1, and the IV Index is the new value used by the network.
  3. The IUT sends a beacon to Lower Tester 1 to relay the IV Index value update. The beacon has the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 1; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.35. Execute Steps 4 and 5 as indicated by the Steps 4 and 5 column in Table 4.35.
- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type	Steps 4 and 5
MESH/SR/PROX/BV-12-C [Send Secure Network Beacon When IV Index Updated]	The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).	The Network ID is set to the current network ID. In Step 1, the IV index field is set to the current IV index. In Step 3, the IV Index field is set to the new IV Index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID, and IV index fields.	0x01 (Secure Network beacon)	N/A

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type	Steps 4 and 5
MESH/SR/PROX/BV-20-C [Send Mesh Private Beacon When IV Index Updated]	The Proxy Privacy parameter is Enabled (0x01) as follows: GATT Proxy state is 0x00. Node Identity feature is either unsupported or supported, but Node Identity state is set to 0x00. Private GATT Proxy state is 0x01. Lower Tester 1 supports the Configuration Client Model.	In Step 1, the IV index field is set to the current IV index. In Step 3, the IV Index field is set to the new IV Index. The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)	1. Lower Tester 1 enables the GATT Proxy state by sending a Config GATT Proxy Set message with GATTProxy field set to 0x01. The IUT responds with a Config GATT Proxy Status message. 2. Repeat Steps 2 and 3, using IV Update Flag set to 0.

Table 4.35: Send Mesh Beacon When IV Index Updated test cases

- Expected Outcome

  Pass verdict

  The IUT sends a beacon to Lower Tester 1 after connecting.

  After receiving the IV Index update from Lower Tester 2, the IUT begins advertising beacons with the IV Update Flag set to 1.
  The Lower Tester receives both beacons with the correct values.

  If Steps 4 and 5 are executed as indicated by the Steps 4 and 5 column in Table 4.35, the IUT does not change the type of the beacon sent when the GATT Proxy state changes.

4.14.5 Send Mesh Beacon with Previous IV Index Value

- Test Purpose

  Verify that the IUT sends Secure Network beacons or Mesh Private beacons after receiving beacons with the previous IV Index value.

- Reference

  [3] 3.3.2, 6.2, 6.6, 6.7, 7.2.2

- Initial Condition

- The IUT has performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
- The IUT is the Proxy Server.
- Lower Tester 1 is a Proxy Client.
- Lower Tester 2 is a node in the mesh network.
- Additional conditions are specified in Table 4.36.
- The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.36.

• Test Procedure

1. The Upper Tester orders the IUT to send beacons. The beacon has the Key Refresh Flag (Flags bit 0) and IV Update Flag (Flags bit 1) disabled; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.36.
2. Lower Tester 1 receives the beacon advertised by the IUT and validates the data in its fields.
3. Lower Tester 2 sends a beacon to the IUT with the IV Index value set to the previous IV Index (e.g., if the current value is 0x00000002, then the beacon sent by Lower Tester 2 contains 0x00000001).
4. The Upper Tester orders the IUT to send beacons.
5. The IUT continues advertising the beacon with the current IV Index value and all values as listed in Step 1.
6. Lower Tester 1 receives the beacon from the IUT and validates the data in its fields.
7. Lower Tester 1 sends a single Mesh Data Proxy Message to the IUT smaller than the negotiated MTU. The Proxy Message does not require reassembly at the IUT. The IV Index of the Proxy PDU is equal to the current IV Index value - 1.
8. The IUT forwards this message to Lower Tester 2 over the advertising bearer.
9. Lower Tester 2 receives a message from the IUT with the exact data contained in the Proxy Message sent from Lower Tester 1 to the IUT.

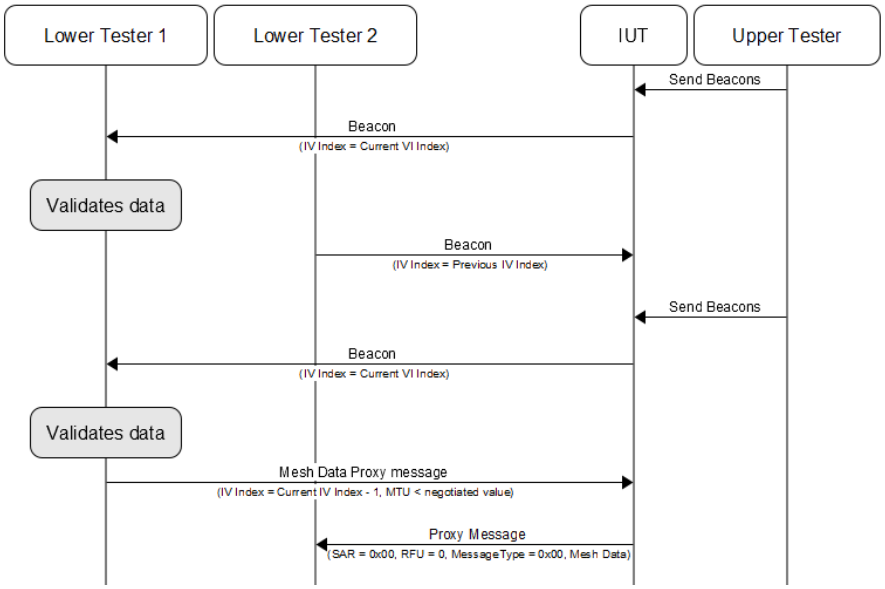


Figure 4.101: MESH/SR/PROX/BV-13-C and MESH/SR/PROX/BV-21-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Additional Beacon Requirements	Beacon Type
MESH/SR/PROX/BV-13-C [Send Secure Network Beacon with Previous IV Index Value]	The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID, and IV index fields.	0x01 (Secure Network beacon)
MESH/SR/PROX/BV-21-C [Send Mesh Private Beacon with Previous IV Index Value]	The Proxy Privacy parameter is Enabled (0x01) as follows: GATT Proxy state is 0x00. Node Identity feature is either unsupported or supported, but Node Identity state is set to 0x00. Private GATT Proxy state is 0x01.	The IV index field is set to the current IV index. The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)

Table 4.36: Send Mesh Beacon with Previous IV Index Value test cases

- Expected Outcome

  Pass verdict

  The IUT sends a beacon to Lower Tester 1 after connecting.

  The IUT continues sending beacons with the current IV Index after receiving a beacon from Lower Tester 2 with the previous IV Index.
  Lower Tester 1 receives each of the beacons from the IUT with the correct values.

  The IUT transmits the Proxy Messages sent by Lower Tester 1 to Lower Tester 2.

4.14.6 IV Update Initiated by Proxy Client

- Test Purpose

  Verify that an IUT acting as Proxy Server can process an IV Update initiated by a Proxy Client.

- Reference

  [3] 3.3.2, 3.10.5, 3.10.5.1, 6.6

- Initial Condition
  - The IUT is in a proxy connection with Lower Tester 1.
  - The IUT is the Proxy Server.
  - Lower Tester 1 is the Proxy Client.
  - Lower Tester 2 is another node in the mesh network.
  - The IV Update Test Mode is activated on the IUT.
  - Additional conditions are specified in Table 4.37.
  - The Mesh Beacon used in the test procedure by each role is specified in Table 4.37.

- Test Procedure

  1. Lower Tester 2 generates a beacon using the same network credentials and current IV Index as the IUT and begins advertising this beacon periodically. The IV Update Flag and the Key Refresh Flag are set to 0.
  2. Lower Tester 2 expects to receive beacons with the same values from the IUT.
  3. Lower Tester 1 increases its IV Index by one, then sends a Proxy PDU containing a new beacon with the new IV Index value, the IV Update Flag set to 1 and the Key Refresh Flag set to 0.
  4. Lower Tester 2 expects to receive the new beacon containing the increased IV Index value and the IV Update Flag set to 1.
  5. Lower Tester 1 sends a Proxy PDU containing a new beacon with the new IV Index value, the IV Update Flag set to 0 and the Key Refresh Flag set to 0.
  6. Lower Tester 2 expects to receive the new beacon containing the increased IV Index value and the IV Update Flag set to 0.

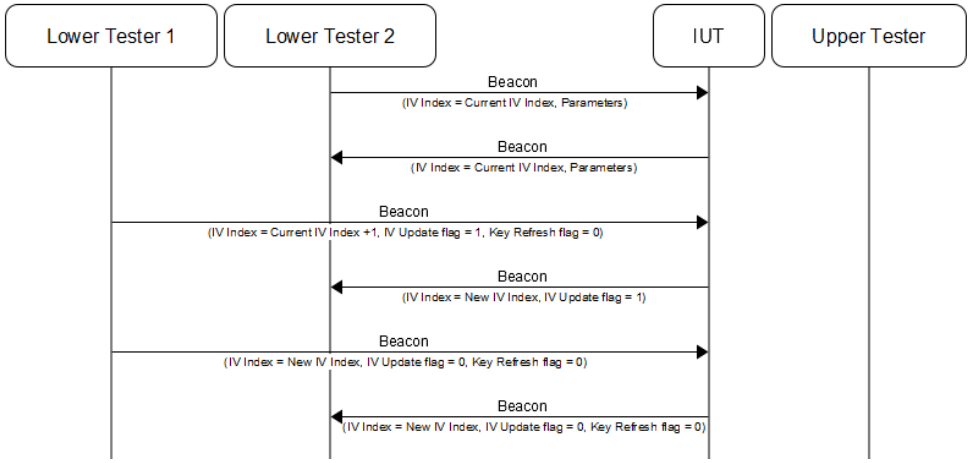


Figure 4.102: MESH/SR/PROX/BV-14-C and MESH/SR/PROX/BV-22-C MSC

- Test Case Configuration

Test Case	Additional Initial Condition	Lower Tester 1 Beacon Type	IUT Beacon Type	Lower Tester 2 Beacon Type
MESH/SR/PROX/BV-14-C [IV Update Initiated by Proxy Client using Secure Network Beacons]	The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).	0x01 (Secure Network beacon)	0x01 (Secure Network beacon)	0x01 (Secure Network beacon)
MESH/SR/PROX/BV-22-C [IV Update Initiated by Proxy Client using Mesh Private Beacons]	Either GATT Proxy state is 0x01 or Private GATT Proxy state is 0x01.	0x02 (Mesh Private beacon)	0x01 (Secure Network beacon) or 0x02 (Mesh Private beacon)	0x01 (Secure Network beacon) or 0x02 (Mesh Private beacon)

Table 4.37: IV Update Initiated by Proxy Client test cases

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends the same beacon as Lower Tester 2.

  In Step 4, the IUT sends the new beacon with the increased IV Index and the IV Update Flag set to 1.

  In Step 6, the IUT sends the new beacon with the increased IV Index and the IV Update Flag set to 0.

MESH/SR/PROX/BV-15-C [Disable Network PDU Relay when Proxy Feature is Disabled]

- Test Purpose

  Verify that an IUT that supports the Proxy feature but that has the feature disabled role does not relay any Network PDU to or from the Proxy Client.

- Reference

  [3] 3.4.6.3

- Initial Condition
  - The IUT has just performed a proxy connection to Lower Tester 1 through a connection-oriented channel from another test case.
  - The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
  - Lower Tester 1 is a Proxy Client.
  - Lower Tester 2 is a node on the mesh network and is provisioned to the same network as Lower Tester 1 and the IUT.
  - The all-nodes fixed group address has been added to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].
- Test Procedure
  1. Lower Tester 2 sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00 ("Proxy Feature is supported and disabled").
  2. Lower Tester 2 expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of ("Proxy Feature is supported and disabled").
  3. Lower Tester 2 sends an access message as an Unsegmented Access Message, smaller than the negotiated MTU, with the destination address set to 0xFFFF.
  4. Lower Tester 1 does not expect any Proxy PDU from the IUT.
  5. The Upper Tester expects an indication from the IUT containing the same access message data sent by Lower Tester 2 in Step 3.
  6. Lower Tester 1 sends an access message as an Unsegmented Access Message, smaller than the negotiated MTU, with the destination address set to 0xFFFF.
  7. Lower Tester 2 does not expect any Proxy PDU from the IUT.
  8. The Upper Tester expects an indication from the IUT containing the same access message data sent by Lower Tester 1 in Step 6.

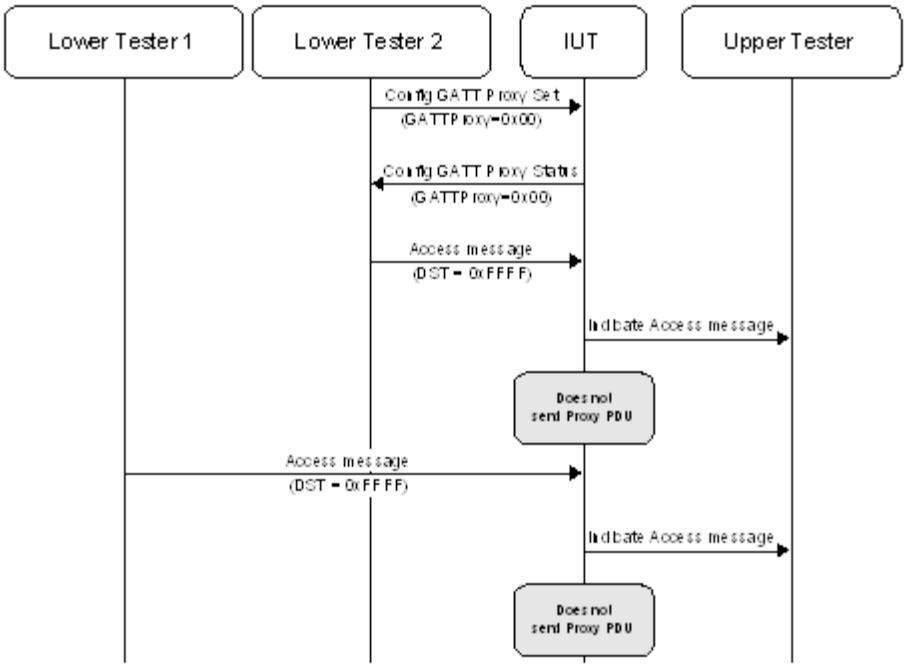


Figure 4.103: MESH/SR/PROX/BV-15-C [Disable Network PDU Relay when Proxy Feature is Disabled] MSC

- Expected Outcome

  Pass verdict

  The IUT does not send any Proxy PDU to Lower Tester 1 or Lower Tester 2.

  The IUT indicates to the Upper Tester that it has received the access messages from Lower Tester 1 and Lower Tester 2.

4.14.7 Send Mesh Beacon when Secondary NetKey Added

- Test Purpose

  Verify that the IUT sends Secure Network beacons or Mesh Private beacons when it gets added to a secondary subnet.

- Reference

  [3] 3.3.2, 6.2, 6.6, 6.7, 7.2.2

- Initial Condition
  - The IUT is in a proxy connection with Lower Tester 1.
  - The IUT is the Proxy Server.
  - Lower Tester 1 is the Proxy Client.
  - Lower Tester 2 is a Configuration Client.

- Additional conditions are specified in Table 4.38.
- The Mesh Beacon used in the test procedure is described by the Beacon Type column in Table 4.38.
- Lower Tester 1 is connected to the IUT.

- Test Procedure

  1. Lower Tester 2 executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The IUT sends a beacon to Lower Tester 1. The beacon has the Key Refresh Flag (Flags bit 0) set to 0 and the IV Update Flag (Flags bit 1) set to 0 and is secured using the new key.

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/SR/PROX/BV-23-C [Send Secure Network Beacon when Secondary NetKey Added]	The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).	0x01 (Secure Network beacon)
MESH/SR/PROX/BV-24-C [Send Mesh Private Beacon when Secondary NetKey Added]	The Proxy Privacy parameter is Enabled (0x01) as follows: GATT Proxy state is 0x00. Node Identity feature is either unsupported or supported, but Node Identity state is set to 0x00. Private GATT Proxy state is 0x01. Lower Tester 1 supports the Configuration Client Model.	0x02 (Mesh Private beacon)

Table 4.38: Send Mesh Beacon When Secondary NetKey Added test cases

- Expected Outcome

  Pass verdict

  The IUT sends a beacon to Lower Tester 1 after Lower Tester 2 sends the new NetKey to the IUT. The beacon is secured with the new NetKey.

MESH/SR/PROX/BI-01-C [Ignore Invalid Message Type]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Server role can properly handle a Proxy PDU with an invalid Message PDU Type.

- Reference

  [3] 6.2, 6.5, 6.6

- Initial Condition

- The IUT has just performed a proxy connection to Lower Tester 1 through a connection oriented channel from another test case.
- The IUT is the Proxy Server. The Mesh Proxy Service is running on the IUT (Proxy state value is 0x01).
- Lower Tester 1 is a Proxy Client.
- Lower Tester 2 is a node on the mesh network and provisioned to the same network as Lower Tester 1 and the IUT.
- There have been no Addresses added to any Proxy Address List.

- Test Procedure

  1. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated MTU with the destination address of Lower Tester 1 and the "MessageType" field of the Proxy PDU set to a random RFU value.
  2. The IUT ignores the Proxy PDU from Lower Tester 2 and does not send any mesh message or Proxy PDU in response.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any mesh message or Proxy PDU in response.

MESH/CL/PROX/BV-01-C [Receive Complete Message from Proxy Server]

- Test Purpose

  Verify that an IUT can receive a message from a Proxy Server where the complete message fits into one PDU.

- Reference

  [3] 6.2, 6.3, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - The Lower Tester is a Proxy Server.
  - The IUT has a mechanism to add its own unicast address to the Proxy accept list.

- Test Procedure

  1. The Lower Tester sends a single mesh message to the IUT smaller than the negotiated MTU and with the DST address of the IUT. The Proxy PDU does not require reassembly at the IUT with SAR field set to 00 ("Complete higher level message"), MessageType equal to 0x00 ("Mesh Data"), and Data field contains the complete mesh data to the IUT.

2. IUT receives a Proxy PDU from the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT receives the Proxy PDU and reports to the Upper Tester that it has received the mesh message contained in the Proxy PDU from the Lower Tester.

MESH/CL/PROX/BV-02-C [Receive Segmented Message from Proxy Server]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can receive a message from a Proxy Server where the complete message does not fit into one PDU.

- Reference

  [3] 6.2, 6.3, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - The Lower Tester is a Proxy Server.

- Test Procedure

  1. The Lower Tester sends a Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the first segment of the mesh data to the IUT.
  2. The Lower Tester then sends a Proxy PDU to the IUT with SAR field set to b11 ("Data field contains the last segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the last segment of the mesh data to the IUT.
  3. The Lower Tester then sends another Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the first segment of the mesh data to the IUT.
  4. The Lower Tester then waits greater than 20 seconds.
  5. The Lower Tester expects the IUT to terminate the connection.

- Expected Outcome

  Pass verdict

  The IUT receives all Proxy PDUs from the Lower Tester and reports to the Upper Tester that it has received the complete mesh message from the Lower Tester.

  In Step 5, the IUT terminates the connection.

MESH/CL/PROX/BV-03-C [Send Complete Message to Proxy Server]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a message to a Proxy Server where the complete message fits into one PDU.

- Reference

  [3] 6.2, 6.3, 6.7

- Initial Condition
  - The The IUT has just performed a proxy connection to the Lower Tester through a connection-oriented channel from another test case.
  - The IUT is the Proxy Client.
  - The Lower Tester is a Proxy Server.
  - There have been no Addresses added to any Proxy Address Filter List.
  - The IUT has a method to initiate sending a non-segmented message to the Lower Tester.
- Test Procedure
  1. The Upper Tester orders the IUT to send a single Proxy PDU to the Lower Tester smaller than the negotiated MTU with the destination address of the Lower Tester. The Proxy PDU does not require reassembly.
  2. The Lower Tester receives the Proxy PDU from the IUT.
- Expected Outcome

  Pass verdict

  The IUT sends a single Proxy PDU to the Lower Tester with SAR field set to b00 ("Complete higher level message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the complete mesh data.

MESH/CL/PROX/BV-04-C [Send Segmented Message to Proxy Server]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a message to a Proxy Server where the complete message does not fit into one PDU.

- Reference

  [3] 6.2, 6.3, 6.7

- Initial Condition
  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.

- The IUT is the Proxy Client.
- The Lower Tester is a Proxy Server.
- There have been no Addresses added to any Proxy Address List.

- Test Procedure

1. The Upper Tester orders the IUT to send a mesh message to the Lower Tester through the Proxy PDU to the IUT which has a larger payload size than the negotiated MTU for that connection and with the destination address of the Lower Tester.
2. The IUT sends a Proxy PDU to the IUT with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the first segment of the mesh data.
3. If the message size in Step 1 was greater than 2 times the negotiated MTU for the Proxy connection then the IUT sends a Proxy PDU to the IUT with SAR field set to b10 ("Data field contains the next segment of a message"), MessageType equal to 0x00 ("Network PDU"), and Data field contains the next segment of the mesh data.
4. Step 3 is repeated until one before the last segment is received.
5. If the message size in Step 1 was not greater than 2 times the negotiated MTU for that Proxy connection then Steps 3 and 4 will not occur and the IUT then sends a Proxy PDU to the Lower Tester with SAR field set to b11 ("Data field contains the last segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the last part of the mesh data to the Lower Tester.
6. The Lower Tester receives all Proxy PDUs from the IUT with the exact data contained in the entire message.

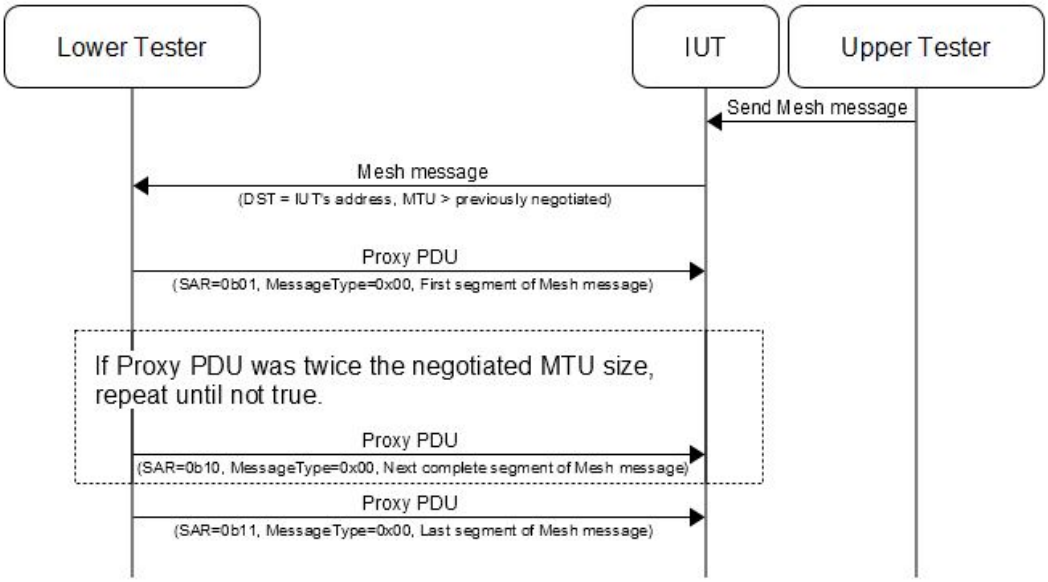


Figure 4.104: MESH/CL/PROX/BV-04-C [Send Segmented Message to Proxy Server] MSC

- Expected Outcome

  Pass verdict

  The IUT sends the Proxy PDUs in Steps 2 with SAR field set to b01 ("Data field contains the first segment of a message"), MessageType equal to 0x0 ("Network PDU"), and the Data field containing first segment of the mesh message.

  The IUT sends all the Proxy PDUs in Steps 3–4 with SAR field set to b10 ("Data field contains the next segment of a message"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the next of the mesh message.

  The IUT sends the last Proxy PDUs in Step 4 with SAR field set to b11 ("Data field contains the last segment of a message"), RFU set to zero, MessageType equal to 0x0 ("Network PDU"), MessageType equal to 0x00 ("Network PDU"), and the Data field contains the final segment of the mesh message.

  All the Proxy PDU data from the mesh message is reassembled.

MESH/CL/PROX/BV-05-C [Send Set Filter Type Proxy Configuration Message]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a Set Filter Type Proxy Configuration Message.

- Reference

  [3] 6.2, 6.3, 6.5, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel

from another test case.

- The IUT is the Proxy Client.
- The Lower Tester is a Proxy Server.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Set Filter Type Proxy PDU to the Lower Tester.
  2. The Lower Tester expects to receive a Set Filter Type Proxy PDU with the DST field with the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (Accept List Filter) or 0x01 (Reject List Filter).

- Expected Outcome

  Pass verdict

  The IUT sends a Set Filter Type Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x00 (Accept List Filter) or 0x01 (Reject List Filter).

MESH/CL/PROX/BV-06-C [Send Add Addresses to Filter Proxy Configuration Message]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send an Add Addresses to Filter Proxy Configuration Message.

- Reference

  [3] 6.2, 6.3, 6.4, 6.5, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - The Lower Tester is a Proxy Server.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Proxy Configuration message with OpCode 0x01 (Add Addresses to Filter) to the Lower Tester.
  2. The Lower Tester expects to receive an Add Addresses to Proxy PDU with the DST field set to the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray equal to n * length of an address.

- Expected Outcome

  Pass verdict

  The IUT sends an Add Addresses Filter Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x01 (Add Addresses to Filter) and the AddressArray equal to n * length of an address.

MESH/CL/PROX/BV-07-C [Send Remove Addresses from Filter Proxy Configuration Message]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol and the Proxy Client role can send a Remove Addresses from Filter Proxy Configuration Message.

- Reference

  [3] 6.2, 6.3, 6.4, 6.5, 6.7

- Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.
  - The Lower Tester is a Proxy Server.
- Test Procedure
  1. The Upper Tester orders the IUT send a Proxy Configuration message with OpCode 0x02 (Remove Addresses from Filter) to the Lower Tester.
  2. The Lower Tester expects to receive a Remove Addresses to Proxy PDU with the DST field set to the unassigned address, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses to Filter) and the AddressArray equal to n * length of an address.
- Expected Outcome

  Pass verdict

  The IUT sends a Remove Addresses Filter Proxy PDU with the DST field set to the unassigned address, SRC address field set to the unicast address of its primary element, SEQ field set to the sequence number of its primary element, a CTL field value set to 1, and a TTL field value set to 0. The Proxy Configuration OpCode field value will be set to 0x02 (Remove Addresses to Filter) and the AddressArray equal to n * length of an address.

4.14.8 Receive Mesh Beacon When Proxy Connection Established

- Test Purpose

  Verify that the Proxy Client IUT can receive Secure Network beacons or Mesh Private beacons when a proxy connection is established.

- Reference

  [3] 3.3.2, 6.2, 6.7

- Initial Condition

  - The IUT is a Proxy Client.
  - The Lower Tester is a Proxy Server.
  - The Mesh Beacon Type used in the test procedure is specified in Table 4.39.

- Test Procedure

  1. The Upper Tester orders the IUT to connect to the Lower Tester.
  2. The Lower Tester sends a beacon to the IUT. The beacon has the Key Refresh Flag and IV Update Flag set to 0; other requirements for the beacon are described by the Additional Initial Condition column in Table 4.39.

- Test Case Configuration

Test Case	Additional Initial Condition	Beacon Type
MESH/CL/PROX/BV-08-C [Receive Secure Network Beacon When Proxy Connection Established]	The Network ID is set to the current network ID. The IV index field is set to the current IV index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID, and IV index fields.	0x01 (Secure Network beacon)
MESH/CL/PROX/BV-11-C [Receive Mesh Private Beacon When Proxy Connection Established]	The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)

Table 4.39: Receive Mesh Beacon When Proxy Connection Established test cases

- Expected Outcome

  Pass verdict

  The IUT receives valid beacon data from the Lower Tester.

4.14.9 Receive Mesh Beacon When IV Index Updated

- Test Purpose

Verify that the IUT can receive Secure Network beacons or Mesh Private beacons when the IV Index is updated.

- Reference

  [3] 3.3.2, 6.2, 6.7

- Initial Condition
  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is a Proxy Client.
  - The Lower Tester is a Proxy Server.
  - The Mesh Beacon Type used in the test procedure is specified in Table 4.40.
- Test Procedure
  1. The Lower Tester sends a beacon to the IUT. The beacon has the Key Refresh Flag and IV Update Flag set to 0; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.40. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.
  2. The Lower Tester sends a beacon to the IUT to indicate the IV Index value update. The beacon has the Key Refresh bit set to 0 and the IV Update bit set to 1 in the Flags field; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.40. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.
  3. The IUT clears its oldest previously accepted IV index and saves a replay protection list for the new IV index.
- Test Case Configuration

Test Case	Additional Beacon Requirements	Beacon Type
MESH/CL/PROX/BV-09-C [Receive Secure Network Beacon When IV Index Updated]	The Network ID is set to the current network ID. In Step 1, the IV index field is set to the current IV index. In Step 2, the IV index field is set to the new IV index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID, and IV index fields.	0x01 (Secure Network beacon)

Test Case	Additional Beacon Requirements	Beacon Type
MESH/CL/PROX/BV-12-C [Receive Mesh Private Beacon When IV Index Updated]	In Step 1, the IV index field is set to the current IV index. In Step 2, the IV index field is set to the new IV index. The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)

Table 4.40: Receive Mesh Beacon When IV Index Updated test cases

- Expected Outcome

  Pass verdict

  After receiving the IV Index update from the Lower Tester, the IUT updates its IV Index.

4.14.10 Receive Mesh Beacon with Previous IV Index Value

- Test Purpose

  Verify that the Proxy Client IUT behaves appropriately when receiving Secure Network beacons or Mesh Private beacons with a previous IV Index value.

- Reference

  [3] 3.3.2, 6.2, 6.7

- Initial Condition

  - The IUT has just performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is a Proxy Client.
  - The Lower Tester is a Proxy Server.
  - The Mesh Beacon Type used in the test procedure is specified in Table 4.41.

- Test Procedure

  1. The Lower Tester sends a beacon to the IUT. The beacon has the Key Refresh bit and IV Update bit of the Flags field set to 0; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.41. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.
  2. The Lower Tester sends a beacon to the IUT to indicate an IV Index value update. The beacon has the Key Refresh bit set to 0 and the IV Update bit set to 1 in the Flags field; other beacon requirements are described by the Additional Beacon Requirements column in Table 4.41. The AD Type of the beacon is set to «Mesh Beacon» and the Beacon Type value is set correctly.
  3. The Lower Tester sends a single Mesh Data Proxy Message to the IUT smaller than the negotiated MTU. The Proxy Message does not require reassembly at the IUT. The IV Index of the Mesh Packet is

equal to the current IV Index value - 1.

- Test Case Configuration

Test Case	Additional Beacon Requirements	Beacon Type
MESH/CL/PROX/BV-10-C [Receive Secure Network Beacon with Previous IV Index Value]	The Network ID is set to the current network ID. In Step 1, the IV index field is set to the current IV index. In Step 2, the IV index field is set to the previous IV index. The Authentication Value is computed using the NetKey, based on the Flags, Network ID, and IV index fields.	0x01 (Secure Network beacon)
MESH/CL/PROX/BV-13-C [Receive Mesh Private Beacon with Previous IV Index Value]	In Step 1, the IV index field is set to the current IV index. In Step 2, the IV index field is set to the previous IV index. The Private Beacon Flags and IV index fields are obfuscated and authenticated based on the value of the Random field using the NetKey.	0x02 (Mesh Private beacon)

Table 4.41: Receive Mesh Beacon with Previous IV Index Value test cases

- Expected Outcome

  Pass verdict

  The IUT receives the packet with the previous IV Index from the Lower Tester.

MESH/CL/PROX/BI-01-C [Ignore Invalid Message Type]

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature and the Proxy Client role can properly handle a Proxy PDU with an invalid Message PDU Type.

- Reference

  [3] 6.2, 6.7

- Initial Condition
  - The IUT has performed a proxy connection to the Lower Tester through a connection oriented channel from another test case.
  - The IUT is the Proxy Client.

- The Lower Tester is a Proxy Server.

- Test Procedure

  1. The Lower Tester sends a single Proxy PDU to the IUT smaller than the negotiated MTU and with the DST address of IUT. The Proxy PDU does not require reassembly at the IUT with SAR field set to 00 ("Complete higher level message"), RFU set to zero, Type equal to a random RFU value.
  2. IUT ignores the Proxy PDU due to the invalid MessageType.

- Expected Outcome

  Pass verdict

  IUT ignores the Proxy PDU due to the invalid MessageType.

4.14.11 Ignore RFU Proxy configuration message

- Test Purpose

  Verify that an IUT that supports the Proxy Protocol feature can properly handle a Proxy configuration message with an RFU opcode.

- Reference

  [3] 6.5, 6.6, 6.7

- Initial Condition

  - The IUT has performed a proxy connection to the Lower Tester through a connection-oriented channel from another test case.
  - Additional initial conditions defined in Table 4.42 apply.

- Test Procedure

  1. The Lower Tester sends a Proxy configuration message to the IUT smaller than the negotiated MTU and with the DST address of the IUT. The Proxy PDU does not require reassembly at the IUT with the SAR field set to 00 ("Complete higher level message"). Opcode field is set to the first RFU value defined by the specification version.
  2. The IUT ignores the Proxy configuration message.

- Test Case Configuration

Test Case	Additional Initial Condition
MESH/SR/PROX/BI-02-C [Ignore RFU Proxy configuration message]	The IUT is a Proxy Server.
MESH/CL/PROX/BI-02-C [Ignore RFU Proxy configuration message]	The IUT is a Proxy Client.

Table 4.42: Additional test conditions for "Ignore RFU Proxy configuration message"

- Expected Outcome

  Pass verdict

The IUT ignores the Proxy configuration message.

The IUT does not terminate the proxy connection with the Lower Tester.

4.15 Mesh Proxy Service

The test group objective is to verify the Mesh Proxy Service.

MESH/SR/MPXS/BV-01-C [Advertise Mesh Proxy Service with Network ID]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with the Network ID.

- Reference

  [3] 6.2, 7.2.2.1, 7.2.2.2, 7.2.2.2.1

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a GAP Observer.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Network ID.
  2. The Lower Tester expects the IUT to send Advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 9 octets of service data.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x00, and the remaining 8 octets are equal to the Network ID derived with the k3 function from the shared network key. The Flags AD indicates that the IUT is in the GAP General Discoverable Mode.

MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Service in the GATT

database.

- Reference

  [3] 6.2, 7.2.2.1, 7.2.2.2

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT and the Lower Tester support GAP Peripheral and GAP Central roles, respectively.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
- Test Procedure
  1. The Lower Tester executes either the procedure in GATT.TS [5] Discover Primary Services by Service UUID - From Server, GATT/SR/GAD/BV-02-C, with the Service UUID of «Mesh Proxy Service» or the procedure in GATT.TS [5] Discover All Primary Services - from Server, GATT/SR/GAD/BV-01-C.
  2. The Lower Tester executes either the procedure in GATT.TS [5] Discover Primary Services by Service UUID - From Server, GATT/SR/GAD/BV-02-C, with the Service UUID of «Mesh Provisioning Service» or the procedure in GATT.TS [5] Discover All Primary Services - from Server, GATT/SR/GAD/BV-01-C.
- Expected Outcome

  Pass verdict

  In Step 1, the IUT returns a single attribute handle range. The starting handle and the ending handle correspond to the Mesh Proxy Service definition in the GATT database of the IUT.

  In Step 2, the IUT returns no instance of the Mesh Provisioning Service.

MESH/SR/MPXS/BV-03-C [Expose Mesh Proxy Data Out Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data Out Characteristic in the GATT database.

- Reference

  [3] 7.2.1, 7.2.2, 7.2.3.2

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.

- The procedure described in MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Service.

- Test Procedure

  The Lower Tester executes either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the handle range of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the handle range of the Mesh Proxy Service and UUID set to «Mesh Proxy Data Out».

- Expected Outcome

  Pass verdict

  The IUT returns a single characteristic with the UUIT equal to «Mesh Proxy Data Out» in the specified handle range. The characteristic properties value has the Notifications flag set to 1.

MESH/SR/MPXS/BV-04-C [Expose Mesh Proxy Data Out Client Characteristic Configuration Descriptor]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data Out Client Characteristic Configuration descriptor in the GATT database.

- Reference

  [3] 7.2.1, 7.2.2, 7.2.3.2

- Initial Condition

  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-03-C [Expose Mesh Proxy Data Out Characteristic] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Data Out characteristic.

- Test Procedure

  The Lower Tester executes one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors - from Server, GATT/SR/GAD/BV-06-C using the handle range of the Mesh Proxy Data Out characteristic.

- Expected Outcome

  Pass verdict

  The IUT returns a descriptor with the UUID equal to «Client Characteristic Configuration» in the specified handle range.

MESH/SR/MPXS/BV-05-C [Support Notifications for Mesh Proxy Data Out Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role supports GATT Notifications on the Mesh Proxy Data Out characteristic.

- Reference

  [3] 7.2.1, 7.2.2, 7.2.3.2

- Initial Condition

  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-04-C [Expose Mesh Proxy Data Out Client Characteristic Configuration Descriptor] has been executed and the Lower Tester retained the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor.

- Test Procedure

  1. The Lower Tester enables notifications on the Mesh Proxy Data Out by executing the preamble procedure in GATT.TS [5] Section 4.2.2.1, using the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor.
  2. The Lower Tester expects the IUT to send an ATT Handle Value Notification packet containing a Secure Network beacon.

- Expected Outcome

  Pass verdict

  In Step 1, the IUT accept the procedure and enables the notifications on the Mesh Proxy Data Out characteristic.

  In Step 2, the IUT sends an ATT Handle Value Notification packet with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value, and the Attribute Value field containing a Secure Network beacon with correct format and valid field values.

MESH/SR/MPXS/BV-06-C [Expose Mesh Proxy Data In Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can expose the Mesh Proxy Data In Client Characteristic Configuration descriptor in the GATT database.

- Reference

  [3] 7.2.1, 7.2.2, 7.2.3.1

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-02-C [Expose Mesh Proxy Service] has been executed and the Lower Tester retained the handle range of the Mesh Proxy Service.

- Test Procedure

  The Lower Tester executes either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the handle range of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/CL/GAD/BV-05-C, with the handle range of the Mesh Proxy Service and UUID set to «Mesh Proxy Data In».

- Expected Outcome

  Pass verdict

  The IUT returns a single characteristic with the UUIT equal to «Mesh Proxy Data In» in the specified handle range. The characteristic properties value has the Write Without Response flag set to 1.

MESH/SR/MPXS/BV-07-C [Support Write Without Response to Mesh Proxy Data In Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role supports GATT Write Without Response on the Mesh Proxy Data In characteristic.

- Reference

  [3] 7.2.1, 7.2.2, 7.2.3.1

- Initial Condition
  - The IUT is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT and the Lower Tester share previously established network security credentials.
  - Execute the steps in setup preamble 4.2.5 to establish an ATT Bearer connection between the IUT and the Lower Tester.
  - The procedure described in MESH/SR/MPXS/BV-06-C [Expose Mesh Proxy Data In Characteristic] has been executed and the Lower Tester retained the handle of the Mesh Proxy Data In characteristic

value.

- The procedure described in MESH/SR/MPXS/BV-05-C [Support Notifications for Mesh Proxy Data Out Characteristic] has been executed.

- Test Procedure

  1. The Lower Tester sends an ATT Write Command packet with the Attribute Handle field set to the handle of the Mesh Proxy Data In characteristic value, and the Attribute Value set to a properly formatted Set Filter Type proxy configuration message.
  2. The Lower Tester expects the IUT to respond with an ATT Handle Value Notification packet with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value, and the Attribute Value field set to a Filter Status proxy configuration message with correct format and parameters.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT responds with a Filter Status message, confirming that it has received the ATT Write Command packet from the Lower Tester in Step 1.

MESH/SR/MPXS/BV-09-C [Advertise Mesh Proxy Service with Node Identity]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with Node Identity.

- Reference

  [3] 7.2.1, 7.2.2

- Initial Condition

  - The IUT exposes the Mesh Proxy Service.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a GAP Observer.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Node Identity.
  2. The Lower Tester expects the IUT to send advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 17 octets of service data.
  3. The Lower Tester does not connect to the IUT, and the IUT stops advertising after maximum 60 seconds.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x01, the next 8 octets contain a Hash value properly computed over the random value with the IdentityKey of the IUT, and the last 8 octets contain the random value used to compute the hash.

  The IUT stops advertising no later than 60 seconds after it has started.

MESH/SR/MPXS/BV-10-C [Advertise Mesh Proxy Service with Private Network Identity]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with the Private Network Identity.

- Reference

  [3] 7.2.2

- Initial Condition

  - The IUT is a GATT Bearer Server and has the Private GATT Proxy State set to 0x01 and the GATT Proxy State set to 0x00.
  - The Lower Tester is a GATT Bearer Client.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a GAP Observer.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Private Network Identity.
  2. The Lower Tester expects the IUT to send Advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 17 octets of service data.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x02, the next 8 octets contain a Hash value properly computed over the random and Network ID values with the IdentityKey, and the last 8 octets contain the random value

used to compute the Hash. The Flags AD indicates that the IUT is in the GAP General Discoverable Mode.

MESH/SR/MPXS/BV-11-C [Advertise Mesh Proxy Service with Private Node Identity]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Server role can properly advertise the Mesh Proxy Service with Private Node Identity.

- Reference

  [3] 7.2.2

- Initial Condition
  - The IUT exposes the Mesh Proxy Service.
  - The IUT supports the GAP Broadcaster role.
  - The Lower Tester is a GAP Observer.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT has the Private Node Identity state set to 0x01 and the Node Identity state set to 0x00.
- Test Procedure
  1. The Upper Tester orders the IUT to start advertising the Mesh Proxy Service with Private Node Identity.
  2. The Lower Tester expects the IUT to send advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 17 octets of service data.
- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends advertising packets with the Advertising Data field containing the required AD Types, formatted correctly and including the «Mesh Proxy Service» and the 17-octet service data. The first octet of the service data is equal to 0x03, the next 8 octets contain a Hash value properly computed over a Random value and the IUT address with the IdentityKey, and the last 8 octets contain the random value used to compute the Hash.

MESH/SR/MPXS/BV-12-C [Respond to Private GATT Proxy Solicitation]

- Test Purpose

  Verify that an On-demand Proxy Server IUT can respond correctly to Private GATT Proxy Solicitation based on the On-Demand Private GATT Proxy state value.

- Reference

  [3] 4.2.11, 4.2.22, 4.4.5, 4.2.41, 4.4.11, 4.4.12

- Initial Condition
  - The IUT supports the On-demand Private Proxy Server model on the Primary Element.
  - The Lower Tester supports the On-demand Private Proxy Client Model.
  - The IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
  - The Proxy and Private Proxy functionalities are disabled.
  - The On-Demand Private GATT Proxy state is set to 0x00.
- Test Procedure

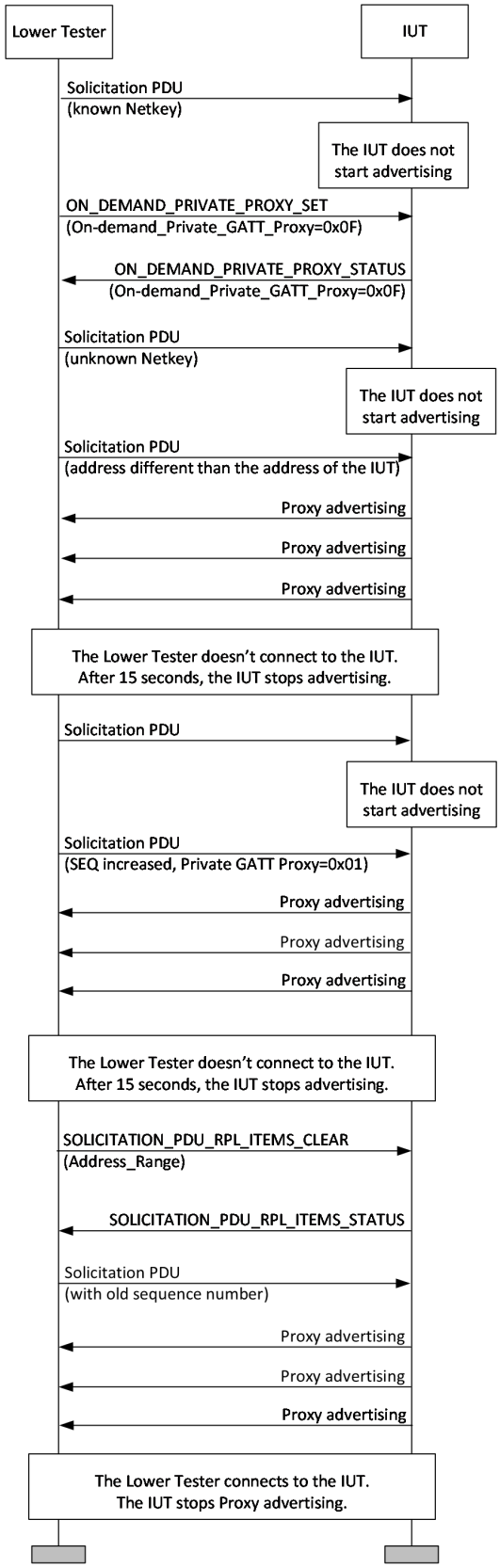


Figure 4.105: Test procedure for MESH/SR/MPXS/BV-12-C MSC

1. The Lower Tester sends a Solicitation PDU to the IUT with a known Netkey.
2. The IUT does not start advertising.
3. The Lower Tester sends an ON_DEMAND_PRIVATE_PROXY_SET message to the IUT with the On-demand_Private_GATT_Proxy field set to 0x0F.
4. The IUT responds with an ON_DEMAND_PRIVATE _PROXY_STATUS message with the On-demand_Private_GATT_Proxy field set as in the incoming message.
5. The Lower Tester sends a Solicitation PDU with an unknown Netkey.
6. The IUT does not start advertising.
7. The Lower Tester sends a valid Solicitation PDU, with the DST field set to an address different than the address of the IUT.
8. The IUT starts Proxy advertising, but the Lower Tester does not connect to the server.
9. After 15 seconds, the IUT stops advertising.
10. The Lower Tester resends the same Solicitation PDU from Step 7.
11. The IUT does not start advertising.
12. The Lower Tester sends a new Solicitation PDU to the IUT (increasing the SEQ field from Step 7) with Private GATT Proxy state set to 0x01.
13. The IUT starts Proxy advertising, but the Lower Tester does not connect to the server.
14. After 15 seconds, the IUT stops advertising.
15. The Lower Tester sends a SOLICITATION_PDU_RPL_ITEMS_CLEAR message with the Address_Range field set to include the unicast address of the Lower Tester.
16. The IUT responds with a SOLICITATION_PDU_RPL_ITEMS_STATUS.
17. The Lower Tester resends the message from Step 7 (with the old sequence number).
18. The IUT starts Proxy advertising.
19. The Lower Tester connects to the server and the IUT stops Proxy advertising.

- Expected Outcome

  Pass verdict

  The IUT only starts Proxy advertising when the On-Demand Private GATT Proxy state is set to a value greater than 0x00 and the Solicitation PDU contains a known Network key. The IUT properly maintains a replay protection list for the Solicitation PDUs that is cleared at the Client's request.

  If the IUT supports multiple subnets, it only advertises for the one corresponding to the network key that is used to authenticate the Solicitation PDU.

MESH/SR/MPXS/BV-13-C [Private GATT Proxy Solicitation with RFU Identification Type Value]

- Test Purpose

  Verify that an On-demand Private Proxy Server IUT does not respond to a Solicitation PDU with Identification Type field set to an RFU value.

- Reference

  [3] 4.2.11, 4.2.22, 4.4.5, 4.4.11, 4.4.12, 6.8

- Initial Condition
  - The IUT supports the On-demand Private Proxy Server model on the Primary Element.
  - The Lower Tester supports the On-demand Private Proxy Client Model.
  - The IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
  - The Private Proxy functionality is disabled.
- Test Procedure

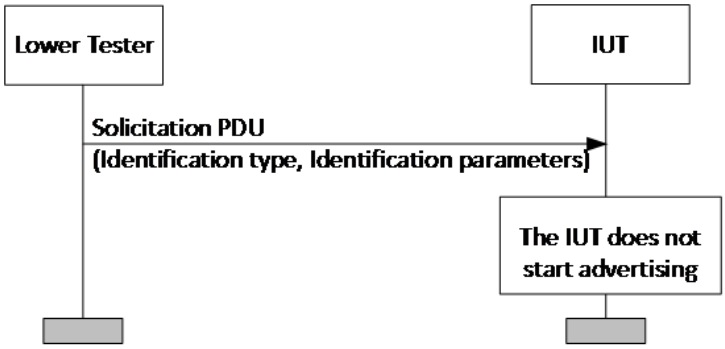


Figure 4.106: Test procedure for MESH/SR/MPXS/BV-13-C MSC

1. The Lower Tester sends a Solicitation PDU to the IUT with Identification Type field set to an RFU value and a correctly configured Network PDU.
2. The IUT does not start advertising.

- Expected Outcome

  Pass verdict

  The IUT does not start Proxy advertising.

MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can find and connect to a GATT Bearer Server.

- Reference

  [3] 7.2.2

- Initial Condition
  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.

- The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to begin scanning for GATT Bearer Server advertisements and automatically connect to devices advertising the Mesh Proxy Service.
  2. The Lower Tester begins sending connectable advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 9 octets of service data. The first octet is 0x00 and the next 8 octets contain the Network ID.
  3. The Lower Tester expects the IUT to establish a GATT connection upon receiving the advertisements.

- Expected Outcome

  Pass verdict

  In Step 3, the IUT establishes a GATT connection with the Lower Tester.

MESH/CL/MPXS/BV-02-C [Discover Mesh Proxy Service]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Service on a GATT Bearer Server.

- Reference

  [3] 7.2.2

- Initial Condition

  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

- Test Procedure

  1. The Upper Tester orders the IUT to discover the Mesh Proxy Service on the Lower Tester either by executing the procedure in GATT.TS [5] Discover Primary Services by Service UUID - By Client, GATT/CL/GAD/BV-02-C, with the Service UUID of «Mesh Proxy Service» or the procedure in GATT.TS [5] Discover All Primary Services - By Client, GATT/CL/GAD/BV-01-C.
  2. The Upper Tester expects the IUT to report the correct attribute handle range of the Mesh Proxy Service from the GATT database on the Lower Tester.

- Expected Outcome

  Pass verdict

In Step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Service on the Lower Tester.

In Step 2, the IUT reports the correct attribute handle range of the Mesh Proxy Service to the Upper Tester of the Mesh Proxy Service.

MESH/CL/MPXS/BV-03-C [Discover Mesh Proxy Data Out Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data Out characteristic on a GATT Bearer Server.

- Reference

  [3] 7.2.2, 7.2.3.2

- Initial Condition
  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

- Test Procedure
  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristics of a Service - by Client, GATT/CL/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Proxy Service, or Discover Characteristics by UUID - by Client, GATT/CL/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Proxy Service and UUID set to «Mesh Proxy Data Out». In the selected procedure, only one pass is needed with the server database defined in Initial Condition.
  2. The Upper Tester expects the IUT to report the correct handle range for the Mesh Proxy Data Out characteristic from the GATT database on the IUT.

- Expected Outcome

  Pass verdict

  In Step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Data Out characteristic on the Lower Tester.

  In Step 2, the IUT reports the correct attribute handle range to the Upper Tester.

MESH/CL/MPXS/BV-04-C [Discover Mesh Proxy Data Out Client Characteristic Configuration Descriptor and Enable and Process Notifications]

- Test Purpose

Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data Out Characteristic Configuration Descriptor on a GATT Bearer Server and can enable and process GATT Notifications for that characteristic.

- Reference

  [3] 7.2.2, 7.2.3.2

- Initial Condition
  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

- Test Procedure
  1. The Upper Tester orders the IUT to execute one pass of the procedure included in GATT.TS [5]: Discover all Characteristic Descriptors, GATT/CL/GAD/BV-06-C using the specified handle range.
  2. The Upper Tester expects the IUT to report the correct attribute handle for the Mesh Proxy Data Out Client Characteristic Configuration Descriptor from the GATT database on the IUT.
  3. The Upper Tester orders the IUT to enable notifications on the Mesh Proxy Data Out characteristic.
  4. The Lower Tester expects the IUT to execute the procedure described in GATT.TS [5]: GATT/CL/GAW/BV-08-C [Write Characteristic Descriptors - by Client] with the handle of the Client Characteristic Configuration Descriptor and the value 0x0001.
  5. The Lower Tester sends an ATT Handle Value Notification packet to the IUT with the Attribute Handle field set to the handle of the Mesh Proxy Data Out characteristic value and the Attribute Value field containing a Secure Network beacon with correct format and valid field values.
  6. The Upper Tester expects the IUT to report the received Secure Network beacon.

- Expected Outcome

  Pass verdict

  In Step 1, the IUT executes the discovery procedure to obtain the handle of the Mesh Proxy Data Out Client Characteristic Configuration Descriptor on the Lower Tester.

  In Step 2, the IUT reports the correct attribute handle to the Upper Tester.

  In Step 4, the IUT executes the write procedure to enable notifications on the Mesh Proxy Data Out characteristics.

  In Step 6, the IUT reports it has received a valid Secure Network beacon.

MESH/CL/MPXS/BV-05-C [Discover Mesh Proxy Data In Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can discover the Mesh Proxy Data In characteristic on a GATT Bearer Server.

- Reference

  [3] 7.2.2, 7.2.3.1

- Initial Condition

  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.

- Test Procedure

  1. The Upper Tester orders the IUT to execute either of the procedures included in GATT.TS [5]: Discover All Characteristic of a Service, GATT/SR/GAD/BV-04-C, with the specified handle range for the instantiation of the Mesh Proxy Service, or Discover Characteristic by UUID, GATT/SR/GAD/BV-05-C, with the specified handle range for the instantiation of the Mesh Proxy Service and UUID set to «Mesh Proxy Data In». In the selected procedure, only one pass is needed with the server database defined in Initial Condition.
  2. The Upper Tester expects the IUT to report the correct handle range for the Mesh Proxy Data In characteristic from the GAT database on the IUT.

- Expected Outcome

  Pass verdict

  In Step 1, the IUT executes the discovery procedure to obtain the attribute handle range of the Mesh Proxy Data In characteristic on the Lower Tester.

  In Step 2, the IUT reports the correct attribute handle range to the Upper Tester.

MESH/CL/MPXS/BV-06-C [Execute Write Without Response to Mesh Proxy Data In Characteristic]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can execute GATT Write Without Response on the Mesh Proxy Data In characteristic.

- Reference

  [3] 7.2.2, 7.2.3, 7.2.3.1

- Initial Condition

- The IUT is a GATT Bearer Client.
- The Lower Tester is a GATT Bearer Server and has the GATT Proxy state set to 0x01.
- The IUT and the Lower Tester share previously established network security credentials.
- The procedure described in MESH/CL/MPXS/BV-01-C [Connect to GATT Proxy Server] has been executed to establish a GATT connection between the IUT and the Lower Tester.
- The procedure described in MESH/CL/MPXS/BV-05-C [Discover Mesh Proxy Data In Characteristic] has been executed.

- Test Procedure

  1. The Upper Tester orders the IUT to send an ATT Write Command packet with the Attribute Handle field set to the handle of the Mesh Proxy Data In characteristic value and the Attribute Value field containing a proxy configuration message.
  2. The Lower Tester expects the IUT to send the proper ATT Write Command packet.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends an ATT Write Command packet with the proper fields.

MESH/CL/MPXS/BV-07-C [Connect to GATT Proxy Server – Server uses Private Network Identity]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can find and connect to a GATT Bearer Server when the Server advertises with Private Network Identity.

- Reference

  [3] 7.2.2

- Initial Condition
  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the Private GATT Proxy State set to 0x01 and the GATT Proxy State set to 0x00.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to begin scanning for GATT Bearer Server advertisements and automatically connect to devices advertising the Mesh Proxy Service.
  2. The Lower Tester begins sending connectable advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh

Proxy Service» and 17 octets of service data. The first octet of the service data is equal to 0x02, the next 8 octets contain a Hash value properly computed over the random and Network ID values with the IdentityKey, and the last 8 octets contain the random value used to compute the Hash.

3. The IUT establishes a GATT connection upon receiving the advertisements.

- Expected Outcome

  Pass verdict

  In Step 3, the IUT establishes a GATT connection with the Lower Tester.

MESH/CL/MPXS/BV-08-C [Connect to GATT Proxy Server – Server uses Private Node Identity]

- Test Purpose

  Verify that an IUT supporting the GATT Bearer Client role can find and connect to a GATT Bearer Server when the Server advertises with Private Node Identity.

- Reference

  [3] 7.2.2

- Initial Condition

  - The IUT is a GATT Bearer Client.
  - The Lower Tester is a GATT Bearer Server and has the Private Node Identity state set to 0x01 and the Node Identity state set to 0x00.
  - The IUT and the Lower Tester support GAP Central and GAP Peripheral roles, respectively.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

  1. The Upper Tester orders the IUT to begin scanning for GATT Bearer Server advertisements and automatically connect to devices advertising the Mesh Proxy Service.
  2. The Lower Tester begins sending connectable advertising packets containing the following AD Types in the Advertising Data field: «Flags», «Incomplete List of 16-bit Service UUIDs» or «Complete List of 16-bit Service UUIDs» including the «Mesh Proxy Service», and «Service Data» containing the «Mesh Proxy Service» and 17 octets of service data. The first octet of the service data is equal to 0x03, the next 8 octets contain a Hash value properly computed over a random value and the Lower Tester address with the IdentityKey, and the last 8 octets contain the random value used to compute the Hash.
  3. The Lower Tester expects the IUT to establish a GATT connection upon receiving the advertisements.

- Expected Outcome

  Pass verdict

  In Step 3, the IUT establishes a GATT connection with the Lower Tester.

MESH/CL/MPXS/BV-09-C [Send GATT Proxy Solicitation]

- Test Purpose

  Verify that a GATT Proxy Client can send a GATT Proxy Solicitation.

- Reference

  [3] 4.2.11, 4.2.22, 4.4.6, 4.4.11, 4.4.12, 6.8

- Initial Condition

  - The Lower Tester supports the On-demand Private Proxy Server model on the Primary Element.
  - The IUT supports the On-demand Private Proxy Client Model.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure

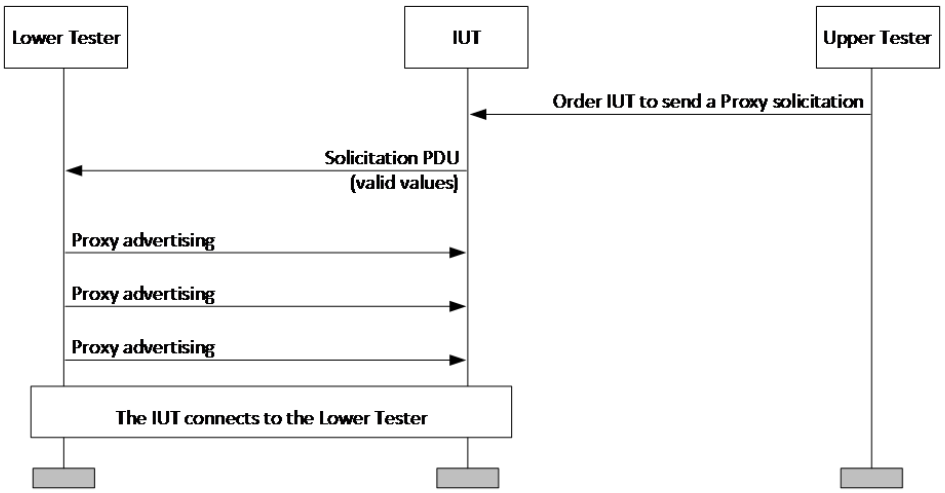


Figure 4.107: Test procedure for MESH/CL/MPXS/BV-09-C MSC

1. The Upper Tester orders the IUT to send a Proxy solicitation with valid parameters.
2. The IUT sends a Solicitation PDU with valid parameters.
3. The Lower Tester starts Proxy advertising.
4. The IUT connects to the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT correctly sends a Solicitation PDU and connects to the GATT Proxy Server.

  The Solicitation PDU has all fields set to correct values: the Identification Type field is set to 0x00; the contained Network PDU has the CTL field set to 1, the TTL field set to 0, and is secured using the managed flooding security material and the proxy solicitation nonce.

4.16 Configuration Model

The test objective is to verify functionality specific to the Configuration Model for both the Configuration Server Model and the Configuration Client Model.

4.16.1 Secure Network Beacon Procedures

This section validates the Secure Network Beacon procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/SNBP/BV-01-C [Respond to Config Beacon Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Beacon Set and Get messages.

- Reference

  [3] 3.9.3, 4.2.10, 4.3.2.1, 4.3.2.2, 4.3.2.3, 4.4.1.2, 4.4.1.2.1, 4.4.2.2.1

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x01.
  2. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x01.
  3. The Lower Tester expects the IUT to start broadcasting Secure Network beacons within 20 seconds.
  4. The Lower Tester sends a Config Beacon Get message to the IUT.
  5. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x01.
  6. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x00 (the node is not beaconing a Secure Network beacon).
  7. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x00.
  8. The Lower Tester scans for 20 seconds to verify that the IUT stops broadcasting Secure Network beacons.
  9. The Lower Tester sends a Config Beacon Get message to the IUT.
  10. The Lower Tester expects to receive a Config Beacon Status message from the IUT with the Beacon field set to 0x00.

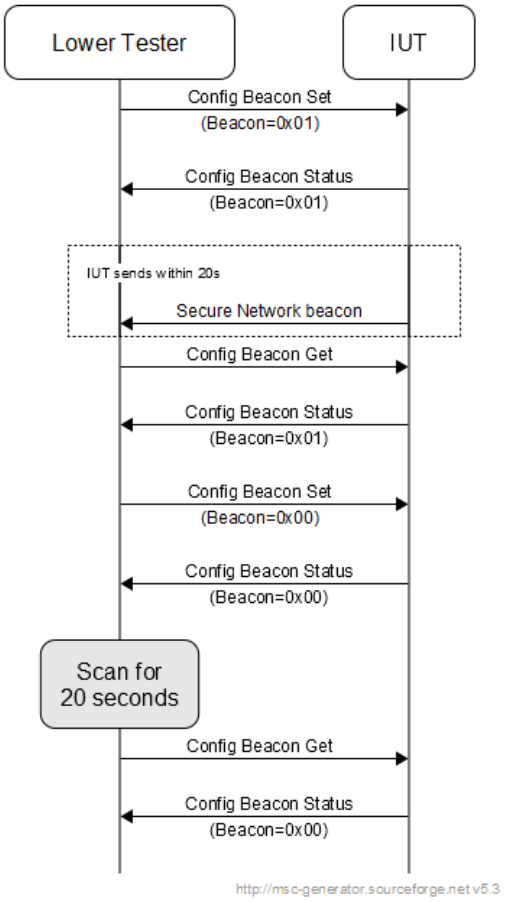


Figure 4.108: MESH/NODE/CFG/SNBP/BV-01-C [Respond to Config Beacon Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2 and Step 5, the IUT sends a Config Beacon Status message with the Beacon field set to 0x01.

  In Step 3, the IUT sends at least one Secure Network beacon.

  In Step 8, the IUT does not send Secure Network beacons.

  In Step 10, the IUT sends a Config Beacon Status message with the Beacon field set to 0x00.

  All messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SNBP/BI-01-C [Ignore invalid Config Beacon Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores invalid Config Beacon Set and Get messages.

- Reference

[3] 3.9.3, 4.2.10, 4.3.2.1, 4.3.2.2, 4.3.2.3, 4.4.1.2.1, 4.4.2.2.1

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Beacon Get message to the IUT.
  2. The Lower Tester expects the IUT to respond with a Config Beacon Status message.
  3. The Lower Tester sends a Config Beacon Set message to the IUT with a Beacon field value of 0x02 (value is Reserved for Future Use).
  4. The Lower Tester expects the IUT to ignore the Config Beacon Set message and not respond with any Config Beacon Status message.
  5. The Lower Tester sends a Config Beacon Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Config Beacon Status message.

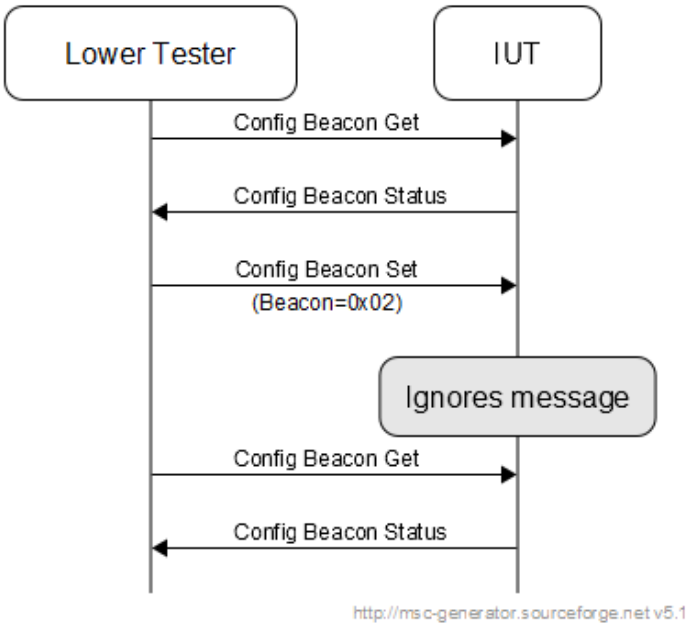


Figure 4.109: MESH/NODE/CFG/SNBP/BI-01-C [Ignore invalid Config Beacon Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT does not send any Config Beacon Status message.

  In Step 6, the Beacon field in the Config Beacon Status message has the same value as the one obtained

in Step 2.

All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/SNBP/BV-01-C [Send Config Beacon Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Beacon Get messages to a Configuration Server.

- Reference

  [3] 4.2.10, 4.3.2.1, 4.3.2.2, 4.4.1.2.1, 4.4.2.2.1

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Beacon Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Beacon Get message and responds with a Config Beacon Status message.
  3. The Upper Tester orders the IUT to send a Config Beacon Set message to the Lower Tester with the Beacon field value set to 0x00.
  4. The Lower Tester expects to receive a Config Beacon Set message and responds by sending a Config Beacon Status message with the Beacon field value set to 0x00.
  5. The Upper Tester orders the IUT to send a Config Beacon Set message to the Lower Tester with the Beacon field value set to 0x01.
  6. The Lower Tester expects to receive a Config Beacon Set message and responds by sending a Config Beacon Status message with the Beacon field value set to 0x01.

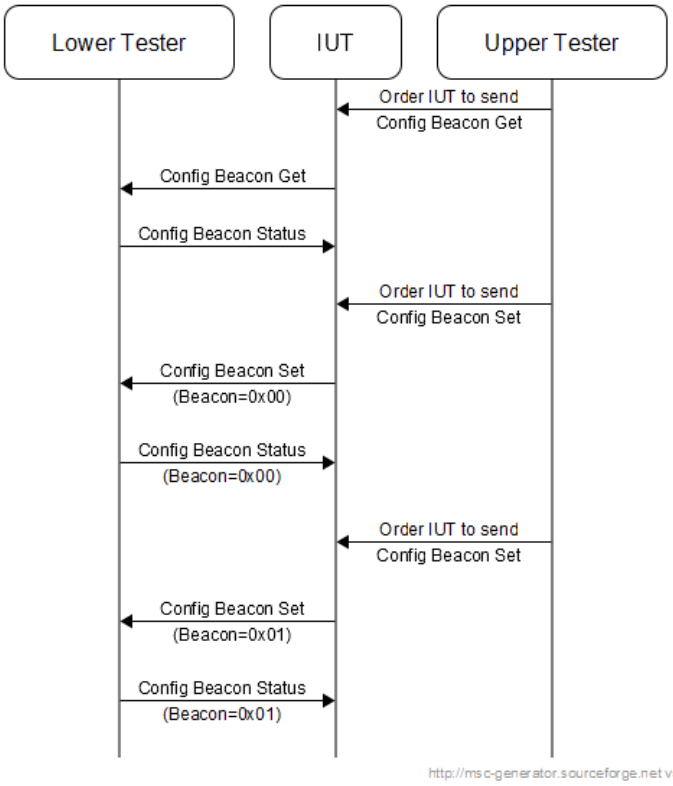


Figure 4.110: MESH/CFGCL/CFG/SNBP/BV-01-C [Send Config Beacon Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a properly formatted Config Beacon Get message.

  In Step 4, the IUT sends a properly formatted Config Beacon Set message with the Beacon field value set to 0x00.

  In Step 6, the IUT sends a properly formatted Config Beacon Set message with the Beacon field value set to 0x01.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.2 Composition Data Procedures

This section validates the Composition Data procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Composition Data

Get messages and that the IUT supports the mandatory models.

- Reference

  [3] 4.2.1, 4.2.1.1, 4.3.2.4, 4.3.2.5, 4.4.1.2, 4.4.1.2.2, 4.4.2.2.2

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT.

- Test Procedure

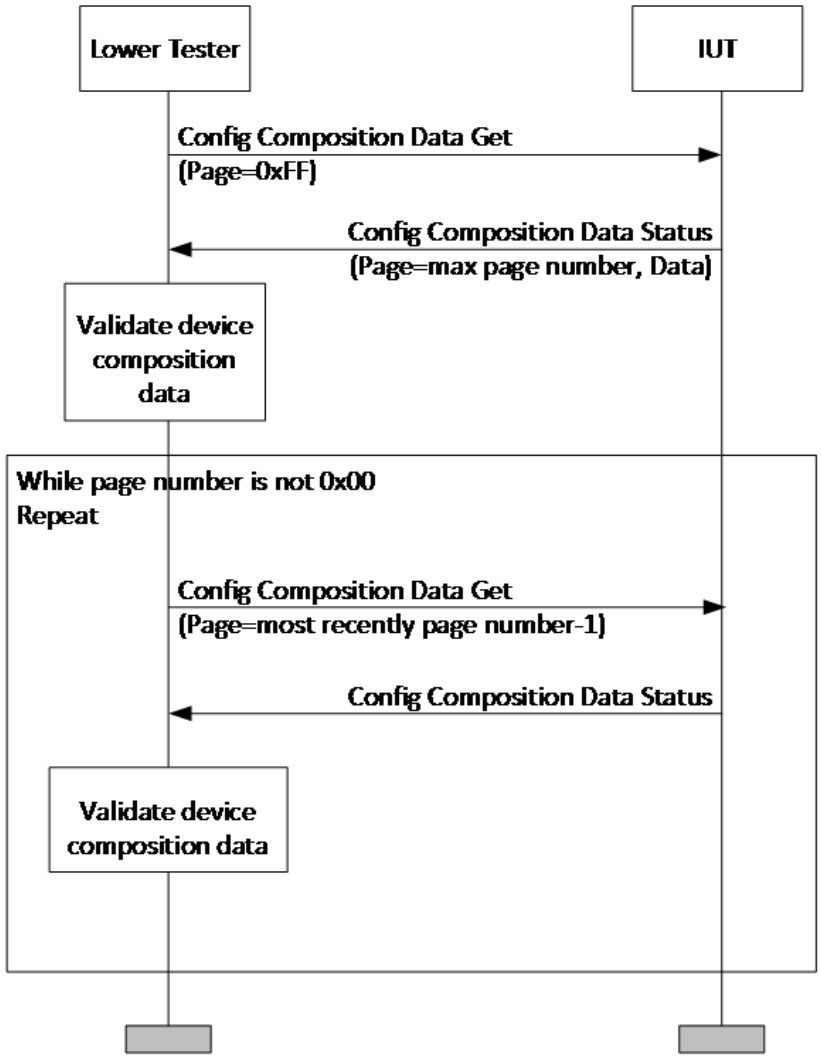


Figure 4.111: Test procedure for MESH/NODE/CFG/COMP/BV-01-C MSC

1. The Lower Tester sends a Config Composition Data Get message to the IUT with the Page field value set to 0xFF.
2. The Lower Tester expects the IUT to respond with a Config Composition Data Status message with the Page field set to the maximum page number supported and the Data field set to the information contained by that Composition Data Page.
3. While the page number received in Step 2 is not 0x00, repeat Steps 1–2, setting the Page field in Step 1 to the most recently received page number minus one.
4. The Lower Tester parses the Composition Data returned by the IUT in all iterations through Step 2 and checks that all fields are valid and consistent and that the mandatory root models are supported.

- Expected Outcome

  Pass verdict

  In all iterations through Step 2, the IUT sends complete and correctly formatted Composition Data. The Configuration Model and the Health Model are supported on the primary Element. All fields from the Composition Data information contain valid values.

  The IUT supports all Composition Data pages that are mandatory based on the supported features, as indicated in the Mesh Protocol ICS [4].

  If supported, the Bridge Configuration Server model is on the primary Element.

  If supported, the Directed Forwarding Configuration Server model is on the primary Element.

  All configuration messages are encrypted using the device key of the IUT.

  All the Model IDs present in the composition data are values allowed by any active Bluetooth specification, i.e., either 16-bit values defined in the list of assigned numbers for Bluetooth Mesh, or 32-bit vendor-assigned values that include a 16-bit Company ID that has been assigned by the Bluetooth SIG.

  Each of the following composition data pages (CDP), if present, satisfy their specific requirements:

  - CDP 0 and 128: the location of the model instances satisfies the requirements stated in the Mesh Protocol Specification [3] for each model based on each model's definition:
    - Some models must be placed on the primary element.
    - When models extend other models, these extended models must also be present on the same element.
  - CDP 1 and 129: the relationship between the model instances satisfies the requirements stated in the Mesh Protocol Specification [3] for each model based on model definitions:
    - Some models are "corresponding", meaning that they both must be present on a node, distributed on elements according to the model definitions.
  - CDP 2 and 130: the model instances identified by the mesh profiles present in CDP 2 (CDP 130) are present in CDP 0 (CDP 128) and their relationships are documented correctly in CDP 1 (CDP 129):
    - Each profile definition requires a set of models to be present, and their presence must be verified on the elements that are associated with each profile (based on the Element_Offset field values).

Fail verdict

The composition data contains at least one 16-bit Model ID value that is not present in the list of assigned numbers for Bluetooth Mesh or a 32-bit value that includes a 16-bit Company ID that has not been assigned by the Bluetooth SIG.

MESH/CFGCL/CFG/COMP/BV-01-C [Send Composition Data Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Composition Data Get messages to a Configuration Server.

- Reference

  [3] 4.3.2.4, 4.4.2.2.2

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

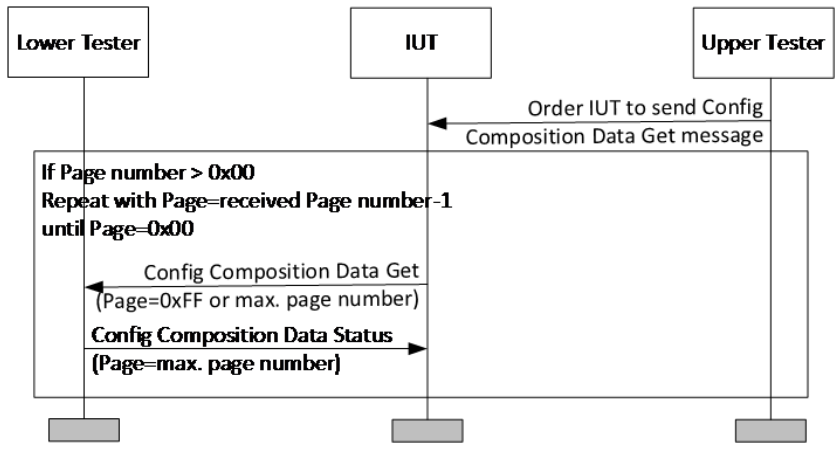


Figure 4.112: Test procedure for MESH/CFGCL/CFG/COMP/BV-01-C MSC

1. The Upper Tester orders the IUT to obtain the device composition data of the Lower Tester.
2. The IUT sends a Config Composition Data Get message with the highest supported page number or 0xFF.
3. The Lower Tester responds with a Config Composition Data Status message with the Page field value set to the highest page number supported by the IUT.
4. If the Page field value sent in Step 3 is greater than 0x00, then repeat Steps 2 and 3, where the IUT requests a Page field value decremented by 1 compared to the previous until 0x00 has been sent by

the Lower Tester.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a properly formatted Config Composition Data Get message to the Lower Tester.

  If applicable, the IUT requests a page number equal to the previously received page number minus 1.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.3 Default TTL Procedure

This section validates the Default TTL procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/DTTL/BV-01-C [Respond to Config Default TTL Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Default TTL Set and Config Default TTL Get messages and the configuration messages are encrypted using the device key used in the received messages.

- Reference

  [3] 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.2, 4.4.1.2.3

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x00.
  2. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x00.
  3. The Lower Tester sends a Config Default TTL Get message to the IUT.
  4. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x00.
  5. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x7F.
  6. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x7F.

7. The Lower Tester sends a Config Default TTL Get message to the IUT.
8. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a TTL field value of 0x7F.

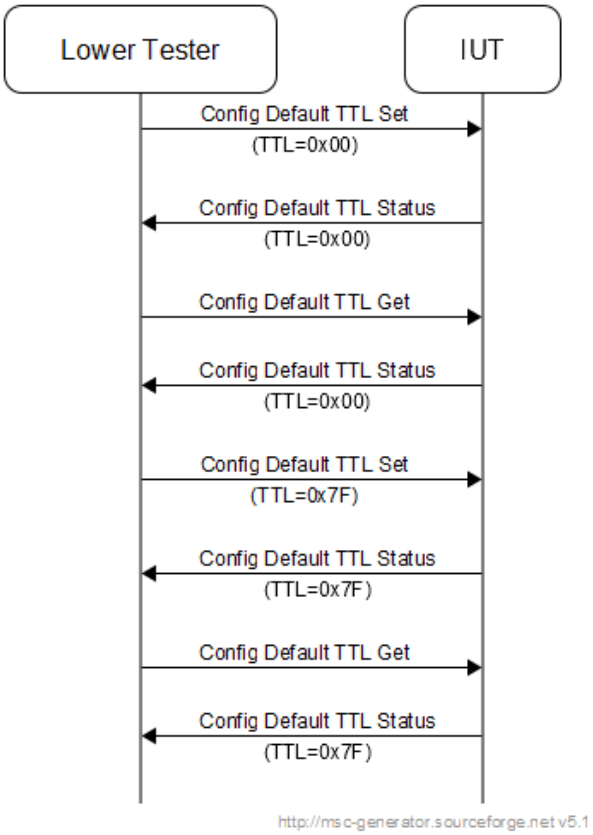


Figure 4.113: MESH/NODE/CFG/DTTL/BV-01-C [Respond to Config Default TTL Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, the IUT responds with a Config Default TTL Status message with the TTL field value equal to 0x00.

  In Steps 6 and 8, the IUT responds with a Config Default TTL Status message with the TTL field value equal to 0x7F.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/DTTL/BI-01-C [Respond to Config Default TTL Set Message (On Secondary Element or With Invalid TTL)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores a Config Default TTL Set

message with invalid fields or if a Config Default TTL Get message is sent to the secondary Element of the IUT.

- Reference

  [3] 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.1, 4.4.1.2.3

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The IUT supports a secondary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Default TTL Get message to the IUT.
  2. The Lower Tester expects to receive a Config Default TTL Status message in response from the IUT with a valid TTL value.
  3. The Lower Tester sends a Config Default TTL Set message to the IUT with the TTL field set to a value of 0x80.
  4. The Lower Tester expects the IUT to ignore the message from Step 3 and not respond with a Config Default TTL Status message.
  5. The Lower Tester sends a Config Default TTL Get message to the secondary Element of the IUT.
  6. The Lower Tester expects the IUT to ignore the message from Step 5 and not respond with a Config Default TTL Status message.
  7. The Lower Tester repeats Steps 3–4, setting the TTL field to 0x01 in Step 3.
  8. The Lower Tester repeats Steps 1–2 to confirm that the value of the Default TTL state has not changed on the IUT.

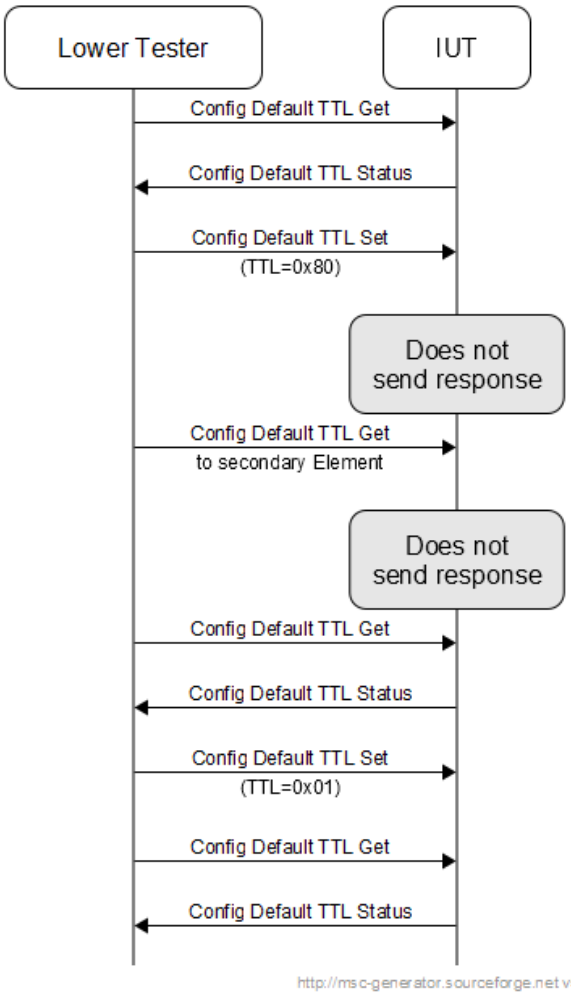


Figure 4.114: MESH/NODE/CFG/DTTL/BI-01-C [Respond to Config Default TTL Set Message (On Secondary Element or With Invalid TTL)] MSC

- Expected Outcome

  Pass verdict

  In both iterations through Step 2, the IUT responds with a Config Default TTL Status message containing a valid value in the TTL field.

  In both iterations through Step 4 and in Step 6, the IUT does not respond to the Config Default TTL Set message with the invalid TTL field value.

  In Step 7, the IUT resends the same message from Step 2.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/DTTL/BV-01-C [Send Config Default TTL Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Default TTL Get

messages to a Configuration Server, and the configuration messages are encrypted using destination device's device key.

- Reference

  [3] 3.7.4.3, 4.3.2.6, 4.3.2.7, 4.3.2.8, 4.4.1.2.3, 4.4.2.2.3

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Default TTL Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Default TTL Get message.
  3. The Lower Tester responds with a Config Default TTL Status message with a valid TTL field value.
  4. The Upper Tester orders the IUT to send a Config Default TTL Set message to the Lower Tester with a valid TTL field value.
  5. The Lower Tester expects to receive a Config Default TTL Set message with the requested TTL field value.
  6. The Lower Tester responds with a Config Default TTL Status message with the TTL field value equal to the value from Step 4.

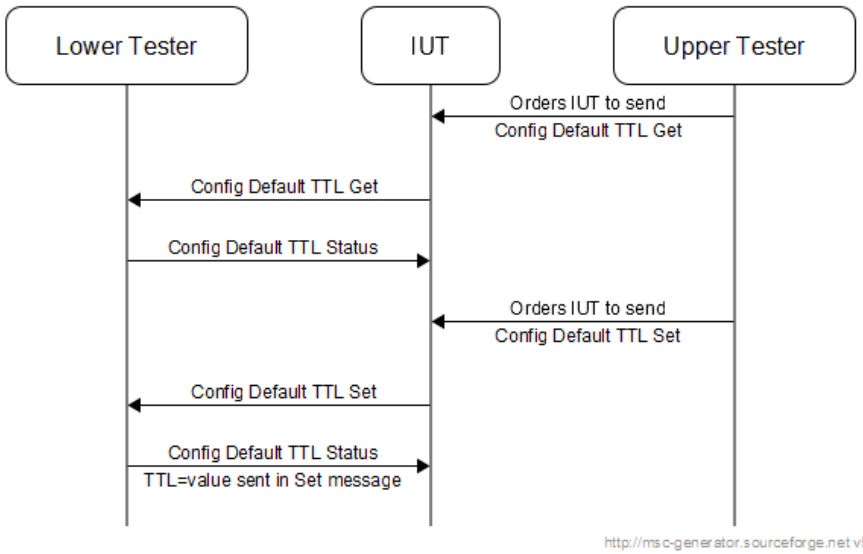


Figure 4.115: MESH/CFGCL/CFG/DTTL/BV-01-C [Send Config Default TTL Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Default TTL Get message to the Lower Tester.

In Step 5, the IUT sends a Config Default TTL Set message to the Lower Tester, with the TTL field value set to the value requested in Step 4.

All configuration messages are encrypted using the device key of the Lower Tester.

4.16.4 GATT Proxy Procedures

This section validates the GATT Proxy procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/GPXY/BV-01-C [Respond to Config GATT Proxy Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config GATT Proxy Set and Get messages.

- Reference

  [3] 4.2.1.1, 4.2.11, 4.2.11.1, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00 ("The Mesh Proxy Service is stopped").
  2. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x00 ("The Mesh Proxy Service is stopped"). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 ("The Mesh Proxy Service is not supported").
  3. The Lower Tester sends a Config GATT Proxy Get message to the IUT.
  4. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x00 ("The Mesh Proxy Service is stopped"). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 ("The Mesh Proxy Service is not supported").
  5. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01 ("The Mesh Proxy Service is running").
  6. If the IUT supports the Mesh Proxy Service, the Lower Tester expects to receive a Config GATT Proxy Status message in response from the IUT with a GATTProxy field value of 0x01 ("The Mesh Proxy Service is running"). Otherwise, the Lower Tester expects a GATTProxy field value of 0x02 ("The Mesh Proxy Service is not supported").

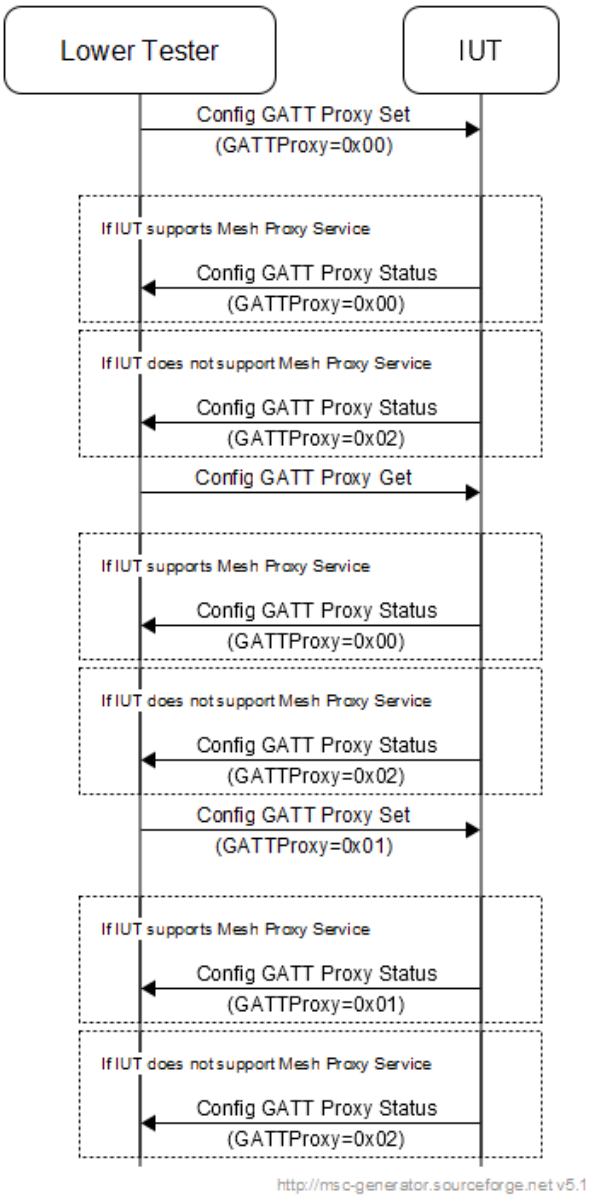


Figure 4.116: MESH/NODE/CFG/GPXY/BV-01-C [Respond to Config GATT Proxy Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, the IUT responds with a Config GATT Proxy Status message with the GATTProxy field value set to 0x00, if the IUT supports the Mesh Proxy Service, or to 0x02, if the IUT does not support the Mesh Proxy Service.

  In Step 6, the IUT responds with a Config GATT Proxy Status message with the GATTProxy field value set to 0x01, if the IUT supports the Mesh Proxy Service, or to 0x02, if the IUT does not support the Mesh Proxy Service.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/GPXY/BV-02-C [Network ID Advertising on Multiple Subnets]

- Test Purpose

  Verify that a Configuration Server IUT can trigger Network ID advertising based on the GATT Proxy state change and can interleave advertising for multiple Subnets.

- Reference

  [3] 4.2.12, 4.3.2.9, 4.3.2.11, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service.
  - The GATT Proxy state is set to 0x00.

- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.
  3. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message and to begin sending interleaved Network ID advertisings for both Subnets it supports.

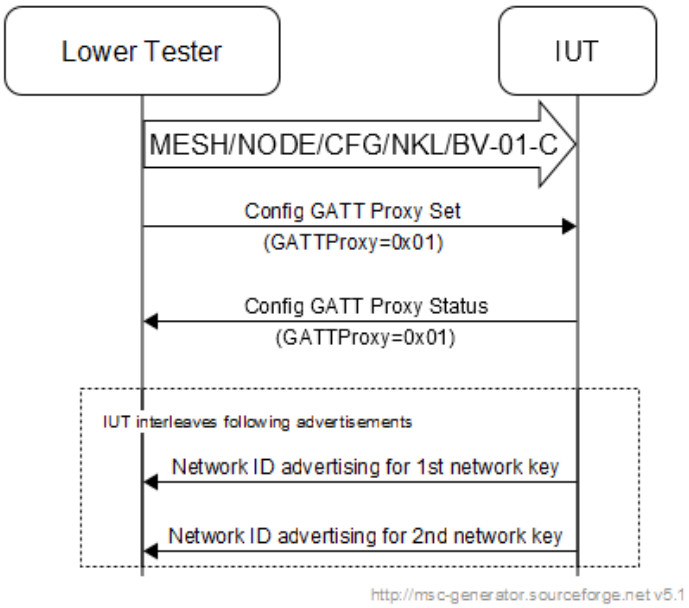


Figure 4.117: MESH/NODE/CFG/GPXY/BV-02-C [Network ID Advertising on Multiple Subnets] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01. The IUT then begins advertising with Network ID with interleaved packets for both Subnets it supports.

MESH/NODE/CFG/GPXY/BI-01-C [Respond to Config GATT Proxy Set Messages Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores Config GATT Proxy Set messages with invalid field values.

- Reference

  [3] 3.1.2, 4.2.1.1, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to

an RFU value (in the range 0x03-0xFF).

2. The Lower Tester expects the IUT to ignore the message and not send any response.

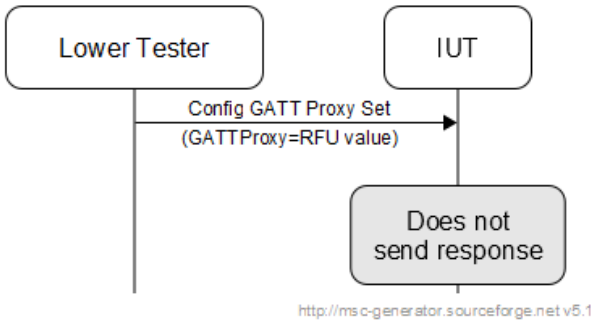


Figure 4.118: MESH/NODE/CFG/GPXY/BI-01-C [Respond to Config GATT Proxy Set Messages Invalid Parameters] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any message in response to the Config GATT Proxy Set message with invalid parameters sent by the Lower Tester in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/GPXY/BV-01-C [Send Config GATT Proxy Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config GATT Proxy Get and Config GATT Proxy Set messages to a Configuration Server.

- Reference

  [3] 4.2.1.1, 4.3.2.10, 4.3.2.11, 4.4.1.2, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config GATT Proxy Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config GATT Proxy Get message.
  3. The Lower Tester sends a Config GATT Proxy Status message in response to the IUT with a valid GATTProxy field value.
  4. The Upper Tester orders the IUT to send a Config GATT Proxy Set message to the Lower Tester with a GATTProxy field value of either 0x00 or 0x01.

5. The Lower Tester expects to receive a Config GATT Proxy Set message with the GATTProxy field value set to the value requested in Step 4.
6. The Lower Tester sends a Config GATT Proxy Status message in response to the IUT with the GATTProxy field value matching the value from the request.

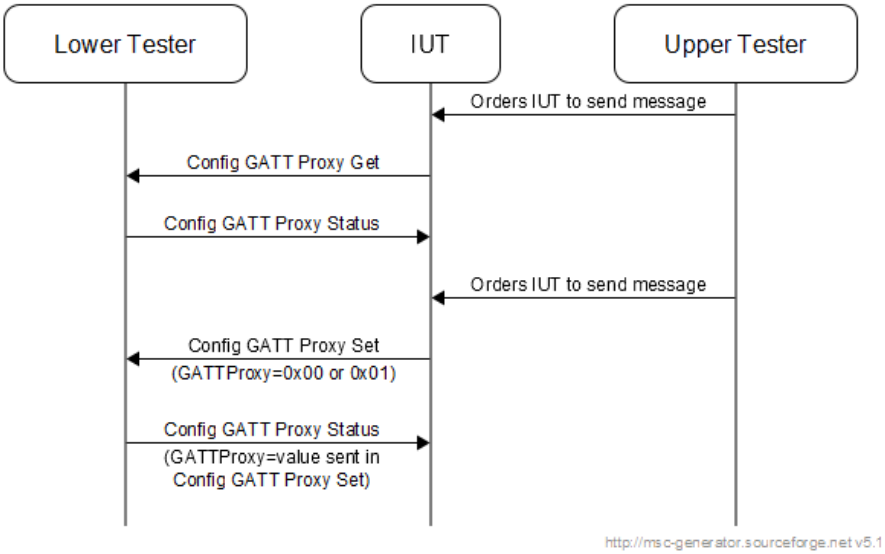


Figure 4.119: MESH/CFGCL/CFG/GPXY/BV-01-C [Send Config GATT Proxy Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config GATT Proxy Get message to the Lower Tester.

  In Step 5, the IUT sends a Config GATT Proxy Set message to the Lower Tester with the GATTProxy field value set to the value requested in Step 4.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.5 Friend Procedures

This section validates the Friend procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/CFGF/BV-01-C [Respond to Config Friend Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Friend Get and Config Friend Set messages.

- Reference

  [3] 3.7.4.3, 4.2.1, 4.2.13, 4.3.2.55, 4.3.2.56, 4.3.2.57, 4.4.1.2, 4.4.1.2.5

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.
  - IUT support for the Friend feature is indicated in the ICS table 7 (Node Capabilities - Upper Transport Layer). If the Friend feature is supported, the feature is disabled.
- Test Procedure
  1. The Lower Tester sends a Config Friend Get message to the IUT.
  2. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x00, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.
  3. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
  4. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x01, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.
  5. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x00.
  6. The Lower Tester expects to receive a Config Friend Status message in response from the IUT with the Friend field set to 0x00, if the Friend feature is supported, or to 0x02, if the Friend feature is not supported.

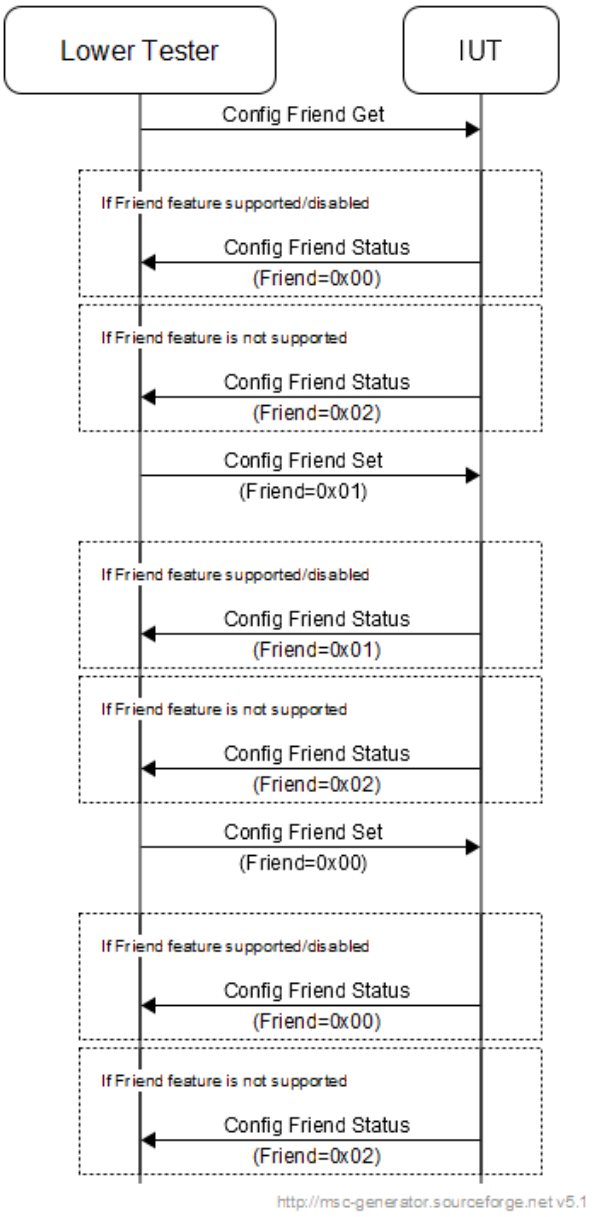


Figure 4.120: MESH/NODE/CFG/CFGF/BV-01-C [Respond to Config Friend Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 6, the IUT responds with a Config Friend Status message with the Friend field value set to 0x00, if the IUT supports the Friend feature, or to 0x02, if the IUT does not support the Friend feature.

  In Step 4, the IUT responds with a Config Friend Status message with the Friend field value set to 0x01, if the IUT supports the Friend feature, or to 0x02, if the IUT does not support the Friend feature.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/CFGF/BI-01-C [Respond to Config Friend Set Messages Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores Config Friend Set messages with invalid field values.

- Reference

  [3] 3.1.2, 4.2.1.1, 4.3.2.56, 4.4.1.2.4, 4.4.2.2.4, 7.2.2.2

- Initial Condition

  - The IUT supports the Configuration Server Model.
  - The IUT supports the Friend feature, and the feature is enabled.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to an RFU value (in the range 0x02-0xFF).
  2. The Lower Tester expects the IUT to ignore the message and not send any response.

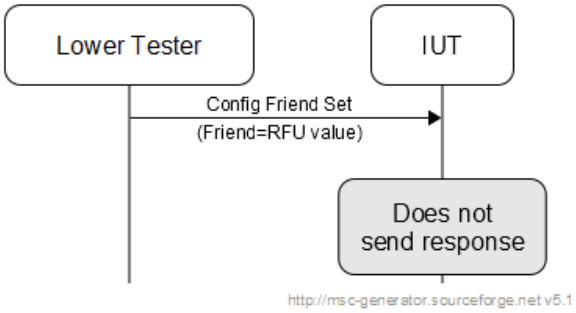


Figure 4.121: MESH/NODE/CFG/CFGF/BI-01-C [Respond to Config Friend Set Messages Invalid Parameters] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any message in response to the Config Friend Set message with invalid parameters sent by the Lower Tester in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/CFGF/BV-01-C [Send Config Friend Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Friend Get and Config Friend Set messages to a Configuration Server.

- Reference

  [3] 4.2.1, 4.3.2.55, 4.3.2.56, 4.3.2.57, 4.4.1.2.5, 4.4.2.2.5

- Initial Condition
  - The IUT supports the Configuration Client Model on the Primary Element.
  - The Lower Tester supports the Configuration Server Model.
  - The Lower Tester supports the Friend feature and the feature is enabled.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Friend Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Friend Get message.
  3. The Lower Tester sends a Config Friend Status message in response to the IUT with a valid Friend field value.
  4. The Upper Tester orders the IUT to send a Config Friend Set message to the Lower Tester with a Friend field value of either 0x00 or 0x01.
  5. The Lower Tester expects to receive a Config Friend Set message with the Friend field value set to the value requested in Step 4.
  6. The Lower Tester sends a Config Friend Status message in response to the IUT with the Friend field value matching the value from the request or 0x02 if the Friend feature is not supported.

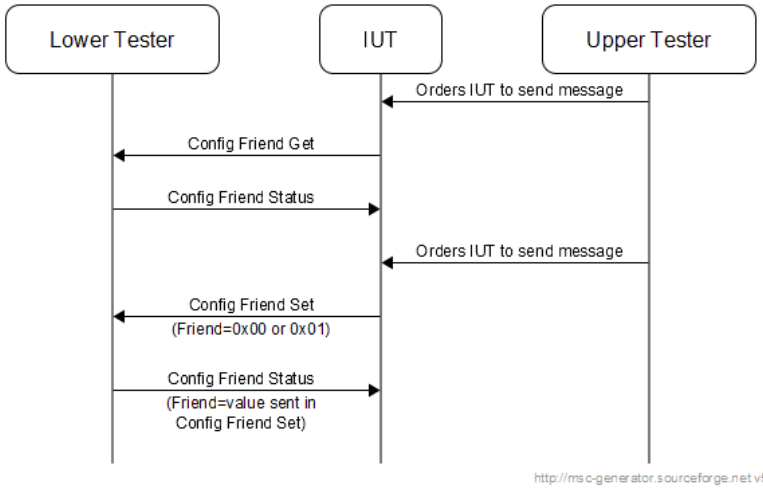


Figure 4.122: MESH/CFGCL/CFG/CFGF/BV-01-C [Send Config Friend Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Friend Get message to the Lower Tester.

  In Step 5, the IUT sends a Config Friend Set message to the Lower Tester with the Friend field value set to the value requested in Step 4.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.6 Relay Procedures

This section validates the Relay procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/CFGR/BV-01-C [Respond to Config Relay Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Relay Get and Config Relay Set messages.

- Reference

  [3] 3.7.4.3, 4.3.2.12, 4.3.2.13, 4.3.2.14, 4.4.1.2, 4.4.1.2.6, 4.4.2.2.6

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.

- IUT support for the Relay feature is indicated in the IXIT [6] parameters. If the Relay feature is supported, the feature is disabled.

- Test Procedure

1. The Lower Tester sends a Config Relay Get message to the IUT.
2. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x00, if the Relay feature is supported, or to 0x02, if the Relay feature is not supported, and the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.
3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01, the RelayRetransmitCount field set to 0b001, and the RelayRetransmitIntervalSteps field set to 0b00001.
4. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b00001, if the Relay feature is supported, or to 0x02, the RelayRetransmitCountfield value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000, if the Relay feature is not supported.
5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00, the RelayRetransmitCount field set to 0b000, and the RelayRetransmitIntervalSteps field set to 0b00000.
6. The Lower Tester expects to receive a Config Relay Status message in response from the IUT with the Relay field set to 0x00, if the Relay feature is supported, or to 0x02, if the Relay feature is not supported, and the RelayRetransmitCount field value set to 0b000, and the Publish RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.

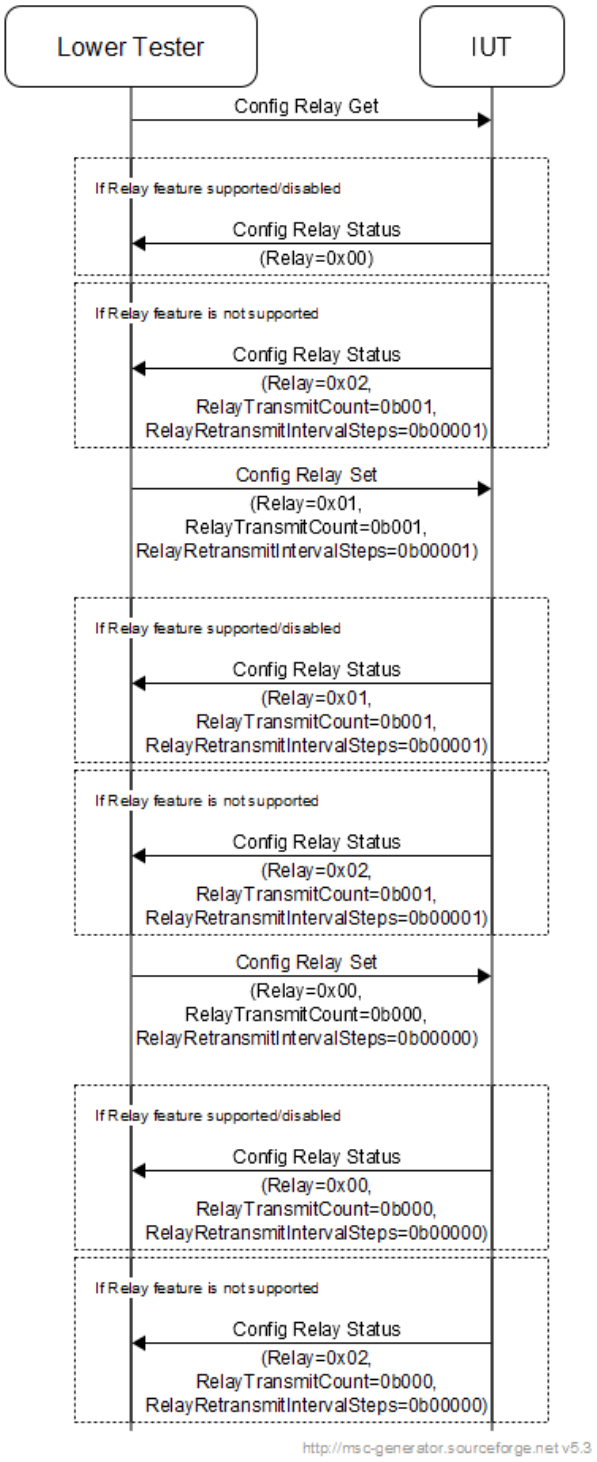


Figure 4.123: MESH/NODE/CFG/CFGR/BV-01-C [Respond to Config Relay Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 6, the IUT responds with a Config Relay Status message with the Relay field value set to 0x00, if the IUT supports the Relay feature, or to 0x02, if the IUT does not support the Relay feature, and

the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000 in both cases.

In Step 4, the IUT responds with a Config Relay Status message with the Relay field value set to 0x01, the RelayRetransmitCount field value set to 0b001, and the RelayRetransmitIntervalSteps field value set to 0b00001, if the IUT supports the Relay feature, or to 0x02, the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000, if the IUT does not support the Relay feature.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/CFGR/BI-01-C [Respond to Config Relay Set Messages Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores Config Relay Set messages with invalid field values.

- Reference

  [3] 3.1.2, 4.3.2.13, 4.3.2.14, 4.4.1.2.6, 4.4.2.2.6

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to an RFU value (in the range 0x03-0xFF), the RelayRetransmitCount field value set to 0b000, and the RelayRetransmitIntervalSteps field value set to 0b00000.
  2. The Lower Tester expects the IUT to ignore the message and not send any response.

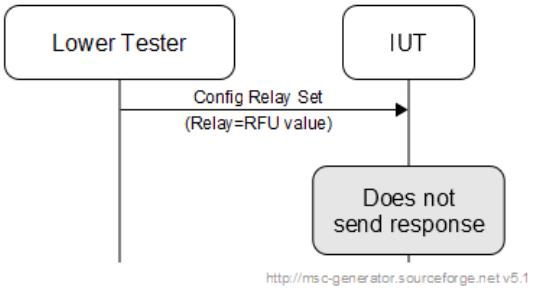


Figure 4.124: MESH/NODE/CFG/CFGR/BI-01-C [Respond to Config Relay Set Messages Invalid Parameters] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not send any message in response to the Config Relay Set message with invalid parameters sent by the Lower Tester in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/CFGR/BV-01-C [Send Config Relay Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Relay Get and Config Relay Set messages to a Configuration Server.

- Reference

  [3] 3.7.4.3, 4.2.8, 4.3.2.12, 4.3.2.13, 4.3.2.14, 4.4.1.2.6, 4.4.2.2.6

- Initial Condition
  - The IUT supports the Configuration Client Model on the Primary Element.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Relay Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Relay Get message.
  3. The Lower Tester sends a Config Relay Status message in response to the IUT with a valid Relay field value, a valid RelayRetransmitCount field value, and a valid RelayRetransmitIntervalSteps field value.
  4. The Upper Tester orders the IUT to send a Config Relay Set message to the Lower Tester with a Relay field value of either 0x00 or 0x01, the RelayRetransmitCount field set to 0b000, and the RelayRetransmitIntervalSteps field set to 0b00000.
  5. The Lower Tester expects to receive a Config Relay Set message with the Relay field value set to the value requested in Step 4.

6. The Lower Tester sends a Config Relay Status message in response to the IUT with the Relay field value matching the value from the request.

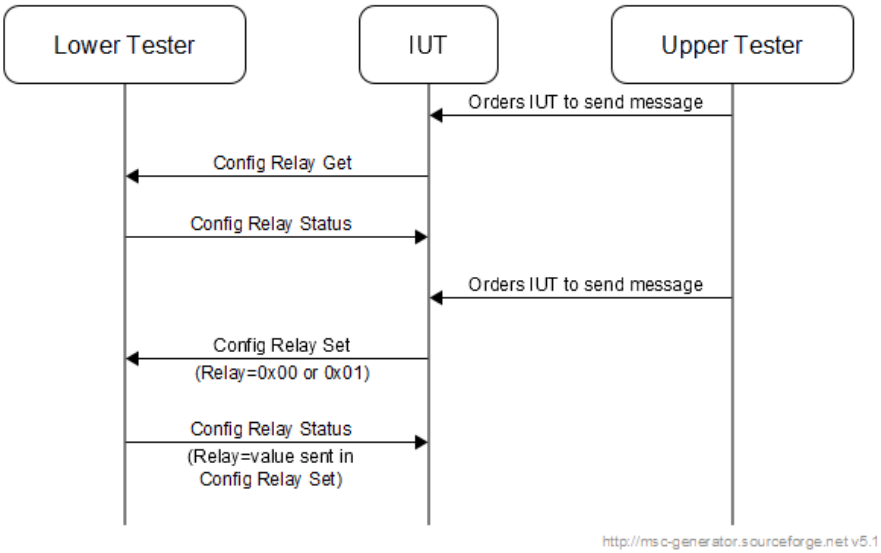


Figure 4.125: MESH/CFGCL/CFG/CFGR/BV-01-C [Send Config Relay Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Relay Get message to the Lower Tester.

  In Step 5, the IUT sends a Config Relay Set message to the Lower Tester with the Relay field value set to the value requested in Step 4.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.7 Model Publication Procedures

This section validates the Model Publication procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Set, Config Model Publication Get and Config Model Publication Virtual Address Set messages.

- Reference

  [3] 3.1.2, 4.2.2.3, 4.3.2, 4.3.2.15, 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition

- The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- IXIT [6] provides ModelID that supports publication with periodic and state change publishing.

• Test Procedure

1. The Lower Tester sends a Config AppKey Add message to the IUT with a NetKeyIndex value of 0, an AppKeyIndex of 0, (so the NetKeyIndex and AppKeyIndex field is set to 0x000000) and a valid 16-octet AppKey value.
2. The Lower Tester expects to receive a Config AppKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in Step 1.
3. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).
4. The Lower Tester expects to receive a Config Model App Status message from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in Step 3. The Lower Tester stores the information for later use.
5. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to the values stored from Step 4, the PublishAddress field set to the address of the Lower Tester, the CredentialFlag field set to 0b0, the RFU field set to 0b111, the Publish TTL field set to 0xFF (use device default TTL), and the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values.
6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field value equal to 0x00, the ElementAddress field value equal to the IUT's unicast address, the PublishAddress field value equal to the address of the Lower Tester, the AppKeyIndex field value equal to the value stored in Step 4, the CredentialFlag field value equal to 0b0, the RFU field value set to 0b000, the Publish TTL field value equal to 0xFF, and the Publish Period, Publish Retransmit Count, Publish Retransmit Interval Steps and Model Identifier fields set to the values received in Step 5.
7. The Lower Tester sends a Config Model Publication Get message with the ElementAddress field value equal to that sent in Step 5 and a Model Identifier field value equal to that used to set the Publication in Step 5.
8. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with all the fields the same as that in Step 6.
9. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to values stored from Step 4, a PublishAddress equal to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b111, Publish TTL field set to a random number between 0x00 and 0x7F, and the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values.

10. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, the ElementAddress field equal to the IUT's unicast address, PublishAddress field value set to Publish Address for the Model, an AppKeyIndex field value equal to the value stored in Step 4, CredentialFlag field value set to 0b0, RFU field value set to 0b000, Publish TTL field value set to the value in Step 9, and the Publish Period, Publish Retransmit Count, Publish Retransmit Interval Steps and Model Identifier fields set to the values received in Step 9.
11. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with the Element Address equal to that in Step 4, the PublishAddress set to a full 128-bit UUID, the AppKeyIndex field set to 0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values and the Model Identifier value as in Step 4.
12. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, the RFU field value set to 0b000 and all other field values equal to those sent in Step 11.
13. Repeat Step 9, but with PublishAddress set to the unassigned address 0x0000.
14. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00.
15. The Lower Tester sends a Config Model Publication Get message to the IUT with the ElementAddress field value equal to that sent in Step 5 and a Model Identifier field value equal to that used to set the Publication in Step 5.
16. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00, ElementAddress and ModelIdentifier fields are set to the values of the incoming message, and all other fields are set to 0x00.

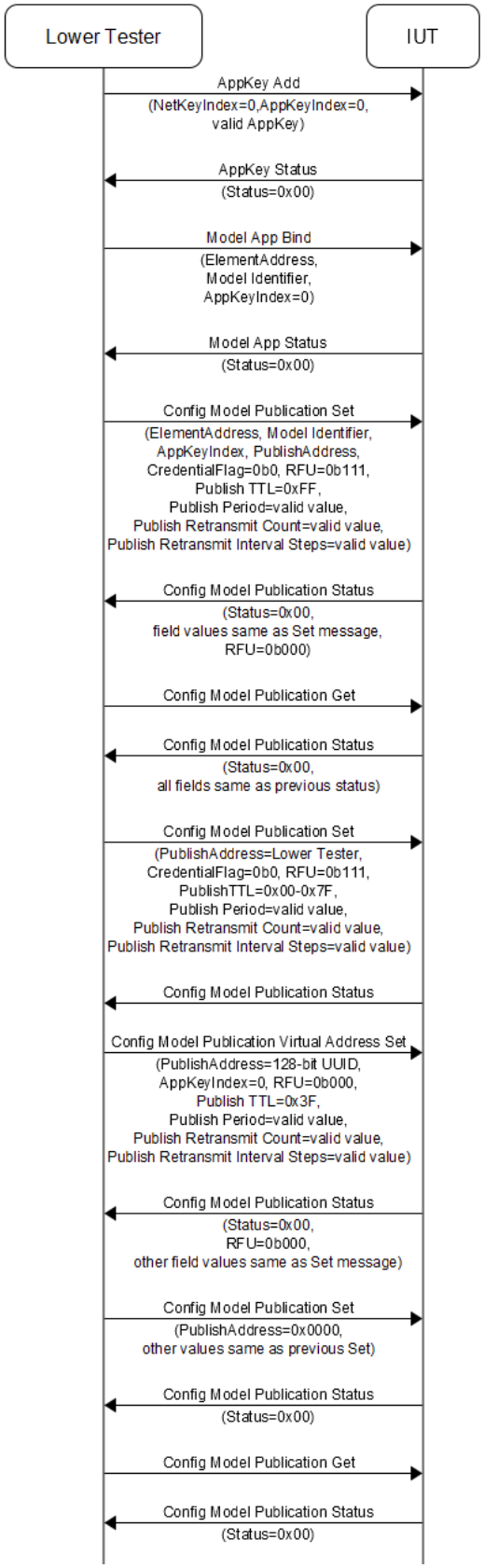


Figure 4.126: MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 6 and Step 8, the IUT sends the Config Model Publication Status message with the Status field set to 0x00, the ElementAddress field set to IUT's unicast address, PublishAddress field set to Publish Address for the Model, AppKeyIndex field value equal to the value stored in Step 4, CredentialFlag field set to 0b0, RFU field set to 0b000, Publish TTL field set to 0xFF, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to the ones requested in Step 5, and the Model Identifier fields set to the Model Identifier as in Step 4.

  In Step 10, the IUT sends the Config Model Publication Status message with the Status field set to 0x00, the ElementAddress field set to IUT's unicast address, PublishAddress field set to Publish Address for the Model, an AppKeyIndex field value equal to the value stored in Step 4, CredentialFlag field set to 0b0, RFU field set to 0b000, Publish TTL field set to the value as in Step 9, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to the ones requested in Step 9, and the Model Identifier field set to the Model Identifier as in Step 9.

  In Step 12, the IUT sends a Config Model Publication Status message with the Status field set to 0x00, the RFU field value set to 0b000, and all other field values matching the values sent in Step 11.

  In Steps 14 and 16, the IUT sends a Config Model Publication Status message with the Status field set to 0x00.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/MP/BV-02-C [Disabling Publishing – AppKey/Config NetKey Deleted]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model disables the publishing of the Status message when the AppKey or NetKey is deleted.

- Reference

  [3] 3.7.4.3, 3.7.6.1.2, 3.7.6.1.3, 4.4.1.2.7, 4.4.1.2.9, 4.4.1.2.10, 4.4.2.2.7

- Initial Condition

  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - IXIT [6] provides ModelID that supports publication with periodic and state change publishing.
  - The IUT has at least two network keys.

- Test Procedure

  1. The Lower Tester runs Step 1 through Step 4 of test case MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] and store the ElementAddress, the Model Identifier of a valid Model on the IUT, and the AppKeyIndex for that Model.
  2. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to the values stored from Step 1, the PublishAddress field set to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b000, the Publish TTL field set to 0xFF (use device default TTL), Publish Period field set to 0x05, the Publish Retransmit Count field set to 0b000, and the Publish Retransmit Interval Steps field set to 0b00000.
  3. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00 and the other fields set to the values received in Step 2.
  4. Follow the instructions in test MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message] to delete the AppKey of the Model.
  5. The Lower Tester sends a Config Model Publication Get message with the ElementAddress field value equal to that sent in Step 2 and a Model Identifier field value equal to that used to set the Publication in Step 2.
  6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with the Status field set to 0x00 and the PublishAddress field set to the unassigned address.
  7. Repeat Steps 1–6, but in Step 4, follow the instructions in another test MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message] to delete the NetKey that the AppKey is bound to (from Step 1).
  8. Repeat Steps 1–6, but in Step 4, execute the procedure described in MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages] to unbind the AppKey from the Model.

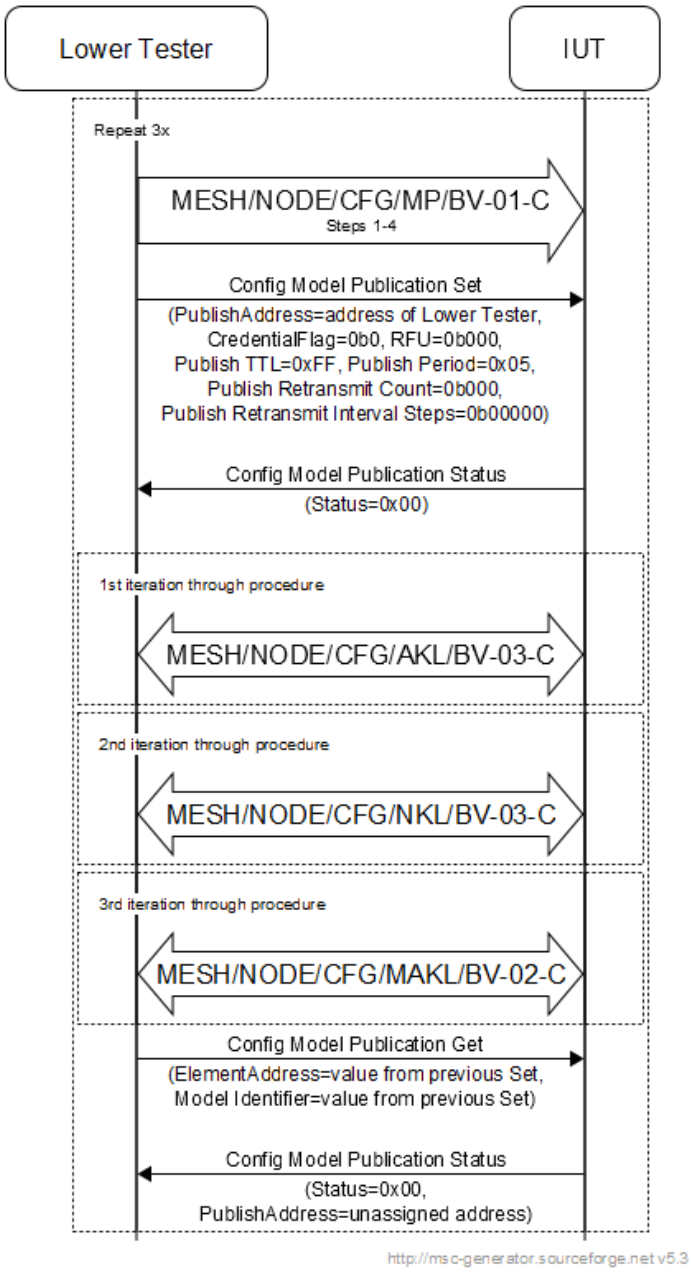


Figure 4.127: MESH/NODE/CFG/MP/BV-02-C [Disabling Publishing – AppKey/Config NetKey Deleted] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Config Model Publication Status message with the Status field set to 0x00 and the other fields set to the values received in Step 2.

  In Step 6, the IUT sends a Config Model Publication Status message with the Status field set to 0x00 and the PublishAddress field set the unassigned address.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MP/BI-01-C [Respond to Config Model Publication Get/Set/Virtual Address Set Messages (Invalid ElementAddress)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Get, Set, and Virtual Address Set messages with an invalid ElementAddress field.

- Reference

  [3] 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition
  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT.

- Test Procedure
  1. The Lower Tester sends a Config Model Publication Set message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table and all other fields set to valid values.
  2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.
  3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table.
  4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.
  5. The Lower Tester sends a Config Model Publication Get message to the IUT with an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table.
  6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

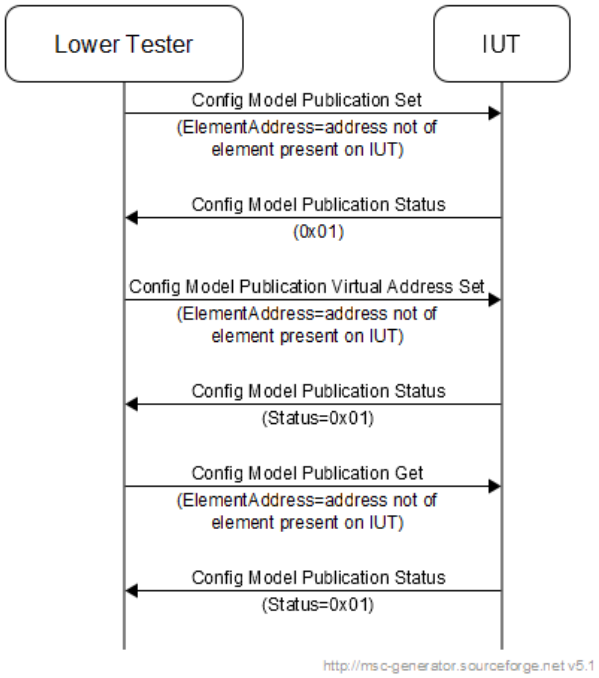


Figure 4.128: MESH/NODE/CFG/MP/BI-01-C [Respond to Config Model Publication Get/Set/Virtual Address Set Messages (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  In Steps 2, 4, and 6, the IUT sends a Config Model Publication Status message with a Status field set to 0x01, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MP/BI-02-C [Respond to Config Model Publication Get/Set/Virtual Address Set Message (Invalid Model Identifier)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Get/Set/Virtual Address Set messages with an invalid Model Identifier field.

- Reference

  [3] 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition

  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.

- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Model list of the IUT.

• Test Procedure

1. The Lower Tester sends a Config Model Publication Set message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table and all other fields set to valid values.
2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).
3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).
5. The Lower Tester sends a Config Model Publication Get message to the IUT with a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
6. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

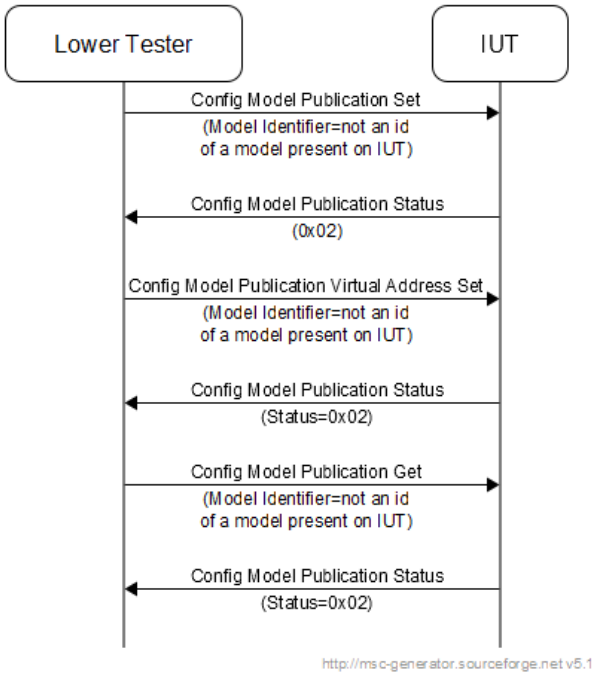


Figure 4.129: MESH/NODE/CFG/MP/BI-02-C [Respond to Config Model Publication Get/Set/Virtual Address Set Message (Invalid Model Identifier)] MSC

• Expected Outcome

Pass verdict

In Steps 2, 4, and 6, the IUT sends a Config Model Publication Status message with a Status field set to

0x02, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MP/BI-03-C [Respond to Config Model Publication Set/Virtual Address Set Messages (Invalid AppKeyIndex)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Publication Set/Virtual Address Set messages with an invalid AppKeyIndex field.

- Reference

  [3] 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition
  - The IUT supports the Configuration Server Model and supports the publish mechanism on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the AppKey list of the IUT.
- Test Procedure
  1. The Lower Tester sends a Config Model Publication Set message to the IUT with a AppKeyIndex value within the range of the field but not one of the AppKeys listed in the IUT's IXIT [6] table and all other fields set to valid values.
  2. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x03 (Invalid AppKey).
  3. The Lower Tester sends a Config Model Publication Virtual Address Set message to the IUT with a AppKeyIndex value within the range of the field but not one of the AppKeys listed in the IUT's IXIT [6] table and all other fields set to valid values.
  4. The Lower Tester expects to receive a Config Model Publication Status message in response from the IUT with a Status field set to 0x03 (Invalid AppKey).
  5. Repeat Steps 1–4 using an AppKeyIndex value that indicates that an AppKey is present in the IUT's list but that has not been bound to the model being configured.

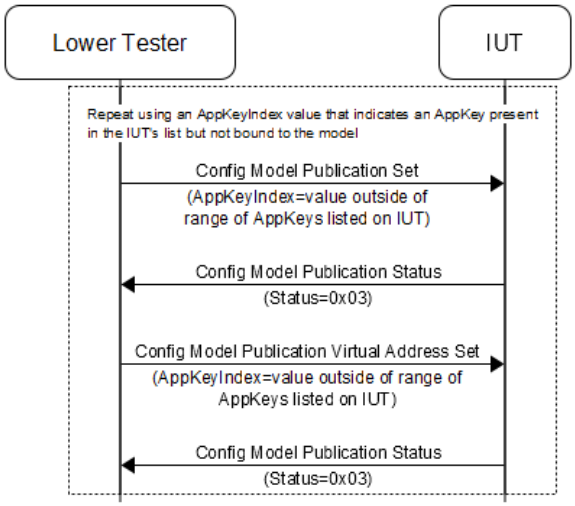


Figure 4.130: MESH/NODE/CFG/MP/BI-03-C [Respond to Config Model Publication Set/Virtual Address Set Messages (Invalid AppKeyIndex)] MSC

- Expected Outcome

  Pass verdict

  In both iterations of Step 2 and Step 4, the IUT sends a Config Model Publication Status message with a Status field set to 0x03, the ElementAddress and Model Identifier fields set to the values of the corresponding fields (i.e., the identically named fields) of the incoming message, and all other fields set to 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/MP/BV-01-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Model Publication Get, Config Model Publication Set and Config Model Publication Virtual Address Set messages to a Configuration Server.

- Reference

  [3] 3.1.2, 3.7.4.3, 4.2.2.1, 4.2.2.3, 4.3.2, 4.3.2.15-4.3.2.18, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

1. The Upper Tester orders the IUT to send a Config Model Publication Get message to the Lower Tester with the Lower Tester's ElementAddress and Model Identifier.
2. The Lower Tester expects to receive a Config Model Publication Get message containing ElementAddress and Model Identifier field values equal to those sent in the Config Model Publication Get message in Step 1.
3. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, the ElementAddress field value as sent in Step 2, the AppKeyIndex field set to 0, the Publish TTL value between 0x00-0x3F, the Publish Period, Publish Retransmit Count and Publish Retransmit Interval Steps fields set to valid values, and the expected Model Identifier as sent in Step 2.
4. The Upper Tester orders the IUT to send a Config Model Publication Set message to the Lower Tester with the ElementAddress value as in Step 1, the PublishAddress set to its own address, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period field set with Step Resolution set to 0b01 and Number of Steps set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in Step 1.
5. The Lower Tester expects to receive a Config Model Publication Set message containing the values as sent in Step 4.
6. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, and all other field values equal to those sent in Step 4.
7. The Upper Tester orders the IUT to send a Config Model Publication Virtual Address Set message to the Lower Tester with the ElementAddress value as in Step 1, the PublishAddress set to a full 128-bit UUID), the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL value set to 0x3F, the Publish Period field set with Step Resolution set to 0b01 and Number of Steps set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in Step 1.
8. The Lower Tester sends a Config Model Publication Status message in response to the IUT with a Status field set to 0x00, and all other field values equal to those sent in Step 7.
9. Repeat Step 4-6 with the PublishAddress field set to a group address.

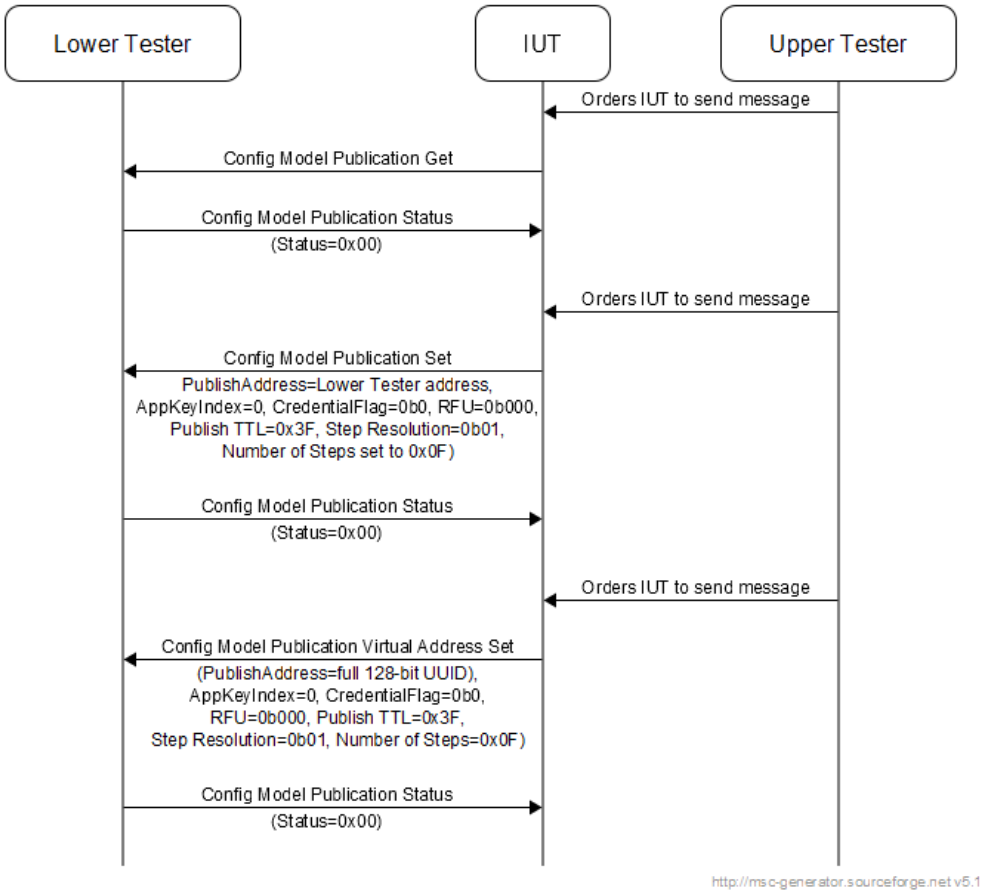


Figure 4.131: MESH/CFGCL/CFG/MP/BV-01-C [Send Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 1, the IUT sends a Config Model Publication Get message to the Lower Tester with the Lower Tester's ElementAddress and Model Identifier.

  In Step 4, the IUT sends a Config Model Publication Set message to the Lower Tester with the ElementAddress value as in Step 1, the PublishAddress set to its own address, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL field value set to 0x3F, the Publish Period field set with the Step Resolution field set to 0b01 and the Number of Steps field set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier field value as in Step 1.

  In Step 7, the IUT sends a Config Model Publication Virtual Address Set message with the ElementAddress field value as in Step 1, the PublishAddress field set to a full 128-bit UUID, the AppKeyIndex field set to 0, the CredentialFlag field set to 0b0, the RFU field set to 0b000, the Publish TTL field value set to 0x3F, the Publish Period field set with the Step Resolution field set to 0b01 and the Number of Steps field set to 0x0F, the Publish Retransmit Count field set to 0b000, the Publish Retransmit Interval Steps field set to 0b00000 and the Model Identifier value as in Step 1.

In Step 9, the IUT sends a Config Model Publication Set message to the Lower Tester with the PublishAddress field set to a group address and all other fields with the same values as in Step 4.

All configuration messages are encrypted and authenticated using the device key of the Lower Tester.

MESH/NODE/CFG/MP/BV-03-C [No Publishing when PublishTTL is set to 0x01]

- Test Purpose

  Verify that the IUT does not publish messages over the air when PublishTTL is set to 0x01.

- Reference

  [3] 3.1.2, 4.2.2.3, 4.3.2, 4.3.2.15, 4.3.2.16, 4.3.2.17, 4.3.2.18, 4.4.1.2, 4.4.1.2.7, 4.4.2.2.7

- Initial Condition

  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials.
  - IXIT [6] provides a Model ID that supports publication (periodic publishing and/or publishing on state change).

- Test Procedure

  1. If the model does not support periodic publishing, skip to Step 6.
  2. The Lower Tester configures the IUT to publish periodically on the available Model ID, with the PublishTTL field set to a value different than 0x01.
  3. The IUT publishes the message of the configured model periodically, with the TTL field set to the value of the PublishTTL state.
  4. The Lower Tester repeats Step 2 using a PublishTTL field value of 0x01.
  5. If the IUT publishes the message of the configured model, the test ends with a Fail verdict.
  6. If the model does not support publishing on state change, skip Steps 7–10.
  7. The Lower Tester configures the IUT to publish on state change on the available Model ID, with the PublishTTL field set to a value different than 0x01.
  8. The IUT is triggered to change the state under test and it publishes the message of the configured model, with the TTL field set to the value of the PublishTTL state.
  9. The Lower Tester repeats Step 7 using a PublishTTL field value of 0x01.
  10. The IUT is again triggered to change the state under test, but if the IUT publishes the message of the configured model, then the test ends with a Fail verdict.

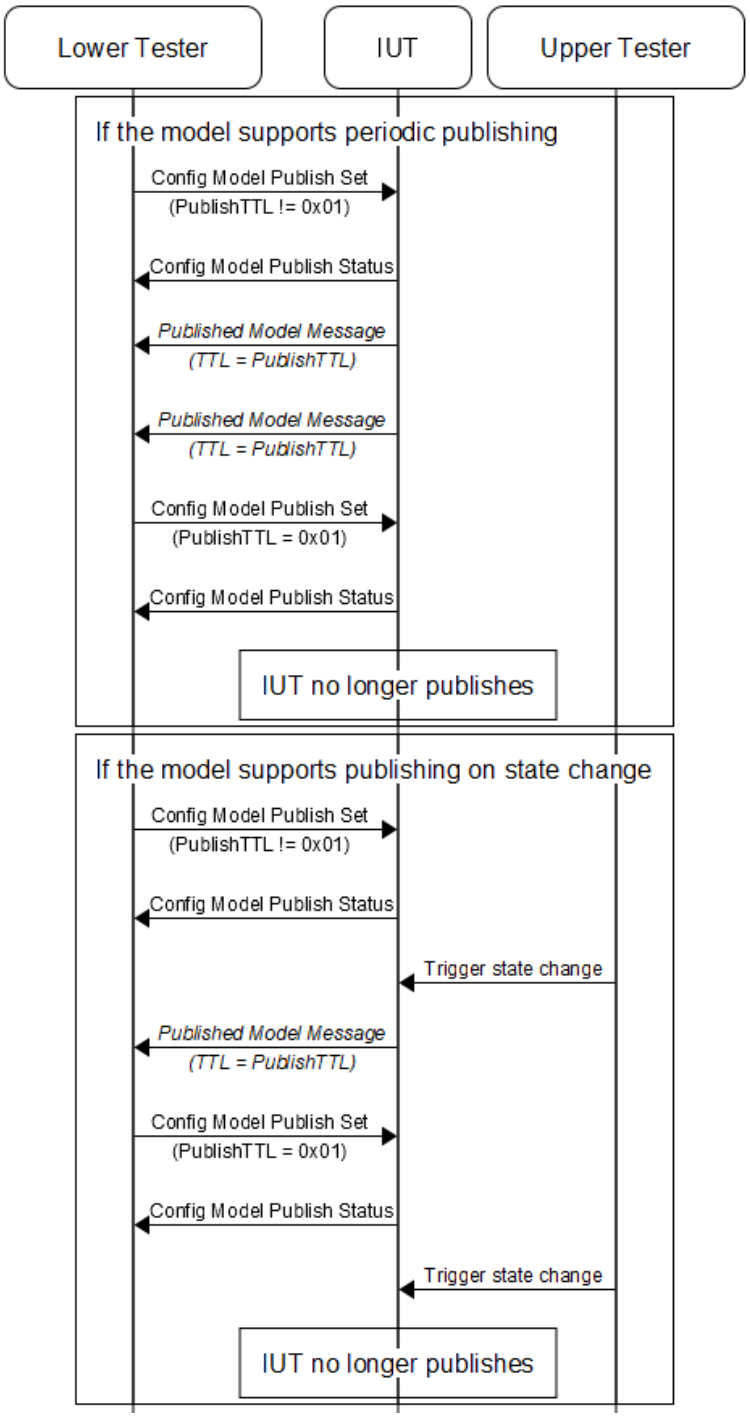


Figure 4.132: MESH/NODE/CFG/MP/BV-03-C [No Publishing when PublishTTL is set to 0x01] MSC

- Expected Outcome

  Pass verdict

  The IUT does not publish messages over the air when the PublishTTL state of the configured model is set to 0x01.

4.16.8 Subscription List Procedures

This section validates the Subscription List procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/SL/BV-01-C [Respond to Config Model Subscription Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Subscription Add/Delete/Delete All/Overwrite messages.

- Reference

  [3] 3.4.2, 3.7.4.3, 4.2.3, 4.3.2.19-4.3.2.30, 4.4.1.2, 4.4.1.2.7, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The IUT has no current subscriptions.
  - IXIT [6] provides ModelID that supports subscription mechanism.

- Test Procedure
  1. The Lower Tester obtains and stores the ElementAddress, and the Model Identifier of a valid Model on the IUT from IXIT [6].
  2. The Lower Tester sends a Config Model Subscription Add message to the IUT with the ElementAddress and Model Identifier values from Step 1 and the Address value of a group address to be added to the Subscription List.
  3. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field and the Model Identifier field set to that in Step 2.
  4. Repeat Steps 2–3 to verify that the IUT responds properly when trying to add the same address twice.
  5. If the Model ID is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the SIG Model ID. If the Model ID is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the Vendor Model ID.
  6. The Lower Tester expects to receive a Config SIG Model Subscription List message (if the message sent in Step 5 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in Step 5 is a Config Vendor Model Subscription Get) in response.
  7. The Lower Tester sends a Config Model Subscription Delete message to the IUT with the ElementAddress, the Model Identifier, and the Address as in Step 2.
  8. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in Step 7.

9. Repeat Step 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm the record is deleted.
10. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status Field set to 0x00 and confirm the record is deleted.
11. Repeat Steps 7–8 to verify that the IUT properly responds when trying to delete the same address twice.
12. Repeat Steps 2–3 to add another address to the Subscription List.
13. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress and the Model Identifier as in Step 2, but a new Address (referred below as new 3) not in Step 12.
14. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in Step 13.
15. Repeat Step 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm that the identified Subscription List is cleared, and the address new 3 in Step 13 is added.
16. Repeat Step 2 to add a new address (say, new 4) not the same as that in Step 13 to the Subscription List. (Now there are two addresses in the Subscription List.)
17. The Lower Tester sends a Config Model Subscription Delete All message to the IUT with the ElementAddress and Model Identifier field values as in Step 16.
18. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with the Status Field set to 0x00, the ElementAddress field and the Model Identifier field set to the incoming message, and the Address field set to unassigned address value 0x0000.
19. Repeat Step 5 to confirm that the identified Subscription List is cleared.
20. The Lower Tester expects to receive a Config SIG Model Subscription List message (if request is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List (if request is Config Vendor Model Subscription Get) in response from the IUT with the Status Field set to 0x00, the ElementAddress field and the Model Identifier field set to the incoming message, and an empty Addresses field.

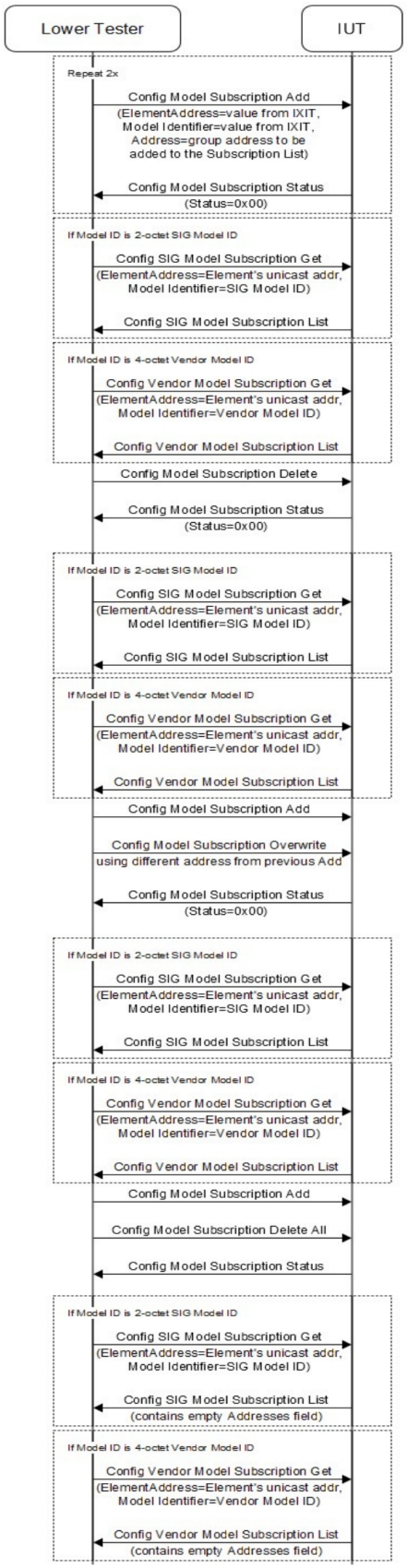


Figure 4.133: MESH/NODE/CFG/SL/BV-01-C [Respond to Config Model Subscription Messages] MSC

- Expected Outcome

  Pass verdict

  In each iteration through Step 3, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the ElementAddress field, and the Address field set to the values in Step 2.

  In Step 5, the IUT sends a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

  In each iteration through Step 8, the IUT sends a Config Model Subscription Status message with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in Step 7.

  In Step 10, IUT sends a Config Model Subscription Status message with the Status Field set to 0x00.

  In Step 14, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the Address, the ElementAddress, and the Model ID fields as defined by the incoming message in Step 13.

  In Step 18, the IUT sends a Config Model Subscription Status message with the Status Field set to 0x00, the ElementAddress field and the Model ID fields set to the incoming message in Step 17, and the Address field set to the unassigned address 0x0000.

  In Step 20, the IUT sends a Config SIG Model Subscription List or a Config Vendor Model Subscription List with an empty Addresses field.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BV-02-C [Respond to Config Model Subscription Virtual Address Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Model Subscription Virtual Address Add/Delete/Overwrite messages.

- Reference

  [3] 4.2.3, 4.3.2.19-4.3.2.30, 4.3.6.26, 4.4.1.2.7, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The IUT has no current subscriptions.

- IXIT [6] provides ModelID that supports subscription mechanism.

- Test Procedure

1. The Lower Tester obtains and stores the ElementAddress, and the Model Identifier of a valid Model on the IUT from IXIT [6].
2. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with the ElementAddress and Model Identifier values from Step 1 and the Address the Label UUID used as the new Address to be added to the Subscription List.
3. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field with the virtual address, and the Model Identifier field value set to that in Step 2.
4. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model Subscription Get with the ElementAddress field the unicast address of the Element and the Model Identifier field the SIG Model ID. If the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model Subscription Get with the ElementAddress field set to the unicast address of the Element and the Model Identifier field set to the Vendor Model ID.
5. The Lower Tester expects to receive a Config SIG Model Subscription List message (if the message sent in Step 4 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in Step 4 is a Config Vendor Model Subscription Get) in response.
6. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the ElementAddress, the Model Identifier, and the Address as in Step 2.
7. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in Step 6.
8. Repeat Steps 4 and 5 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm the record is deleted.
9. Repeat Step 2 to add a new Label UUID address (say, new 2) to the Subscription List.
10. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the ElementAddress, and the Model Identifier as in Step 2, but the value of a new Label UUID not in Step 9.
11. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, ElementAddress, and Model Identifier fields as defined by the incoming message in Step 10. The Address field is set to the virtual address which references the Label UUID used in Step 10.
12. Repeat Step 4 (the Lower Tester sending a Config SIG Model Subscription Get or a Config Vendor Model Subscription Get) to confirm that the identified Subscription List has one entry equal to Label UUID from Step 10.

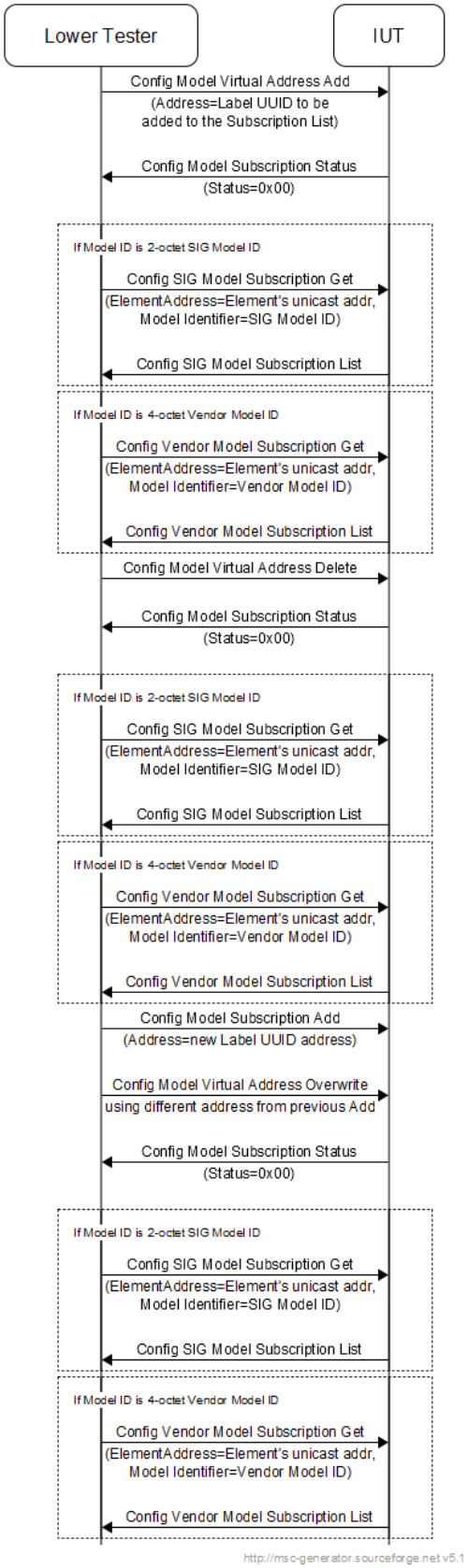


Figure 4.134: MESH/NODE/CFG/SL/BV-02-C [Respond to Config Model Subscription Virtual Address Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 3 the IUT sends a Config Model Subscription Status message with the Status field value of 0x00, the ElementAddress field and the Address field set to the values in Step 2.

  In Step 4 the IUT sends a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

  In Step 5, the IUT sends a Config SIG Model Subscription List message (if the message sent in Step 4 is Config SIG Model Subscription Get) or a Config Vendor Model Subscription List message (If the message sent in Step 4 is a Config Vendor Model Subscription Get) in response with the current values of the identified Subscription List state (added in Step 2), setting the ElementAddress and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00. The Addresses field is set to the value of the virtual address which references the Label UUID used in Step 2.

  In Step 7, the IUT sends a Config Model Subscription Status message with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as defined by the incoming message in Step 6.

  In Step 9, IUT sends a Config Model Subscription Status message with a Status Field set to 0x00.

  In Step 11, the IUT sends a Config Model Subscription Status message with a Status field value of 0x00, the Address, the ElementAddress, and the Model Identifier fields as defined by the incoming message in Step 11.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BV-03-C [Respond to Config Vendor Model Subscription Get Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Vendor Model Subscription Get messages.

- Reference

  [3] 3.7.4.3, 4.2.3, 4.3.2.19, 4.3.2.23, 4.3.2.26, 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.

- The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6], which supports subscription mechanism.

- Test Procedure

1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress field and the Model Identifier field set to a valid value, and the Address field set to a valid group address.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the Address, ElementAddress, and Model Identifier fields as in Step 1.
3. The Lower Tester sends a Config Model Subscription Add message to the IUT with the same ElementAddress and Model Identifier values as in Step 1 and the Address field set to another group address different from the one in used Step 1.
4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x00, the ElementAddress field, the Address field, and the Model Identifier field set to that in Step 3.
5. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with the ElementAddress and Model Identifier of the Vendor Model identified in the IXIT [6].
6. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x00, the ElementAddress and Model Identifier field values equal to the values sent in Step 1, and the addresses of the two group addresses added in Step 1 and Step 3.

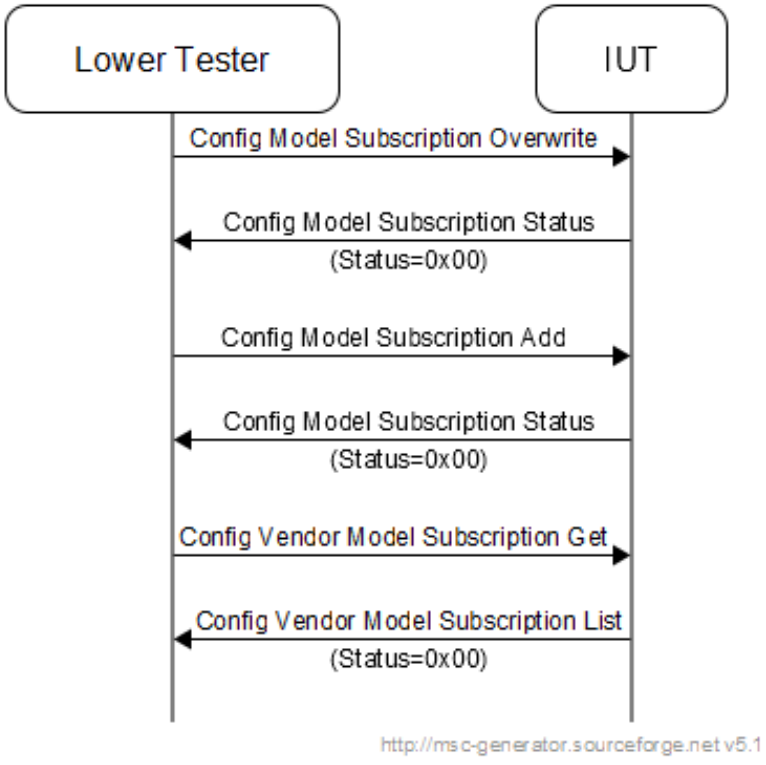


Figure 4.135: MESH/NODE/CFG/SL/BV-03-C [Respond to Config Vendor Model Subscription Get Message] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Vendor Model Subscription List message in response to the Lower Tester with the Status field set to 0x00, the ElementAddress and Model Identifier field values equal to the values sent in Step 1, and the addresses of the two group addresses added in Step 1 and Step 3.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-01-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid ElementAddress)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Add message or a Config Model Subscription Virtual Address Add message with an invalid ElementAddress field.

- Reference

  [3] 4.3.2.19, 4.3.2.20, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT.

- Test Procedure
  1. The Lower Tester sends a Config Model Subscription Add message to the IUT with a valid 16-bit Model Identifier value, the Address field set to a valid group address, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).
  3. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with ElementAddress field and Model Identifier field set to the same as in Step 1 and the Address field the Label UUID used as the new address to be added to the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).

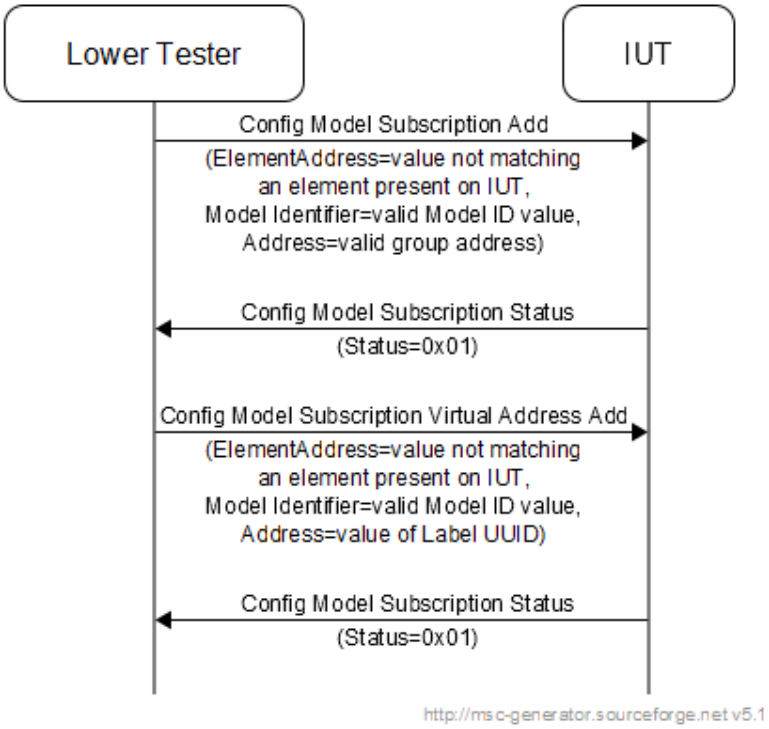


Figure 4.136: MESH/NODE/CFG/SL/BI-01-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  In Step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01, the Address field set to the unassigned address, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-02-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid Model Identifier)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Add message or a Config Model Subscription Virtual Address Add message with an invalid Model Identifier field.

- Reference

  [3] 4.3.2.19, 4.3.2.20, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Model list of the IUT.
- Test Procedure
  1. The Lower Tester sends a Config Model Subscription Add message to the IUT with a valid ElementAddress, the Address field set to a valid group address, and a Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).
  3. The Lower Tester sends a Config Model Subscription Virtual Address Add message to the IUT with the ElementAddress field and Model Identifier field set to the same as in Step 1 and the Address field the Label UUID used as the new address to be added to the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

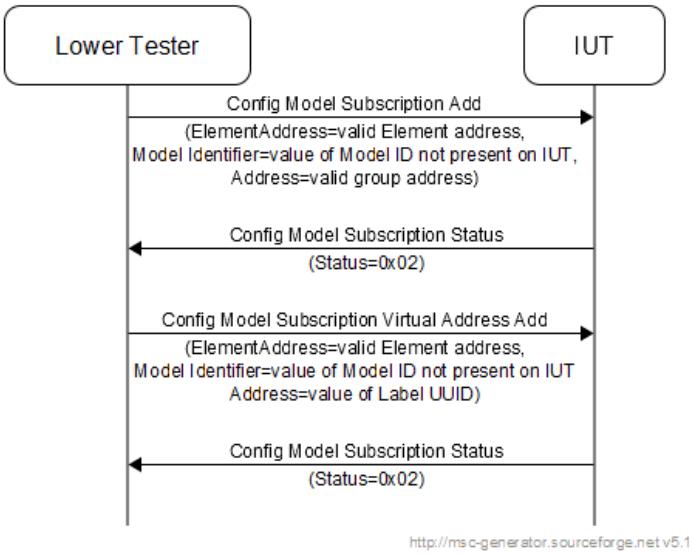


Figure 4.137: MESH/NODE/CFG/SL/BI-02-C [Respond to Config Model Subscription Add / Virtual Address Add Message (Invalid Model Identifier)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In Step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-03-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid ElementAddress)]

- Test Purpose

  Verify that the IUT can responds with the correct error status to a Config Model Subscription Delete message or a Config Model Subscription Virtual Address Delete message with an invalid ElementAddress value.

- Reference

  [3] 4.3.2.21, 4.3.2.22, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Element list of the IUT.

- Test Procedure
  1. The Lower Tester sends a Config Model Subscription Delete message to the IUT with a valid 16-bit Model Identifier value, the Address field set to a valid group address, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).
  3. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the ElementAddress field and Model Identifier field set to the same as in Step 1 and the Address field the Label UUID used as the new address to be removed from the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).

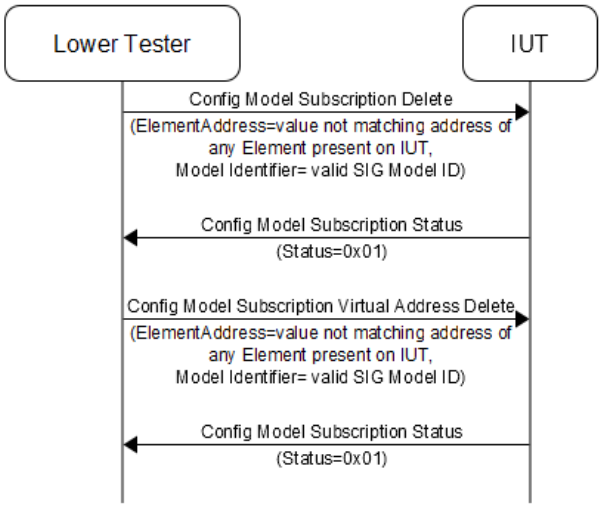


Figure 4.138: MESH/NODE/CFG/SL/BI-03-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01 (Invalid Address), and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  In Step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x01 (Invalid Address), the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-04-C [Respond to Config SIG Model Subscription Get Message (Invalid ElementAddress)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config SIG Model Subscription Get message with an invalid ElementAddress field.

- Reference

  [3] 4.3.2.27, 4.3.2.28, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.

- The Lower Tester has access to the Element list of the IUT.
- The IUT has one or more SIG Models stored on the device, the 16-bit SIG Model IDs.

- Test Procedure

  1. The Lower Tester sends a Config SIG Model Subscription Get message to the IUT with a valid 16-bit SIG Model Identifier value, and an ElementAddress value within the range of the field but not one of the Elements listed in the IUT's IXIT [6] table.
  2. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier field values equal to the values sent in Step 1, and the Addresses field set to an empty list.

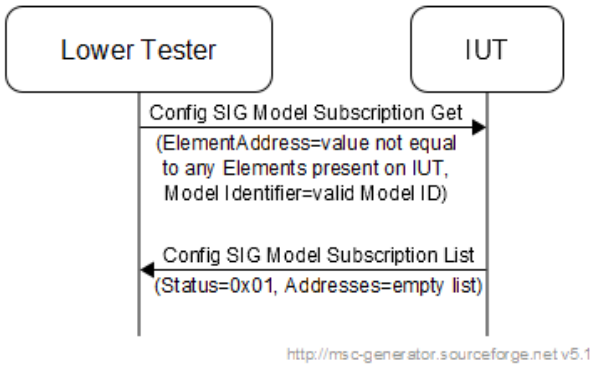


Figure 4.139: MESH/NODE/CFG/SL/BI-04-C [Respond to Config SIG Model Subscription Get Message (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config SIG Model Subscription List message with the Status field set to 0x01, the ElementAddress and the Model Identifier field values equal to the values sent in Step 1, and the Addresses field set to an empty list.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-05-C [Respond to Config SIG Model Subscription Get Message (Invalid Model Identifier)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config SIG Model Subscription Get message with an invalid Model Identifier field.

- Reference

  [3] 4.3.2.27, 4.3.2.28, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition

- The IUT supports the Configuration Server Model on the primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Model list of the IUT.
- The IUT has one or more SIG Models stored on the device, the 16-bit SIG Model IDs.

- Test Procedure

1. The Lower Tester sends a Config SIG Model Subscription Get message to the IUT with a valid ElementAddress value, and a 16-bit Model Identifier value not matching any SIG Model identified in the IXIT [6] a valid value.
2. The Lower Tester expects to receive a Config SIG Model Subscription List message in response from the IUT with a Status field set to 0x02 (Invalid Model), an ElementAddress and the Model Identifier field values equal to the values sent in Step 1 and the Addresses field set to an empty list.

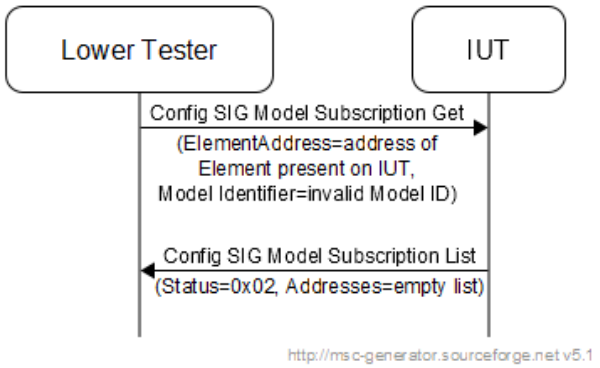


Figure 4.140: MESH/NODE/CFG/SL/BI-05-C [Respond to Config SIG Model Subscription Get Message (Invalid Model Identifier)] MSC

- Expected Outcome

Pass verdict

The IUT sends a Config Model Subscription List message with the Status field set to 0x02, the ElementAddress field and the Model Identifier field equal to that in Step 1, and the Addresses field set to an empty list.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-06-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid Model Identifier)]

- Test Purpose

Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Delete message or a Config Model Subscription Virtual Address Delete message with an

invalid Model Identifier field.

- Reference

  [3] 4.3.2.21, 4.3.2.22, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Model list of the IUT.
- Test Procedure
  1. The Lower Tester sends a Config Model Subscription Delete message to the IUT with a valid ElementAddress, the Address field set to a group address to be removed from IUT's subscription list, and Model Identifier value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model), an Address field equal to the group address set in Step 1.
  3. The Lower Tester sends a Config Model Subscription Virtual Address Delete message to the IUT with the same ElementAddress and Model Identifier as that in Step 1 and the Address field set to a Label UUID which identifies the Address to be removed from the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

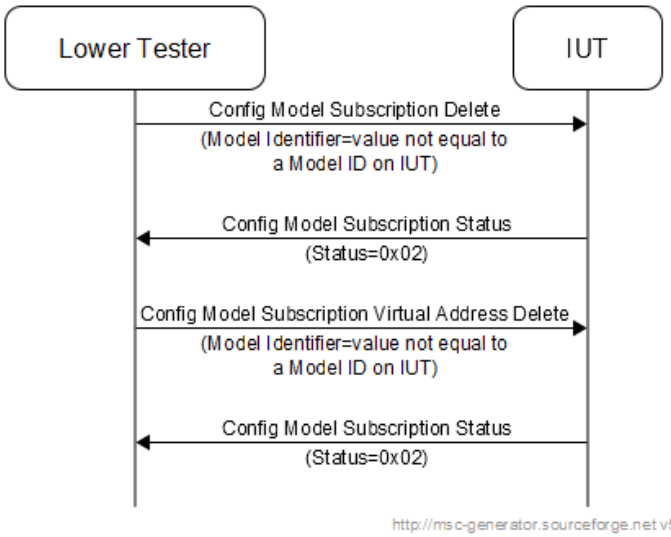


Figure 4.141: MESH/NODE/CFG/SL/BI-06-C [Respond to Config Model Subscription Delete / Virtual Address Delete Message (Invalid Model Identifier)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address fields set to the group address received in Step 1, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  In Step 4, the IUT sends a Config Model Subscription Status message with the Status field set to 0x02, the Address fields set to the unassigned address, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-07-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid ElementAddress)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Overwrite message or a Config Model Subscription Virtual Address Overwrite message with an invalid ElementAddress field.

- Reference

  [3] 4.3.2.23, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The Lower Tester has no current subscriptions.

- Test Procedure
  1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with an ElementAddress field value not of an Element on the IUT, a valid 16-bit Model Identifier value, and the Address value of a valid group address.
  2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x01 (Invalid Address), an ElementAddress field with a value equal to that in Step 1, Address field with a value of a valid group address, and a Model Identifier field value equal to that in Step 1.
  3. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the same ElementAddress and Model Identifier as that in Step 1, and the Address field set to a Label UUID which identifies the Address to be added to the Subscription List.
  4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x01 (Invalid Address).

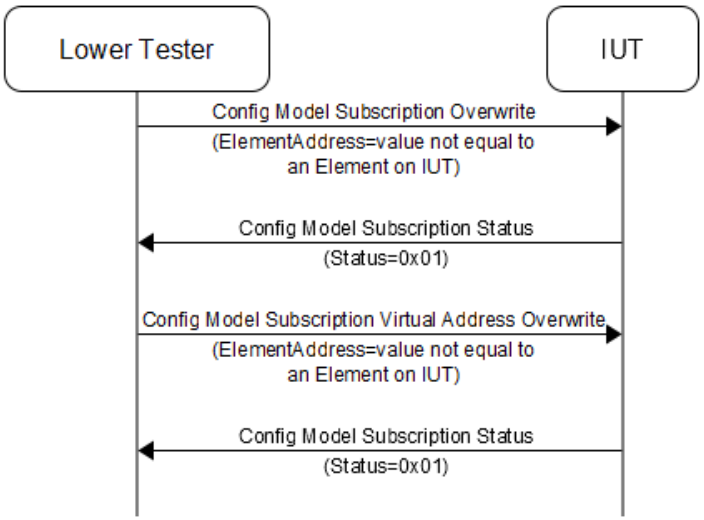


Figure 4.142: MESH/NODE/CFG/SL/BI-07-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Status message with the Status field value of 0x01, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  In Step 4, the IUT sends a Config Model Subscription Status message with the Status field value of 0x01, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-08-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid Model Identifier)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model Subscription Overwrite message or a Config Model Subscription Virtual Address Overwrite message with an invalid Model Identifier field.

- Reference

  [3] 4.3.2.23, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.

- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has no current subscriptions.

- Test Procedure

1. The Lower Tester sends a Config Model Subscription Overwrite message to the IUT with the ElementAddress field value of the primary Element on the IUT, a valid 16-bit Model Identifier value that is not valid for the IUT (or an invalid Model Identifier not defined, such as 0x0004, if the IUT supports all the defined Model Identifier), and the Address field set to a valid group address.
2. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status Field set to 0x02 (Invalid Model), and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.
3. The Lower Tester sends a Config Model Subscription Virtual Address Overwrite message to the IUT with the same parameters as that in Step 1.
4. The Lower Tester expects to receive a Config Model Subscription Status message in response from the IUT with a Status field set to 0x02 (Invalid Model).

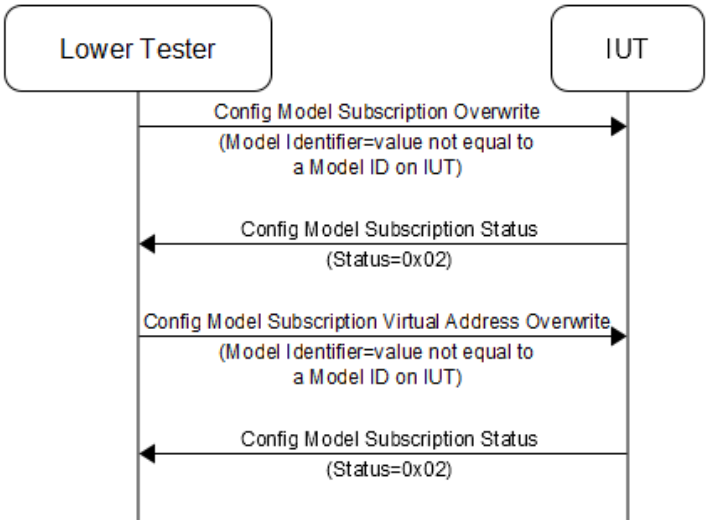


Figure 4.143: MESH/NODE/CFG/SL/BI-08-C [Respond to Config Model Subscription Overwrite / Virtual Address Overwrite Message (Invalid Model Identifier)] MSC

- Expected Outcome

Pass verdict

In Step 2, the IUT sends a Config Model Subscription Status message with the Status field value of 0x02, and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

In Step 4, the IUT sends a Config Model Subscription Status message with the Status field value of 0x02, the Address field set to the unassigned address and other fields to the values of the corresponding fields (i.e., the identically named fields) of the incoming message.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-09-C [Respond to Config Vendor Model Subscription Get Message (Invalid ElementAddress)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can return the correct Status/Error Code to a Config Vendor Model Subscription Get message with an ElementAddress field value that does not match any stored values.

- Reference

  [3] 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The Lower Tester has previously discovered the primary Element address of the IUT and is using that Element Address for all communication.
  - The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.
  - The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6].

- Test Procedure
  1. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with a valid Vendor Model a valid Model Identifier value, and an ElementAddress value not in the Element address range of the IUT identified in the IXIT [6].
  2. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x01 (Invalid Address), the ElementAddress and Model Identifier field values equal to the values sent in Step 1, and the Addresses field set to an empty list.

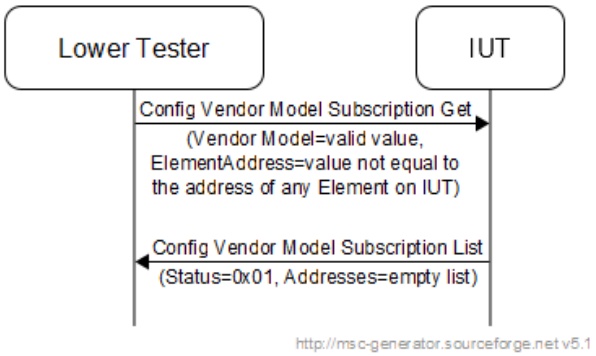


Figure 4.144: MESH/NODE/CFG/SL/BI-09-C [Respond to Config Vendor Model Subscription Get Message (Invalid ElementAddress)] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Vendor Model Subscription List message with the Status field set to 0x01, the ElementAddress field and the Model Identifier field value equal to that in Step 1, and the Addresses field set to an empty list.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/SL/BI-10-C [Respond to Config Vendor Model Subscription Get Message (Invalid Model Identifier)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Vendor Model Subscription Get message with an invalid Model Identifier field.

- Reference

  [3] 4.3.2.29, 4.3.2.30, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Server Model.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester knows the ElementAddress and Model Identifier of a Vendor Model stored on the IUT.
  - The IUT has one or more Vendor Models stored on the device, the 32-bit Vendor Model IDs specified in the IXIT [6].
- Test Procedure

1. The Lower Tester sends a Config Vendor Model Subscription Get message to the IUT with an ElementAddress of an element in IUT and a valid Vendor Model Identifier not matching any Vendor Model identified in the IXIT [6].
2. The Lower Tester expects to receive a Config Vendor Model Subscription List message in response from the IUT with the Status field set to 0x02 (Invalid Model), and the ElementAddress and Model Identifier field values equal to the values sent in Step 1 and the Addresses field set to an empty list.

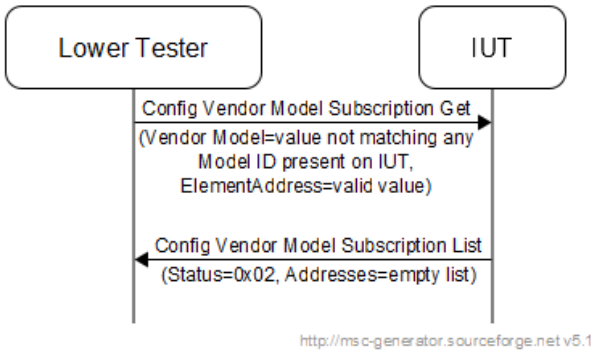


Figure 4.145: MESH/NODE/CFG/SL/BI-10-C [Respond to Config Vendor Model Subscription Get Message (Invalid Model Identifier)] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Vendor Model Subscription List message with the Status field set to 0x02, the ElementAddress field and the Model Identifier field equal to that in Step 1, and the Addresses field set to an empty list.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/SL/BV-01-C [Send Config Model Subscription Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Model Subscription Add/Get/Delete/Delete All/Overwrite messages to a Configuration Server.

- Reference

  [3] 3.7.4.3, 4.3.2.19-4.3.2.30, 4.4.2.2.8

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

1. The Upper Tester orders the IUT to send a Config Model Subscription Add message to the Lower Tester with the ElementAddress and Model Identifier field values and an Address value equal to a group address.
2. The Lower Tester expects to receive a Config Model Subscription Add message.
3. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
4. The Upper Tester orders the IUT to send a Config Model Subscription Delete message to the Lower Tester with the ElementAddress, Address and Model Identifier field values from Step 1.
5. The Lower Tester expects to receive a Config Model Subscription Delete message.
6. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
7. The Upper Tester orders the IUT to send a Config Model Subscription Overwrite message to the Lower Tester with ElementAddress, Address and Model Identifier field values from Step 1.
8. The Lower Tester expects to receive a Config Model Subscription Overwrite message.
9. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
10. The Upper Tester orders the IUT to send a Config Model Subscription Delete All message to the Lower Tester with the ElementAddress as above and a valid Model Identifier.
11. The Lower Tester expects to receive a Config Model Subscription Delete All message.
12. The Lower Tester responds with a Config Model Subscription Status message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Address field to 0x0000, and setting the Status field to 0x00.
13. The Upper Tester orders the IUT to send a Config SIG Model Subscription Get message to the Lower Tester with the ElementAddress field set to the unicast address of the Lower Tester, and the Model Identifier field set to a 2-octet value which identifies the Model within the Element.
14. The Lower Tester expects to receive a Config SIG Model Subscription Get message.
15. The Lower Tester responds with a Config SIG Model Subscription List message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Addresses field to an empty list, and setting the Status field to 0x00.
16. The Upper Tester orders the IUT to send a Config Vendor Model Subscription Get message to the Lower Tester with the ElementAddress field set to the unicast address of the Lower Tester, and the Model Identifier field set to a 4-octet value which identifies the Model within the Element.
17. The Lower Tester expects to receive a Config Vendor Model Subscription Get message.
18. The Lower Tester responds with a Config Vendor Model Subscription List message, setting the ElementAddress and Model Identifier fields as defined by the incoming message, setting the Addresses field to an empty list, and setting the Status field to 0x00.

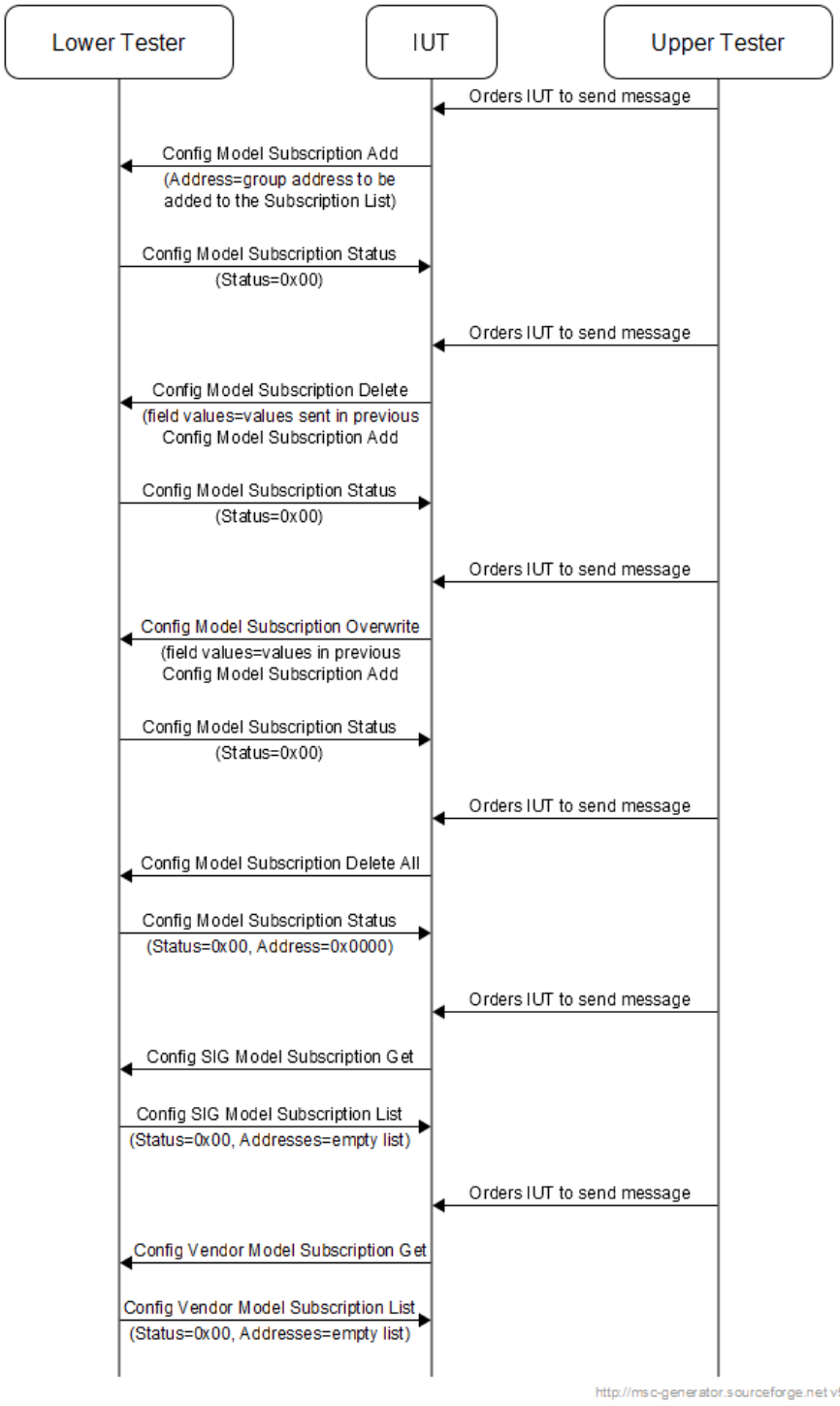


Figure 4.146: MESH/CFGCL/CFG/SL/BV-01-C [Send Config Model Subscription Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Add message to the Lower Tester with the ElementAddress and Model Identifier field values equal to those in Step 1, the Address field equal to a

group address.

In Step 6, the IUT sends a Config Model Subscription Delete message to the Lower Tester with the ElementAddress equal to that of the Lower Tester's, the Address field equal to a group address, and the Model Identifier equal to that of the model on the Lower Tester.
In Step 9, the IUT sends a Config Model Subscription Overwrite message to the Lower Tester with the ElementAddress equal to that of the Lower Tester's, the Address field equal to a group address, and the Model Identifier equal to that of the model on the Lower Tester.

In Step 12, the IUT sends a Config Model Subscription Delete All message to the Lower Tester with the ElementAddress and the Model Identifier field values equal to those in Step 1.

In Step 15, the IUT sends a Config SIG Model Subscription Get message with the ElementAddress equal to that of the Lower Tester's and the Model Identifier equal to the 2-octet value that identifies the Model within the Element on the Lower Tester.

In Step 18, the IUT sends a Config Vendor Model Subscription Get message with the ElementAddress equal to that of the Lower Tester's and the Model Identifier equal to the 4-octet value that identifies the Model within the Element on the Lower Tester.

All configuration messages are encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/SL/BV-02-C [Send Config Model Subscription Virtual Address Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Model Subscription Virtual Address Add/Delete/Overwrite messages to a Configuration Server.

- Reference

  [3] 3.7.4.3, 4.3.2.20, 4.3.2.22, 4.3.2.24, 4.3.2.26, 4.4.1.2.8, 4.4.2.2.8

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Add message to the Lower Tester with the ElementAddress field set to the Lower Tester's unicast address and Model Identifier field values and an Address value equal to a valid 16-octet Label UUID.
  2. The Lower Tester expects to receive a Config Model Subscription Virtual Address Add message.
  3. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

4. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Delete message to the Lower Tester with the ElementAddress, Model Identifier field, and the Address set to the same value as in Step 1.
5. The Lower Tester expects to receive a Config Model Subscription Virtual Address Delete message.
6. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.
7. The Upper Tester orders the IUT to send a Config Model Subscription Virtual Address Overwrite message to the Lower Tester with ElementAddress and Model Identifier field values from Step 1, and an Address value equal to a valid 16-octet Label UUID.
8. The Lower Tester expects to receive a Config Model Subscription Virtual Address Overwrite message.
9. The Lower Tester responds with a Config Model Subscription Status message setting the Address, ElementAddress, and Model Identifier fields as defined by the incoming message and setting the Status field to 0x00.

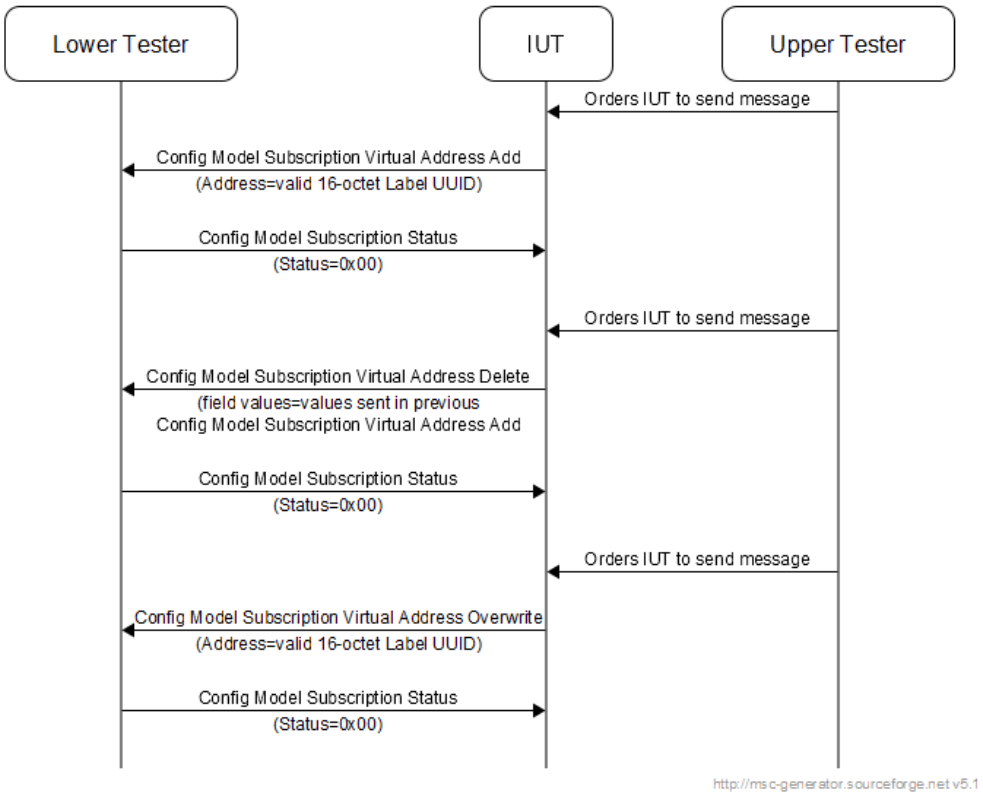


Figure 4.147: MESH/CFGCL/CFG/SL/BV-02-C [Send Config Model Subscription Virtual Address Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model Subscription Virtual Address Add message with the ElementAddress, the Model Identifier and the Address fields values equal to those in Step 1.

In Step 6, the IUT sends a Config Model Subscription Virtual Address Delete message with the ElementAddress, the Model Identifier and the Address fields values equal to those in Step 4.

In Step 9, the IUT sends a Config Model Subscription Virtual Address Overwrite message with the ElementAddress, the Model Identifier and the Address fields values equal to those in Step 7.

All configuration messages are encrypted using the device key of the Lower Tester.

4.16.9 NetKey List Procedures

This section validates the NetKey List procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config NetKey Add and Get Messages.

- Reference

  [3] 3.8.5.1, 4.2.4, 4.3.1.1, 4. 4.3.2.31, 4.3.2.34, 4.3.2.35, 4.3.2.36, 4.4.1.2, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.

- Test Procedure

  1. The Lower Tester sends a Config NetKey Get message to the IUT.
  2. The Lower Tester receives a NetKey List message from the IUT with the NetKeyIndexes field equal to 0x0000.
  3. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 0x000 and the same NetKey value.
  4. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x00 and the NetKeyIndex set to 0x000.
  5. If the IUT does not support a second subnet, skip the following steps.
  6. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 1 (one) and a new NetKey value.
  7. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to

0x00 and the NetKeyIndex set to the value used in Step 6.

8. The Lower Tester sends a Config NetKey Get message to the IUT.
9. The Lower Tester receives a NetKey List message from the IUT using the generic segmentation and reassembly mechanism with the NetKeyIndexes field equal to the two NetKey Indexes present on the IUT.

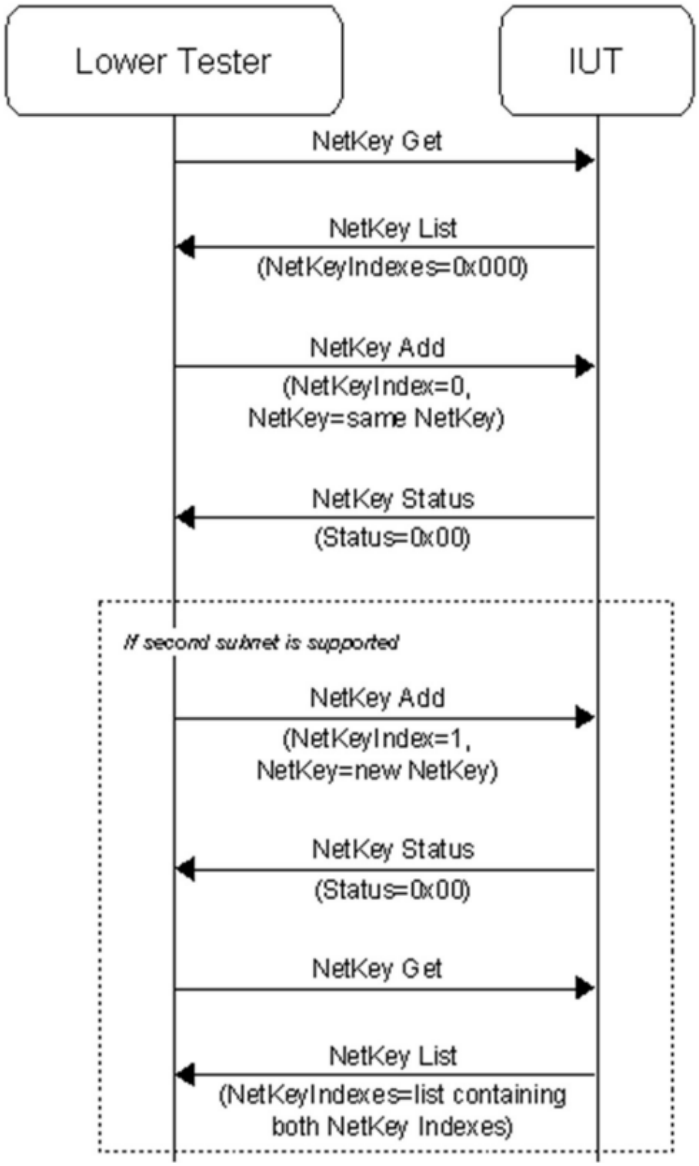


Figure 4.148: MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a NetKey List message with the NetKeyIndexes field indicating only the NetKeyIndex value of 0x000.

In Steps 4 and 7, the IUT sends a Config NetKey Status message with a Status field set to 0x00 and expected NetKeyIndex value of 0x000 and 0x001, respectively.

In Step 9, the IUT sends a NetKey List message with the NetKeyIndexes field indicating both NetKey indexes present on the IUT.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config NetKey Update message.

- Reference

  [3] 4.3.1.1, 4.3.2.31, 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.
  - The IUT is in Key Refresh state 0x00.
- Test Procedure
  1. The Lower Tester sends for a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0 and a new valid NetKey value.
  2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex field set to 0.
  3. Return the IUT to the original state (i.e., newly provisioned state).

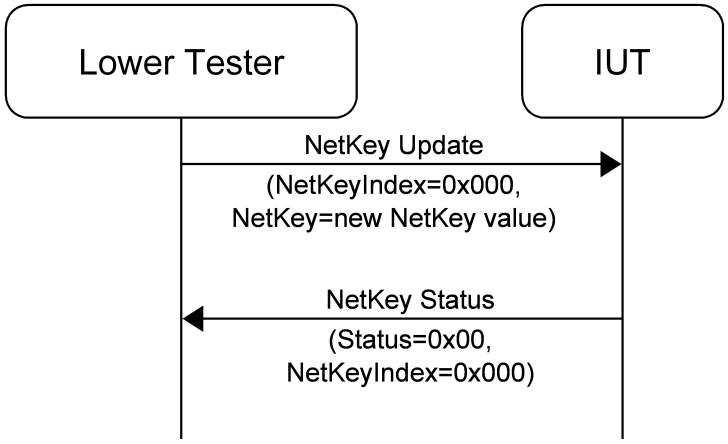


Figure 4.149: MESH/NODE/CFG/NKL/BV-02-C [Respond to Config NetKey Update Message] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config NetKey Status Messages with a Status field set to 0x00 and expected NetKeyIndex field value of 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config NetKey Delete message.

- Reference

  [3] 4.3.2.33, 4.3.2.34, 4.3.2.36, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has a list of two NetKeys with NetKeyIndex of 0 and 1.

- Test Procedure

  1. The Lower Tester sends a Config NetKey Delete message to the IUT with a NetKeyIndex value of 1.
  2. The Lower Tester receives a Config NetKey Status message in response from the IUT with a Status field set to 0x00 and the NetKeyIndex field set to the value used in Step 1.

3. The Lower Tester sends a Config NetKey Get message to the IUT.
4. The Lower Tester receives a NetKey List message from the IUT with the NetKeyIndexes field equal to 0.
5. The Lower Tester repeats Steps 1–2, attempting to delete the same key again.

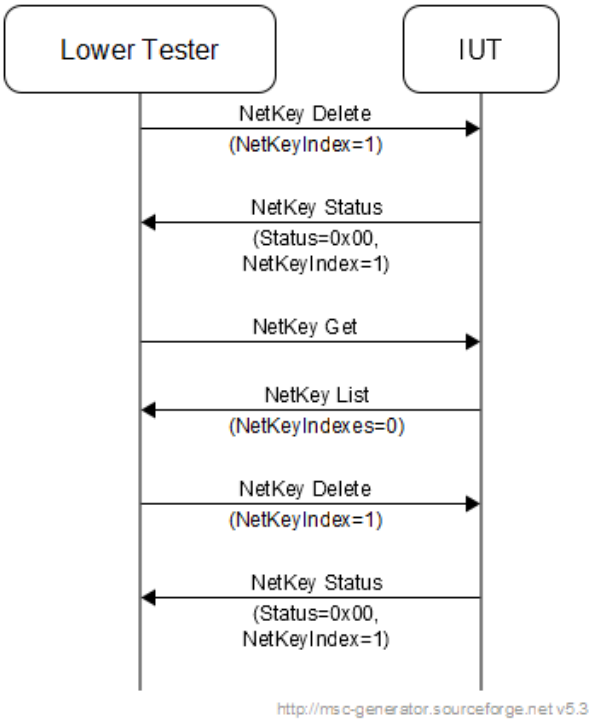


Figure 4.150: MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message] MSC

- Expected Outcome

  Pass verdict

  In both iterations of Step 2, the IUT sends a Config NetKey Status message with a Status field set to 0x00 and a NetKeyIndex field value of 1.

  In Step 4, the IUT sends a NetKey List message with the NetKeyIndexes field equal to 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BV-04-C [Receive Config NetKey Delete Message for Last NetKey]

- Test Purpose

  Verify that the IUT supporting the Configuration Server Model with the last NetKey present can respond to a Config NetKey Delete message.

- Reference

  [3] 4.2.4, 4.3.2.33, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.
- Test Procedure
  1. The Lower Tester sends a Config NetKey Delete message to the IUT with a NetKeyIndex value of 0.
  2. The Lower Tester receives a Config NetKey Status message from the IUT with the Status field set to 0x0C (Cannot Remove), and the NetKeyIndex field set to 0.

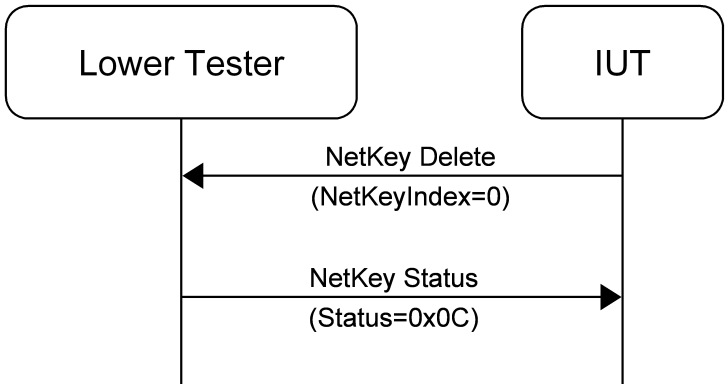


Figure 4.151: MESH/NODE/CFG/NKL/BV-04-C [Receive Config NetKey Delete Message for Last NetKey] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the Config NetKey Status message received from the IUT with the Status field set to 0x0C, and the NetKeyIndex field set to 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BI-01-C [Respond to Config NetKey Update Message (Invalid NetKeyIndex)]

- Test Purpose

  Verify that the IUT can respond with the correct error status to a Config NetKey Update message with an invalid NetKeyIndex value.

- Reference

  [3] 4.3.1.1, 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.
- Test Procedure
  1. The Lower Tester sends a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 2 and a valid NetKey value.
  2. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex set to value used in Step 1.

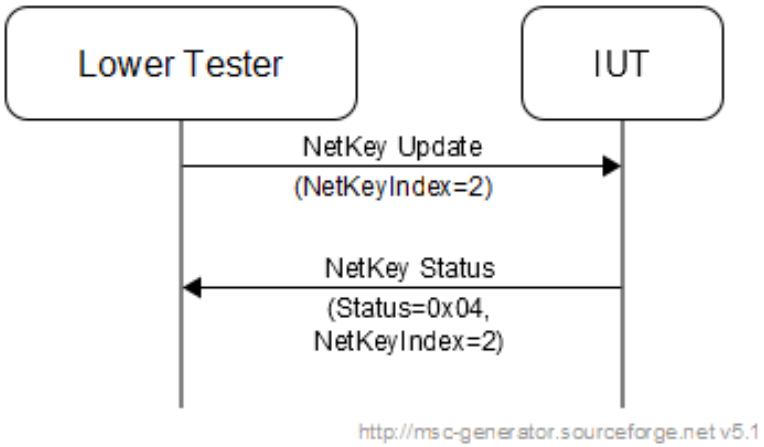


Figure 4.152: MESH/NODE/CFG/NKL/BI-01-C [Respond to Config NetKey Update Message (Invalid NetKeyIndex)] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config NetKey Status message with a Status field set to 0x04 and expected NetKeyIndex in the incoming Config NetKey Update message.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BI-02-C [Respond to Config NetKey Add Message (NetKeyIndex Already Stored)]

- Test Purpose

  Verify that the IUT can respond with the correct error status to a Config NetKey Add message with a

NetKeyIndex already stored.

- Reference

  [3] 4.3.2.31, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on the primary subnet.

- Test Procedure

  1. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 0 and a different NetKey value.
  2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains the Status field set to 0x06 (Key Index Already Stored), and the NetKeyIndex field set to 0.

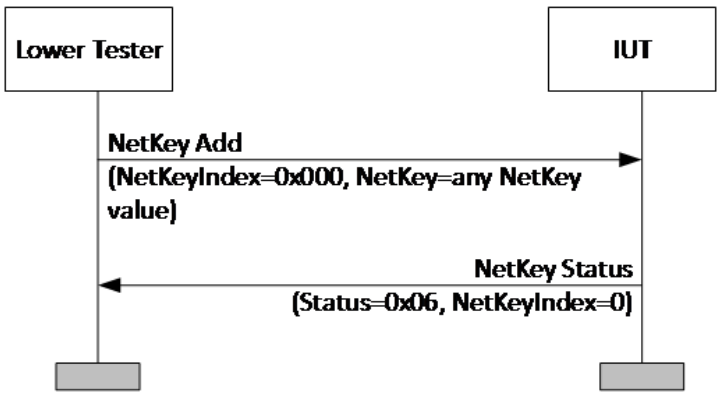


Figure 4.153: MESH/NODE/CFG/NKL/BI-02-C [Respond to Config NetKey Add Message (NetKeyIndex Already Stored)] MSC

- Expected Outcome

  Pass verdict

  The Config NetKey Add is expected to fail and the IUT returns a Config NetKey Status message. The message contains the Status field set to 0x06, and the NetKeyIndex field set to 0.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NKL/BI-03-C [Respond to Config NetKey Add Message (Insufficient Resources)]

- Test Purpose

  Verify that the IUT can respond with the correct error status to a Config NetKey Add message, if it doesn't have sufficient resources to handle it.

- Reference

  [3] 4.3.2.31, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured with the maximum number of network keys supported as defined in the IXIT [6] parameters. The NetKeyIndex value of 1 has not been used.

- Test Procedure
  1. The Lower Tester sends a Config NetKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config NetKey Add message contains a NetKeyIndex value of 1 and a different NetKey value.
  2. The Lower Tester receives a Config NetKey Status message from the IUT. The message contains the Status field set to 0x05, and the NetKeyIndex field set to 1.

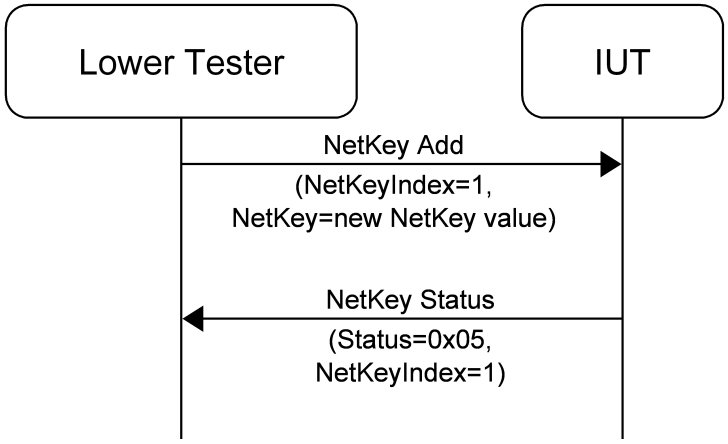


Figure 4.154: MESH/NODE/CFG/NKL/BI-03-C [Respond to Config NetKey Add Message (Insufficient Resources)] MSC

- Expected Outcome

  Pass verdict

  The Config NetKey Add in Step 1 is expected to fail and the IUT returns a Config NetKey Status message. The message contains the Status field set to 0x05, and the NetKeyIndex field set to 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/NKL/BV-01-C [Send Config NetKey Add/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Add and Config NetKey Get messages to a Configuration Server.

- Reference

  [3] 4.3.2.31, 4.3.2.34, 4.3.2.35, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config NetKey Add message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and NetKey.
  2. The Lower Tester receives a Config NetKey Add message using the generic segmentation and reassembly mechanism.

3. The Lower Tester sends a Config NetKey Status message in response to the IUT.
4. The Upper Tester orders the IUT to send a Config NetKey Get message to the Lower Tester.
5. The Lower Tester receives a Config NetKey Get message using the generic segmentation and reassembly mechanism.
6. The Lower Tester sends a NetKey List message in response to the IUT.

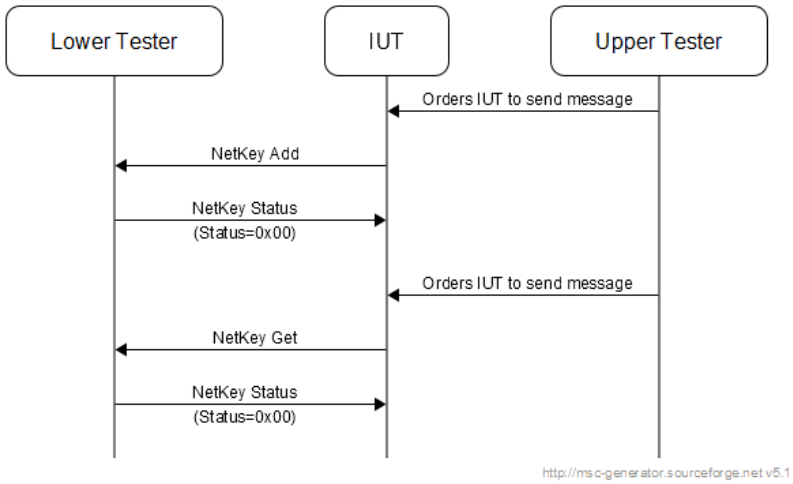


Figure 4.155: MESH/CFGCL/CFG/NKL/BV-01-C [Send Config NetKey Add/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 1, the IUT sends a Config NetKey Add message with a 2-octet unique global NetKeyIndex and the16-octet NetKey. The key is not the same as the one defined in the sample data.

  In Step 4, the IUT sends a Config NetKey Get message.

  The Config NetKey Add message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Update messages to a Configuration Server.

- Reference

  [3] 4.3.2.32, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config NetKey Update message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and a valid NetKey value.
  2. The Lower Tester receives a Config NetKey Update message using the generic segmentation and reassembly mechanism.
  3. The Lower Tester sends a Config NetKey Status message in response to the IUT.

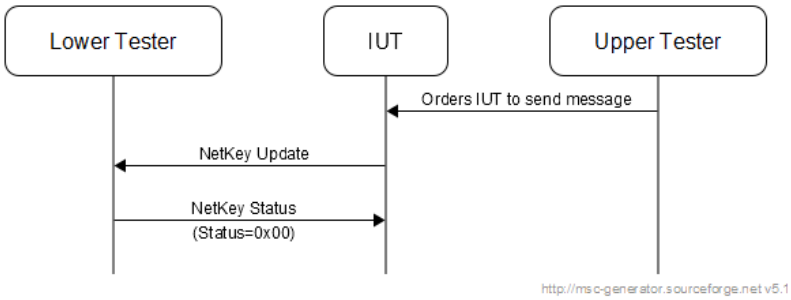


Figure 4.156: MESH/CFGCL/CFG/NKL/BV-02-C [Send Config NetKey Update Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config NetKey Update message with a 2-octet existing global NetKeyIndex and the 16-octet NetKey. The key is not the same as the one defined in the sample data.

  The Config NetKey Update message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/NKL/BV-03-C [Send Config NetKey Delete Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config NetKey Delete messages to a Configuration Server.

- Reference

  [3] 4.3.2.33, 4.3.2.34, 4.4.1.2.9, 4.4.2.2.9

- Initial Condition

  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
  - The Lower Tester has a list of two NetKeys with NetKeyIndex of 0 and 1.

- Test Procedure

1. The Upper Tester orders the IUT to generate a Config NetKey Delete message to the Lower Tester using a NetKeyIndex value of 1.
2. The Lower Tester expects to receive a Config NetKey Delete message from the IUT.
3. The Lower Tester sends a Config NetKey Status message in response to the IUT.

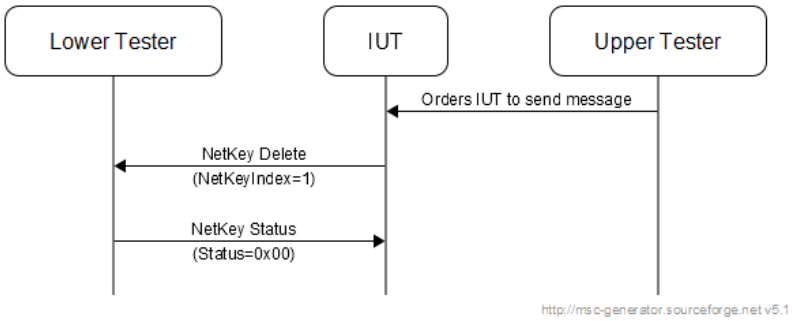


Figure 4.157: MESH/CFGCL/CFG/NKL/BV-03-C [Send Config NetKey Delete Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config NetKey Delete message with a 2-octet existing global NetKeyIndex value of 1.

  The Config NetKey Delete message is encrypted using the device key of the Lower Tester.

4.16.10 AppKey List Procedures

This section validates the AppKey List procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/AKL/BV-01-C [Respond to Config AppKey Add/Get Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Add and Get Messages.

- Reference

  [3] 4.3.2.37, 4.3.2.40, 4.3.2.41, 4.3.2.42, 4.4.1.2, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.

- Test Procedure

  1. The Lower Tester sends a Config AppKey Add message to the IUT with a NetKeyIndex value of 0x123, an AppKeyIndex of 0x456 and a 16-octet AppKey value using the generic segmentation and reassembly mechanism.
  2. The Lower Tester expects to receive a Config AppKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in Step 1.
  3. The Lower Tester sends a Config AppKey Get message to the IUT. The message contains a NetKeyIndex field set to 0x123.
  4. The Lower Tester receives an AppKey List message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex field and the AppKeyIndexes field set to the values used in Step 1.
  5. The Lower Tester repeats Steps 1–2, attempting to add the same application key with the same indexes.

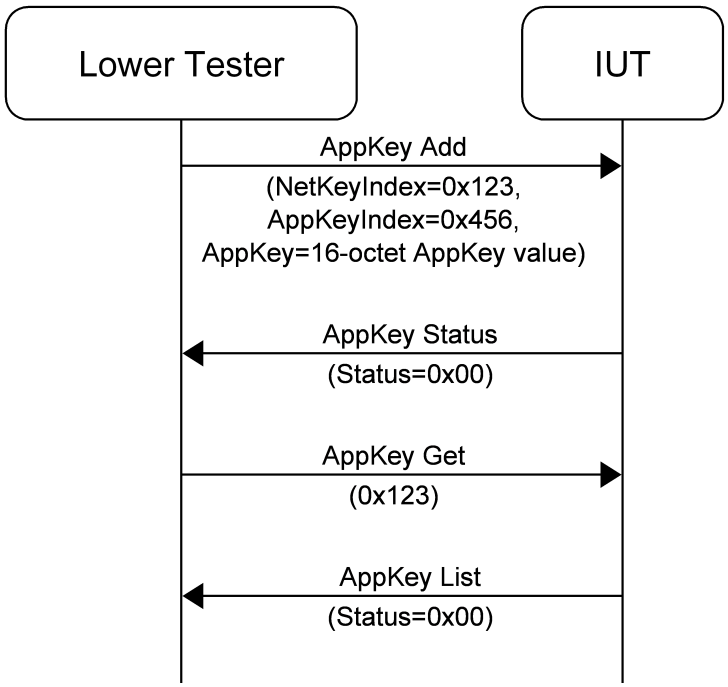


Figure 4.158: MESH/NODE/CFG/AKL/BV-01-C [Respond to Config AppKey Add/Get Message] MSC

- Expected Outcome

  Pass verdict

  In both iterations through Step 2, the IUT sends a Config AppKey Status message with the Status field set to 0x00, the NetKeyIndex value set to 0x123, and the AppKeyIndex value set to 0x456.

  In Step 4, the IUT sends an AppKey List message with a Status field set to 0x00, the NetKeyIndex value set to 0x123, and the AppKeyIndex value set to 0x456.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Update message.

- Reference

  [3] 4.3.2.38, 4.3.2.40, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.

- Test Procedure
  1. The Lower Tester sends a Config AppKey Add message to the IUT using the generic segmentation and reassembly mechanism. The Config AppKey Update message contains a NetKeyIndex value of 0x123 and AppKeyIndex value of 0x456 in the NetKeyIndex and AppKeyIndex field, and a valid new AppKey value in the AppKey field.
  2. The Lower Tester receives a Config AppKey Status message in response from the IUT with a Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.
  3. The Lower Tester sends for a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x0123 and a new valid NetKey value.
  4. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex set to value used in Step 3.
  5. The Lower Tester sends a Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The Config AppKey Update message contains a NetKeyIndex value of 0x123 and AppKeyIndex value of 0x456 in the NetKeyIndex and AppKeyIndex field, and a valid new AppKey value.
  6. The Lower Tester receives a Config AppKey Status message in response from the IUT with the Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.
  7. The Lower Tester sends a Config AppKey Get message to the IUT. The message contains a NetKeyIndex field set to 0x123.
  8. The Lower Tester receives an AppKey List message from the IUT. The message contains a Status field set to 0x00, a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.

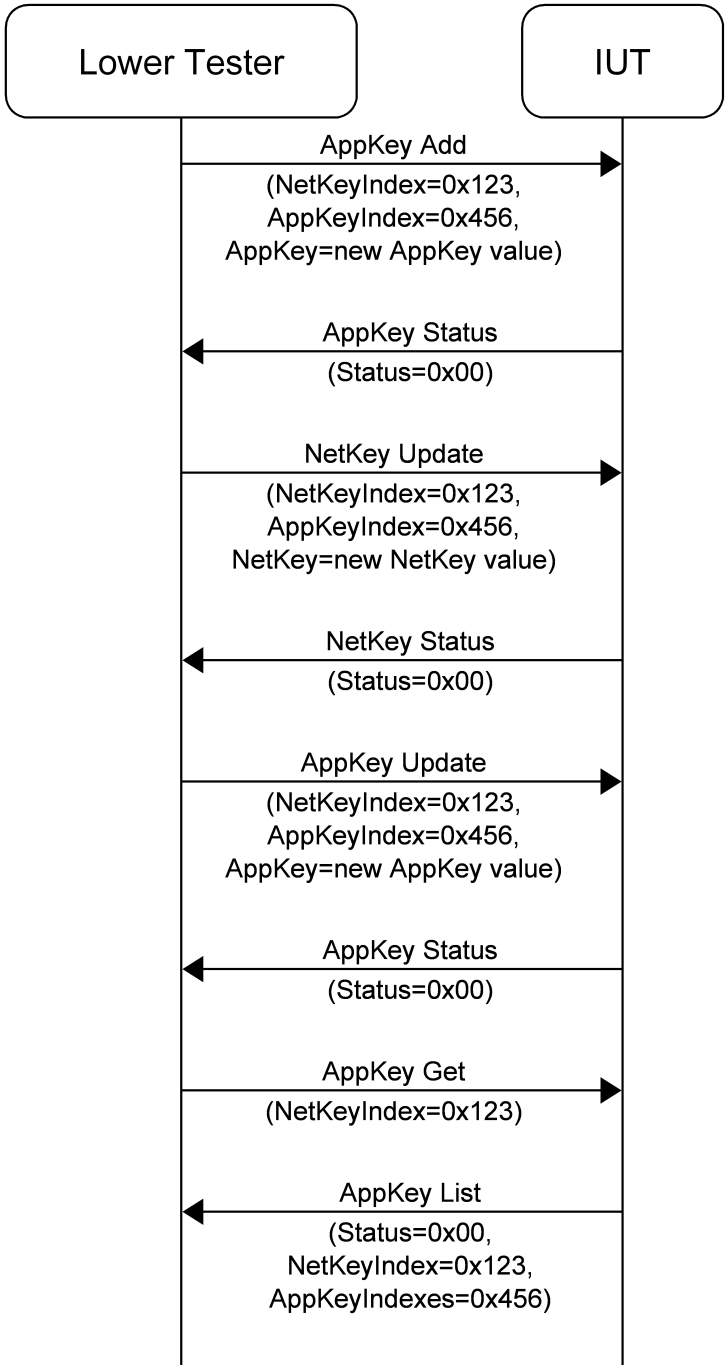


Figure 4.159: MESH/NODE/CFG/AKL/BV-02-C [Respond to Config AppKey Update Message] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 6, the IUT sends a Config AppKey Status message with a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to 0x236145.

  In Step 8, the IUT sends an AppKey List message with the Status field set to 0x00, a NetKeyIndex value of

0x123 and an AppKeyIndex value of 0x456.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Delete message.

- Reference

  [3] 4.3.2.39, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured only on a subnet with the NetKeyIndex set to 0x123.
  - The IUT has a list of at least two AppKeys (with AppKeyIndex field value set to 0x000 and 0x456) bound to the NetKey (with NetKeyIndex field value set to 0x123).

- Test Procedure

  1. The Lower Tester sends a Config AppKey Delete message to the IUT with the NetKeyIndex of the NetKey to which the AppKey is bound and an AppKeyIndex value of 0x123.
  2. The Lower Tester expects to receive a Config AppKey Status message in response from the IUT with a Status field set to 0x00 and the NetKeyIndex and AppKeyIndex field set to value used in Step 1.
  3. The Lower Tester repeats Steps 1–2, attempting to delete the same key.

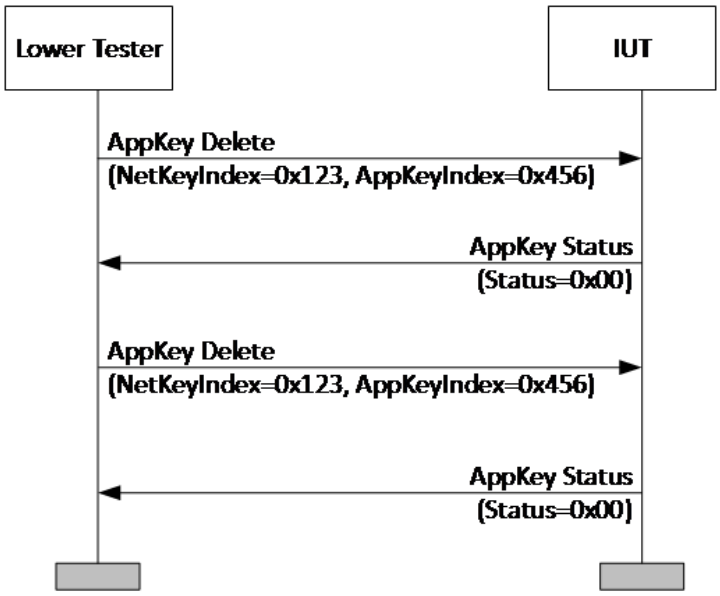


Figure 4.160: MESH/NODE/CFG/AKL/BV-03-C [Respond to Config AppKey Delete Message] MSC

- Expected Outcome

  Pass verdict

  In both iterations through Step 2, the IUT sends the Config AppKey Status message with a Status field set to 0x00 and NetKeyIndex and AppKeyIndex values equal to those received in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BI-01-C [Respond to Config AppKey Add Message (Invalid NetKey Index)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Add message with invalid Config AppKey Add parameters.

- Reference

  [3] 4.3.1.1, 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- IUT does not have AppKeyIndex of 0x000 in AppKey list and does not have NetKeyIndex of 0xFFF in NetKey list.

- Test Procedure

  1. The Lower Tester sends a Config AppKey Add message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0xFFF, an AppKeyIndex value of 0x000, and a valid AppKey value.
  2. The Lower Tester receives a Config AppKey Status message from the IUT. The message contains a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex value matching the value in Step 1.

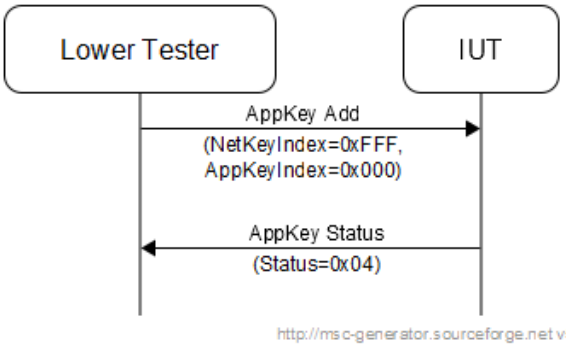


Figure 4.161: MESH/NODE/CFG/AKL/BI-01-C [Respond to Config AppKey Add Message (Invalid NetKey Index)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends the Config AppKey Status message with a Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 1.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BI-02-C [Respond to Config AppKey Update Message (Invalid AppKey Index and NetKey Index)]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config AppKey Update message with invalid Config AppKey Update parameters.

- Reference

  [3] 4.3.1.1, 4.3.2.38, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.

- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT has an AppKey with AppKeyIndex 0x000 bound to NetKeyIndex 0x000.
- If the IUT supports a second subnet, it is also configured on a subnet with the NetKeyIndex of 0x001, and has an AppKey with AppKeyIndex 0x001 bound to NetKeyIndex 0x001.
- The IUT does not have AppKeyIndex of 0xFFF in AppKey list.
- The IUT does not have NetKeyIndex of 0xFFF in NetKey list.

• Test Procedure

1. The Lower Tester sends a Config NetKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x000 and a new valid NetKey value.
2. The Lower Tester expects to receive a Config NetKey Status message from the IUT. The message contains a Status field set to 0x00 and the NetKeyIndex set to value used in Step 1.
3. The Lower Tester sends a Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0x000 and an AppKeyIndex value of 0xFFF.
4. The Lower Tester expects the IUT to send a Config AppKey Status message with a Status field set to 0x03 (Invalid AppKey) and the NetKeyIndex and AppKeyIndex field value matching the value in Step 3.
5. The Lower Tester sends a Config AppKey Update message to the IUT using the generic segmentation and reassembly mechanism. The message contains a NetKeyIndex value of 0xFFF and an AppKeyIndex value of 0x000.
6. The Lower Tester expects the IUT to send a Config AppKey Status message with a Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex field value matching the value in Step 5.
7. If the IUT does not support a second subnet, skip Steps 8–9.
8. The Lower Tester sends a Config AppKey Update message to the IUT. The message contains a NetKeyIndex value of 0x000, an AppKeyIndex value of 0x001 and a valid AppKey.
9. The Lower Tester expects the IUT to send a Config AppKey Status message with a Status field set to 0x11 (Invalid Binding) and the NetKeyIndex and AppKeyIndex field value matching the value in Step 8.

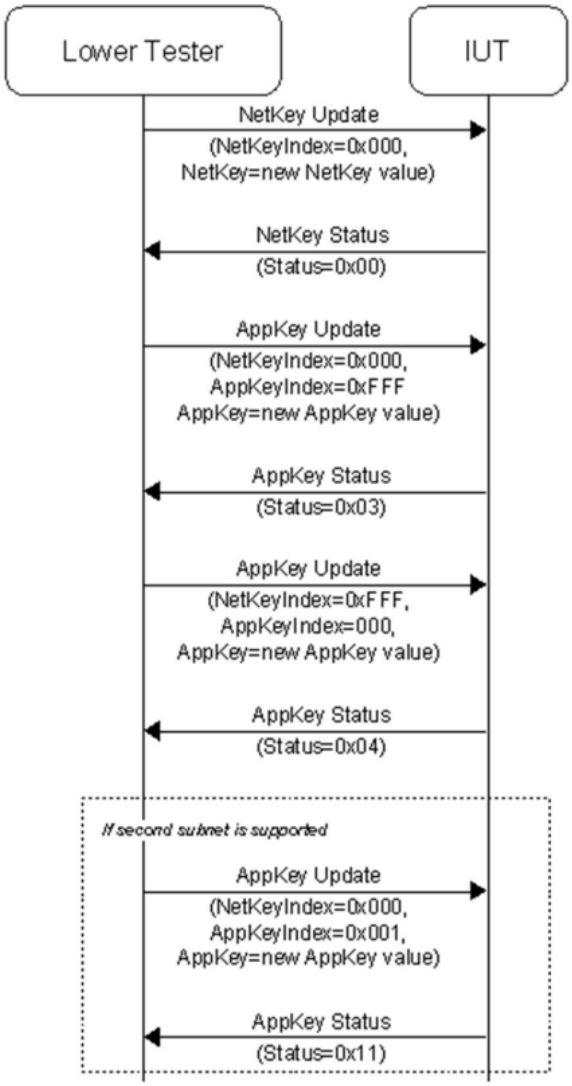


Figure 4.162: MESH/NODE/CFG/AKL/BI-02-C [Respond to Config AppKey Update Message (Invalid AppKey Index and NetKey Index)] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends a Config AppKey Status message with a Status field set to 0x03 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 3.

  In Step 6, the IUT sends a Config AppKey Status message with a Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 5.

  In Step 9, the IUT sends a Config AppKey Status message with a Status field set to 0x11 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 8.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BI-03-C [Respond to Config AppKey Get Message with Invalid NetKey Index]

- Test Purpose

  Verify that a Configuration Server IUT can responds to a Config AppKey Get message with invalid values.

- Reference

  [3] 4.3.2.41, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - IUT does not have NetKeyIndex of 0xFFF in NetKey list.

- Test Procedure

  1. The Lower Tester sends a Config AppKey Get message to the IUT with a NetKeyIndex value different from any of NetKeyindexes stored on the IUT (e.g., 0xFFF).
  2. The Lower Tester expects to receive an AppKey List message in response from the IUT with the Status field set to 0x04 (Invalid NetKey), the NetKeyIndex field as defined by the incoming message, and the AppKeyIndexes field to a zero-length (empty) list.

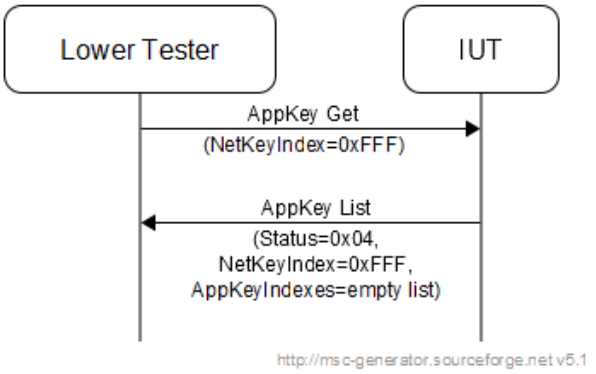


Figure 4.163: MESH/NODE/CFG/AKL/BI-03-C [Respond to Config AppKey Get Message with Invalid NetKey Index] MSC

- Expected Outcome

  Pass verdict

  The IUT sends an AppKey List message with the Status field set to 0x04, the NetKeyIndex field as defined by the incoming message, and the AppKeyIndexes field to a zero-length (empty) list.

All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/AKL/BI-04-C [Respond to Config AppKey Add Message (NetKeyIndex and AppKeyIndex Already Stored)]

- Test Purpose

  Verify that the IUT can respond to a Config AppKey Add message with NetKeyIndex and AppKeyIndex already stored.

- Reference

  [3] 3.8.5, 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT is configured on the primary subnet.
  - If the IUT supports a second subnet, it is also configured on a subnet with the NetKeyIndex of 0x001.
- Test Procedure
  1. The Lower Tester sends a Config AppKey Add message to the IUT with a NetKeyIndex value of 0x000, an AppKeyIndex value of 0x000, and a valid 16-octet AppKey value using the generic segmentation and reassembly mechanism.
  2. The Lower Tester expects the IUT to respond with a Config AppKey Status message with the Status field set to 0x00 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 1.
  3. Repeat Step 1 with the same NetKeyIndex and AppKeyIndex but different AppKey value.
  4. The Lower Tester expects the IUT to respond with a Config AppKey Status message with the Status field set to 0x06 (Key Index Already Stored), and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 3.
  5. If the IUT does not support a second subnet, skip Steps 6–7.
  6. Repeat Step 1 with the same AppKey, the same AppKeyIndex, but NetKeyIndex field set to 0x001.
  7. The Lower Tester expects the IUT to respond with a Config AppKey Status message with the Status field set to 0x04 (Invalid NetKey) and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 5.

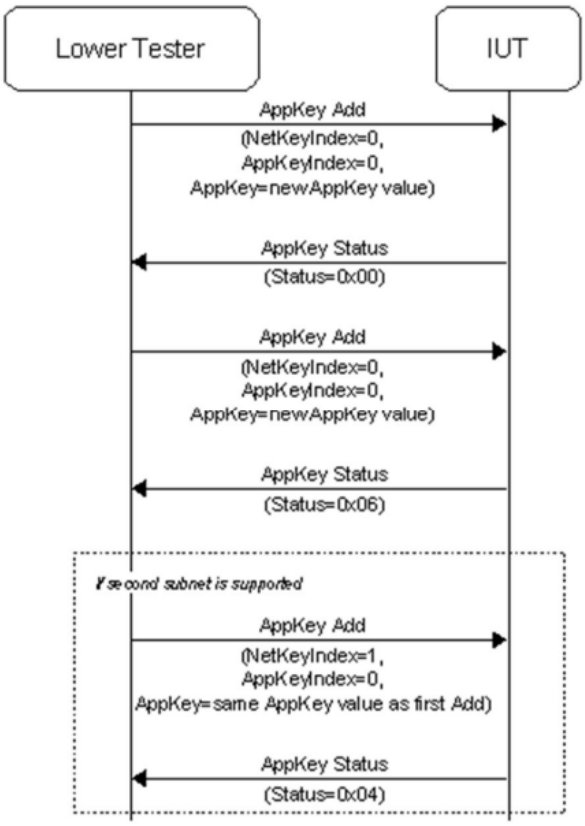


Figure 4.164: MESH/NODE/CFG/AKL/BI-04-C [Respond to Config AppKey Add Message (NetKeyIndex and AppKeyIndex Already Stored)] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config AppKey Status message with the Status field set to 0x00, and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 1.

  In Step 4, the IUT sends a Config AppKey Status message with the Status field set to 0x06, and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 3.

  In Step 6, the IUT sends a Config AppKey Status message with the Status field set to 0x04 and the NetKeyIndex and AppKeyIndex values equal to those sent in Step 5.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/AKL/BV-01-C [Send Config AppKey Add Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Add messages to a Configuration Server.

- Reference

[3] 4.3.2.37, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config AppKey Add message to the Lower Tester with valid NetKeyIndex and AppKeyIndex and a valid 16-octet AppKey value. The message is transmitted via the generic segmentation and reassembly mechanism.
  2. The Lower Tester receives a Config AppKey Add message from the IUT using the generic segmentation and reassembly mechanism.
  3. The Lower Tester sends a Config AppKey Status message in response to the IUT.
- Expected Outcome

  Pass verdict

  The IUT sends a Config AppKey Add message containing a valid NetKeyIndex and AppKeyIndex and a valid 16-byte AppKey value. The key is not the same as the one defined in the sample data.

  The Config AppKey Add message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/AKL/BV-02-C [Send Config AppKey Update Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Update messages to a Configuration Server.

- Reference

  [3] 3.10.4, 4.3.2.38, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config AppKey Update message to the Lower Tester using the generic segmentation and reassembly mechanism. The message contains a valid NetKeyIndex and AppKeyIndex and AppKey value.

2. The Lower Tester receives a Config AppKey Update message using the generic segmentation and reassembly mechanism.
3. The Lower Tester sends a Config AppKey Status message in response to the IUT.

- Expected Outcome

  Pass verdict

  The IUT sends a Config AppKey Update messages with a valid NetKeyIndex and AppKeyIndex and a valid AppKey. The key is not the same as the one defined in the sample data.

  The Config AppKey Update message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/AKL/BV-03-C [Send Config AppKey Delete Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Delete messages to a Configuration Server.

- Reference

  [3] 4.3.2.39, 4.3.2.40, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
  - The Lower Tester has a NetKeyIndex 0x123 with a NetKey and an AppKeyIndex 0x456 with an AppKey bound to the NetKey.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config AppKey Delete message to the Lower Tester with a NetKeyIndex value equal to 0x123 and a AppKeyIndex value equal to 0x456.
  2. The Lower Tester receives a Config AppKey Delete message.
  3. The Lower Tester sends a Config AppKey Status message in response to the IUT.

- Expected Outcome

  Pass verdict

  The IUT sends a Config AppKey Delete message with a NetKeyIndex value of 0x123 and an AppKeyIndex value of 0x456.

  The Config AppKey Delete message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/AKL/BV-04-C [Send Config AppKey Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config AppKey Get messages to a Configuration Server.

- Reference

  [3] 4.3.2.41, 4.3.2.42, 4.4.1.2.10, 4.4.2.2.10

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure
  1. The Upper Tester orders the IUT to send a Config AppKey Get message to the Lower Tester with a NetKeyIndex value of 0x000.
  2. The Lower Tester receives a Config AppKey Get message with the NetKeyIndex field value set to 0x000.
  3. The Lower Tester sends an AppKey List message in response to the Config AppKey Get message.

- Expected Outcome

  Pass verdict

  The IUT sends a Config AppKey Get message with a NetKeyIndex value of 0x000.

  The Config AppKey Get message is encrypted using the device key of the Lower Tester.

4.16.11 Model to AppKey List Procedures

This section validates the Model to AppKey List procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model App Bind and SIG/Config Vendor Model App Get Messages.

- Reference

  [3] 4.3.2.46, 4.3.2.48-4.3.2.52, 4.4.1.2, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has an AppKey in the AppKey list with AppKeyIndex of 0x000.
  - Select any model instance on the IUT that uses application keys and that has not been bound to the AppKey with AppKeyIndex 0x000.
- Test Procedure
  1. The Lower Tester sends a message to the IUT for the selected model instance using AppKeyIndex of 0x000. The IUT does not respond.
  2. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to the Model Identifier of the selected model instance and AppKeyIndex field set to zero (0).
  3. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in Step 2.
  4. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model App Get message to the IUT with the same ElementAddress and the same Model Identifier; if the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model App Get message to the IUT with the same ElementAddress and the same Model Identifier.
  5. The Lower Tester receives a Config SIG Model App List message (if the message sent in Step 4 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in Step 4 is Config Vendor Model App Get) in response from the IUT with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in Step 4, Model Identifier field value set to the Model Identifier used in Step 4, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex added in Step 2 is included in the list.
  6. The Lower Tester resends the message in Step 1 and the IUT responds to the message.

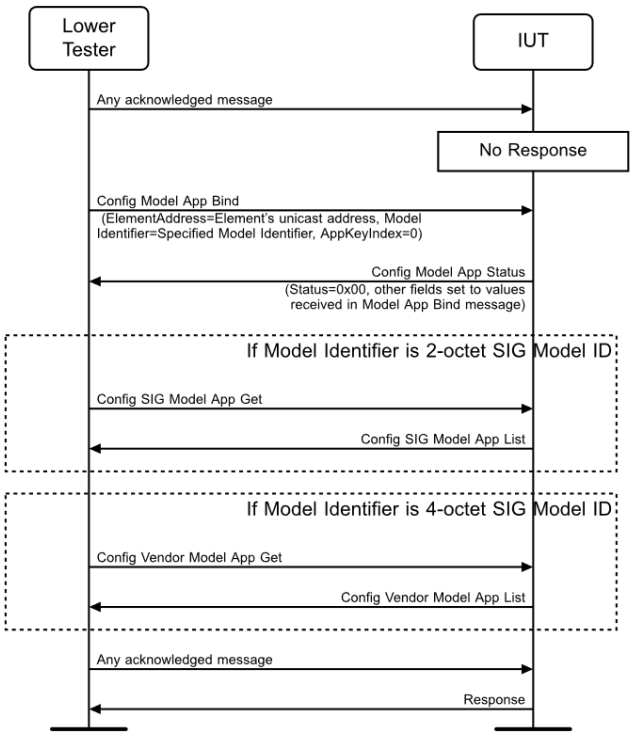


Figure 4.165: MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Config Model App Status message with a Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex set to 0 (zero), and the Model Identifier as referenced in Step 2.

  In Step 5, the IUT sends a Config SIG Model App List message (if the message sent in Step 4 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in Step 4 is Config Vendor Model App Get) with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in Step 4, Model Identifier field value set to the Model Identifier used in Step 4, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex added in Step 2 is included in the list.

  The IUT only responds to the message after it has been bound to the AppKey.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Model App

Unbind and SIG/Config Vendor Model App Get Messages.

- Reference

  [3] 4.3.2.47-4.3.2.52, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - Test case MESH/NODE/CFG/MAKL/BV-01-C [Respond to Config Model App Bind and SIG/Config Vendor Model App Get Messages] has been run. That is, the AppKey identified by AppKeyIndex has been bound to a selected model instance on the IUT.

- Test Procedure

  1. The Lower Tester sends a Config Model App Unbind message to the IUT with a ElementAddress field set to the unicast address of the Element, the Model Identifier field set to the Model Identifier of the selected model instance, and the AppKeyIndex field set to 0 (zero).
  2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex matching that in Step 1, and the Model Identifier as referenced in Step 1.
  3. If the Model Identifier is a 2-octet SIG Model ID, then the Lower Tester sends a Config SIG Model App Get message to the IUT with the same ElementAddress and the same Model Identifier; If the Model Identifier is a 4-octet Vendor Model ID, then the Lower Tester sends a Config Vendor Model App Get message to the IUT with the same ElementAddress and the same Model Identifier.
  4. The Lower Tester receives a Config SIG Model App List message (if the message sent in Step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in Step 3 is Config Vendor Model App Get) in response from the IUT with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in Step 3, Model Identifier field value set to the Model Identifier used in Step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex of 0x000 is not included in the list.
  5. The Lower Tester sends a message to the IUT for the selected model instance using AppKeyIndex of 0x000. The IUT does not respond.

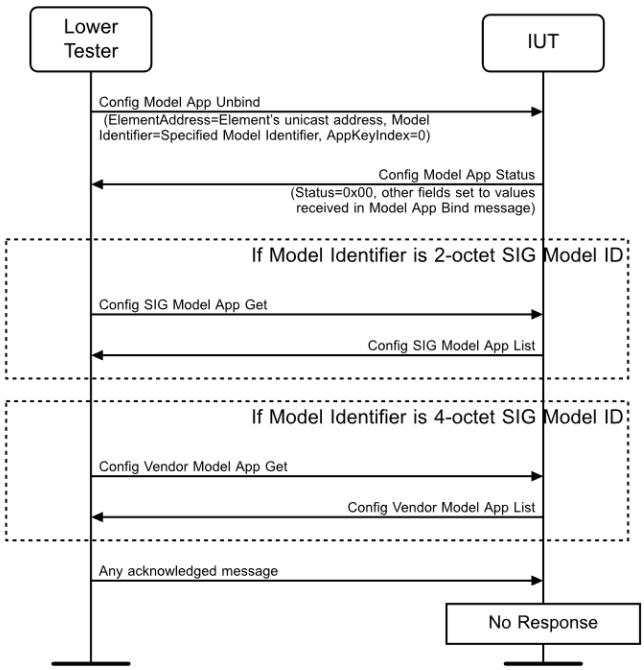


Figure 4.166: MESH/NODE/CFG/MAKL/BV-02-C [Respond to Config Model App Unbind and SIG/Config Vendor Model App Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model App Status message with a Status field set to 0x00, the ElementAddress value matching the unicast address of the Element, the AppKeyIndex matching that in Step 1, and the Model Identifier as referenced in Step 1.

  In Step 4, the IUT sends Config SIG Model App List message (if the message sent in Step 3 is Config SIG Model App Get) or a Config Vendor Model App List message (if the message sent in Step 3 is Config Vendor Model App Get) with the Status field value set to 0x00, the ElementAddress field value set to the Element Address used in Step 3, Model Identifier field value set to the Model Identifier used in Step 3, and the AppKeyIndexes field containing all AppKey indexes bound to the Model, and the AppKeyIndex 0x000 is not included in the list.

  The IUT does not respond to messages sent to the selected model instance for which the AppKey has been unbound.

  All configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MAKL/BI-01-C [Respond to Config Model App Bind Message with Invalid Values]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config Model App Bind message with values that cannot be accepted.

- Reference

  [3] 4.3.2.46, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least two AppKeys in the AppKey list and bound to a Model.
- Test Procedure
  1. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to an invalid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid AppKeyIndex in IUT.
  2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 1.
  3. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to an address which is not the address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid value in IUT.
  4. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x01 (Invalid Address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 3.
  5. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a value (e.g., 0xFFF) which is not stored in IUT.
  6. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x03 (Invalid AppKey) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 5.
  7. The Lower Tester sends a Config Model App Bind message to the IUT with an ElementAddress field set to the address of the Element, the Model Identifier field set to the Model Identifier of the Configuration Server Model and the AppKeyIndex field set to a valid value which is stored on the IUT.
  8. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x0D (Cannot Bind) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 7.

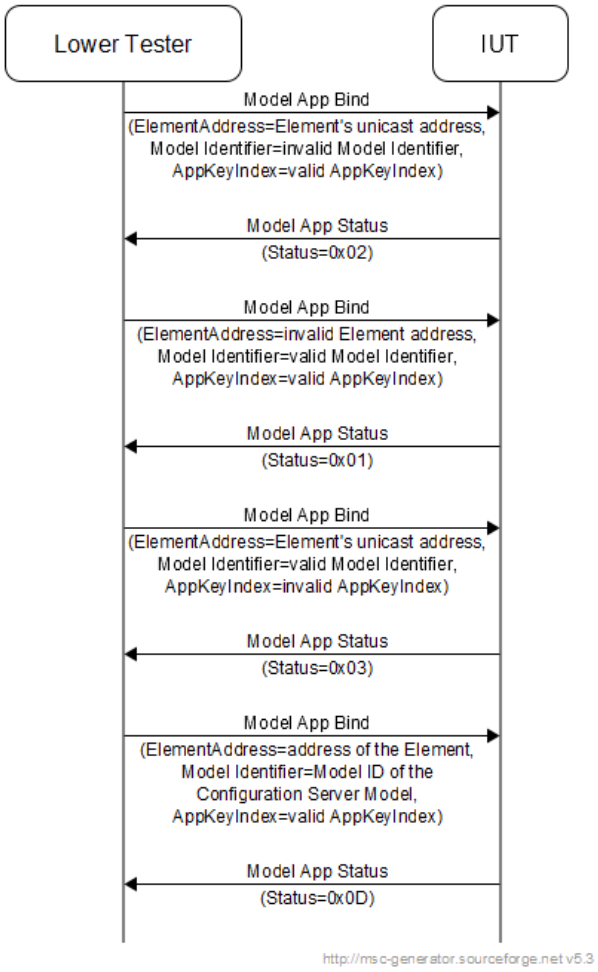


Figure 4.167: MESH/NODE/CFG/MAKL/BI-01-C [Respond to Config Model App Bind Message with Invalid Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, IUT sends a Config Model App Status message with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 1.

  In Step 4, IUT sends a Config Model App Status message with the Status field set to 0x01 (Invalid address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 3.

  In Step 6, IUT sends a Config Model App Status message with the Status field set to 0x03 (Invalid AppKey) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 5.

  In Step 8, IUT sends a Config Model App Status message with the Status field set to 0x0D (Cannot Bind) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding

fields of the incoming message in Step 7.

The Config Model App Status message is encrypted using the device key of the IUT.

MESH/NODE/CFG/MAKL/BI-02-C [Respond to Config Model App Unbind Message with Invalid Values]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config Model App Unbind message with values that cannot be accepted.

- Reference

  [3] 4.3.2.47, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure
  1. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6], and an invalid AppKeyIndex field value.
  2. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x03 (Invalid AppKey), and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 1.
  3. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to an invalid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid AppKeyIndex in IUT.
  4. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 3.
  5. The Lower Tester sends a Config Model App Unbind message to the IUT with an ElementAddress field set to an address which is not the address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to a valid value in IUT.
  6. The Lower Tester receives a Config Model App Status message in response from the IUT with the Status field set to 0x01 (Invalid Address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 5.

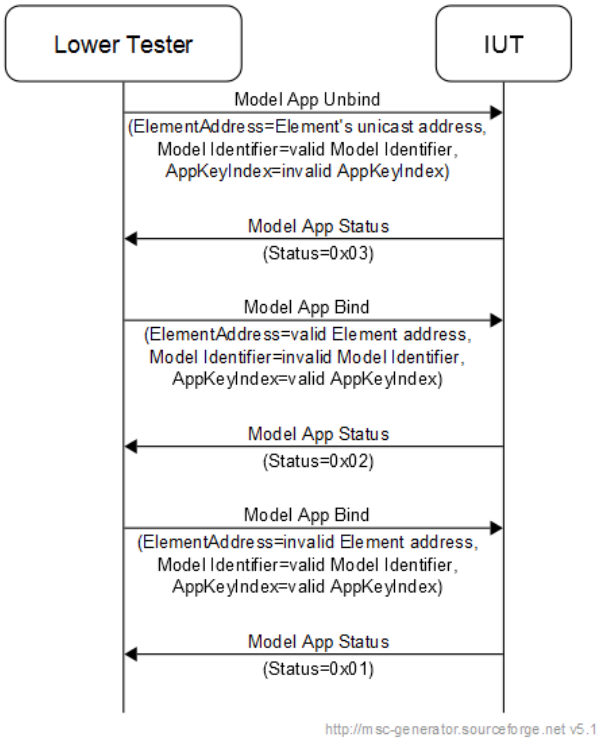


Figure 4.168: MESH/NODE/CFG/MAKL/BI-02-C [Respond to Config Model App Unbind Message with Invalid Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Model App Status message with a Status field set to 0x03 (Invalid AppKey), and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 1.

  In Step 4, IUT sends a Config Model App Status message with the Status field set to 0x02 (Invalid Model) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 3.

  In Step 6, IUT sends a Config Model App Status message with the Status field set to 0x01 (Invalid address) and other fields (ElementAddress, AppKeyIndex and Model Identifier) set to the values of the corresponding fields of the incoming message in Step 5.

  The Config Model App Status messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MAKL/BI-03-C [Respond to Config SIG Model App Get Message with Invalid Values]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config SIG Model App Get message with invalid values.

- Reference

  [3] 4.3.2.49, 4.3.2.50, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element and supports a SIG Model.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least one AppKey bound to a Model.
- Test Procedure
  1. The Lower Tester sends a Config SIG Model App Get message to the IUT with the ElementAddress field set to the unicast address of the Element and Model Identifier field set to an invalid Model Identifier value.
  2. The Lower Tester receives a Config SIG Model App List message in response from the IUT with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 1, and the AppKeyIndexes field to a zero-length (empty) list.
  3. The Lower Tester sends a Config SIG Model App Get message to the IUT with an ElementAddress field set to an address which is not the address of any of the Elements of the IUT, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6].
  4. The Lower Tester receives a Config SIG Model App List message in response from the IUT with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 3, and the AppKeyIndexes field to a zero-length (empty) list.

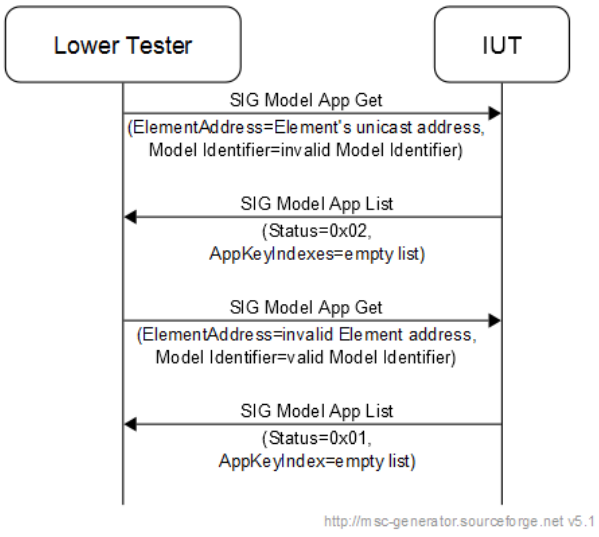


Figure 4.169: MESH/NODE/CFG/MAKL/BI-03-C [Respond to Config SIG Model App Get Message with Invalid Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config SIG Model App List message in response to the Lower Tester with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 1, and the AppKeyIndexes field to a zero-length (empty) list.

  In Step 4, the IUT sends a Config SIG Model App List message in response to the Lower Tester with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 3, and the AppKeyIndexes field to a zero-length (empty) list.

  The Config SIG Model App List messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/MAKL/BI-04-C [Respond to Config Vendor Model App Get Message with Invalid Values]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config Vendor Model App Get message with invalid values.

- Reference

  [3] 4.3.2.51, 4.3.2.52, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element and supports a Vendor Model.
  - The Lower Tester supports Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has at least one AppKey bound to a Model.
- Test Procedure
  1. The Lower Tester sends a Config Vendor Model App Get message to the IUT with the ElementAddress field set to the unicast address of the Element and Model Identifier field set to a Model Identifier value not defined in the IXIT [6].
  2. The Lower Tester receives a Config Vendor Model App List message in response from the IUT with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 1, and the AppKeyIndexes field to a zero-length (empty) list.
  3. The Lower Tester sends a Config Vendor Model App Get message to the IUT with an ElementAddress field set to an address which is not the address ofany of the Elements of the IUT, the Model Identifier

field set to a valid Model Identifier as specified in the IXIT [6].

4. The Lower Tester receives a Config Vendor Model App List message in response from the IUT with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 3, and the AppKeyIndexes field to a zero-length (empty) list.

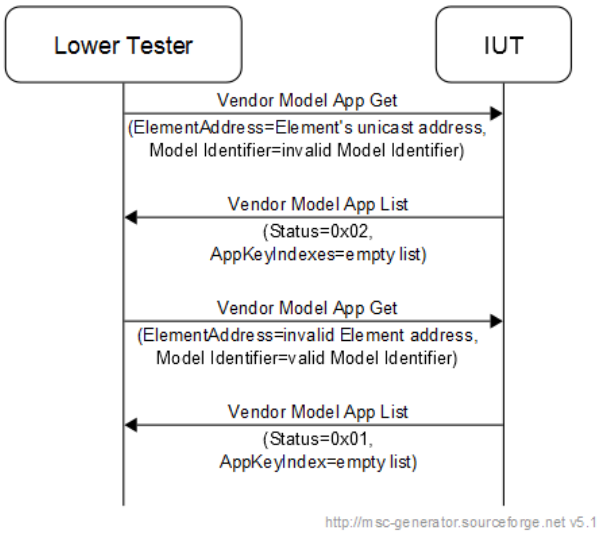


Figure 4.170: MESH/NODE/CFG/MAKL/BI-04-C [Respond to Config Vendor Model App Get Message with Invalid Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Vendor Model App List message in response to the Lower Tester with the Status field set to 0x02 (Invalid Model), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 1, and the AppKeyIndexes field to a zero-length (empty) list.

  In Step 4, the IUT sends a Config Vendor Model App List message in response to the Lower Tester with the Status field set to 0x01 (Invalid Address), ElementAddress and Model Identifier fields to the corresponding ElementAddress and Model Identifier of the incoming message in Step 3, and the AppKeyIndexes field to a zero-length (empty) list.

  The Config Vendor Model App List messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/MAKL/BV-01-C [Send Config Model App Bind Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Model App Bind messages to a Configuration Server.

- Reference

[3] 4.3.2.46, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Model App Bind message to the Lower Tester with an ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).
  2. The Lower Tester receives the Config Model App Bind message with the expected values.
  3. The Lower Tester sends a Config Model App Status message in response to the IUT.

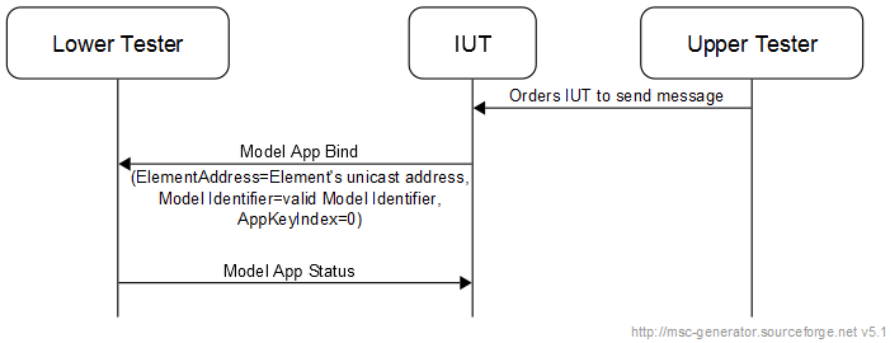


Figure 4.171: MESH/CFGCL/CFG/MAKL/BV-01-C [Send Config Model App Bind Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Model App Bind message from the IUT containing all valid fields - the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6] and AppKeyIndex field set to zero (0).

  The Config Model App Bind messages are encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/MAKL/BV-02-C [Send Config Model App Unbind Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Model App Unbind messages to a Configuration Server.

- Reference

  [3] 4.3.2.47, 4.3.2.48, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports Configuration Client Model.
  - The Lower Tester supports Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Model App Unbind message to the Lower Tester with the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier as specified in the IXIT [6], and the AppKeyIndex field set to 0 (zero).
  2. The Lower Tester receives the Config Model App Unbind message.
  3. The Lower Tester sends a Config Model App Status message in response to the IUT.

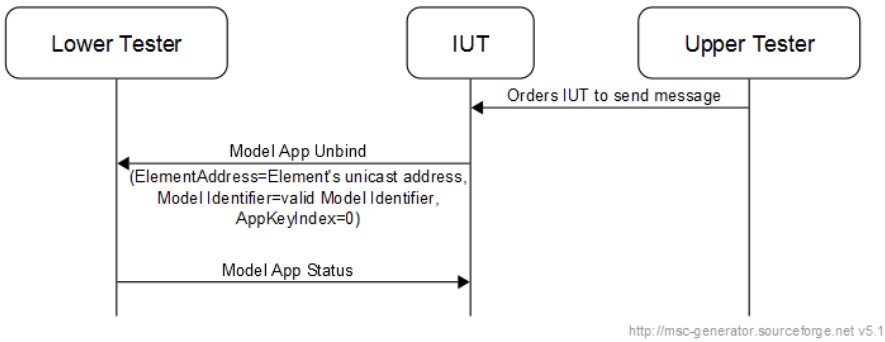


Figure 4.172: MESH/CFGCL/CFG/MAKL/BV-02-C [Send Config Model App Unbind Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Model App Unbind message from the IUT containing all valid fields - the ElementAddress field set to the unicast address of the Element, the Model Identifier field set to a valid Model Identifier, and the AppKeyIndex field set to 0 (zero).

  The Config Model App Unbind message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/MAKL/BV-03-C [Send Config SIG Model App Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config SIG Model App Get messages to a Configuration Server.

- Reference

  [3] 4.3.2.49, 4.3.2.50, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition

- The IUT supports Configuration Client Model.
- The Lower Tester supports Configuration Server Model supporting SIG Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

1. The Upper Tester orders the IUT to send a Config SIG Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and Model Identifier set to the SIG Model ID that identifies the Model within the Element.
2. The Lower Tester receives the Config SIG Model App Get message containing the expected ElementAddress and Model Identifier values.
3. The Lower Tester sends a Config SIG Model App List message in response to the IUT.

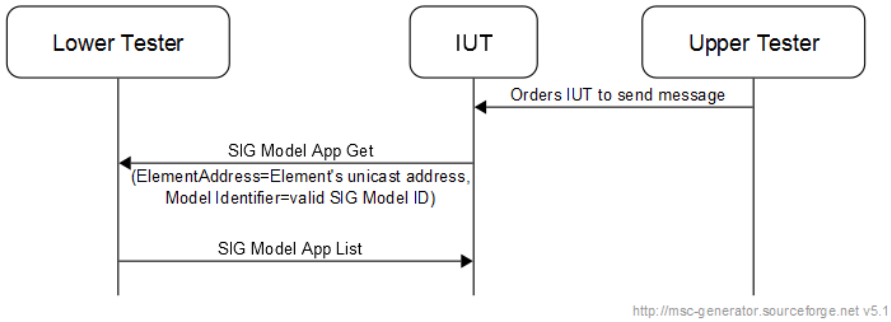


Figure 4.173: MESH/CFGCL/CFG/MAKL/BV-03-C [Send Config SIG Model App Get Messages] MSC

- Expected Outcome

Pass verdict

The IUT sends a Config SIG Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and the SIG Model Identifier the SIG Model ID that identifies the Model within the Element.

The Config SIG Model App Get message is encrypted using the device key of the Lower Tester.

MESH/CFGCL/CFG/MAKL/BV-04-C [Send Config Vendor Model App Get Messages]

- Test Purpose

Verify that an IUT node supporting the Configuration Client Model can send Config Vendor Model App Get messages to a Configuration Server.

- Reference

[3] 4.3.2.51, 4.3.2.52, 4.4.1.2.11, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Client Model.

- The Lower Tester supports the Configuration Server Model.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

1. The Upper Tester orders the IUT to send a Config Vendor Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and Model Identifier the Vendor Model ID that identifies the Model within the Element.
2. The Lower Tester receives the Config Vendor Model App Get message containing the expected ElementAddress and Model Identifier values.
3. The Lower Tester sends a Config Vendor Model App List message in response to the IUT.

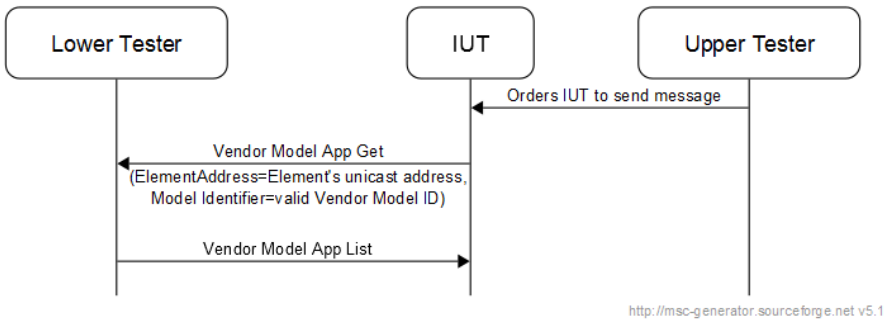


Figure 4.174: MESH/CFGCL/CFG/MAKL/BV-04-C [Send Config Vendor Model App Get Messages] MSC

- Expected Outcome

Pass verdict

The IUT sends a Config Vendor Model App Get message to the Lower Tester with the ElementAddress set to the address of the Element and the Model Identifier the Vendor Model ID that identifies the Model within the Element.

The Config Vendor Model App Get message is encrypted using the device key of the Lower Tester.

4.16.12 Node Identity Procedures

This section validates the Node Identity procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/NID/BV-01-C [Receive Config Node Identity Get/Set Messages]

- Test Purpose

Verify that a Configuration Server IUT can respond to a Config Node Identity Get/Set message.

- Reference

[3] 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The IUT supports the Mesh Proxy Service.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Node Identity Set message with the NetKeyIndex field set to 0x000, and the Identity field set to 0x00 which means Mesh Proxy Service is supported, but Node Identity for a Subnet is stopped.
  2. The Lower Tester expects the IUT to receive the Config Node Identity Set message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
  3. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in Step 1 to the IUT.
  4. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
  5. The Lower Tester sends a Config Node Identity Set message with the NetKeyIndex field set to 0x000, and the Identity field set to 0x01 which means Mesh Proxy Service is supported, and Node Identity for a Subnet is running.
  6. The Lower Tester expects the IUT to receive the Config Node Identity Set message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.
  7. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in Step 5 to the IUT.
  8. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

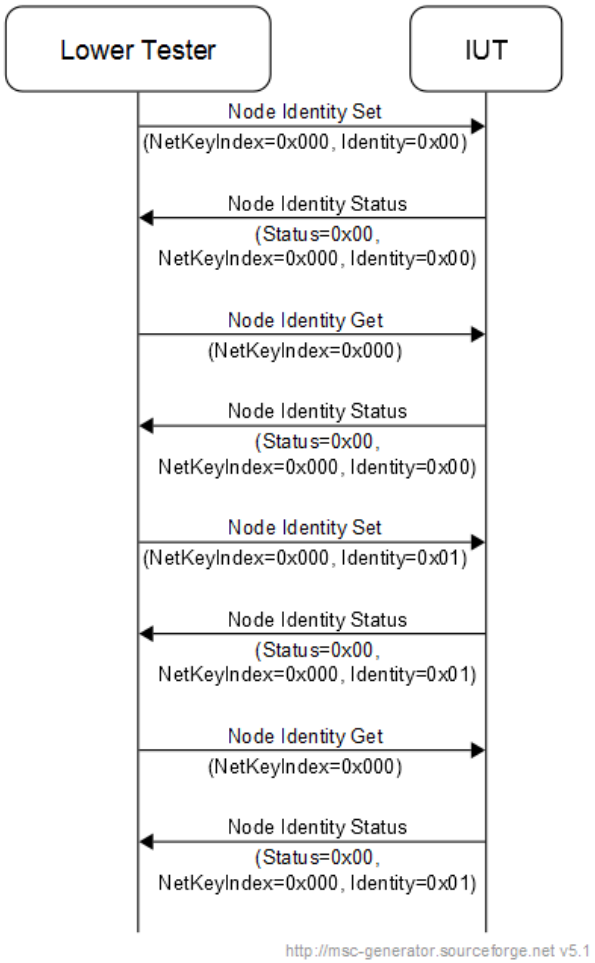


Figure 4.175: MESH/NODE/CFG/NID/BV-01-C [Receive Config Node Identity Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, IUT responds with a Config Node Identity Status message with Status field value set to 0x00, NetKeyIndex field value set to 0x000and Identity field value set to 0x00.

  In Steps 6 and 8, IUT responds with a Config Node Identity Status message with Status field value set to 0x00, NetKeyIndex field value set to 0x000and Identity field value set to 0x01.

  The Config Node Identity Status messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NID/BV-02-C [Node Identity Advertising on Multiple Subnets]

- Test Purpose

  Verify that a Configuration Server IUT can trigger Node Identity advertising based on the Node Identity state change and can interleave advertising for multiple Subnets.

- Reference

[3] 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT exposes the Mesh Proxy Service.
  - The IUT supports multiple subnets.
- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT.
  2. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set randomly to 0x00 or 0x01 and the Identity field set to 0x01.
  3. The IUT responds with a Config Node Identity Status message indicating the state change and begins advertising with Node Identity only for the Subnet identified by the NetKeyIndex sent in Step 2.
  4. The Lower Tester sends a Config Node Identity Get message to the IUT with the NetKeyIndex field set to the index of the other network key than the one used in Step 2.
  5. The IUT responds with a Config Node Identity Status message with the Identity field set to 0x00.
  6. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set to the same value as in Step 2 and the Identity field set to 0x00.
  7. The IUT responds with a Config Node Identity Status message indicating the state change and stops advertising with Node Identity.
  8. The Upper Tester orders the IUT to begin advertising with Node Identity, simulating user interaction.
  9. The IUT begins sending interleaved Node Identity advertisings for both Subnets it supports.
  10. The Lower Tester reads the Node Identity states for both network keys, and the IUT responds with the Identity field value of 0x01 in both cases.
  11. After 60 seconds of advertising, the IUT stops advertising with Node Identity on both subnets.
  12. The Lower Tester reads the Node Identity states for both network keys, and the IUT responds with the Identity field value of 0x00 in both cases.

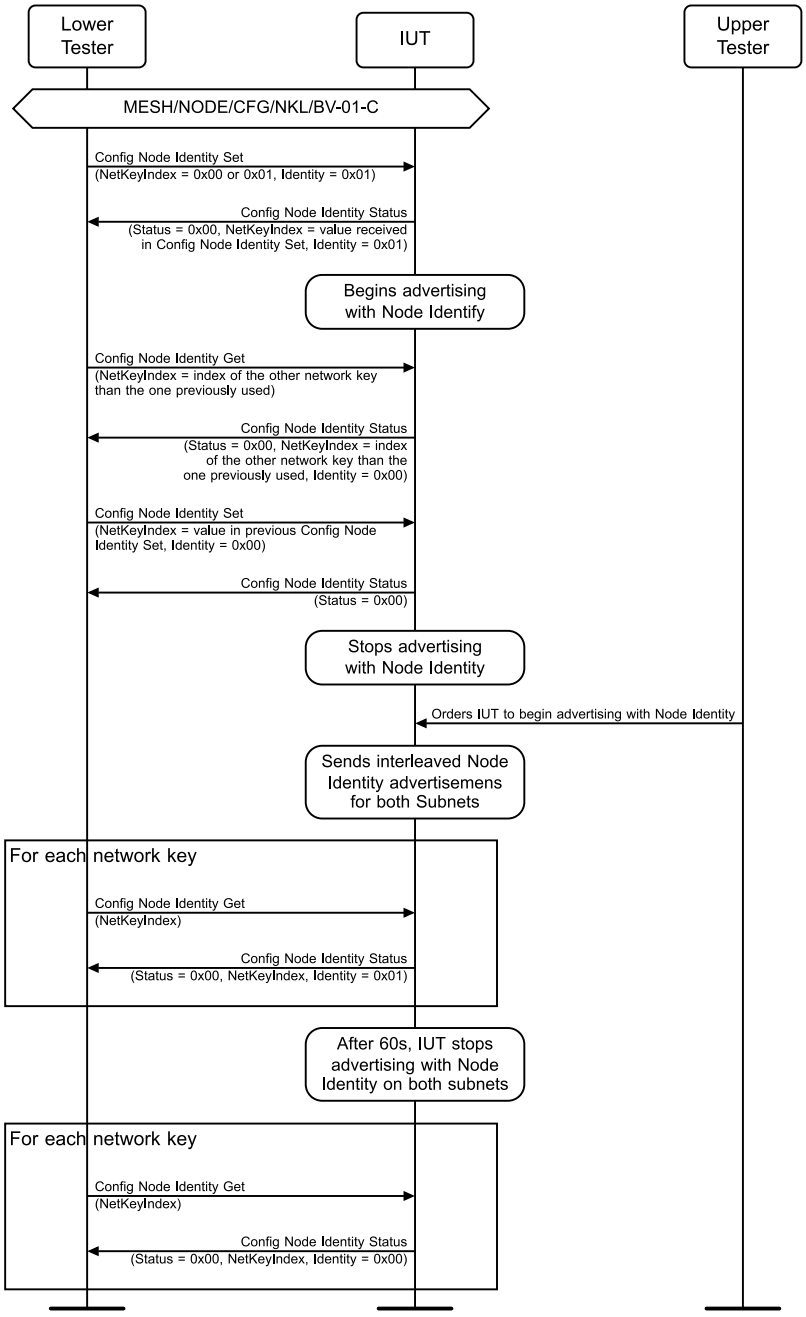


Figure 4.176: MESH/NODE/CFG/NID/BV-02-C [Node Identity Advertising on Multiple Subnets] MSC

- Expected Outcome

  Pass verdict

  The IUT responds to all messages as expected and advertises on the correct subnets.

MESH/NODE/CFG/NID/BV-03-C [Node Identity Config NetKey Deleted]

- Test Purpose

Verify that a Configuration Server IUT sets the Node Identity state to 0x00 when NetKey is deleted.

- Reference

  [3] 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service.
  - The IUT supports multiple subnets.

- Test Procedure
  1. The Lower Tester executes the procedure described in MESH/NODE/CFG/NKL/BV-01-C [Respond to Config NetKey Add and Get Messages] to add a second network key on the IUT with NetKeyIndex 1.
  2. The Lower Tester sends a Config Node Identity Set message to the IUT with the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.
  3. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x00, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.
  4. The Lower Tester deletes a NetKey from IUT's NetKey List, following the procedure described in MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]. The NetKeyIndex field is set to 0x0001.
  5. The Lower Tester sends a Config Node Identity Get message to the IUT with the NetKeyIndex field set to 0x0001.
  6. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the NetKeyIndex field set to 0x0001 and the Identity field set to 0x00.

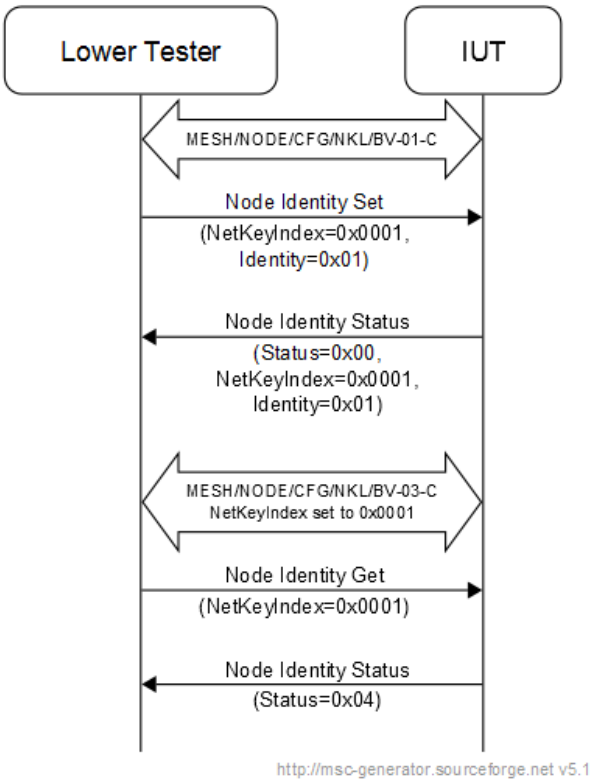


Figure 4.177: MESH/NODE/CFG/NID/BV-03-C [Node Identity Config NetKey Deleted] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Config Node Identity Status message with the Status field set to 0x00, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x01.

  In Step 6, the IUT sends a Config Node Identity Status message with the Status field set to 0x04, the NetKeyIndex field set to 0x0001 and the Identity field set to 0x00.

  All the Configuration messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NID/BI-01-C [Receive Config Node Identity Get/Set Messages – Invalid Parameter]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config Node Identity Get/Set message with invalid parameters.

- Reference

  [3] 3.1.2, 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

- Initial Condition

- The IUT supports the Configuration Server Model on the Primary Element.
- The Lower Tester supports the Configuration Client Model.
- The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- The IUT supports the Mesh Proxy Service.

• Test Procedure

1. The Lower Tester sends a Config Node Identity Set message with a NetKeyIndex which doesn't exist in the IUT's NetKey list, and an Identity field set to 0x00.
2. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), and the NetKeyIndex and Identity fields as defined in the incoming message in Step 1.
3. The Lower Tester sends a Config Node Identity Get message with the same invalid NetKeyIndex used in Step 1 to the IUT.
4. The Lower Tester expects the IUT to respond with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the Identity field set to 0x00, and the NetKeyIndex as defined in the incoming message in Step 6.
5. The Lower Tester sends a Config Node Identity Set message with a valid NetKeyIndex and the Identity field set to a random number between 0x03 and 0xFF (RFU).
6. The Lower Tester expects the IUT to ignore the message and does not respond with any messages.

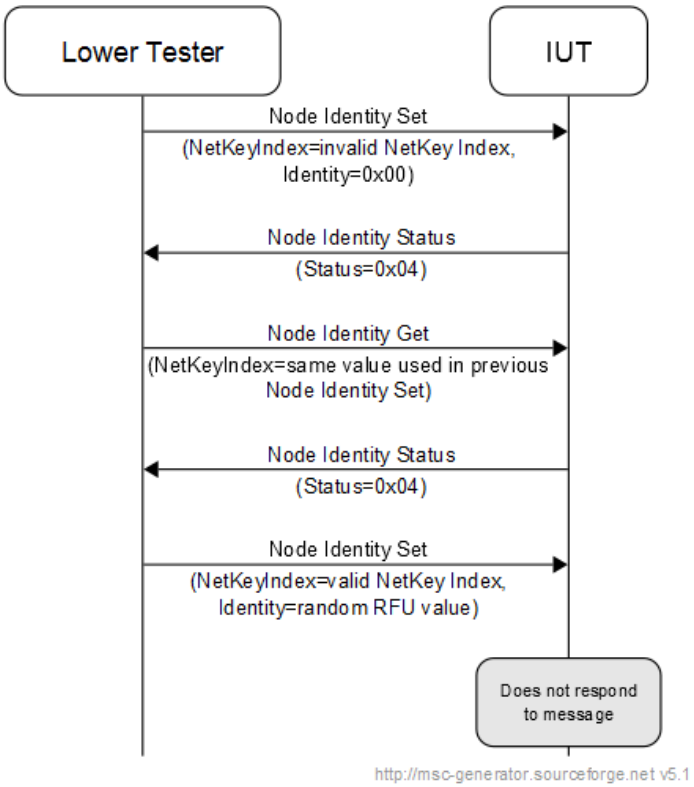


Figure 4.178: MESH/NODE/CFG/NID/BI-01-C [Receive Config Node Identity Get/Set Messages – Invalid Parameter] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT responds with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), and the NetKeyIndex and Identity fields as defined in the incoming message in Step 1.

  In Step 4, the IUT responds with a Config Node Identity Status message with the Status field set to 0x04 (Invalid NetKey), the Identity field set to 0x00, and the NetKeyIndex as defined in the incoming message in Step 3.

  In Step 6, the IUT does not respond with any messages.

  The Config Node Identity Status messages are encrypted using the device key of the IUT.

MESH/NODE/CFG/NID/BI-03-C [Receive Config Node Identity Set Message – Change to Not Supported]

- Test Purpose

  Verify that a Configuration Server IUT supporting the Mesh Proxy Service ignores the Config Node Identity Set message to change the Node Identify to be not supported.

- Reference

[3] 3.1.2, 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.1.2, 4.4.1.2.12

- Initial Condition
  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by the NetKeyIndex of 0x000 and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT supports the Mesh Proxy Service and has the Identity field set to 0x00.
- Test Procedure
  1. The Lower Tester sends a Config Node Identity Set message with a valid NetKeyIndex which is an index that identifies the global NetKey Index of the NetKey of the Node Identity state, and an Identity field set to 0x02 which means Mesh Proxy Service is not supported.
  2. The Lower Tester expects the IUT to ignore the message and does not respond with any messages.
  3. The Lower Tester sends a Config Node Identity Get message with the same NetKeyIndex used in Step 1 to the IUT.
  4. The Lower Tester expects the IUT to receive the Config Node Identity Get message and respond with a Config Node Identity Status message with the Identity field and the NetKeyIndex field as defined by the incoming message and setting the Status field to 0x00.

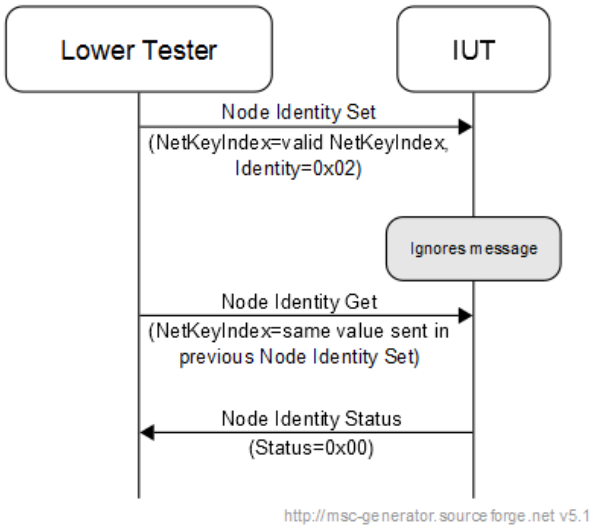


Figure 4.179: MESH/NODE/CFG/NID/BI-03-C [Receive Config Node Identity Set Message – Change to Not Supported] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT does not respond with any messages.

  In Step 4, the IUT responds with a Config Node Identity Status message with the Status field set to 0x00,

the NetKeyIndex field set to 0x000 and the Identity field set to 0x00.

The Config Node Identity Status messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/NID/BV-01-C [Send Config Node Identity Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Node Identity Get/Set messages to a Configuration Server.

- Reference

  [3] 4.2.12, 4.3.2.43, 4.3.2.44, 4.3.2.45, 4.4.2.2.11

- Initial Condition
  - The IUT supports the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The Lower Tester supports the Mesh Proxy Service.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Node Identity Set message to the Lower Tester with a valid NetKeyIndex which is an index that identifies the global NetKey Index of the NetKey of the Node Identity state, and an Identity field set to 0x00.
  2. The Lower Tester expects to receive a Config Node Identity Set message with the fields set with the values defined in Step 1.
  3. The Lower Tester responds with a Config Node Identity Status message.
  4. The Upper Tester orders the IUT to send a Config Node Identity Set message to the Lower Tester with the same NetKeyIndex value as in Step 1, and an Identity field set to 0x01.
  5. The Lower Tester expects to receive a Config Node Identity Set message with the fields set with the values defined in Step 3.
  6. The Lower Tester responds with a Config Node Identity Status message.
  7. The Upper Tester orders the IUT to send the Config Node Identity Get message to the Lower Tester with the same NetKeyIndex value as in Step 1.
  8. The Lower Tester expects to receive a Config Node Identity Get message with the fields set with the values defined in Step 5.
  9. The Lower Tester responds with a Config Node Identity Status message.

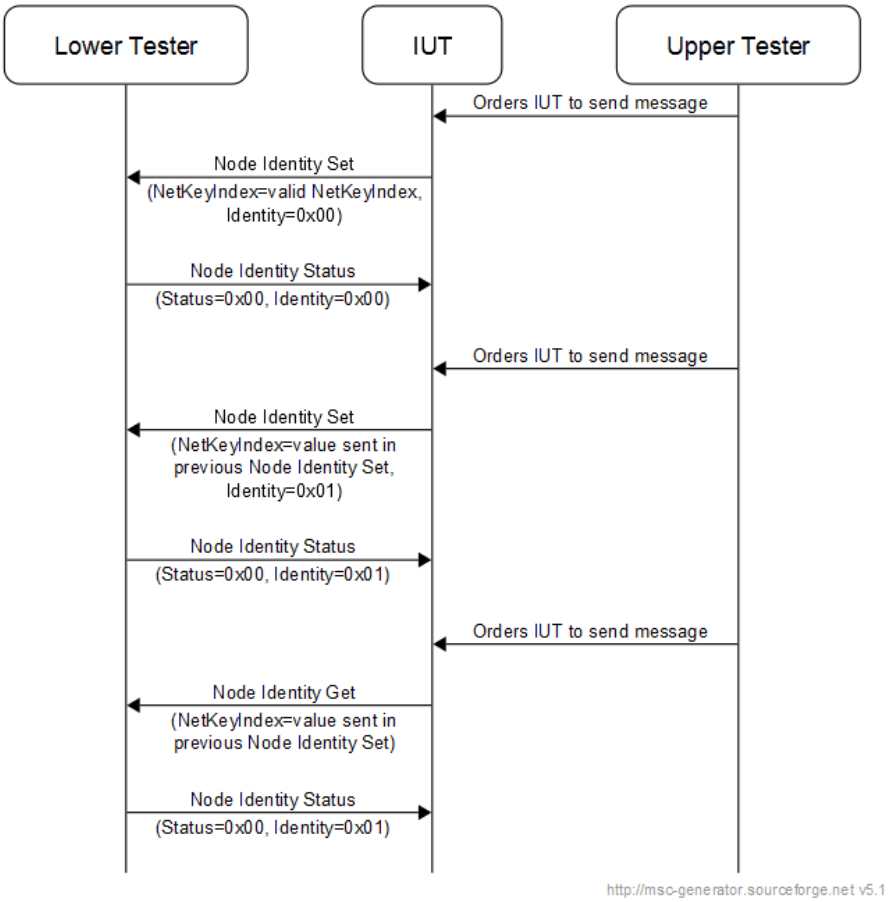


Figure 4.180: MESH/CFGCL/CFG/NID/BV-01-C [Send Config Node Identity Get/Set Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sent a properly formatted Config Node Identity Set message with a valid NetKeyIndex field and the Identity field value set to 0x00.

  In Step 5, the IUT sent a properly formatted Config Node Identity Set message with the Identity field value set to 0x01.

  In Step 8, the IUT sent a properly formatted Config Node Identity Get message with the same NetKeyIndex field value as in Step 1.

  The Config Node Identity Set messages are encrypted using the device key of the IUT.

4.16.13 Reset Procedures

This section validates the Reset procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/RST/BV-01-C [Receive Config Node Reset Message]

- Test Purpose

  Verify that a Configuration Server IUT can respond to a Config Node Reset message.

- Reference

  [3] 3.10.7, 4.3.2.53, 4.3.2.54, 4.4.1.2, 4.4.1.2.13

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Beacon Set message to the IUT with the Beacon field set to 0x01.
  2. The Lower Tester receives a Config Beacon Status message from the IUT with the Beacon field value set to 0x01.
  3. The Lower Tester sends a Config Node Reset message to the IUT.
  4. The Lower Tester receives a Config Node Reset Status message in response from the IUT.

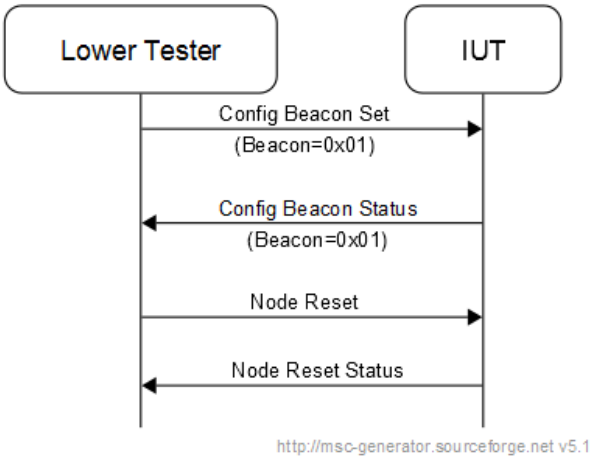


Figure 4.181: MESH/NODE/CFG/RST/BV-01-C [Receive Config Node Reset Message] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends a Config Node Reset Status message to the Lower Tester.

  The Config Node Reset Status message is encrypted using the device key of the IUT.

MESH/CFGCL/CFG/RST/BV-01-C [Send Config Node Reset Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Node Reset messages to a Configuration Server.

- Reference

  [3] 3.10.7, 4.3.2.53, 4.3.2.54, 4.4.1.2.13, 4.4.2.2.13

- Initial Condition

  - The Lower Tester supports the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Node Reset message to the Lower Tester.
  2. The Lower Tester sends a Config Node Reset Status message in response to the IUT.

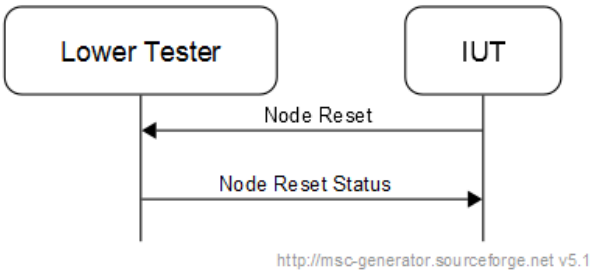


Figure 4.182: MESH/CFGCL/CFG/RST/BV-01-C [Send Config Node Reset Messages] MSC

- Expected Outcome

  Pass verdict

  The IUT sends a Config Node Reset message to the Lower Tester.

  The Config Node Reset message is encrypted using the device key of the Lower Tester.

4.16.14 Heartbeat Publication Procedures

This section validates the Heartbeat Publication procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/HBP/BV-01-C [Receive Config Heartbeat Publication Set/Get messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat

Publication Set message and update the identified Heartbeat Publication state.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.1, 4.3.2.61, 4.3.2.62, 4.3.2.63, 4.4.1.2, 4.4.1.2.15, 4.4.2.2.15

- Initial Condition
  - The IUT is a node supporting the Configuration Server Model on the primary Element.
  - The Lower Tester supports the Configuration Client Model.
  - The IUT has been provisioned and has a NetKey identified by NetKeyIndex of 0x000.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with the CountLog field set to 0x0C, the PeriodLog field set to 0x04, the TTL field set to 0x00, the Features field set to 0x0000, the Destination field set to the Lower Tester's unicast address, and the NetKeyIndex field set to 0x0000.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
  3. The Lower Tester sends a Config Heartbeat Publication Get message to the IUT.
  4. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the same field values received in Step 2. The CountLog field value has been decremented to 0x0B if the IUT has already sent a Heartbeat message after Step 2.

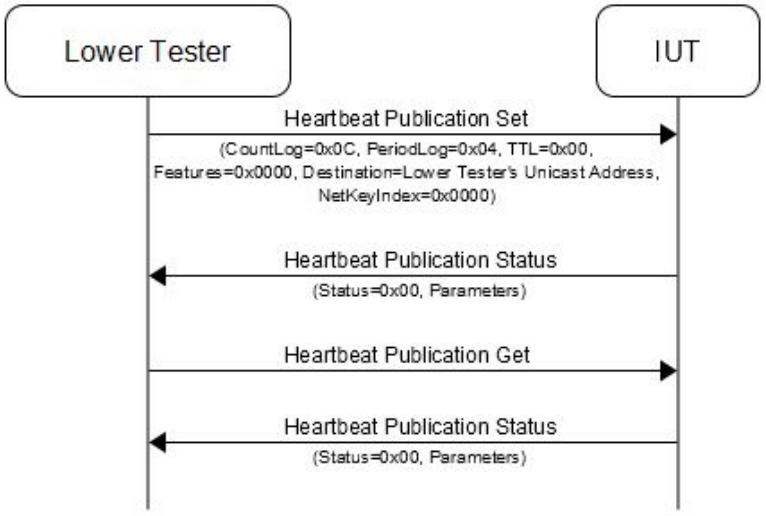


Figure 4.183: MESH/NODE/CFG/HBP/BV-01-C [Receive Config Heartbeat Publication Set/Get messages] MSC

- Expected Outcome

  Pass verdict

  In Steps 2 and 4, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message sent by the Lower Tester in Step 1, with the CountLog field decremented in Step 4 if a Heartbeat message has already been sent.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BV-02-C [Periodic publishing of Heartbeat messages]

- Test Purpose

  Verify that the IUT can send Heartbeat messages as configured by a node that supports the Configuration Client Model.

- Reference

  [3] 3.1.2, 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.1, 4.2.17.6, 4.3.2.62, 4.3.2.63, 4.4.1.2.15, 4.4.2.2.15

- Initial Condition
  - The IUT is a node supporting the Configuration Server Model on the primary Element.
  - The IUT's primary Element has a dynamically-assigned group address programmed into.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.

- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the Destination field set to the Lower Tester's unicast address, the CountLog field set to 0x02, the PeriodLog field set to 0x02, the TTL field set to 0x7F, the Features field set to 0xFFF0, and the NetKeyIndex field set to the Index of the NetKey of the IUT.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0000 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester expects the IUT to send two Heartbeat messages to the address set in the Destination field of the Config Heartbeat Publication Set message sent in Step 1, with the InitTTL field set to 0x7F. The Lower Tester expects to receive the Heartbeat messages within an interval of 2 seconds (+/- 250 ms).
4. The Lower Tester repeats Steps 1–3 using the Lower Tester's dynamically-assigned group address as the Destination field of the Config Heartbeat Publication Set message sent in Step 1.
5. The Lower Tester repeats Steps 1–2 using the unassigned address (0x0000) instead of its unicast address in the Destination field of the Config Heartbeat Publication Set message sent in Step 1.
6. Wait 2.5 seconds, and the Lower Tester expects the IUT not to send any other Heartbeat messages.

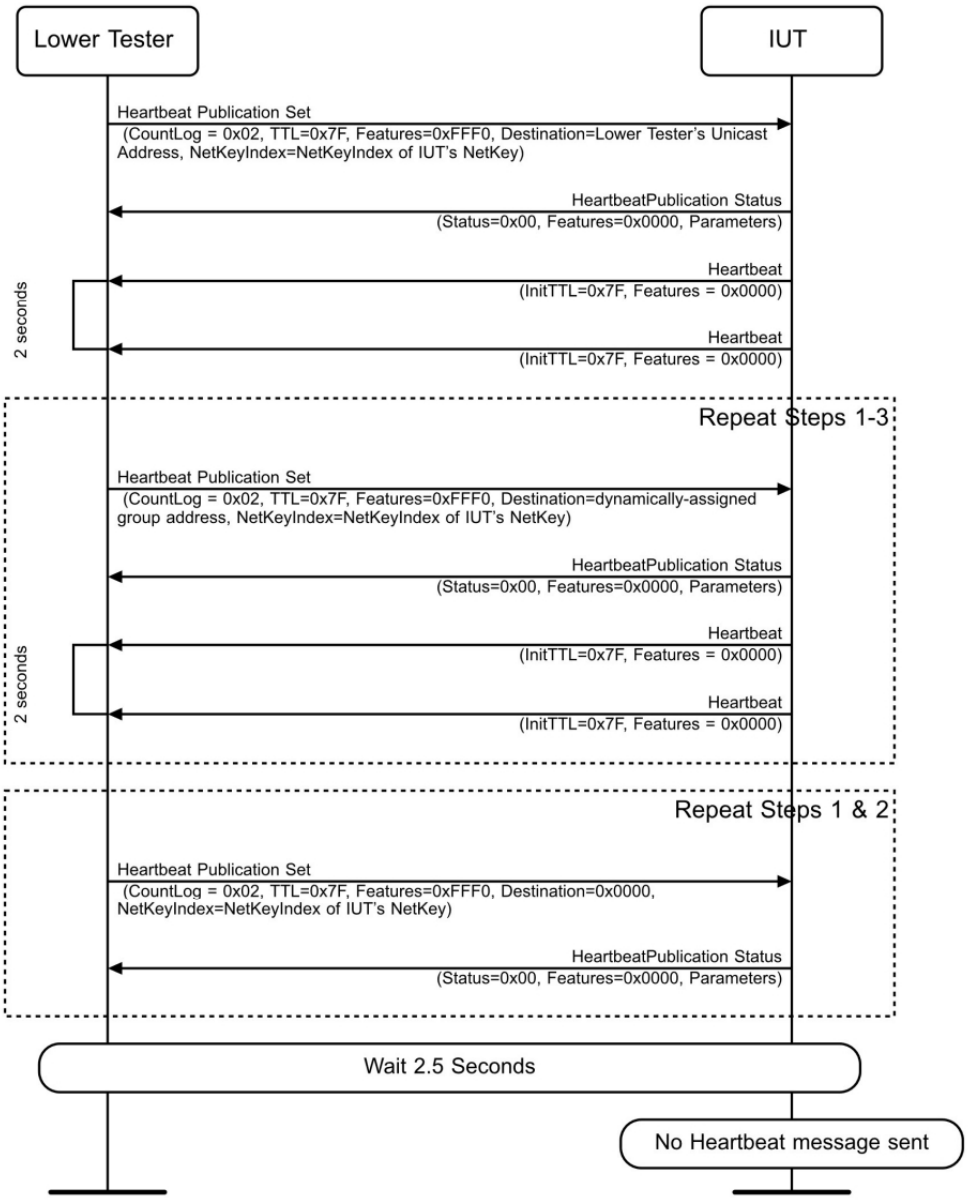


Figure 4.184: MESH/NODE/CFG/HBP/BV-02-C [Periodic publishing of Heartbeat messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0000 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message.
  In Step 3, the IUT sends two Heartbeat messages with the DST field set to the address from the Destination field of the Config Heartbeat Publication Set message. The time interval between the two Heartbeat messages is 2 seconds. The Opcode field of the Transport Control message is set to 0x0A.

  In Step 4, the IUT sends the Heartbeat messages as in Step 2 and Step 3.

In Step 5, the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the values of the CountLog, PeriodLog, and TTL fields set to 0x00.

In Step 6, the IUT does not send any other Heartbeat messages.

All configuration messages are encrypted and authenticated using the device key of the IUT.

The NetKey used by Transport Control message is the NetKey from Step 1.

MESH/NODE/CFG/HBP/BV-03-C [Triggered publishing of Heartbeat messages – Relay State changes]

- Test Purpose

  Verify that the IUT can send Heartbeat Messages triggered by the Relay State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.8, 4.3.2.13, 4.3.2.14, 4.3.2.62, 4.3.2.63, 4.4.1.2.6, 4.4.2.2.6

- Initial Condition
  - The IUT supports the Relay feature and the feature is disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0001 and all the other fields set to valid values.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
  3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.
  4. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and to send a Heartbeat message with the Relay bit of the Features field set to 1.
  5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.
  6. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and to send a Heartbeat message with the Relay bit of the Features field set to 0.
  7. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.
  8. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and not to send any Heartbeat message.
  9. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.

10. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
11. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.
12. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and not to send any Heartbeat message.

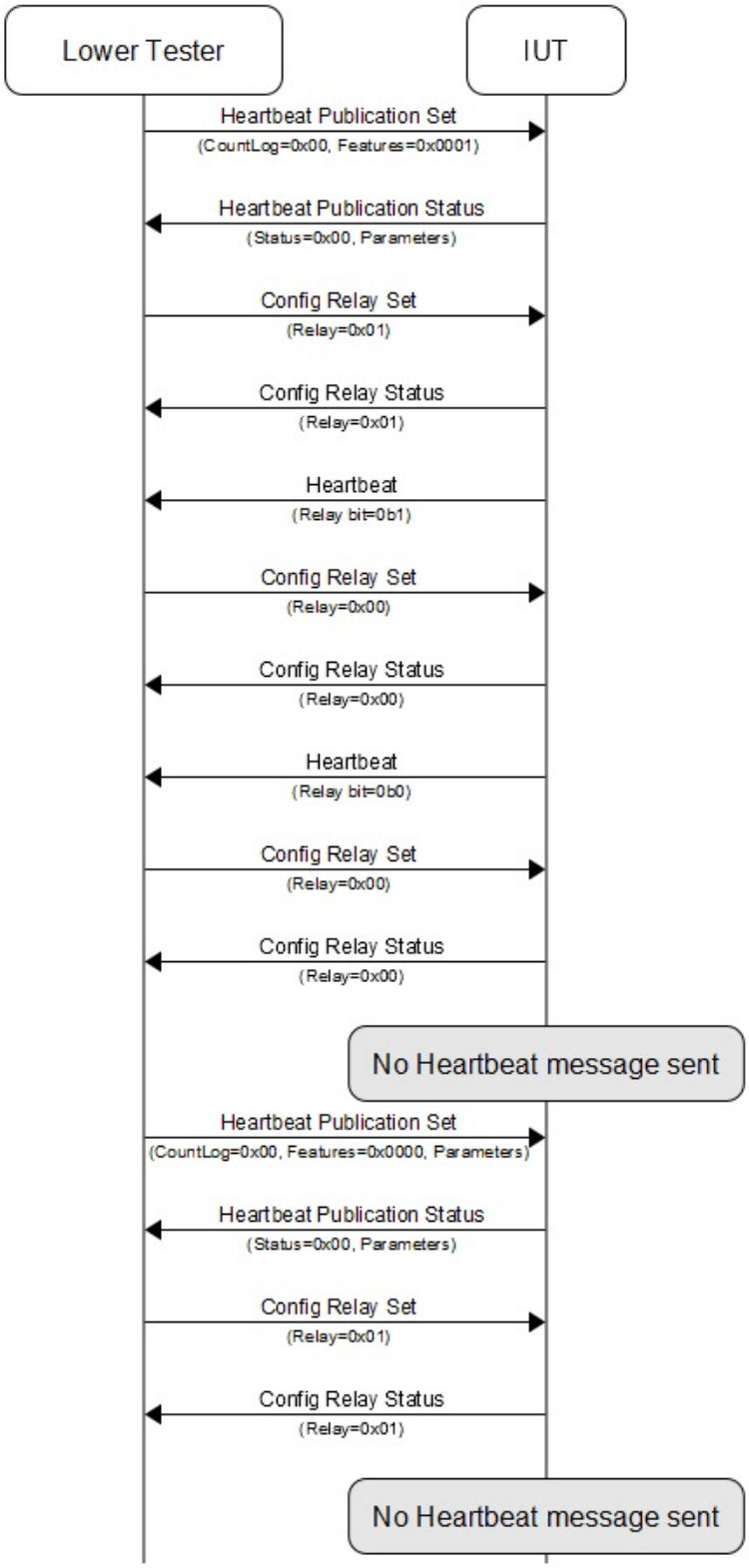


Figure 4.185: MESH/NODE/CFG/HBP/BV-03-C [Triggered publishing of Heartbeat messages – Relay State changes] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 1.

  In Step 4, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and a Heartbeat message with the Relay bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in Step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message. The Opcode field of the Transport Control message is set to 0x0A and the NetKey used is from Step 1.

  In Step 6, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and a Heartbeat message with the Relay bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in Step 4.

  In Step 8, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and does not send a Heartbeat message.

  In Step 10, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 9.

  In Step 12, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and does not send a Heartbeat message.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BV-04-C [Triggered publishing of Heartbeat messages - Friend state changes]

- Test Purpose

  Verify that the IUT can send Heartbeat Messages triggered by the Friend State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.13, 4.3.2.56, 4.3.2.57, 4.3.2.62, 4.3.2.63, 4.4.1.2.5, 4.4.2.2.5

- Initial Condition
  - The IUT has the Friend feature disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.

- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0004 and all the other fields set to valid values.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
4. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x01 and send a Heartbeat message with the Friend bit of the Features field set to 1.
5. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x00.
6. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x00 and send a Heartbeat message with the Friend bit of the Features field set to 0.
7. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.
8. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
9. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x01.
10. The Lower Tester expects the IUT to respond with a Config Friend Status message with the Friend field set to 0x01, but not to send any Heartbeat message.

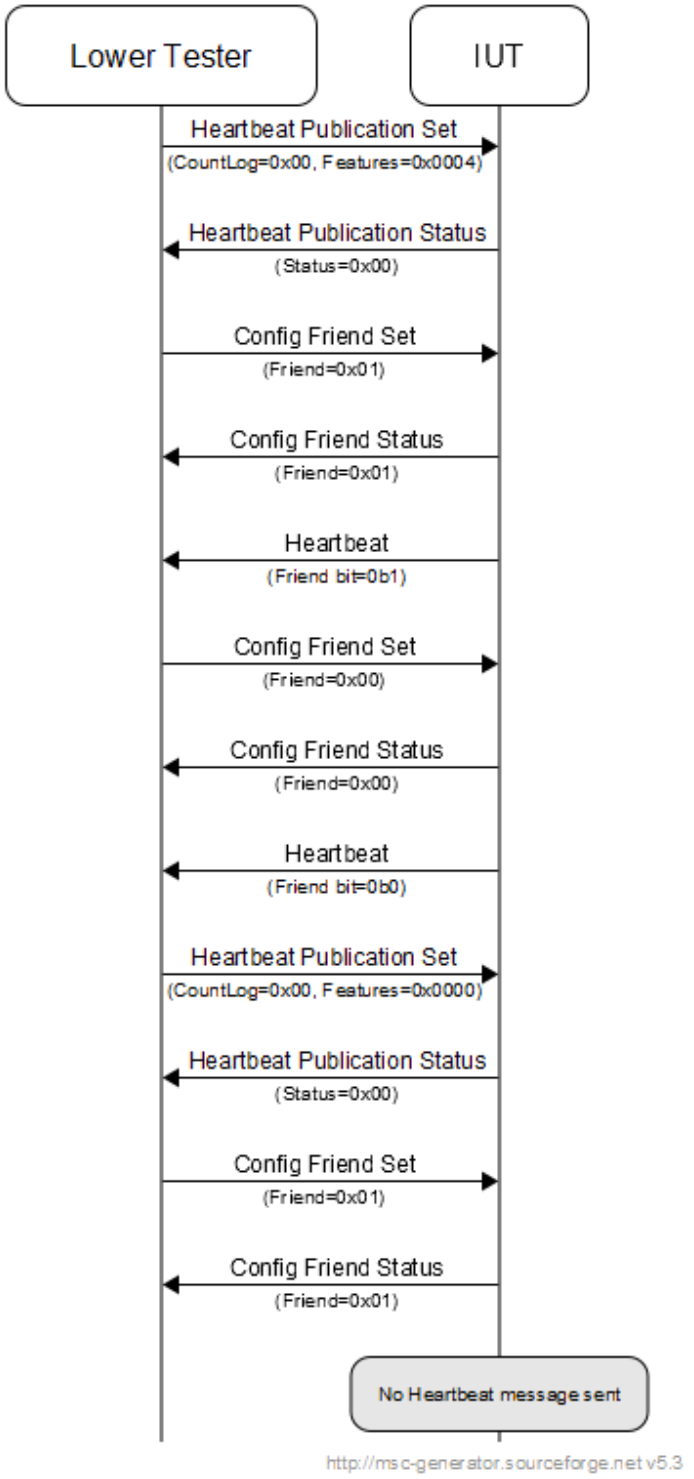


Figure 4.186: MESH/NODE/CFG/HBP/BV-04-C [Triggered publishing of Heartbeat messages - Friend state changes] MSC

- Expected Outcome

  Pass verdict

In Step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 1.

In Step 4, the IUT responds with a Config Friend Status message with the Friend field set to 0x01 and sends a Heartbeat message with the Friend bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in Step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message.

In Step 6, the IUT responds with a Config Friend Status message with the Friend field set to 0x00 and sends a Heartbeat message with the Friend bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in Step 4.

In Step 8, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 7.

In Step 10, the IUT responds with a Config Friend Status message with the Friend field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BV-05-C [Triggered publishing of Heartbeat messages - Low Power State changes]

- Test Purpose

  Verify that the IUT supporting Low Power feature can send Heartbeat Messages triggered by the Low Power State changes on the node, when configured to do so by a node supporting the Configuration Client Model.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.5, 4.3.2.62, 4.3.2.63

- Initial Condition
  - The IUT has the Low Power feature enabled.
  - Lower Tester 1 is a node supporting the Friend feature and the feature is enabled.
  - Lower Tester 2 is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT, Lower Tester 1, and Lower Tester 2 share previously established network security credentials.
  - Lower Tester 2 has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The initial state of the Low Power bit of the Heartbeat Publication Features Values is set (i.e., 0b1).

- Test Procedure

1. The Upper Tester orders the IUT to establish a friendship with Lower Tester 1, following the procedure described in MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
2. Lower Tester 2 sends a Config Heartbeat Publication Set message to the IUT, with the Destination field set to Lower Tester 2's unicast address, the CountLog field set to 0x00, the Features field set to 0x0008 and all the other fields set to valid values.
3. The Upper Tester orders the IUT to send the Friend Poll message with the FSN field set to 0 to Lower Tester 1.
4. Lower Tester 1 forwards the Config Heartbeat Publication Set message to the IUT.
5. Lower Tester 2 expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
6. The Upper Tester orders the IUT to send four Friend Poll messages with the FSN field set to 1, 0, 1, and 0, respectively.
7. Lower Tester 1 does not respond to the four Friend Poll messages within the PollTimeout (in Friend Request message) from the IUT, which will cause the friendship to be terminated.
8. Lower Tester 2 expects the IUT to send a Heartbeat message with the Low Power bit of the Features field set to 0.
9. Repeat Step 1 to re-establish the friendship.
10. Lower Tester 2 expects the IUT to send a Heartbeat message with the Low Power bit of the Features field set to 1.
11. Lower Tester 2 sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.
12. Repeat Steps 3–4.
13. Lower Tester 2 expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
14. Repeat Step 6.
15. Lower Tester 2 expects the IUT not to send any Heartbeat message.

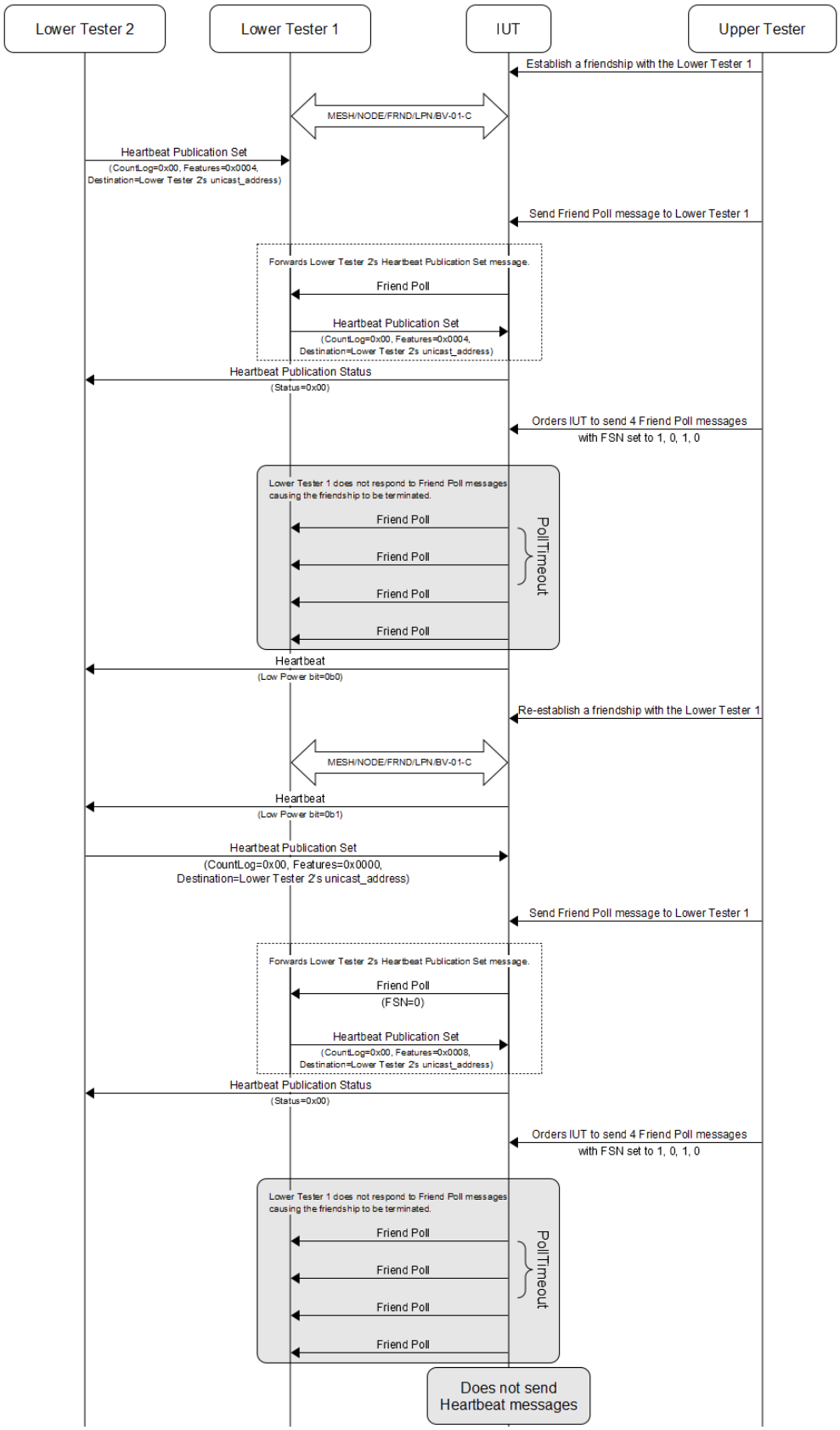


Figure 4.187: MESH/NODE/CFG/HBP/BV-05-C [Triggered publishing of Heartbeat messages - Low Power State changes] MSC

- Expected Outcome

  Pass verdict

  In Step 5, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message in Step 2.

  In Step 8, the IUT sends a Heartbeat message with the Low Power bit of the Features field set to 0. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message in Step 2, the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message and the NetKey used is the NetKey identified by the NetKeyIndex used in Step 2.

  In Step 10, the IUT sends a Heartbeat message with the Low Power bit of the Features field set to 1. The Heartbeat message has the DST field and the InitTTL field as in Step 2.

  In Step 13, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message in Step 11.

  In Step 15, the IUT does not send a Heartbeat message.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BV-06-C [Disabling publishing of Heartbeat messages – Config NetKey Deleted]

- Test Purpose

  Verify that the IUT disables Heartbeat Messages publishing when NetKey used in Heartbeat Publication is deleted because of the processing of the Config NetKey Delete message.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.17.2, 4.2.17.3, 4.3.2.3.3, 4.3.2.62, 4.4.1.2.9

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The IUT has a list of two NetKeys with NetKeyIndex of 0 and 1.

- Test Procedure

  1. The Lower Tester sets its Heartbeat Subscription Source to IUT's unicast address and Heartbeat Subscription PeriodLog to 0x11, and then sends a Config Heartbeat Publication Set message with the

Destination field set to the Lower Tester's unicast address, the CountLog field set to 0xFF, the PeriodLog field set to 0x01, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to NetKey Index of the NetKey to the IUT.

2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester expects the IUT to start sending periodical Heartbeat messages to the Lower Tester's unicast address every second.
4. The Lower Tester deletes the NetKey from IUT's NetKey List, following the procedure described in MESH/NODE/CFG/NKL/BV-03-C [Respond to Config NetKey Delete Message]. The NetKeyIndex field is an index that identifies the global NetKey Index of the NetKey specified in Step 1.
5. The Lower Tester expects the IUT to stop sending periodical Heartbeat messages. Wait 4 seconds, and the Lower Tester does not receive any Heartbeat messages.
6. The Lower Tester sends a Config Heartbeat Publication Set message with the Destination field set to the Lower Tester's unicast address, the CountLog field set to 0xFF, the PeriodLog field set to 0x01, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to NetKey Index of the NetKey just deleted to the IUT.
7. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x04 (Invalid NetKey) and the Destination, CountLog, PeriodLog, and TTL fields set to the values of corresponding fields of the incoming message.

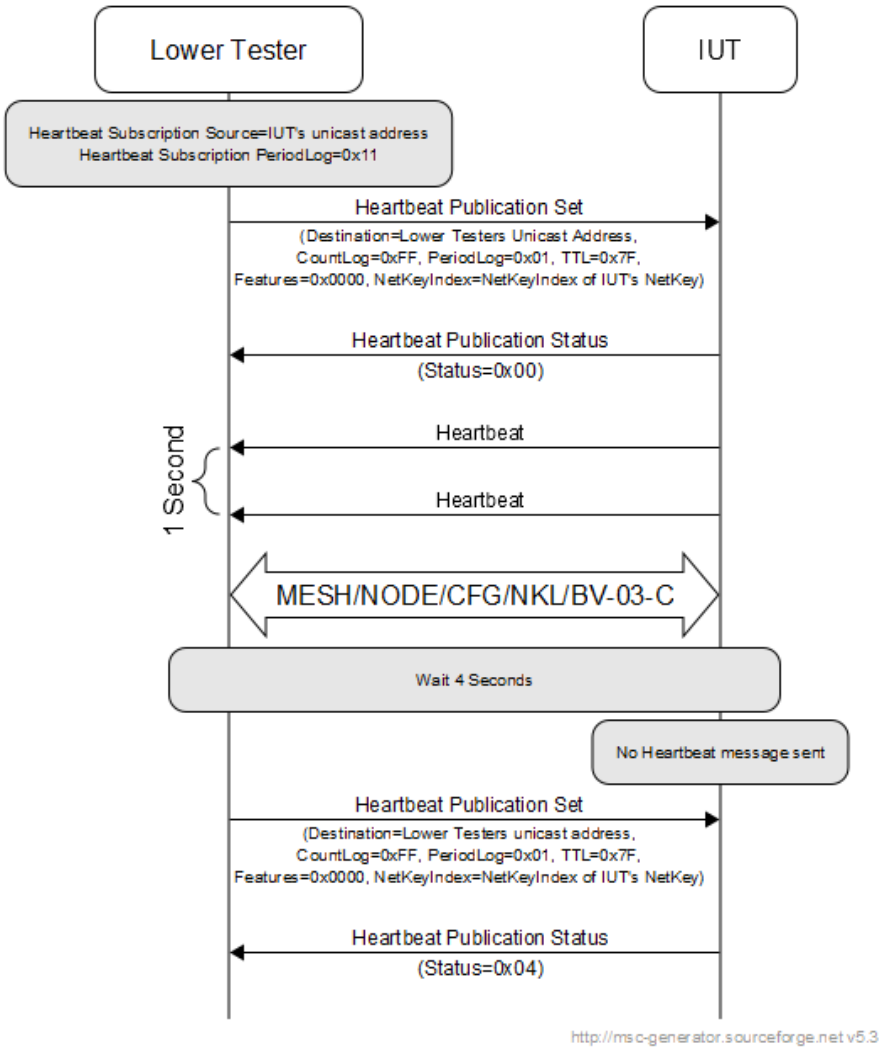


Figure 4.188: MESH/NODE/CFG/HBP/BV-06-C [Disabling publishing of Heartbeat messages – Config NetKey Deleted] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT starts sending periodical Heartbeat message every second and the NetKey used is the NetKey identified by the NetKeyIndex used in Step 1.

  In Step 5, the IUT stops sending periodical Heartbeat messages.

  In Step 7, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x04 (Invalid NeyKey) and the Destination, CountLog, PeriodLog, and TTL fields set to the values of corresponding fields of the incoming message.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BV-07-C [Triggered publishing of Heartbeat messages – Proxy state changes]

- Test Purpose

  Verify that the IUT can send Heartbeat Messages triggered by the Proxy state changes on the node, when configured to do so by a node supporting the Configuration Client Model.

- Reference

  [3] 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.11, 4.3.2.10, 4.3.2.11, 4.3.2.62, 4.3.2.63, 4.4.1.2.4, 4.4.2.2.4

- Initial Condition
  - The IUT supports the Proxy feature and the feature is disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0002 and all the other fields set to valid values.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
  3. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.
  4. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and to send a Heartbeat message with the Proxy bit of the Features field set to 1.
  5. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00.
  6. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and to send a Heartbeat message with the Proxy bit of the Features field set to 0.
  7. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00.
  8. The Lower Tester expects the IUT to respond with a Config GATTProxy Status message with the GATT Proxy field set to 0x00 and not to send any Heartbeat message.
  9. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT, with the CountLog field set to 0x00, the Features field set to 0x0000 and all the other fields set to valid values.
  10. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.

11. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x01.
12. The Lower Tester expects the IUT to respond with a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and not to send any Heartbeat message.

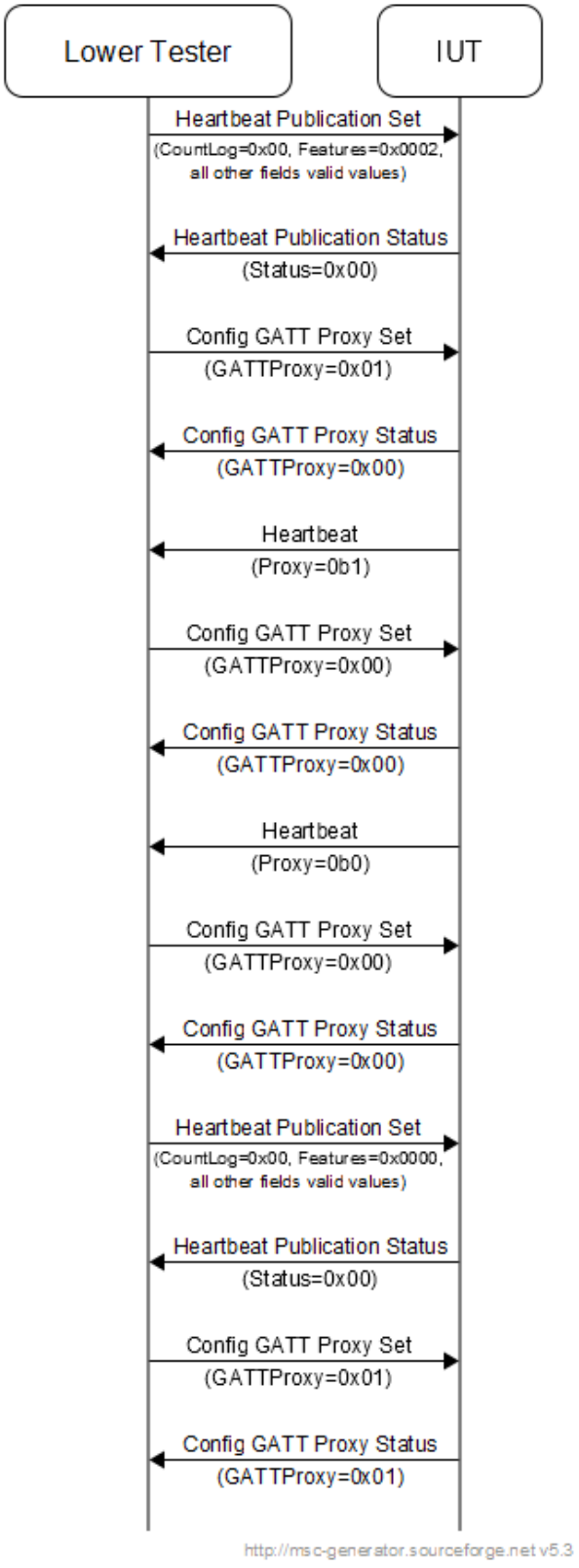


Figure 4.189: MESH/NODE/CFG/HBP/BV-07-C [Triggered publishing of Heartbeat messages – Proxy state changes] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 1.

  In Step 4, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and a Heartbeat message with the Proxy bit of the Features field set to 1. The Heartbeat message has the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in Step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set message. The Opcode field of the Transport Control message is set to 0x0A and the NetKey used is from Step 1.

  In Step 6, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and a Heartbeat message with the Proxy bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field as in Step 4.

  In Step 8, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x00 and does not send a Heartbeat message.

  In Step 10, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 9.

  In Step 12, the IUT sends a Config GATT Proxy Status message with the GATTProxy field set to 0x01 and does not send a Heartbeat message.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BI-01-C [Respond to Config Heartbeat Publication Set – Invalid Publication Features RFU Values]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat Publication Set message with invalid Publication Features RFU Values.

- Reference

  [3] 3.1.2, 3.6.5.10, 3.6.7.1, 3.6.7.2, 4.2.8, 4.2.17.5, 4.3.1, 4.3.2.13, 4.3.2.14, 4.3.2.62, 4.3.2.63, 4.4.1.2.6, 4.4.2.2.6

- Initial Condition
  - The IUT has the Relay mode disabled.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.

- The IUT and the Lower Tester share previously established network security credentials.
- The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

• Test Procedure

1. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with CountLog set to 0x00, the Features field set to 0xFFF9 (All RFU bits and Relay bit are set, Friend and Low Power bits are not set) and all the other fields set to valid values.
2. The Lower Tester expects the IUT to respond with a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0001 and the other fields set to the values of the corresponding fields from the received Config Heartbeat Publication Set message.
3. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.
4. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01 and to send a Heartbeat message with the Relay bit of the Features field set to 1.
5. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x00.
6. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x00 and to send a Heartbeat message with the Relay bit of the Features field set to 0.
7. The Lower Tester sends a Config Heartbeat Publication Set message to the IUT with CountLog set to 0x00, the Features field set to 0xFFF8 (All RFU bits are set and all Relay, Friend and Low Power bits are not set) and all the other fields set to valid values.
8. The Lower Tester sends a Config Relay Set message to the IUT with the Relay field set to 0x01.
9. The Lower Tester expects the IUT to respond with a Config Relay Status message with the Relay field set to 0x01, but not to send any Heartbeat message.

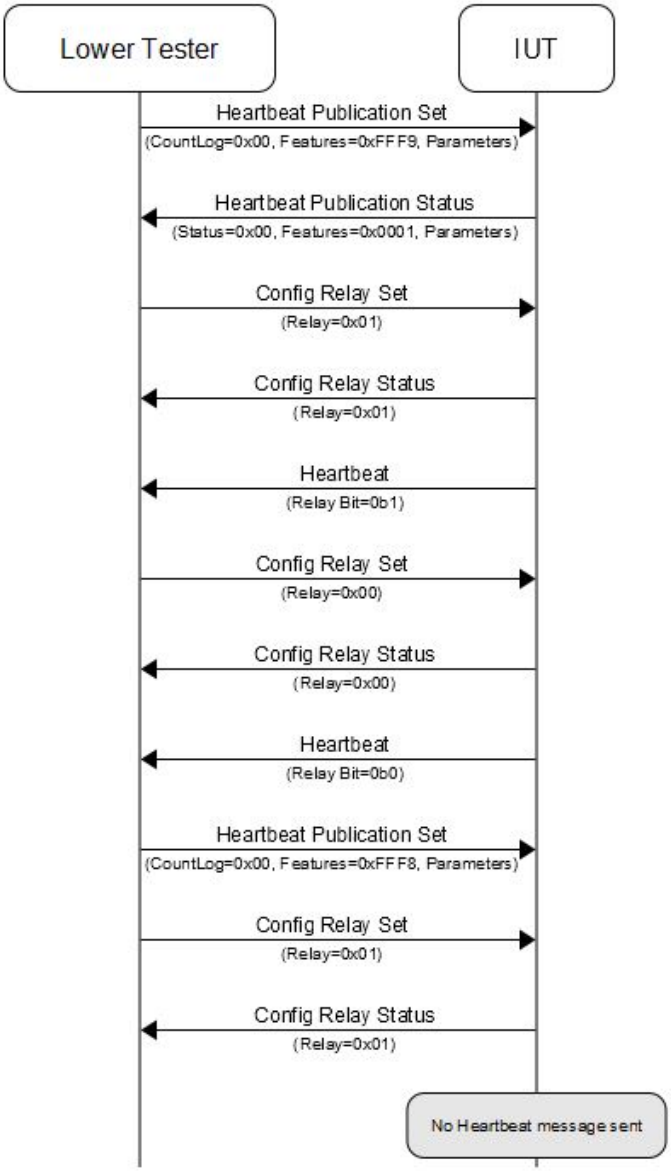


Figure 4.190: MESH/NODE/CFG/HBP/BI-01-C [Respond to Config Heartbeat Publication Set – Invalid Publication Features RFU Values] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Publication Status message with the Status field set to 0x00, the Features field set to 0x0001 and the other fields set to the values of the corresponding fields from the Config Heartbeat Publication Set message received in Step 1.

  In Step 4, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and a Heartbeat message with the Relay bit of the Features field set to 1. The Heartbeat message is published with the DST field set to the address contained in the Destination field of the Config Heartbeat Publication Set message received in Step 1, and the InitTTL field set to the TTL field of the same Config Heartbeat Publication Set

message.

In Step 6, the IUT sends a Config Relay Status message with the Relay field set to 0x00 and a Heartbeat message with the Relay bit of the Features field set to 0. The Heartbeat message has the DST field and the InitTTL field values set as set in Step 4.

In Step 9, the IUT sends a Config Relay Status message with the Relay field set to 0x01 and does not send a Heartbeat message.

All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBP/BI-02-C [Respond to Config Heartbeat Publication Get – Wrongly Encrypted by AppKey]

- Test Purpose

  Verify that an IUT can ignore an improperly encrypted Config Heartbeat Publication Get message.

- Reference

  [3] 3.7.4.3, 3.7.4.4, 4.3.2.61

- Initial Condition
  - The IUT is a node supporting the Configuration Server Model.
  - The Lower Tester is aw node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
  - The AppKey is known to the IUT within the context of the NetKey used.
- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Publication Get message to the IUT. This Configuration message is supposed to be encrypted by IUT's DevKey, but in this step, the Lower Tester intentionally encrypts the message using the AppKey.
  2. The Lower Tester expects the IUT not to send any response message.

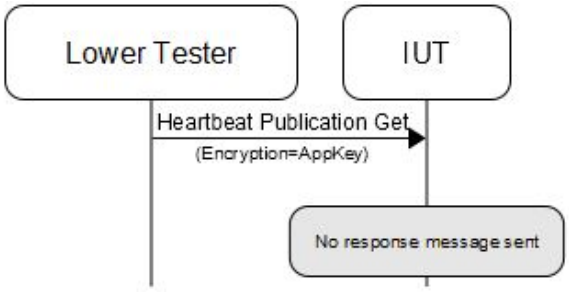


Figure 4.191: MESH/NODE/CFG/HBP/BI-02-C [Respond to Config Heartbeat Publication Get – Wrongly Encrypted by AppKey] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT ignores the incoming message encrypted by a wrong key and does not send any response message back.

MESH/CFGCL/CFG/HBP/BV-01-C [Send Config Heartbeat Publication Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can transmit a Config Heartbeat Publication Get/Set message.

- Reference

  [3] 3.6.7.1, 3.6.7.2, 4.2.17.1-4.2.17.6, 4.3.2.61, 4.3.2.62, 4.4.2.2.15

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials.
- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Heartbeat Publication Set message with the Destination field set to IUT's unicast address, the CountLog field set to 0x11, the PeriodLog field set to 0x02, the TTL field set to 0x7F, the Features field set to 0x0000, and the NetKeyIndex field set to valid NetKey Index to the Lower Tester.
  2. The Lower Tester expects to receive a Config Heartbeat Publication Set message from the IUT with all the fields set to the values in Step 1.
  3. The Lower Tester responds with a Config Heartbeat Publication Status message.
  4. The Upper Tester orders the IUT to send a Config Heartbeat Publication Get message to the Lower Tester.
  5. The Lower Tester expects to receive a Config Heartbeat Publication Get message from the IUT without parameters.

6. The Lower Tester responds with a Config Heartbeat Publication Status message.
7. Repeat Step 1 and 2 two more times with the Destination field set to the unassigned address, and a dynamically assigned group address.

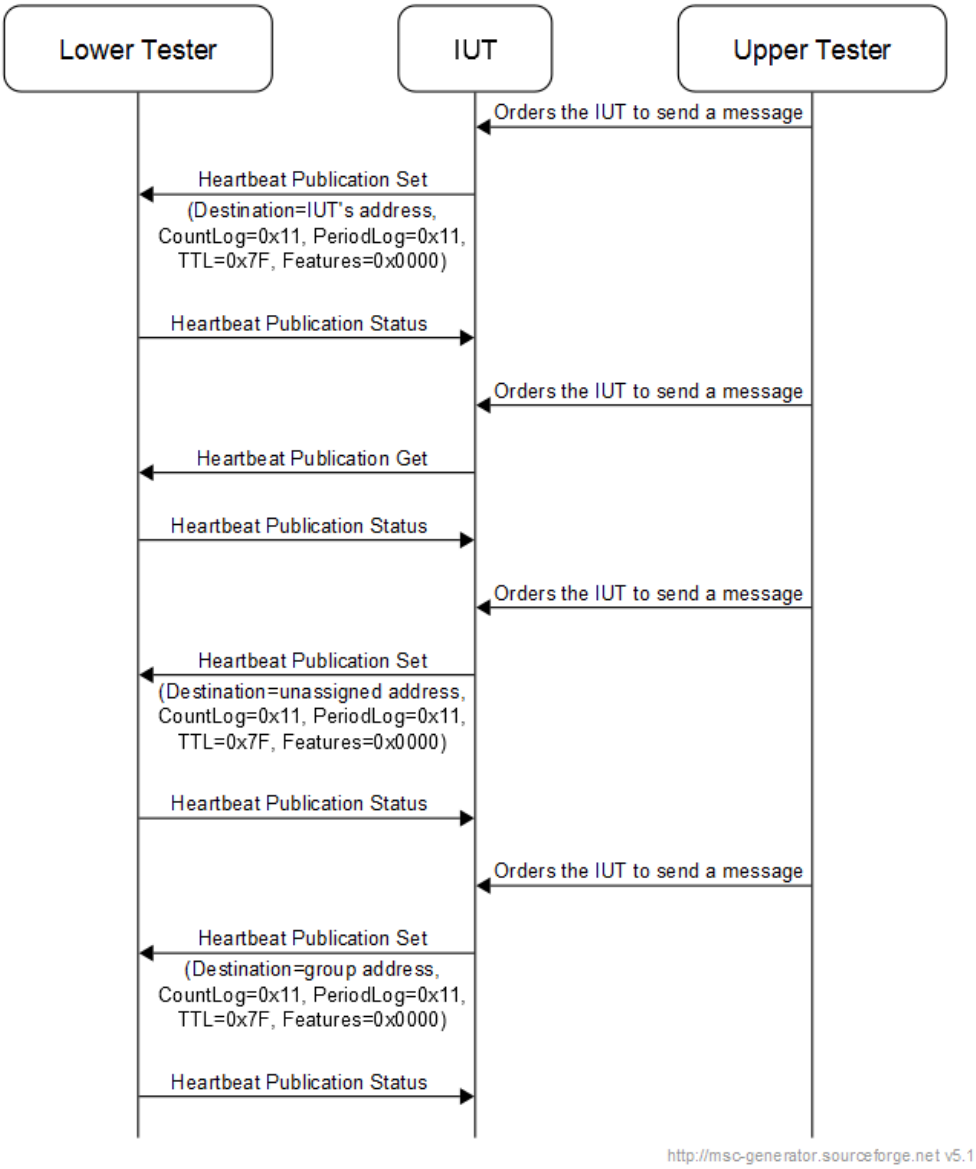


Figure 4.192: MESH/CFGCL/CFG/HBP/BV-01-C [Send Config Heartbeat Publication Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Publication Set message with all the fields set to the values in Step 1.

  In Step 4, the IUT sends a Config Heartbeat Publication Get message without parameters.

In Step 5, the IUT sends Config Heartbeat Publication Set message.

All configuration messages are encrypted and authenticated using the device key of the Lower Tester.

4.16.15 Heartbeat Subscription Procedures

This section validates the Heartbeat Subscription procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/HBS/BV-01-C [Receive Config Heartbeat Subscription Set/Get message]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to a Config Heartbeat Subscription Set/Get message, and update the identified Heartbeat Subscription state.

- Reference

  [3] 3.7.5.2, 4.2.18.3, 4.2.18.4, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16

- Initial Condition
  - The IUT is a node supporting the Configuration Server Model on the Primary Element.
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The Lower Tester supports the Configuration Server Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address and the PeriodLog field set to 0x11.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the values of the corresponding fields from the received Config Heartbeat Subscription Set message in Step 1, and each of the other 4 fields set to 1-octet value (which will be verified in the next test cases.)
  3. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  4. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same fields as the message received in Step 2.
  5. Repeat Step 1, but with Destination field set to the unassigned address 0x0000, and the PeriodLog field set to 0x10.
  6. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog and MaxHops fields set to 0x00.

7. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
8. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.
9. Repeat Step 1, but with the Source field set to the unassigned address 0x0000, and the PeriodLog field set to 0x10.
10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog and MaxHops fields are set to 0x00.
11. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
12. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

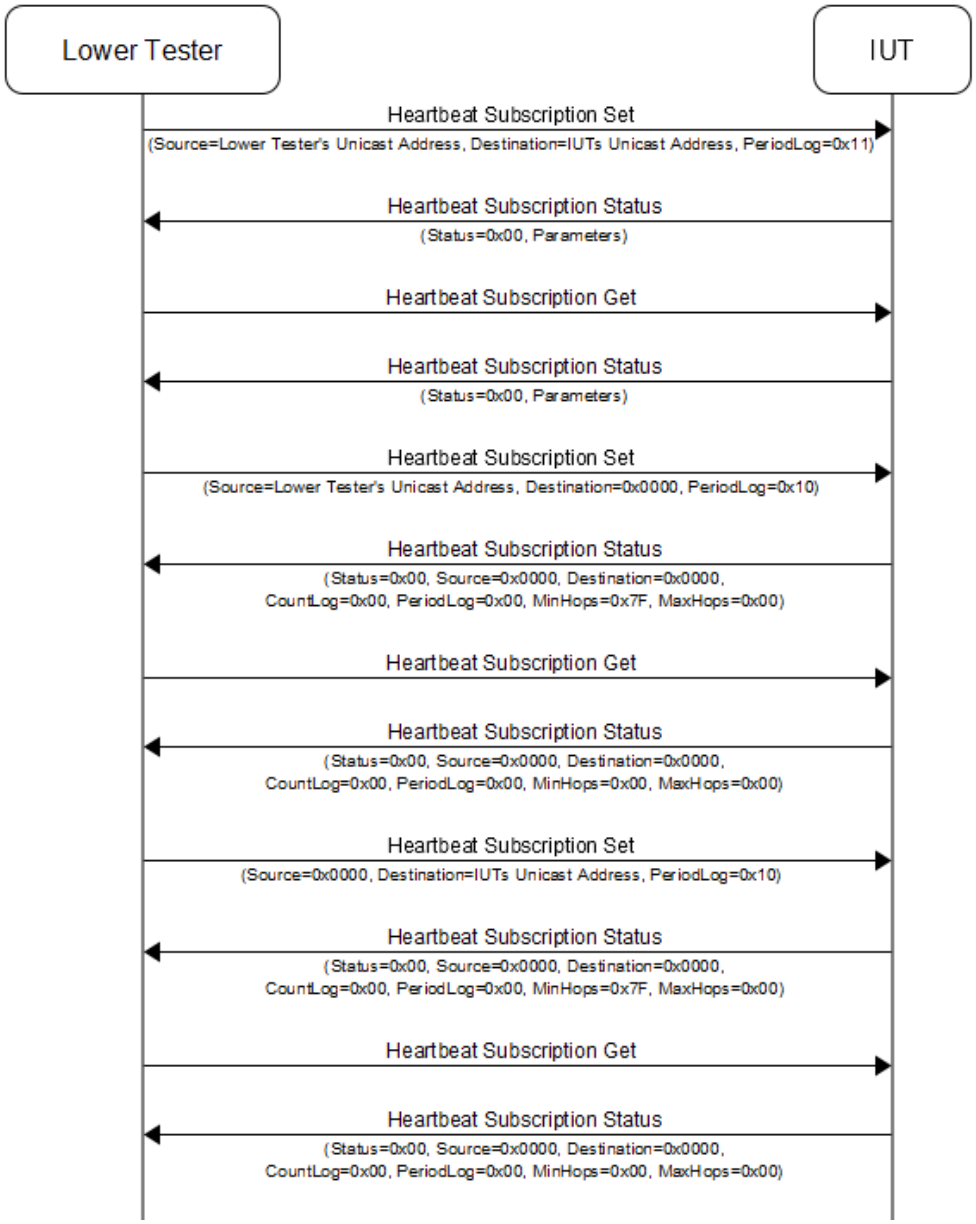


Figure 4.193: MESH/NODE/CFG/HBS/BV-01-C [Receive Config Heartbeat Subscription Set/Get message] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the values of the corresponding fields from the received Config Heartbeat Subscription Set message in Step 1, and each of the other 4 fields set to 1-octet value.

  In Step 4, the IUT sends a Config Heartbeat Subscription Status message with the same fields as the message received in Step 2.

  In Steps 6 and 10 the IUT sends a Config Heartbeat Subscription Status message with a Config Heartbeat

Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, the MinHops field set to 0x7F, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

In Steps 8 and 12 the IUT sends a Config Heartbeat Subscription Status message with a Config Heartbeat Subscription Status message with the Status field set to 0x00, the Source and Destination fields set to the unassigned address 0x0000, and the CountLog, PeriodLog, MinHops and MaxHops fields set to 0x00.

All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBS/BV-02-C [Receive Heartbeat messages – Controlled by PeriodLog State]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can receive the periodic Heartbeat messages controlled by Heartbeat Subscription PeriodLog state.

- Reference

  [3] 3.6.7.2, 4.2.18.1-4.2.18.4, 4.3.2.64 - 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by a NetKeyIndex.
  - The IUT and the Lower Testers share previously established network security credentials.
  - The Lower Testers have access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and the PeriodLog field set to 0x00.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
  3. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x0000.
  4. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  5. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
  6. Repeat Step 1, but with the PeriodLog field set to 0x02.
  7. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.
  8. Within 2 seconds, repeat Step 3.
  9. After 2 seconds, the Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.

10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and the PeriodLog field set to 0x00.
11. Repeat Steps 3 and 4.
12. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and the PeriodLog field set to 0x00.
13. Repeat Steps 1–12, using the same address as used in Step 1 (for the Destination field) and in Step 3 (for the DST field), but in this iteration expect the CountLog field in Step 2 to be equal to 0x01.
14. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and the PeriodLog field set to 0x10.
15. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.
16. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x0000.
17. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
18. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.
19. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and the PeriodLog field set to 0x00.
20. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.

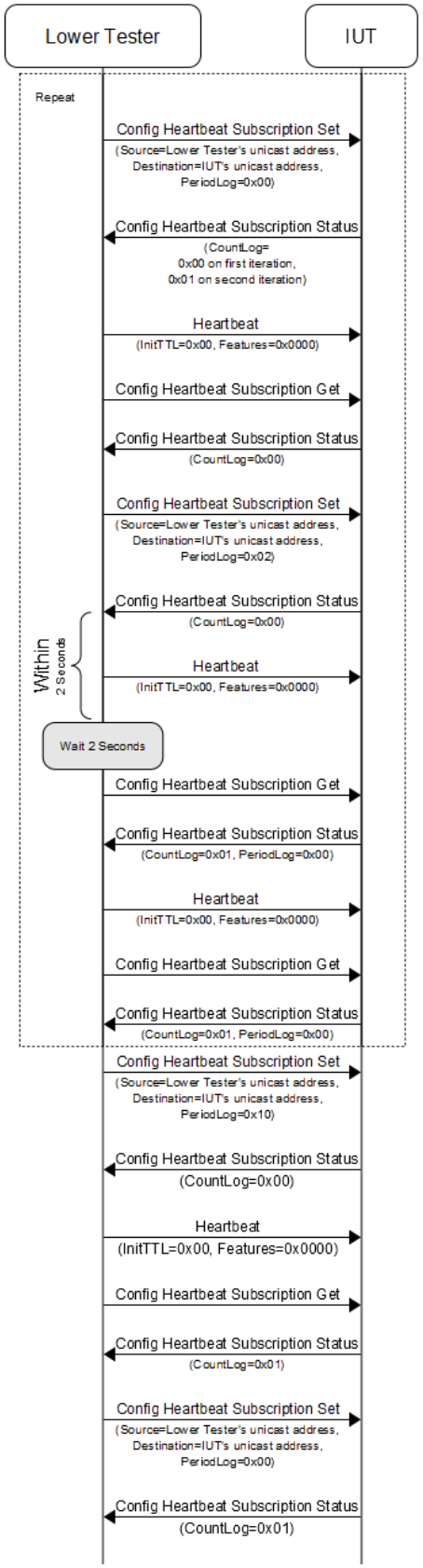


Figure 4.194: MESH/NODE/CFG/HBS/BV-02-C [Receive Heartbeat messages – Controlled by PeriodLog State] MSC

- Expected Outcome

  Pass verdict

  In the first iteration through Steps 1–12, the following are true:

  In Step 2, 5, and 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

  In Steps 10 and 12, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and PeriodLog field set to 0x00.

  In the second iteration through Steps 1–12, the following are true:

  In Step 2, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.

  In Steps 5 and 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

  In Steps 10 and 12, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01 and PeriodLog field set to 0x00.

  In Step 15, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

  In Steps 18 and 20, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBS/BV-03-C [Not receiving Heartbeat messages – Unassigned or Mismatched SRC field]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model cannot receive the periodic Heartbeat messages if Source/Destination field is set to an unassigned address or Source mismatches with SRC field.

- Reference

  [3] 3.6.7.2, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to an address which is not the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and PeriodLog field is set to 0x02.
  2. Within 2 seconds, the Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x00 and the Features field set to 0x0000.
  3. After 2 seconds, the Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  4. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
  5. Repeat Step 1, but with the Source field set to the Lower Tester's unicast address and the Destination field set to an address which is not the IUT's unicast address.
  6. Repeat Steps 2 and 3.
  7. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the CountLog field is set to 0x00.
  8. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the unassigned address 0x0000, Destination field set to the IUT's unicast address, and PeriodLog field is set to 0x02.
  9. Repeat Steps 2 and 3.
  10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the Source and Destination fields are set to the unassigned address, and the values of the CountLog, PeriodLog, MinHops, and MaxHops fields are set to 0x00.
  11. Repeat Step 8, but with the Source field set to the Lower Tester's unicast address, and Destination field set to the unassigned address 0x0000.
  12. Repeat Steps 2 and 3.
  13. The Lower Tester expects the IUT to respond with the same Config Heartbeat Subscription Status message as that in Step 10.

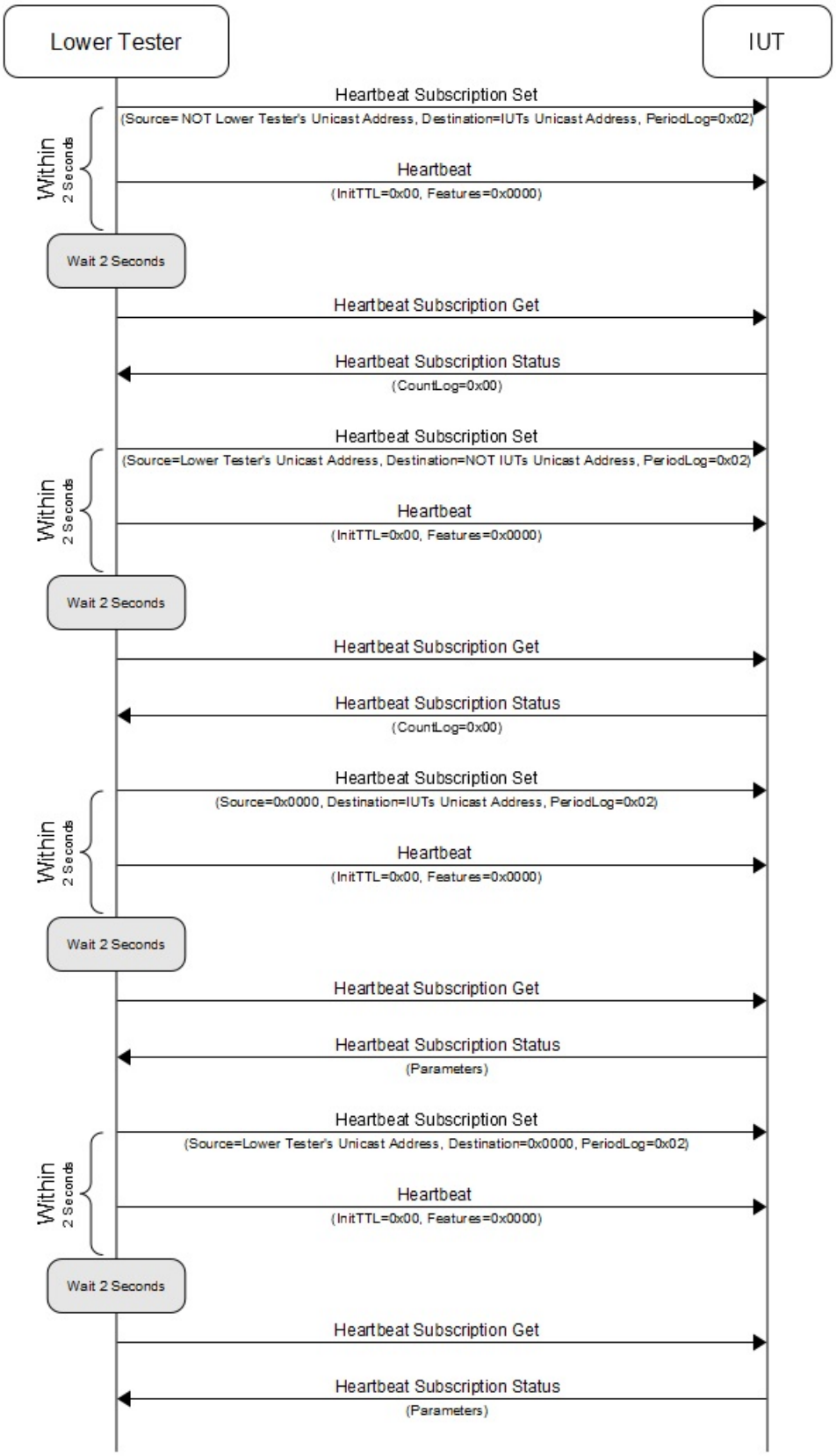


Figure 4.195: MESH/NODE/CFG/HBS/BV-03-C [Not receiving Heartbeat messages – Unassigned or Mismatched SRC field] MSC

- Expected Outcome

  Pass verdict

  In Steps 4 and 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00.

  In Steps 10 and 13, the IUT responds with a Config Heartbeat Subscription Status message, and in the message the Source and Destination fields are set to the unassigned address, and the CountLog, PeriodLog, MinHops, and MaxHops fields are set to 0x00.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBS/BV-04-C [Receive Heartbeat messages – CountLog and Hops]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can receive the periodic Heartbeat messages and update the state accordingly.

- Reference

  [3] 3.6.7.2, 4.2.18.3, 4.2.18.4, 4.2.18.5, 4.2.18.6, 4.3.2.64-4.3.2.66, 4.4.2.2.15

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model and the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to IUT's unicast address, and the PeriodLog field is set to 0x00.
  2. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to IUT's unicast address, and the PeriodLog field is set to 0x11.
  3. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  4. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message with the CountLog field set to 0x00, the MinHops field set to 0x7F and the MaxHops field set to 0x00.
  5. The Lower Tester sends a Heartbeat message to the IUT with the InitTTL field set to 0x02 and the Features field set to 0x0000.
  6. The Lower Tester sends a Config Heartbeat Subscription Get message to IUT.
  7. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x01, the MinHops field is set to 0x01 and the MaxHops field

is set to 0x01.

8. Repeat Steps 5 and 6.
9. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x02.
10. Repeat Steps 5 and 6.
11. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x02.
12. Repeat Steps 5 and 6.
13. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x03.
14. Repeat Steps 5 and 6.
15. The Lower Tester expects the IUT to reply with a Config Heartbeat Subscription Status message. In the message, the CountLog field is set to 0x03.

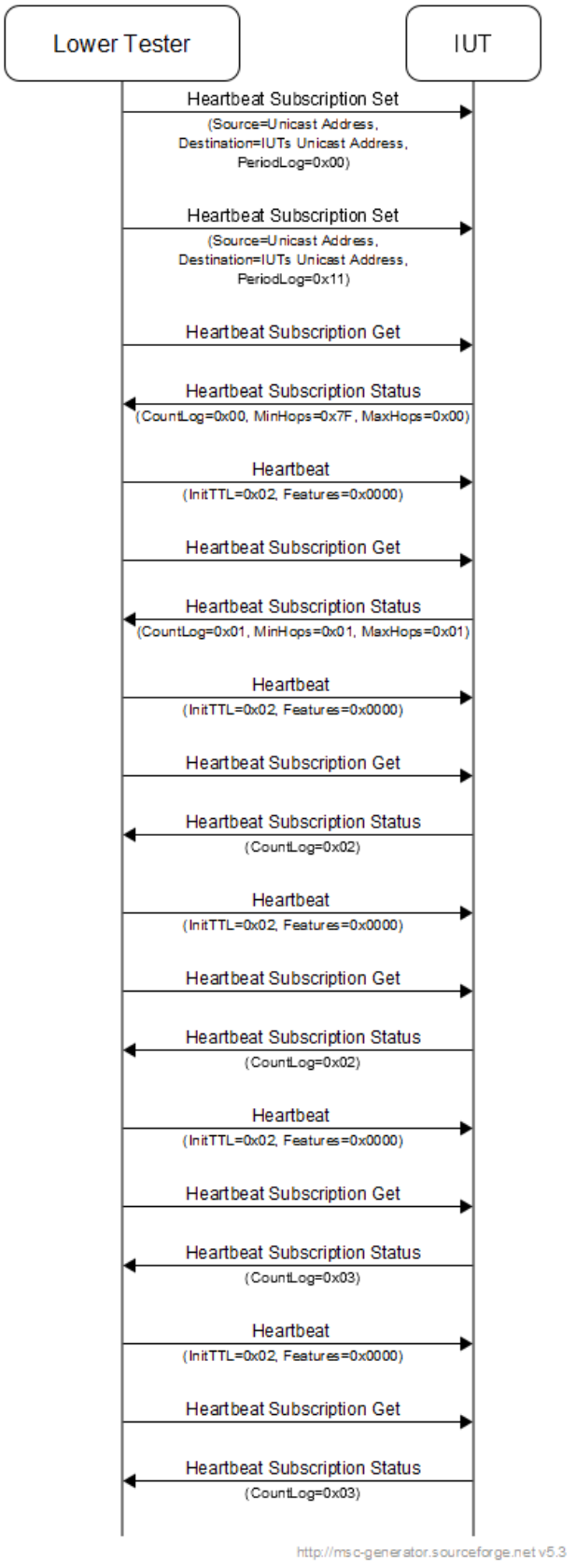


Figure 4.196: MESH/NODE/CFG/HBS/BV-04-C [Receive Heartbeat messages – CountLog and Hops] MSC

- Expected Outcome

  Pass verdict

  In Step 4, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x00, the MinHops field set to 0x7F and the MaxHops field set to 0x00.

  In Step 7, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x01, the MinHops field set to 0x01 and the MaxHops field set to 0x01.

  In Step 9, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x02.

  In Step 11, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x02.

  In Step 13, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x03.

  In Step 15, the IUT sends a Config Heartbeat Subscription Status message with the CountLog field set to 0x03.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBS/BV-05-C [Receive Heartbeat messages – Friendship]

- Test Purpose

  Verify that an IUT acting as Friend node can cache periodic Heartbeat messages and forward them to its Low Power node if the DST of the messages is Low Power node's unicast address or a dynamically-assigned group address that has been added to the Friend Subscription List.

- Reference

  [3] 3.6.5.10, 3.6.7.2, 4.2.13, 4.2.18.1, 4.2.18.2, 4.3.2.65, 4.4.1.2.16, 4.4.2.2.16

- Initial Condition
  - The IUT has the Friend feature enabled.
  - Lower Tester 1 is a node supporting the Low Power feature and the feature is enabled.
  - Lower Tester 2 is a node supporting the Configuration Client Model.
  - The IUT has been provisioned.
  - The IUT and the Lower Testers share previously established network security credentials.
  - Lower Tester 2 has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. Lower Tester 1 establishes a friendship with the IUT, following the procedure described in MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].

2. Lower Tester 1 sends a Friend Subscription List Add message to the IUT with the AddressList field value containing Lower Tester 1's group address and a random TransactionNumber field value.
3. Lower Tester 1 expects to receive a Friend Subscription List Confirm message with a TransactionNumber field equal to that of the value it sent in Step 2 from the IUT.
4. Lower Tester 2 sends a Config Heartbeat Subscription Set message with the Source field set to Lower Tester 2's unicast address, the Destination field set to LPN's (Lower Tester 1's) unicast address, and the PeriodLog field set to 0x11. The DST field of the Network PDU containing this message is set to the unicast address of the Lower Tester 1, therefore the IUT is expected to cache the PDU in its Friend Queue.
5. Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.
6. Lower Tester 1 expects the IUT to forward the message received and cached in Step 4.
7. Lower Tester 2 sends a Heartbeat message, with the DST field set to the unicast address of the LPN, and with the InitTTL field set to 0x7F and the Features field set to 0x0002.
8. Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.
9. Lower Tester 1 expects the IUT to forward the Heartbeat message received and cached in Step 7.
10. Repeat Steps 4–6, but with the Destination field set to LPN's (Lower Tester 1's) group address.
11. Lower Tester 2 sends a Heartbeat message to the IUT with the DST field set to the LPN's group address.
12. Lower Tester 1 sends a Friend Poll message to the IUT with the FSN field toggled.
13. Lower Tester 1 expects the IUT to forward the Heartbeat message received and cached in Step 11.

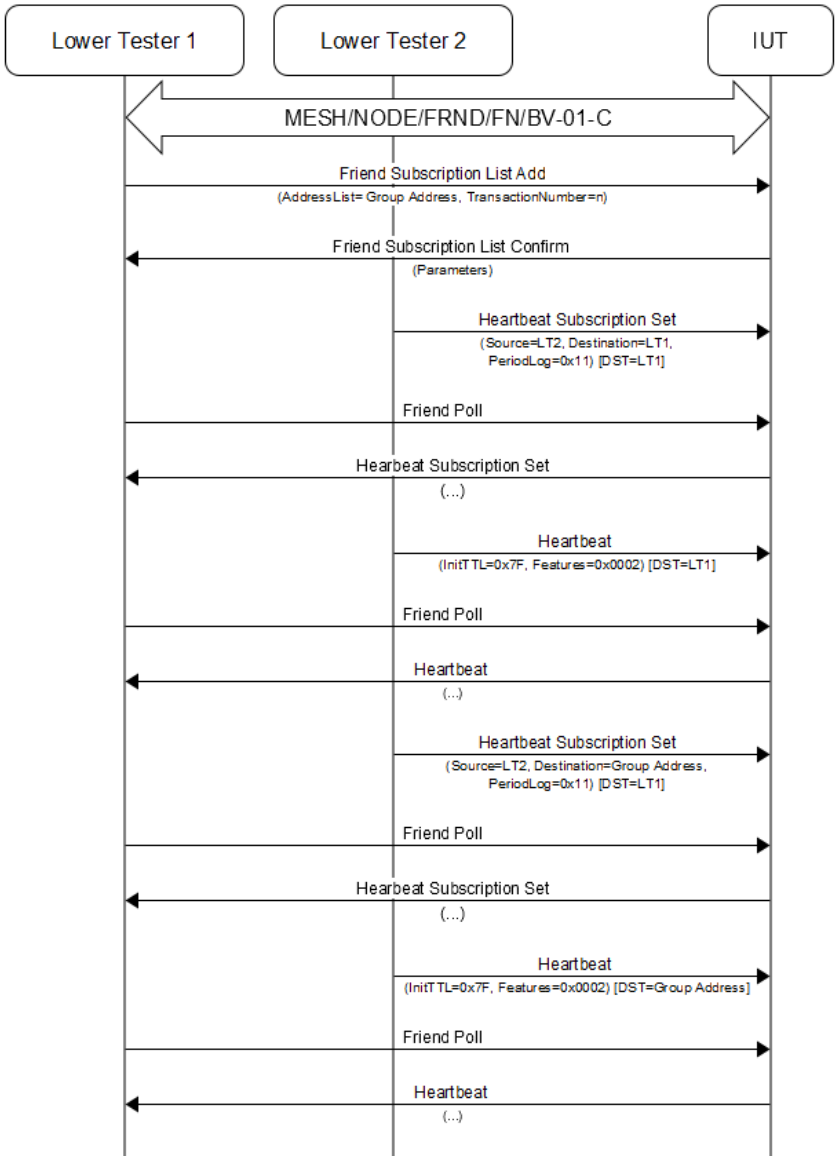


Figure 4.197: MESH/NODE/CFG/HBS/BV-05-C [Receive Heartbeat messages – Friendship] MSC

- Expected Outcome

  Pass verdict

  In Steps 9 and 13, the IUT forwards the Heartbeat message to Lower Tester 1.

  All configuration messages are encrypted and authenticated using the device key of the IUT.

MESH/NODE/CFG/HBS/BI-01-C [Ignore Config Heartbeat Subscription Set Message –Invalid Source Address or Destination Address]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores the Config Heartbeat Subscription Set message if with invalid Source field.

- Reference

  [3] 3.6.7.2, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.1.2.16, 4.4.2.2.16

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Client Model.
  - The IUT has been provisioned with a NetKey identified by a NetKeyIndex of 0x000.
  - The IUT and the Lower Tester share previously established network security credentials.
  - The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure
  1. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's unicast address, the Destination field set to the IUT's unicast address, and PeriodLog field set to 0x01.
  2. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message, and in the message, the Source field and the Destination field are set as set in Step 1.
  3. The Lower Tester sends a Config Heartbeat Subscription Set message to IUT with the Source field set to the Lower Tester's group address, the Destination field set to the IUT's unicast address, and PeriodLog field is set to 0x05.
  4. The Lower Tester expects the IUT not to respond with any messages.
  5. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in Step 2.
  7. Repeat Step 3, but with the Source field set to the all nodes address.
  8. The Lower Tester expects the IUT not to respond with any messages.
  9. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  10. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in Step 2.
  11. Repeat Step 3, but with the Source field set to a virtual address.
  12. The Lower Tester expects the IUT not to respond with any messages.
  13. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  14. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in Step 2.
  15. Repeat Step 1, but with the Destination field set to a virtual address.
  16. The Lower Tester expects the IUT not to respond with any messages.
  17. The Lower Tester sends a Config Heartbeat Subscription Get message to the IUT.
  18. The Lower Tester expects the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in Step 2.

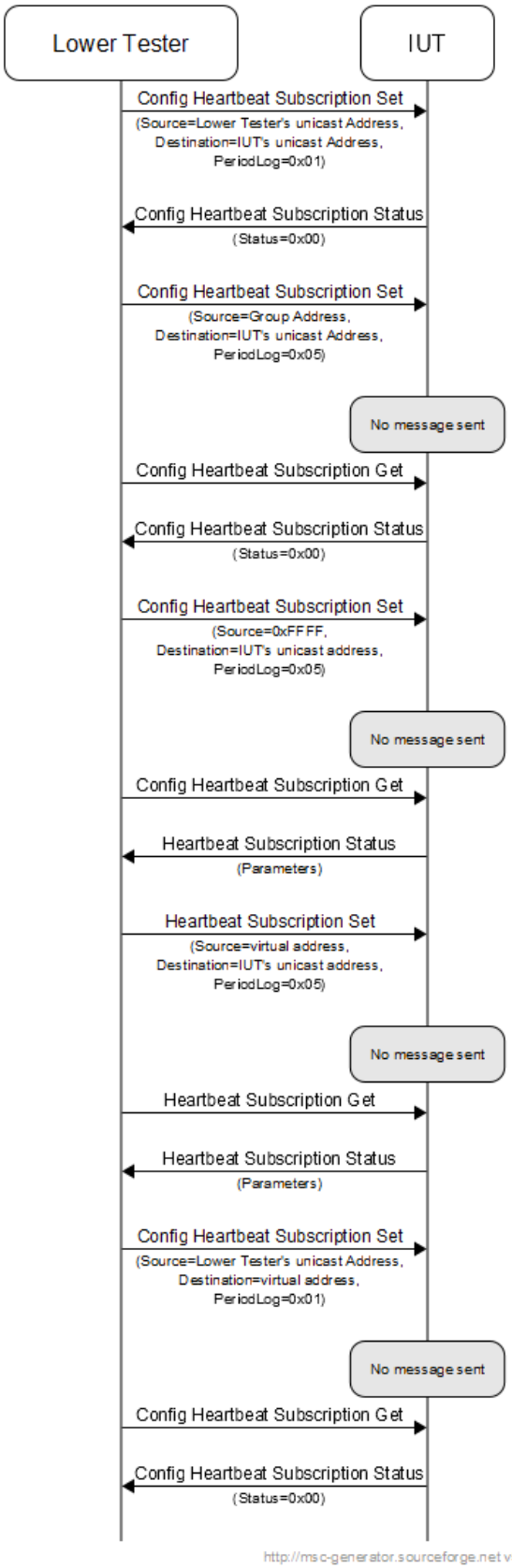


Figure 4.198: MESH/NODE/CFG/HBS/BI-01-C [Ignore Config Heartbeat Subscription Set Message –Invalid Source Address or Destination Address] MSC

- Expected Outcome

  Pass verdict

  In Steps 4, 8, 12, and 16 the IUT does not respond with any messages.

  In Steps 6, 10, 14, and 18 the IUT to respond with a Config Heartbeat Subscription Status message with the same Status, Source and Destination fields as that in Step 2.

MESH/CFGCL/CFG/HBS/BV-01-C [Send Config Heartbeat Subscription Set/Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can transmit a Config Heartbeat Subscription Set/Get message.

- Reference

  [3] 3.6.7.1, 4.2.18.1, 4.2.18.2, 4.3.2.64, 4.3.2.65, 4.3.2.66, 4.4.2.2.16

- Initial Condition
  - The Lower Tester is a node supporting the Configuration Server Model.
  - The IUT and the Lower Tester share previously established network security credentials.

- Test Procedure
  1. The Upper Tester orders the IUT to send a Config Heartbeat Subscription Set message to the Lower Tester with Source field set to the IUT's unicast address, the Destination field set to the Lower Tester's unicast address, and the PeriodLog field set to 0x11.
  2. The Lower Tester receives the Config Heartbeat Subscription Set message and replies with a Config Heartbeat Subscription Status message.
  3. The Upper Tester orders the IUT to send a Config Heartbeat Subscription Get message to the Lower Tester.
  4. The Lower Tester receives the Config Heartbeat Subscription Get message and replies with a Config Heartbeat Subscription Status message.
  5. Repeat Steps 1–2 two times first with the Destination field set to unassigned address 0x0000, then a dynamically-assigned group address.

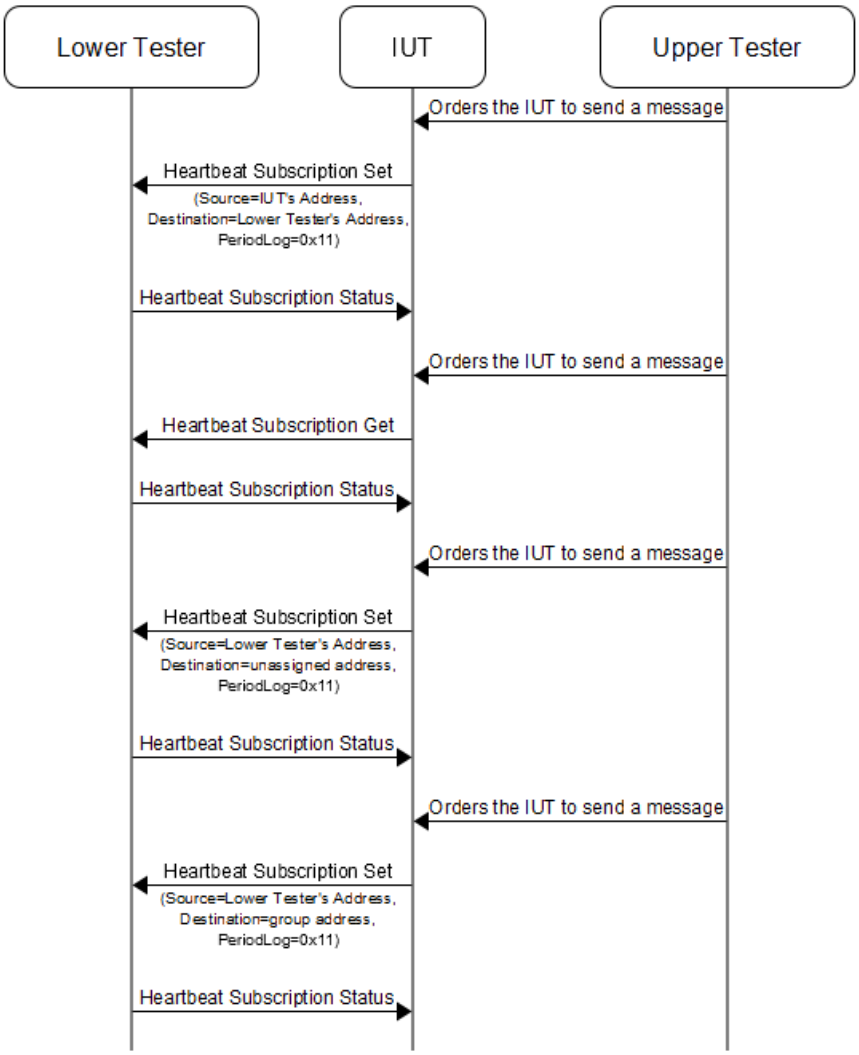


Figure 4.199: MESH/CFGCL/CFG/HBS/BV-01-C [Send Config Heartbeat Subscription Set/Get Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Heartbeat Subscription Set message with Source field set to the IUT's unicast address, the Destination field set to the Lower Tester's unicast address, and the PeriodLog field set to 0x11.

  In Step 4, the IUT sends a Config Heartbeat Subscription Get message without parameters.

  In Step 5, the IUT sends a Config Heartbeat Subscription Set message with the values specified in Step 5.

  All configuration messages are encrypted and authenticated using the device key of the Lower Tester.

4.16.16 Network Transmit Procedures

This section validates the Network Transmit procedures for both the Configuration Server Model and the Configuration Client Model.

MESH/NODE/CFG/NTX/BV-01-C [Respond to Config Network Transmit Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model can respond to Config Network Transmit Get and Config Network Transmit Set messages.

- Reference

  [3] 3.7.4.3, 4.3.2.69, 4.3.2.70, 4.3.2.71, 4.4.1.2

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element.
  - The Lower Tester acts as Configuration Client.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Network Transmit Get message to the IUT.
  2. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT.
  3. The Lower Tester sends a Config Network Transmit Set message to the IUT with the NetworkTransmitCount and the NetworkTransmitIntervalSteps fields set to values different than the ones received in Step 2.
  4. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT with the field values equal to the ones sent in Step 3.
  5. The Lower Tester sends a Config Network Transmit Get message to the IUT.
  6. The Lower Tester expects to receive a Config Network Transmit Status message in response from the IUT with the field values equal to the ones sent in Step 3.

- Expected Outcome

  Pass verdict

  In Steps 2, 4, and 6, the IUT sends a Config Network Transmit Status message to the Lower Tester. The field values in Steps 4 and 6 are equal to the values requested in Step 3.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/NTX/BV-01-C [Send Config Network Transmit Get/Set Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Network Transmit Get and Config Network Transmit Set messages to a Configuration Server.

- Reference

  [3] 3.7.4.3, 4.3.2.69, 4.3.2.70, 4.3.2.71, 4.4.2.2

- Initial Condition

  - The IUT supports the Configuration Client Model on the Primary Element.
  - The Lower Tester acts as Configuration Server.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Network Transmit Get message to the Lower Tester.
  2. The Lower Tester expects to receive a Config Network Transmit Get message from the IUT.
  3. The Lower Tester sends a Config Network Transmit Status message in response to the IUT.
  4. The Upper Tester orders the IUT to send a Config Network Transmit Set message to the Lower Tester with random values for the RelayRetransmitCount and RelayRetransmitIntervalSteps fields.
  5. The Lower Tester expects to receive a Config Network Transmit Set message from the IUT with the values requested in Step 4.
  6. The Lower Tester sends a Config Network Transmit Status message in response to the IUT with the values received in Step 5.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Network Transmit Get message to the Lower Tester.

  In Step 5, the IUT sends a Config Network Transmit Set message to the Lower Tester with the requested values.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.16.17 Low Power Node PollTimeout Procedures

This section validates the Low Power Node PollTimeout procedures for both the Configuration Server Model and the Configuration Client Model. Additional valid behavior tests for the Configuration Server Model are covered by MESH/NODE/FRND/FN/BV-21-C [Friend Node Respond to Config Low Power Node PollTimeout Get].

MESH/NODE/CFG/LPNPT/BI-01-C [Ignore Invalid Config Low Power Node PollTimeout Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Server Model ignores invalid Config Low Power Node PollTimeout Get messages.

- Reference

  [3] 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17

- Initial Condition

  - The IUT supports the Configuration Server Model on the Primary Element and the Friend feature.
  - The Lower Tester acts as Configuration Client.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

  1. The Lower Tester sends a Config Low Power Node PollTimeout Get message to the IUT with the LPNAddress field set to the unassigned address.
  2. The Lower Tester expects no response from the IUT.
  3. The Lower Tester repeats Steps 1–2 two more times, setting the LPNAddress field to a group address and then to a virtual address.

- Expected Outcome

  Pass verdict

  In all iterations through Step 2, the IUT does not respond to the invalid messages.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CFGCL/CFG/LPNPT/BV-01-C [Send Config Low Power Node PollTimeout Get Messages]

- Test Purpose

  Verify that an IUT node supporting the Configuration Client Model can send Config Low Power Node PollTimeout Get messages to a Configuration Server.

- Reference

  [3] 4.3.2.67, 4.3.2.68, 4.4.1.2.17, 4.4.1.2.18, 4.4.2.2.17

- Initial Condition

  - The IUT supports the Configuration Client Model on the Primary Element.

- The Lower Tester acts as Configuration Server.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

  1. The Upper Tester orders the IUT to send a Config Low Power Node PollTimeout Get message to the Lower Tester, for a given LPN unicast address.
  2. The Lower Tester expects to receive a Config Low Power Node PollTimeout Get message from the IUT, with the LPNAddress field set to the LPN unicast address requested in Step 1, and responds with a Config Low Power Node PollTimeout Status message with valid parameters.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Config Low Power Node PollTimeout Get message to the Lower Tester with the requested LPNAddress field value.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.17 Health Model

The test group objective is to verify functionality related to both the Health Server Model and the Health Client Model.

4.17.1 Current Fault State

This section validates the behavior of the Current Fault State and its messages for both the Health Server Model and the Health Client Model.

MESH/SR/HM/CFS/BV-01-C [Send Health Current Status Messages - No Faults]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can publish unsolicited Health Current Status messages periodically when it has no active faults.

- Reference

  [3] 3.7.4.3, 4.3.3.1, 4.3.3.9, 4.4.3.1, 4.4.3.2.1, 4.4.3.2.3

- Initial Condition
  - The IUT's FaultArray contains no records.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and

Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- Test Procedure

1. Execute Steps 1–6 in test MESH/NODE/CFG/MP/BV-01-C [Respond to Config Model Publication Set/Get and Config Model Publication Virtual Address Set Messages]. In Step 5, the Lower Tester used a Publish Period value of 0x48 (8 seconds).
2. The Lower Tester sends a Health Period Set Unacknowledged message to the IUT with the Fast Period Log field set to 0x02.
3. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 8 seconds and no re-transmission.
4. The Lower Tester sends a Config Model Publication Set message to the IUT with the ElementAddress, Model Identifier, and AppKeyIndex fields set to values stored from Step 1, a PublishAddress equal to the address of the Lower Tester, CredentialFlag field set to 0b0, RFU field set to 0b111, Publish TTL field set to a random number between 0x00 and 0x7F, and Publish Period field set to 0x48, the Publish Retransmit Count field set to 0b001, and the Publish Retransmit Interval Steps field set to 0b00001.
5. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 8 seconds with one retransmit 100 ms later.

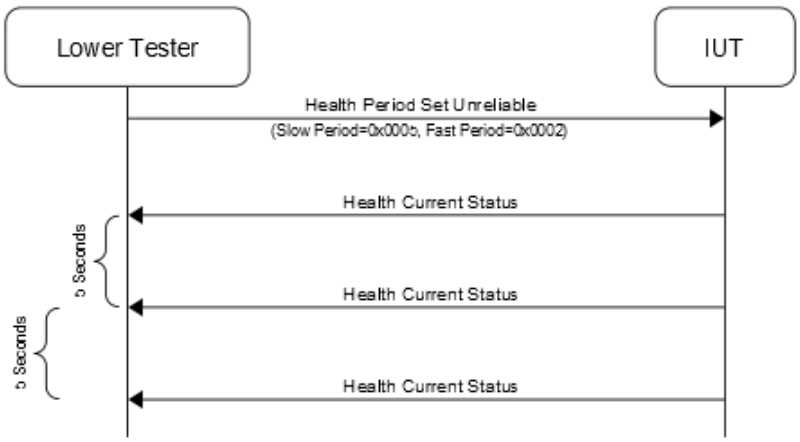


Figure 4.200: MESH/SR/HM/CFS/BV-01-C [Send Health Current Status Messages - No Faults] MSC

- Expected Outcome

Pass verdict

In Step 3, the IUT sends an unsolicited Health Current Status message every 8 seconds with the Company ID field set to the CID field of the Composition Data state, and an empty FaultArray field and without re-transmit.

In Step 5, the IUT sends an unsolicited Health Current Status message every 8 seconds with one retransmit 100 ms later.

All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/SR/HM/CFS/BV-02-C [Send Health Current Status Messages - With Faults]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can publish unsolicited Health Current Status messages periodically when it has active faults.

- Reference

  [3] 3.7.4.3, 4.3.3.1, 4.3.3.9, 4.4.3.1, 4.4.3.2.1, 4.4.3.2.3

- Initial Condition
  - The Lower Tester supports the Health Client Model.
  - The IUT's Publish Period state contains 0x48 (i.e., 8 seconds).
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages]).

- Test Procedure
  1. The Lower Tester sends a Health Period Set Unacknowledged message to the IUT with the Fast Period Log field set to 0x02.
  2. The Upper Tester generates some faults on IUT such that the FaultArray field contains records.
  3. The Lower Tester expects the IUT to send an unsolicited Health Current Status message every 2 seconds.

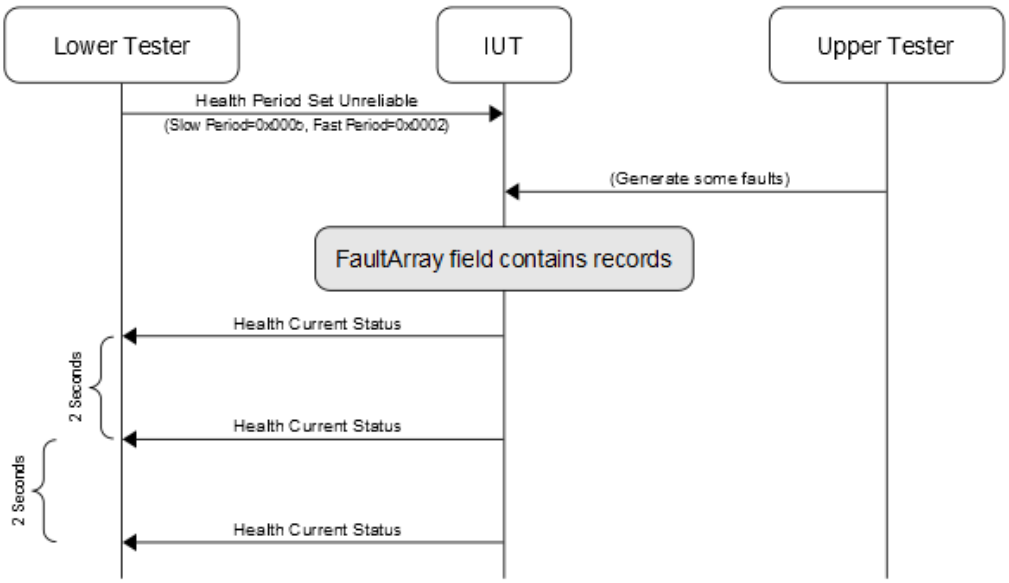


Figure 4.201: MESH/SR/HM/CFS/BV-02-C [Send Health Current Status Messages - With Faults] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends an unsolicited Health Current Status message every 2 seconds with the Company ID field set to the CID field of the Composition Data state, and the FaultArray field set to a sequence of faults representing a sequence of faults in the Current Fault FaultArray.

  All Health messages are encrypted with the shared application key bound to the Health Models.

4.17.2 Registered Fault State

This section validates the behavior of the Registered Fault State and its messages for both the Health Server Model and the Health Client Model.

MESH/SR/HM/RFS/BV-01-C [Respond to Health Fault Get]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can respond to Health Fault Get messages encrypted using the AppKey used in the message received.

- Reference

  [3] 3.7.5.2, 4.3.3.2, 4.3.3.7, 4.4.3.1, 4.4.3.2.2

- Initial Condition

  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.

- The Lower Tester uses the Device Composition Data of the IUT to store the Company ID of the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
- The Lower Tester has subscribed the IUT on the Health Server Model to a random group or virtual address.

- Test Procedure

1. The Lower Tester sends a Health Fault Get message to the IUT, with the Company ID field set to the Company ID declared by the IUT.
2. The Lower Tester expects the IUT to respond with a Health Fault Status message.
3. Repeat Steps 1–2, but in the DST field of the Health Fault Get message, use the address to which the IUT has been subscribed for the Health Server Model.

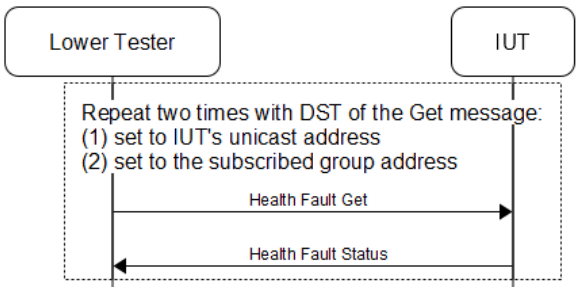


Figure 4.202: MESH/SR/HM/RFS/BV-01-C [Respond to Health Fault Get] MSC

- Expected Outcome

Pass verdict

In Step 2, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to the ID of the most recently performed test and the FaultArray field containing a sequence of faults representing a sequence of faults in the Registered Fault FaultArray. The destination of the Health Fault Status message is the Lower Tester's unicast address.

All Health messages are encrypted with the shared application key bound to the Health Models.

All of the above apply also when the group address is used as DST instead of the unicast address of the IUT.

MESH/SR/HM/RFS/BV-02-C [Respond to Health Fault Clear and Health Fault Clear Unacknowledged Messages]

- Test Purpose

Verify that an IUT node supporting the Health Server Model can respond to Health Fault Clear messages and can clear the Registered Fault state upon receiving Health Fault Clear and Health Fault Clear Unacknowledged messages.

- Reference

[3] 3.7.4.3, 4.3.3.3, 4.3.3.4, 4.3.3.7, 4.4.3.1, 4.4.3.2.2

- Initial Condition
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
- Test Procedure
  1. The Upper Tester generates some faults on the IUT, such that the IUT's Registered Fault state is not empty.
  2. The Lower Tester sends a Health Fault Get message to the IUT using the Company ID of the IUT.
  3. The Lower Tester sends a Health Fault Clear message to the IUT using the Company ID of the IUT.
  4. The Lower Tester expects the IUT to respond with a Health Fault Status message.
  5. Repeat Step 1.
  6. The Lower Tester sends a Health Fault Clear Unacknowledged message to the IUT using the Company ID of the IUT.
  7. The Lower Tester expects the IUT not to respond with any messages.
  8. The Lower Tester sends a Health Fault Get message to the IUT using the Company ID of the IUT.
  9. The Lower Tester expects the IUT to respond with a Health Fault Status message.

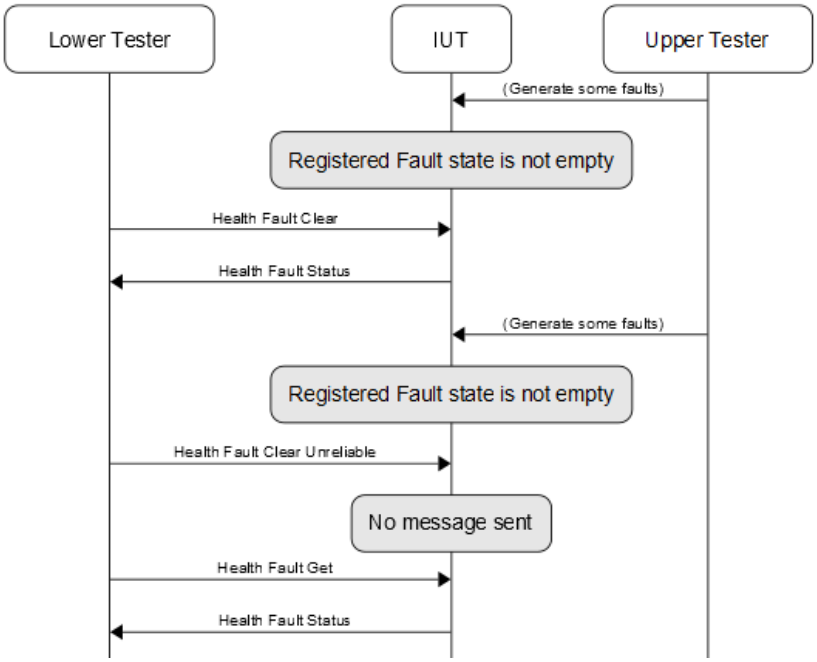


Figure 4.203: MESH/SR/HM/RFS/BV-02-C [Respond to Health Fault Clear and Health Fault Clear Unacknowledged Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state and an empty FaultArray field.

  In Step 7, the IUT does not respond with any messages.

  In Step 9, the IUT sends a Health Fault Status message with an empty FaultArray field.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/SR/HM/RFS/BV-03-C [Respond to Health Fault Test and Health Fault Test Unacknowledged Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can respond to Health Fault Test messages and can execute a self-test upon receiving Health Fault Test and Health Fault Test Unacknowledged messages.

- Reference

  [3] 4.3.3.5, 4.3.3.6, 4.3.3.7, 4.4.3.1, 4.4.3.2.2

- Initial Condition
  - The IUT is a Node and has a primary Element.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- Test Procedure
  1. The Lower Tester sends a Health Fault Test message with the Company ID field set to the value of the CID field of the Composition Data state and Test ID field specified in the IXIT [6].
  2. The Lower Tester expects the IUT to respond with a Health Fault Status message.
  3. The Lower Tester sends a Health Fault Test Unacknowledged message with the Company ID field set to the value of the CID field of the Composition Data state and Test ID field set as specified in the IXIT [6].

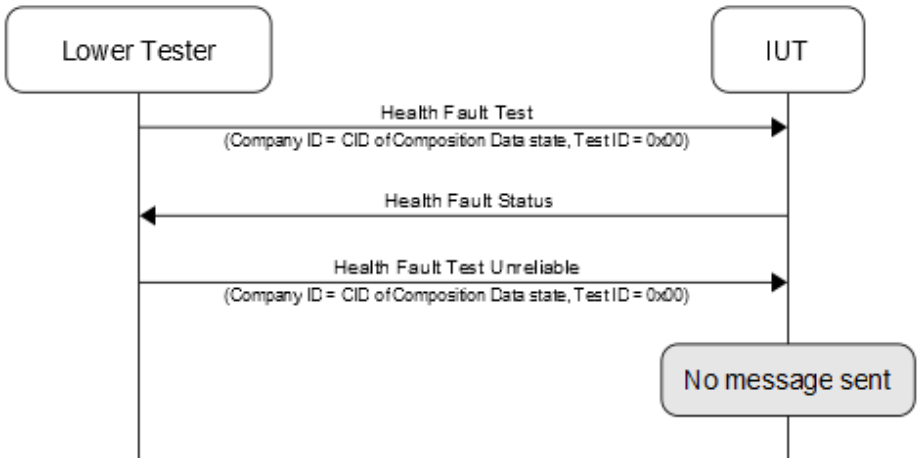


Figure 4.204: MESH/SR/HM/RFS/BV-03-C [Respond to Health Fault Test and Health Fault Test Unacknowledged Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to the ID of the performed test and a FaultArray field containing a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

  In Step 3, the IUT does not respond with any response messages.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/SR/HM/RFS/BI-01-C [Ignore Invalid Health Fault Test message]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model ignores Health Fault Test messages with invalid parameters.

- Reference

  [3] 4.3.3.5, 4.3.3.7, 4.4.1.3, 4.4.3.1, 4.4.3.2.2

- Initial Condition
  - The IUT is a Node and has a primary Element.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- The valid Test IDs the IUT supports are defined in IXIT [6].

- Test Procedure

  1. The Lower Tester sends a Health Fault Test message with the Company ID field set to the value of the CID field of the Composition Data state, but the Test ID doesn't exist.
  2. The Lower Tester expects the IUT not to respond with any message.
  3. The Lower Tester sends a Health Fault Test message with a valid Test ID, but wrong Company ID.
  4. The Lower Tester expects the IUT not to respond with any message.
  5. The Lower Tester sends a Health Fault Test message with Test ID 0x00 (Standard test) and a valid Company ID.
  6. The Lower Tester expects the IUT to respond with a Health Fault Status message.

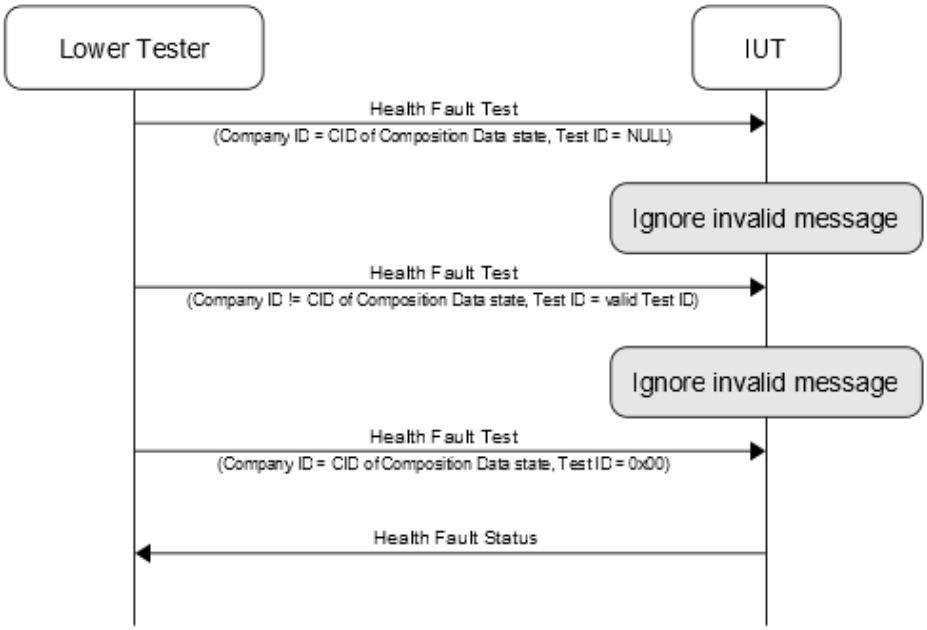


Figure 4.205: MESH/SR/HM/RFS/BI-01-C [Ignore Invalid Health Fault Test message] MSC

- Expected Outcome

  Pass verdict

  In Step 2 and Step 4, the IUT does not send any response messages.

  In Step 6, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/SR/HM/RFS/BI-02-C [Ignore Invalid Health Fault Test Unacknowledged message]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model ignores Health Fault Test Unacknowledged

messages with invalid parameters.

- Reference

  [3] 4.3.3.6, 4.4.3.1, 4.4.3.2.2

- Initial Condition
  - The IUT is a Node and has a primary Element.
  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
  - The valid Test IDs the IUT supports are defined in IXIT [6].
- Test Procedure
  1. The Lower Tester sends a Health Fault Test Unacknowledged message with the Company ID field set to the value of the CID field of the Composition Data state, but the Test ID doesn't exist.
  2. The Lower Tester expects the IUT not to respond with any messages.
  3. The Lower Tester sends a Health Fault Test Unacknowledged message with a valid Test ID, but wrong Company ID.
  4. The Lower Tester expects the IUT not to respond with any messages.
  5. The Lower Tester sends a Health Fault Test message with a valid Test ID 0x00 (Standard test) and a valid Company ID.
  6. The Lower Tester expects the IUT to respond with a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

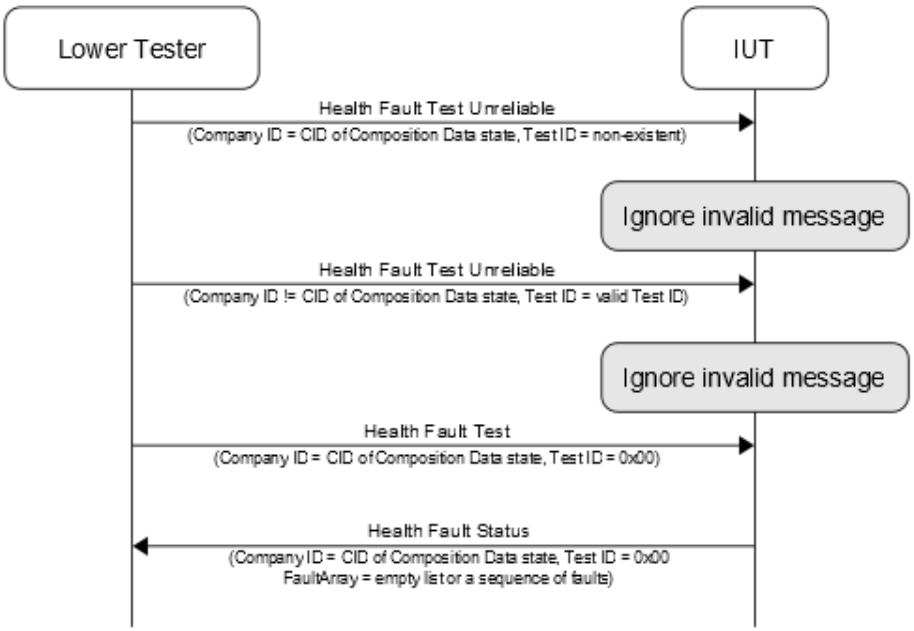


Figure 4.206: MESH/SR/HM/RFS/BI-02-C [Ignore Invalid Health Fault Test Unacknowledged message] MSC

- Expected Outcome

  Pass verdict

  In Step 2 and Step 4, the IUT does not send any response messages.

  In Step 6, the IUT sends a Health Fault Status message with the Company ID field set to the value of the CID field of the Composition Data state, the Test ID field set to 0x00 and a FaultArray field containing an empty list or a sequence of faults representing a sequence of faults in the Registered Fault FaultArray.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/CL/HM/RFS/BV-01-C [Send Health Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Client Model can send Health Fault Get, Health Fault Clear, Health Fault Clear Unacknowledged, Health Fault Test, and Health Fault Test Unacknowledged messages to a Health Server.

- Reference

  [3] 4.3.3.2-4.3.3.6, 4.4.4.1, 4.4.4.2.2

- Initial Condition

  - The Lower Tester supports the Health Server Model.
  - The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.

- The Upper Tester indicates a valid Company ID to the IUT when ordering it to send a message.

- Test Procedure

1. The Upper Tester orders the IUT to send a Health Fault Get message to the Lower Tester.
2. The Lower Tester sends a Health Fault Status message in response to the IUT.
3. The Upper Tester orders the IUT to send a Health Fault Clear message to the Lower Tester.
4. The Lower Tester sends a Health Fault Status message in response to the IUT.
5. The Upper Tester orders the IUT to send a Health Fault Clear Unacknowledged message to the Lower Tester.
6. The Lower Tester does not respond to the Health Fault Clear Unacknowledged message.
7. The Upper Tester orders the IUT to send a Health Fault Test message to the Lower Tester with the Test ID field set to 0x00, and the Company ID field set to the Bluetooth-assigned Company Identifier.
8. The Lower Tester sends a Health Fault Status message in response to the IUT.
9. The Upper Tester orders the IUT to send a Health Fault Test Unacknowledged message to the Lower Tester with the Test ID field set to the 0x00, and the Company ID field set to the Bluetooth-assigned Company Identifier.
10. The Lower Tester does not respond to the Health Fault Test Unacknowledged message.

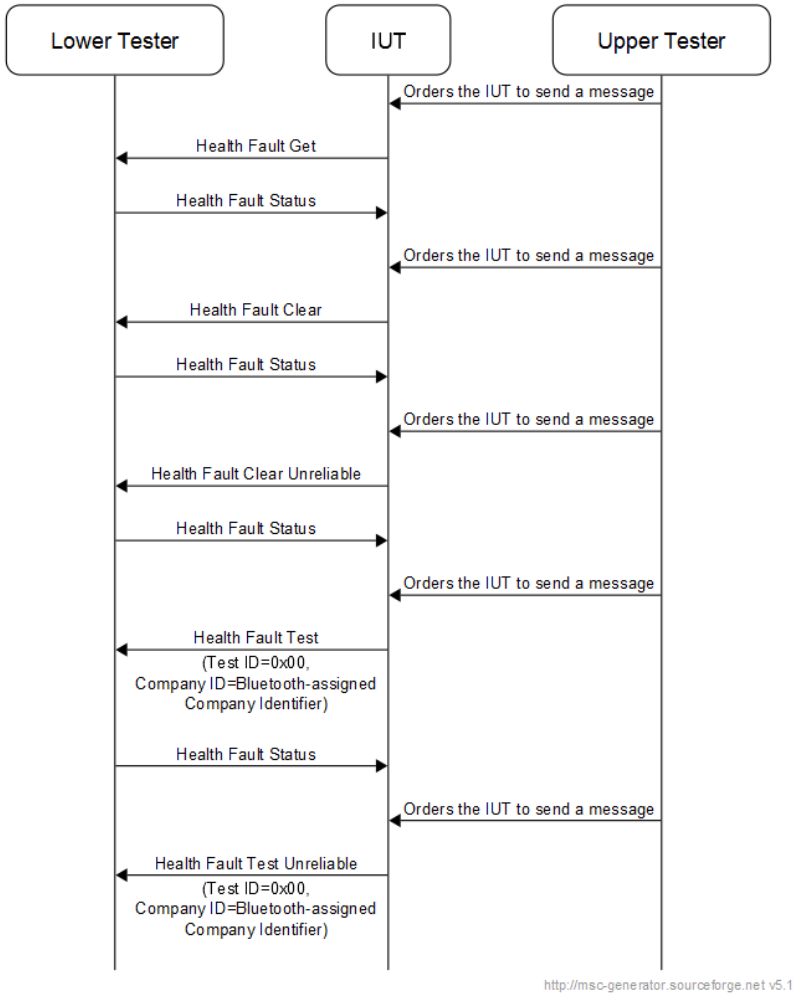


Figure 4.207: MESH/CL/HM/RFS/BV-01-C [Send Health Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 1, the IUT sends a properly formatted Health Fault Get message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

  In Step 3, the IUT sends a properly formatted Health Fault Clear message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

  In Step 5, the IUT sends a properly formatted Health Fault Clear Unacknowledged message with the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

  In Step 7, the IUT sends a properly formatted Health Fault Test message with the Test ID field set to 0x00 and the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

  In Step 9, the IUT sends a properly formatted Health Fault Test Unacknowledged message with the Test ID field set to 0x00 and the Company ID field set to the 16-bit Bluetooth-assigned Company Identifier.

  All Health messages are encrypted with the shared application key bound to the Health Models.

4.17.3 Health Period State

This section validates the behavior of the Health Period State and its messages for both the Health Server Model and the Health Client Model.

MESH/SR/HM/HPS/BV-01-C [Respond to Health Period Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can respond to Health Period Get and Health Period Set messages and can process but not respond to Health Period Set Unacknowledged messages.

- Reference

  [3] 4.3.3.8-4.3.3.11, 4.4.3.1, 4.4.3.2.3

- Initial Condition

  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- Test Procedure

  1. The Lower Tester sends a Health Period Set message with the Fast Period Log field set to 0x02.
  2. The Lower Tester expects the IUT to respond with a Health Period Status message with the Fast Period Log field set to 0x02.

3. The Lower Tester sends a Health Period Set Unacknowledged message with the Fast Period Log field set to 0x03.
4. The Lower Tester expects no response from the IUT.
5. The Lower Tester sends a Health Period Get message to the IUT.
6. The Lower Tester expects the IUT to respond with a Health Period Status message with the Fast Period Log field set to 0x03.

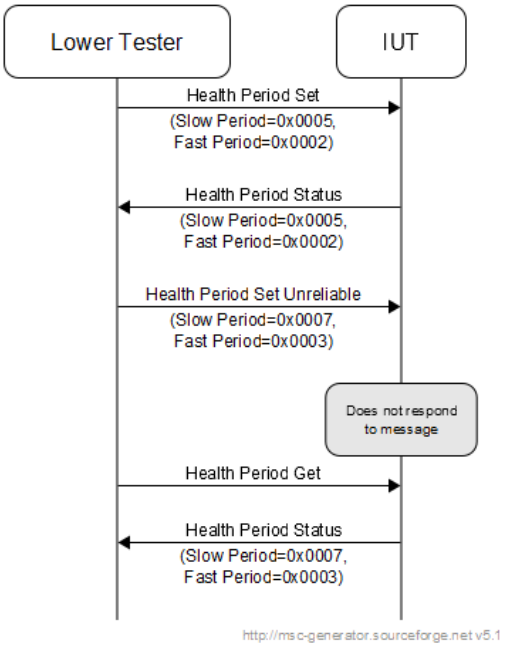


Figure 4.208: MESH/SR/HM/HPS/BV-01-C [Respond to Health Period Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a response Health Period Status message with the Fast Period Log field set to 0x02.

  In Step 4, the IUT does not send any response messages.

  In Step 6, the IUT sends a response Health Period Status message with the Fast Period Log field set to 0x03.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/CL/HM/HPS/BV-01-C [Send Health Period Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Client Model can send Health Period Get, Health Period Set and Health Period Set Unacknowledged messages to a Health Server.

- Reference

[3] 4.3.3.8, 4.3.3.9, 4.3.3.10, 4.3.3.11, 4.4.4.1, 4.4.4.2.3

- Initial Condition
  - The Lower Tester supports the Health Server Model.
  - The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
  - The IUT uses the Device Composition Data of the Lower Tester to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).
- Test Procedure
  1. The Upper Tester orders the IUT to send a Health Period Get message to the Lower Tester.
  2. The Lower Tester responds with a Health Period Status message.
  3. The Upper Tester orders the IUT to send a Health Period Set message to the Lower Tester.
  4. The Lower Tester responds with a Health Period Status message.
  5. The Upper Tester orders the IUT to send a Health Period Set Unacknowledged message to the Lower Tester.

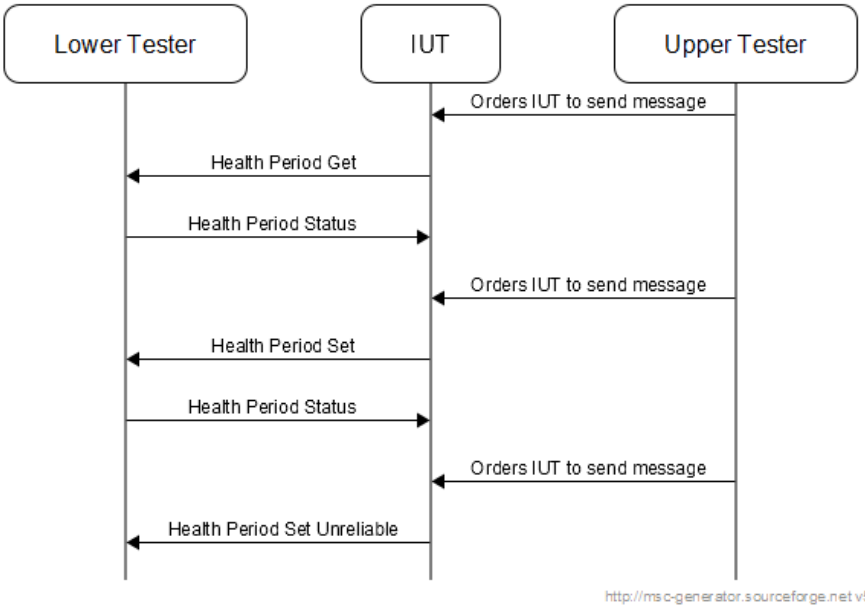


Figure 4.209: MESH/CL/HM/HPS/BV-01-C [Send Health Period Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 1, the IUT sends a properly formatted Health Period Get message.

  In Step 3, the IUT sends a properly formatted Health Period Set message with the Fast Period Log field set to a 1-octet value.

In Step 5, the IUT sends a properly formatted Health Period Set Unacknowledged message with the Fast Period Log field set to a 1-octet value.

All Health messages are encrypted with the shared application key bound to the Health Models.

4.17.4 Attention Timer State

This section validates the behavior of the Attention Timer State and its messages for both the Health Server Model and the Health Client Model.

MESH/SR/HM/ATS/BV-01-C [Respond to Attention Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Server Model can respond to Health Attention Get and Health Attention Set messages and can process but not respond to Health Attention Set Unacknowledged messages.

- Reference

  [3] 4.2.9, 4.3.3.12-4.3.3.15, 4.4.3.1, 4.4.3.2.4

- Initial Condition

  - The Lower Tester supports the Health Client Model.
  - The IUT has been provisioned and the IUT and the Lower Testers share previously established network security credentials and an application key bound to the Health Models.
  - The Lower Tester uses the Device Composition Data of the IUT to store the ElementAddress and Company ID on the IUT (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- Test Procedure

  1. The Lower Tester sends a Health Attention Set message with the Attention field set to 0x05.
  2. The Lower Tester expects the IUT to respond with a Health Attention Status message with the Attention value less than or equal to the value received in Step 1.
  3. The Lower Tester sends a Health Attention Set Unacknowledged message with the Attention field set to 0x07.
  4. The Lower Tester expects the IUT not to respond with a Health Attention Status message.
  5. The Lower Tester sends a Health Attention Get message to the IUT.
  6. The Lower Tester expects the IUT to respond with a Health Attention Status message with the Attention field equal to or less than the value sent in Step 3.

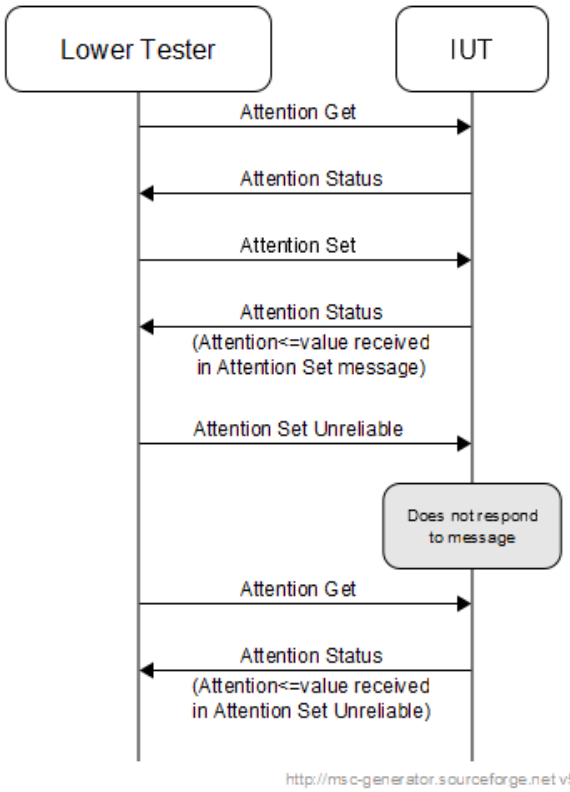


Figure 4.210: MESH/SR/HM/ATS/BV-01-C [Respond to Attention Messages] MSC

- Expected Outcome

  Pass verdict

  In Step 2, the IUT responds with a Health Attention Status message with the Attention field set to 0x05.

  In Step 4, the IUT does not respond with any message.

  In Step 6, the IUT responds with a Health Attention Status message from the IUT with the Attention field set to 0x07.

  All Health messages are encrypted with the shared application key bound to the Health Models.

MESH/CL/HM/ATS/BV-01-C [Send Attention Messages]

- Test Purpose

  Verify that an IUT node supporting the Health Client Model can send Health Attention Get, Health Attention Set, and Health Attention Set Unacknowledged messages to a Health Server.

- Reference

  [3] 4.2.9, 4.3.3.12, 4.3.3.13, 4.3.3.14, 4.3.3.15, 4.4.4.1, 4.4.4.2.3

- Initial Condition

- The Lower Tester supports the Health Server Model.
- The IUT and the Lower Tester share previously established network security credentials and an application key bound to the Health Models.
- The IUT uses the Device Composition Data of the Lower Tester to store the ElementAddress and Company ID on the Lower Tester (e.g., by executing the steps in test case MESH/NODE/CFG/COMP/BV-01-C [Respond to Config Composition Data Get Messages] or by other means).

- Test Procedure

1. The Upper Tester orders the IUT to send a Health Attention Get message to the Lower Tester.
2. The Lower Tester responds with a Health Attention Status message.
3. The Upper Tester orders the IUT to send a Health Attention Set message to the Lower Tester.
4. The Lower Tester responds with a Health Attention Status message with an Attention value less than or equal to the Attention value received in Step 3.
5. The Upper Tester orders the IUT to send a Health Attention Set Unacknowledged message to the Lower Tester.

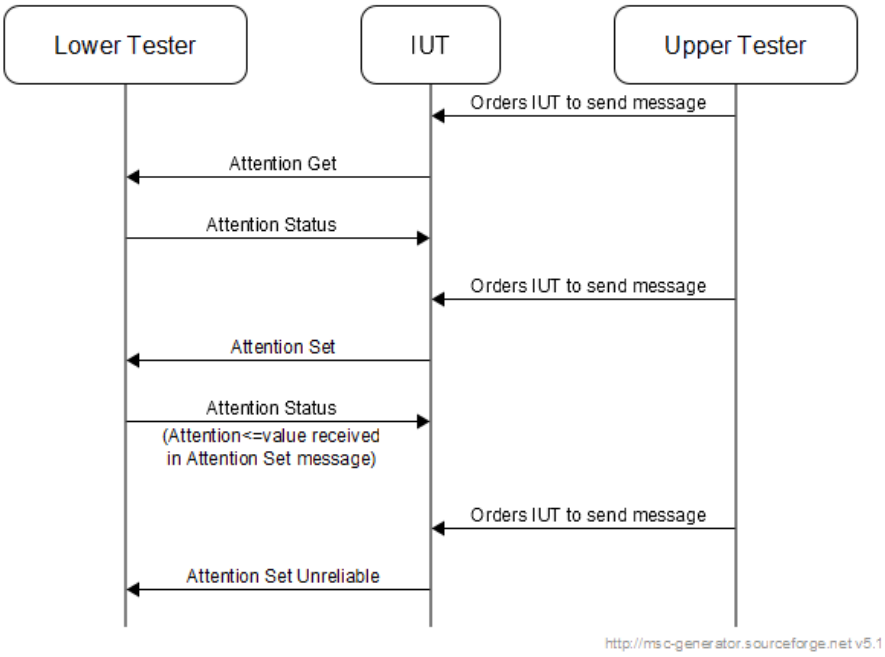


Figure 4.211: MESH/CL/HM/ATS/BV-01-C [Send Attention Messages] MSC

- Expected Outcome

Pass verdict

In Step 1, the IUT sends a properly formatted Health Attention Get message with no parameters.

In Step 3, the IUT sends a properly formatted Health Attention Set message.

In Step 5, the IUT sends a properly formatted Health Attention Set Unacknowledged message.

All Health messages are encrypted with the shared application key bound to the Health Models.

4.18 Directed Forwarding

This section contains tests for the Directed Forwarding functionality including Directed Forwarding Initialization, Directed Forwarding Discovery, Directed Forwarding Establishment, Directed Forwarding Confirmation, Directed Forwarding Dependents Update, Directed Forwarding Echo, Directed Forwarding Solicitation, Forwarding Using Fixed Paths, and Directed Proxy.

4.18.1 Verify Forwarding Table on the primary subnet

- Test Purpose

  Verify that the IUT responds to Forwarding Table procedures with the status messages correctly representing the paths stored in the Forwarding Table on the IUT.

- Reference

  [10] 4.4.5.6.5

- Initial Condition
  - The steps are run by using the values specified by each test case in the corresponding table.
- Test Procedure

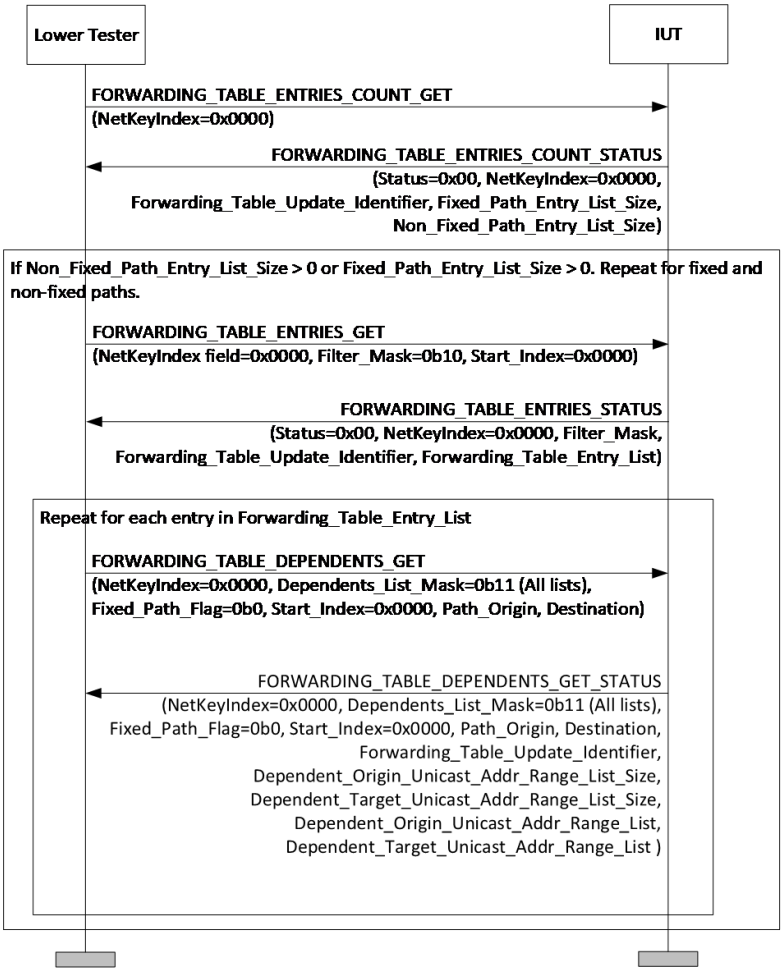


Figure 4.212: Test procedure for Forwarding Table on the primary subnet MSC

1. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
2. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success); NetKeyIndex field set to 0x0000; Forwarding_Table_Update_Identifier set to the specified Forwarding_Table_Update_Identifier value if present, otherwise set to any valid value; Fixed_Path_Entries_Count field set to the specified Fixed_Path_Entries_Count value, otherwise set to 0x0000; and Non_Fixed_Path_Entries_Count field set to the specified Non_Fixed_Path_Entries_Count value, otherwise set to 0x0000.
3. If the Non_Fixed_Path_Entries_Count field is 0, skip Steps 4–8.
4. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET message with NetKeyIndex field set to 0x0000, Filter_Mask set to 0b10 (Non_Fixed_Paths), and Start_Index set to 0x0000.
5. The IUT sends a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to 0b10 (Non_Fixed_Paths), Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to the value from the input table. Forwarding_Table_Entry_List contains Non_Fixed_Path_Entries_Count entries. Each entry *i* in the interval [0, Non_Fixed_Path_Entries_Count) will contain the following fields:

- Fixed_Path_Flag set to 0b0
- Unicast_Destination_Flag set to 0b0 if Multicast_Destination$_i$ field is present, otherwise set to 0b1
- Backward_Path_Validated_Flag set to Backward_Path_Validated$_i$
- Bearer_Toward_Path_Origin_Indicator set to 0b1 if Bearer_Toward_Path_Origin$_i$ is present, otherwise set to 0b0
- Bearer_Toward_Path_Target_Indicator set to 0b1 if Bearer_Toward_Path_Target$_i$ is present, otherwise set to 0b0
- Dependent_Origin_List_Size_Indicator set to 0b01 if Dependent_Origin_List$_i$ is present, otherwise set to 0b00
- Dependent_Target_List_Size_Indicator set to 0b01 if Dependent_Target_List$_i$ is present, otherwise set to 0b00
- Lane_Counter set to Lane_Counter$_i$
- Path_Remaining_Time set to a value near the Path Lifetime value
- Path_Origin_Forwarding_Number set to Path_Origin_Forwarding_Number$_i$
- Path_Origin_Unicast_Addr_Range set to Path_Origin_Unicast_Addr_Range$_i$
- Dependent_Origin_List_Size set to the size of Dependent_Origin_List$_i$ if present, otherwise excluded
- Bearer_Toward_Path_Origin set to Bearer_Toward_Path_Origin$_i$ if present, otherwise excluded
- Path_Target_Unicast_Addr_Range set to Path_Target_Unicast_Addr_Range$_i$ if present, otherwise excluded
- Multicast_Destination set to Multicast_Destination$_i$ if present, otherwise excluded
- Dependent_Target_List_Size set to the size of Dependent_Target_List$_i$ if present, otherwise excluded
- Bearer_Toward_Path_Target set to Bearer_Toward_Path_Target$_i$ if present, otherwise excluded

6. Repeat Steps 7 and 8 for each *i* in the interval [0, Non_Fixed_Path_Entries_Count).
7. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Fixed_Path_Flag set to 0b0, Start_Index set to 0x0000, Path_Origin set to the RangeStart field from Path_Origin_Unicast_Addr_Range$_i$, and Destination set to the RangeStart field from Path_Target_Unicast_Addr_Range$_i$.
8. The IUT sends a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with the following fields:

    - Status set to 0x00 (Success)
    - NetKeyIndex field set to 0x0000
    - Dependents_List_Mask set to 0b11 (All lists)
    - Fixed_Path_Flag set to 0b0
    - Start_Index set to 0x0000
    - Path_Origin set to the RangeStart field from Path_Origin_Unicast_Addr_Range$_i$
    - Destination set to the RangeStart field from Path_Target_Unicast_Addr_Range$_i$
    - Forwarding_Table_Update_Identifier set to the value from the input table
    - Dependent_Origin_Unicast_Addr_Range_List_Size set to Dependent_Origin_List_Size$_i$ if present, otherwise set to 0x00
    - Dependent_Target_Unicast_Addr_Range_List_Size set to Dependent_Target_List_Size$_i$ if

present, otherwise set to 0x00
- Dependent_Origin_Unicast_Addr_Range_List set to Dependent_Origin_List$_i$ if present, otherwise excluded
- Dependent_Target_Unicast_Addr_Range_List set to Dependent_Target_List$_i$ if present, otherwise excluded

9. If Fixed_Path_Entries_Count is 0, skip Steps 10–12.
10. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET message with NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1 (Fixed_Paths), and Start_Index set to 0x0000.
11. The IUT sends a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1 (Fixed_Paths), Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to the value from the input table. Forwarding_Table_Entry_List contains Fixed_Path_Entries_Count entries. Each entry *i* will contain the following fields:

    - Fixed_Path_Flag set to 0b1
    - Unicast_Destination_Flag set to 0b0 if Multicast_Destination$_i$ field is present, 0b1 otherwise
    - Backward_Path_Validated_Flag set to Backward_Path_Validated$_i$
    - Bearer_Toward_Path_Origin_Indicator set to 0b1 if Bearer_Toward_Path_Origin$_i$ is present, otherwise set to 0b0
    - Bearer_Toward_Path_Target_Indicator set to 0b1 if Bearer_Toward_Path_Target$_i$ is present, otherwise set to 0b0
    - Dependent_Origin_List_Size_Indicator set to 0b01 if Dependent_Origin_List_Size$_i$ is greater than 0, otherwise it is set to 0b00
    - Dependent_Target_List_Size_Indicator set to 0b01 if Dependent_Target_List_Size$_i$ is greater than 0, otherwise it is set to 0b00
    - Path_Origin_Unicast_Addr_Range set to Path_Origin_Unicast_Addr_Range$_i$
    - Dependent_Origin_List_Size set to Dependent_Origin_List_Size$_i$ if value is greater than 0, otherwise not present
    - Bearer_Toward_Path_Origin set to Bearer_Toward_Path_Origin$_i$ if present
    - Path_Target_Unicast_Addr_Range set to Path_Target_Unicast_Addr_Range$_i$, if present
    - Multicast_Destination set to Multicast_Destination$_i$, if present
    - Dependent_Target_List_Size set to Dependent_Target_List_Size$_i$ if value is greater than 0, otherwise not present
    - Bearer_Toward_Path_Target set to Bearer_Toward_Path_Target$_i$ if present

12. Repeat Steps 7 and 8 for each *i* in the interval [0, Non_Fixed_Path_Entries_Count) with Fixed_Path_Flag set to 0b1.

- Expected Outcome

  Pass verdict

  The IUT responds to each message from the Lower Tester with the expected fields. The messages are encrypted with the device key.

4.18.2 Directed Forwarding Initialization

This subsection verifies the Directed Forwarding Initialization procedure.

MESH/NODE/DF/INIT/BV-01-C [Send PATH_REQUEST, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT can send a PATH_REQUEST message to establish a path toward a unicast Path Target and that the IUT allocates a separate forwarding number value space for each supported subnet.

- Reference

  [10] 3.6.8.2.1

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - If supported, the IUT is configured on a second subnet with NetKeyIndex of 0x0001.
  - The IUT did not send any directed forwarding control messages.
  - The Max Concurrent Init state is configured to 0x01 on the IUT for all configured subnets.

- Test Procedure

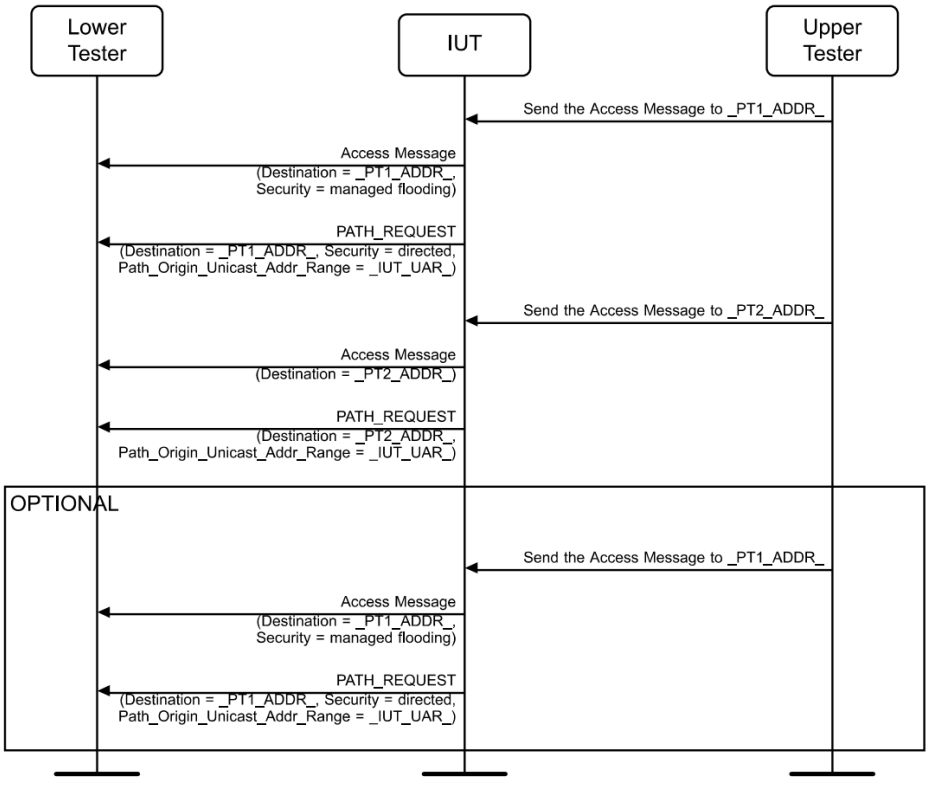


Figure 4.213: MESH/NODE/DF/INIT/BV-01-C MSC

1. The Upper Tester commands the IUT to send an access message to _PT1_ADDR_ on the primary subnet.
2. The IUT sends the access message to _PT1_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT1_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
3. The Upper Tester commands the IUT to send an access message to _PT2_ADDR_.
4. The IUT sends the access message to _PT2_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with Destination set to _PT2_ADDR_, Path_Origin_Forwarding_Number set to 0x01, and the other fields as in Step 2.
5. If the IUT is configured on the second subnet, repeat Steps 1 and 2 on the second subnet.

- Expected Outcome

  Pass verdict

  If expected, the IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet in Step 2 and on the secondary subnet (if supported) in Step 3.

  The IUT sends the access message secured with managed flooding security credentials on the primary subnet in the first iteration through Step 2 and on the secondary subnet (if supported) in the second iteration through Step 2 (as described in Step 5).

4.18.2.1 Send PATH_REQUEST for Dependent, IUT as Path Origin

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node can send a PATH_REQUEST message to establish a path toward a Path Target on behalf of a dependent node.

- Reference

  [10] 3.6.8.2.1

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.43 apply.
  - Lower Tester 2 simulates two elements.

- Test Procedure

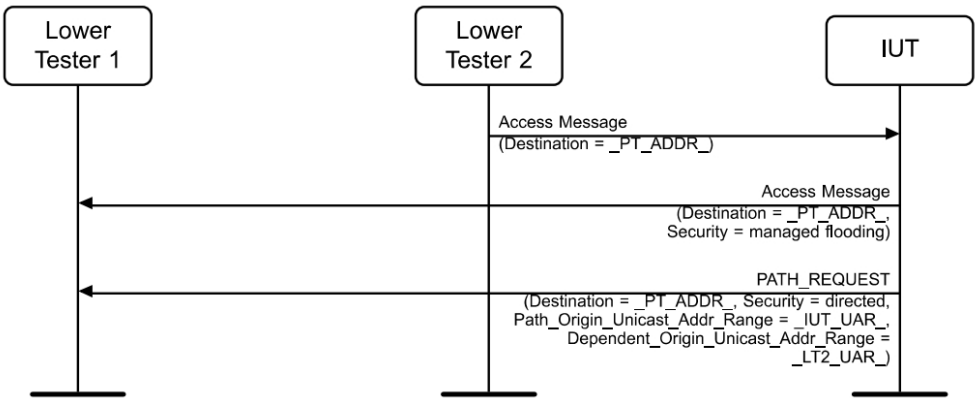


Figure 4.214: Send PATH_REQUEST for Dependent, IUT as Path Origin MSC

1. Lower Tester 2 sends an access message with SRC set to _LT2_ADDR_ and DST set to _PT_ADDR_ secured as specified in Table 4.43.
2. The IUT sends the access message to _PT_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b1, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, and Dependent_Origin_Unicast_Addr_Range set to the value in Table 4.43.

- Test Case Configuration

TCID	Additional Initial Condition	Dependent Origin Unicast Address Range
MESH/NODE/DF/INIT/BV-02-C [Send PATH_REQUEST, IUT as Directed Friend Path Origin]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [10] Section 3.8.5.3.1. Lower Tester 2 uses only friendship security credentials to send access messages.	_LT2_UAR_

TCID	Additional Initial Condition	Dependent Origin Unicast Address Range
MESH/NODE/DF/INIT/BV-03-C [Send PATH_REQUEST, IUT as Path Origin for Proxy Client]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 uses only managed flooding security credentials to send access messages.	_LT2_UAR_
MESH/NODE/DF/INIT/BV-04-C [Send PATH_REQUEST, IUT as Path Origin for Directed Proxy Client]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	_LT2_UAR_
MESH/NODE/DF/INIT/BV-11-C [Send PATH_REQUEST, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). Lower Tester 1 is only part of the first subnet (index 0), and Lower Tester 2 is only part of the second subnet (index 1). Directed forwarding is enabled on the IUT on the first subnet (index 0). The IUT is configured with the bridging table entry (0x01, 10, 01, _LT2_SEC_ADDR_, _PT_ADDR_) by running preamble 4.2.7.	_LT2_SEC_ADDR_

Table 4.43: Additional test conditions for Send PATH_REQUEST for Dependent, IUT as Path Origin

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT sends the access message secured with managed flooding security credentials on the primary subnet.

MESH/NODE/DF/INIT/BV-05-C [Multiple Lanes, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT sets up multiple lanes as a path origin.

- Reference

  [10] 3.6.8.2.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured on the primary subnet with the Wanted Lanes state set to 0x02, the Path Discovery Interval state set to 0b0 (5 seconds), and Lane Discovery Guard state set to 0b1 (10 seconds).
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.

- Test Procedure

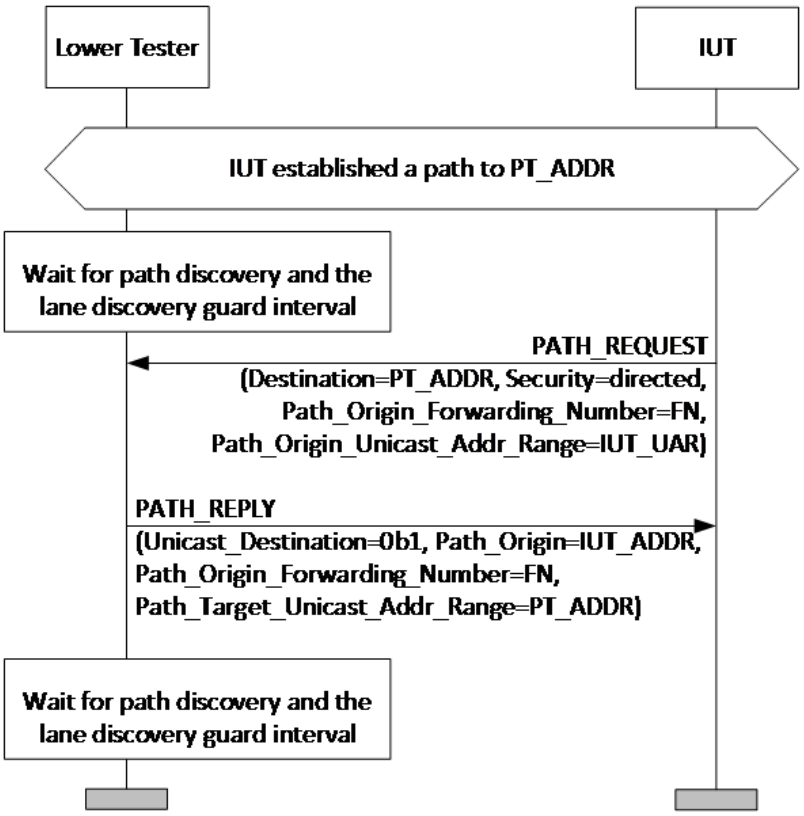


Figure 4.215: MESH/NODE/DF/INIT/BV-05-C MSC

1. After the path discovery and the lane discovery guard interval, the IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b0, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
2. Lower Tester 2 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to

_IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.

3. After the path discovery and the lane discovery guard interval, the IUT does not send any PATH_REQUEST messages.

- Expected Outcome

  Pass verdict

  In Step 1, the IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0, only after the Path_Discovery_Interval timeout has expired. The message is secured with directed security credentials on the primary subnet.

  In Step 3, the IUT does not send a PATH_REQUEST message.

  The IUT reports a Forwarding Table entry for the path to _PT_ADDR_ with the expected fields.

MESH/NODE/DF/INIT/BV-06-C [Path Discovery Retry Interval Timeout, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT controls when the Directed Path Initialization is executed again after the procedure fails.

- Reference

  [10] 3.6.8.2.1, 4.4.7.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Path Discovery Retry Interval state is set to 20 (20 seconds).
- Test Procedure

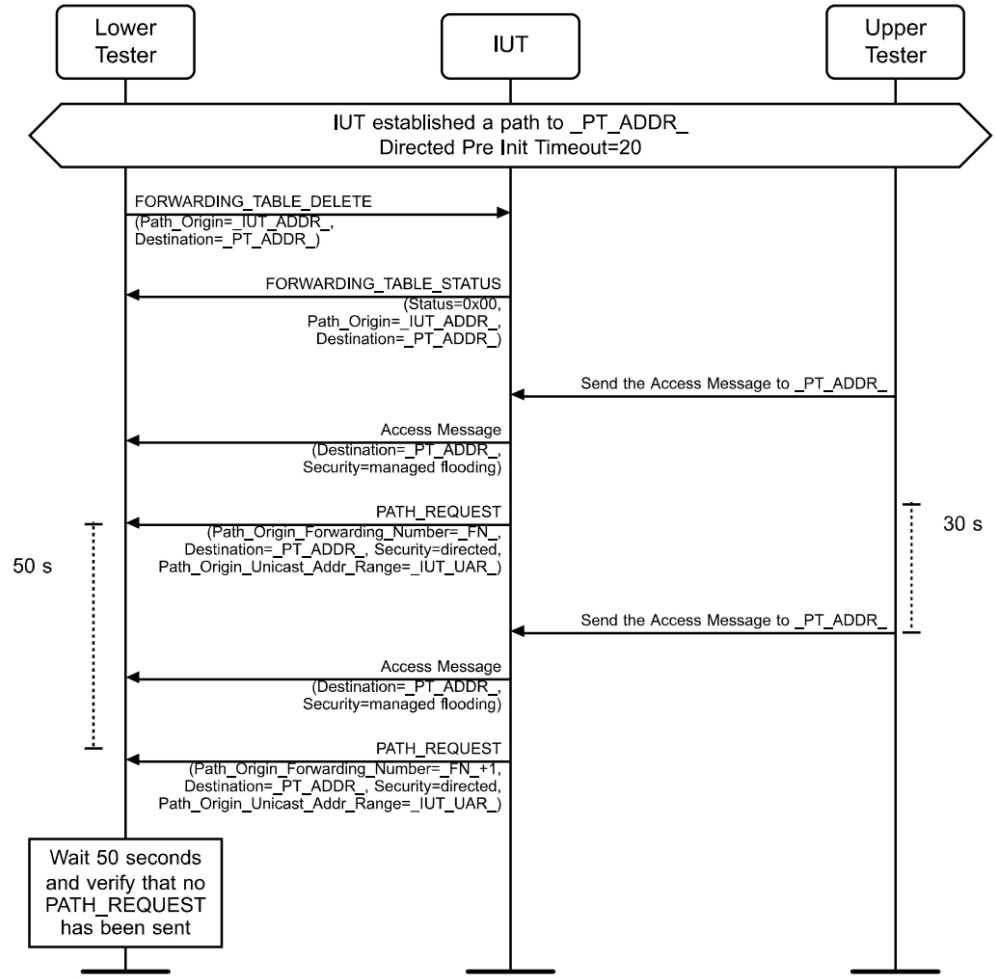


Figure 4.216: MESH/NODE/DF/INIT/BV-06-C MSC

1. The Upper Tester commands the IUT to send an access message to _PT_ADDR_.
2. The IUT sends the access message to _PT_ADDR_.
3. The IUT also immediately sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
4. After 30 seconds since the PATH_REQUEST message from Step 3, but before the Path Discovery Retry Interval expires, the Upper Tester commands the IUT to send an access message to _PT_ADDR_.
5. The IUT sends the access message to _PT_ADDR_ but does not immediately send any PATH_REQUEST messages (not until 50 seconds have passed since the PATH_REQUEST message from Step 3).
6. After 50 seconds since the PATH_REQUEST message from Step 3, the IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1,

Path_Origin_Forwarding_Number set to _FN_ + 1, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.

7. The Lower Tester waits for 50 seconds and expects no PATH_REQUEST message from the IUT.

- Expected Outcome

  Pass verdict

  In Steps 3 and 6, the IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The time interval between the two PATH_REQUEST messages (from Step 3 and Step 6) is 50 seconds (i.e., equal to the Path Discovery Interval plus the Path Discovery Retry Interval).

  In Steps 5 and 7, the IUT does not send a PATH_REQUEST message.

  The IUT sends the access message secured with managed flooding security credentials.

MESH/NODE/DF/INIT/BV-07-C [Path Monitoring Interval Timeout on Path Expiry, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT controls when the Directed Path Initialization is executed again before the path expiry.

- Reference

  [10] 3.6.8.2.1, 4.4.7.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Path Monitoring Interval state is set to 10 (10 seconds).
  - Path Monitoring test mode is activated on the IUT (via HW or SW interface).
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9.
- Test Procedure

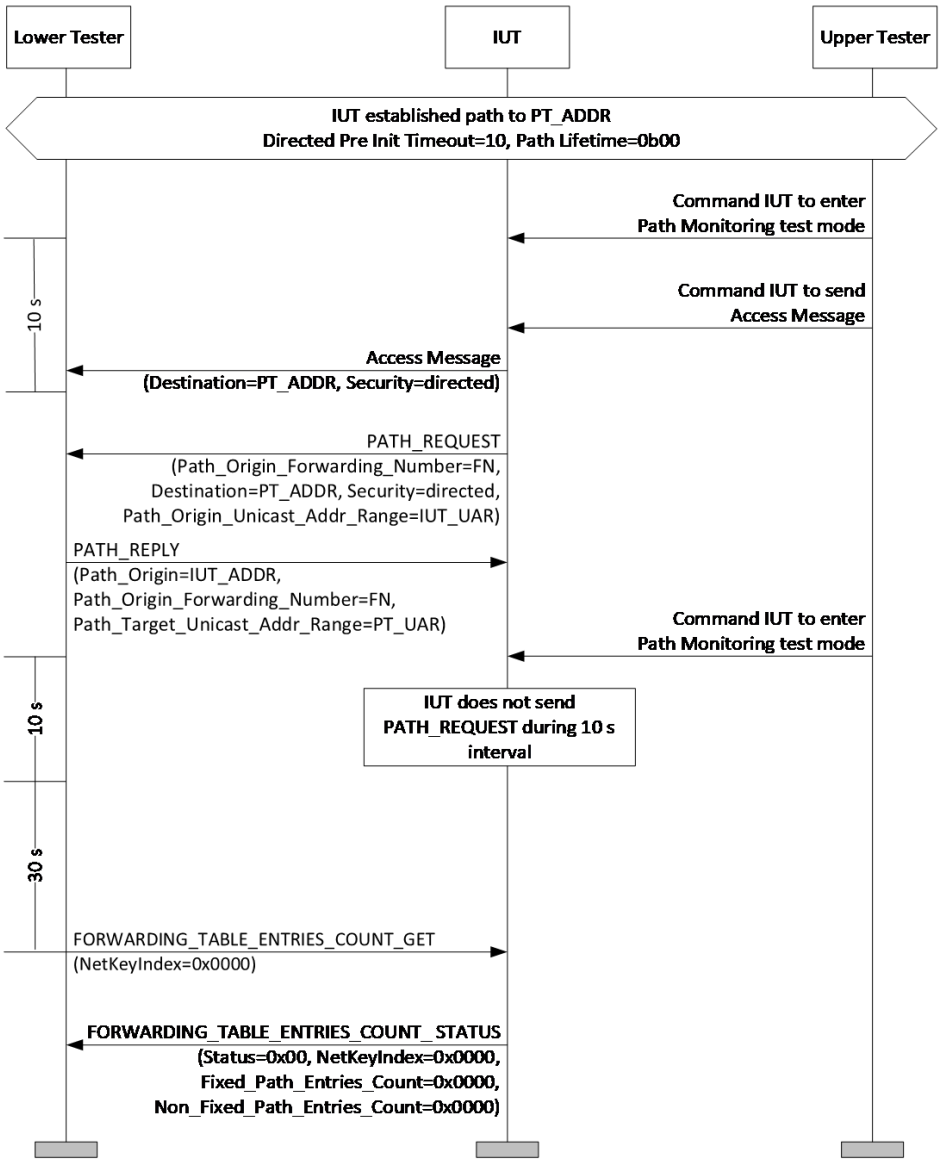


Figure 4.217: Test Procedure for MESH/NODE/DF/INIT/BV-07-C MSC

1. The Upper Tester commands the IUT to enter Path Monitoring test mode.
2. Within 10 seconds after entering Path Monitoring test mode in Step 1, the Upper Tester commands the IUT to send the access message to _PT_ADDR_ secured with directed material.
3. Ten seconds after entering Path Monitoring test mode in Step 1, the IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
4. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and

Path_Target_Unicast_Addr_Range set to _PT_UAR_.

5. The Upper Tester commands the IUT to enter Path Monitoring test mode.
6. Ten seconds after entering Path Monitoring test mode in Step 5, the IUT does not send any PATH_REQUEST messages.
7. Thirty seconds after Step 6, the Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
8. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000.

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet. The message is sent after 10 seconds of Step 1.

  In Step 6, the IUT does not send a PATH_REQUEST message.

  The IUT sends the access message secured with directed security credentials.

MESH/NODE/DF/INIT/BV-08-C [Path Monitoring Interval Timeout on Power-Up, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT controls when the Directed Path Initialization is executed on power-up.

- Reference

  [10] 3.6.8.2.1, 4.4.7.2

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured with the default values for the Directed Publish Policy state.
  - The IUT is configured for model publication to _PT_ADDR_ on the ModelID provided in the IXIT [6].

- Test Procedure

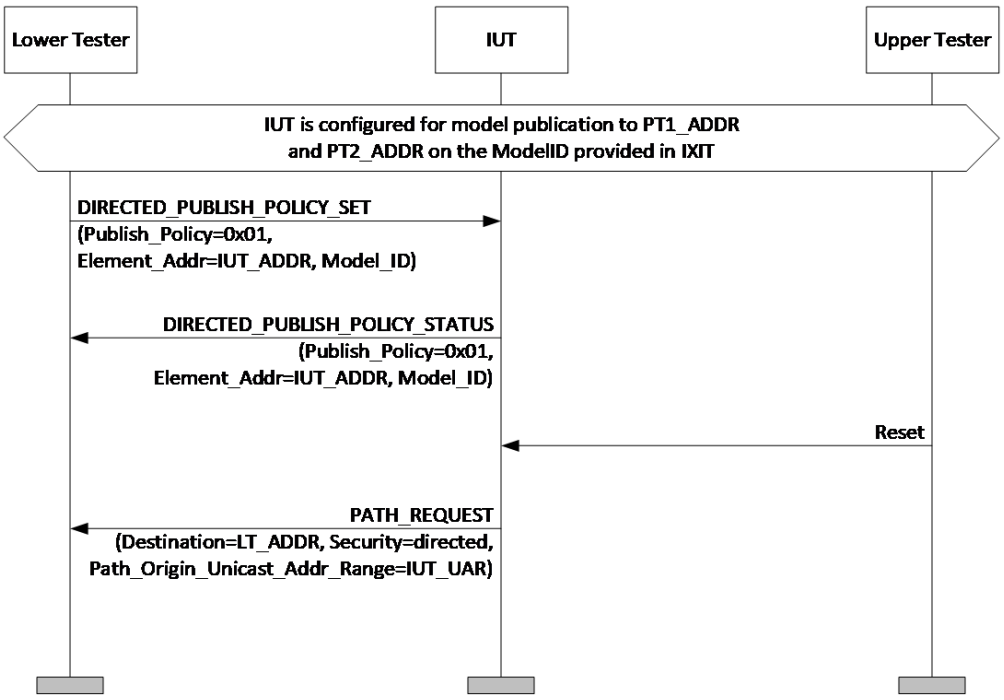


Figure 4.218: MESH/NODE/DF/INIT/BV-08-C MSC

1. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_SET message with Directed_Publish_Policy set to 0x01(Directed Forwarding), Element_Addr set to _IUT_ADDR_, and Model_ID set to the ModelID value from the IXIT [6].
2. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x00 (Success), Directed_Publish_Policy set to 0x01 (Directed Forwarding), Element_Addr set to _IUT_ADDR_, and Model_ID set to ModelID.
3. The Upper Tester orders the IUT to reset itself.
4. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
5. Repeat Steps 1 and 2 with Directed_Publish_Policy set to 0x00 (Managed Flooding).
6. The Upper Tester orders the IUT to reset itself.
7. The IUT publishes an access message to _PT_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT sends the PATH_REQUEST messages with the expected fields and TTL set to 0. The messages are secured with directed security credentials on the primary subnet.

  In Step 7, the IUT sends the access messages secured with managed flooding security credentials.

MESH/NODE/DF/INIT/BV-09-C [Transmit Access Message for Existing Non-fixed Path Without PATH_REQUEST]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node does not send a PATH_REQUEST message after transmitting an access message if there is an existing Forwarding Table entry for a non-fixed path.

- Reference

  [10] 3.7.4.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.

- Test Procedure

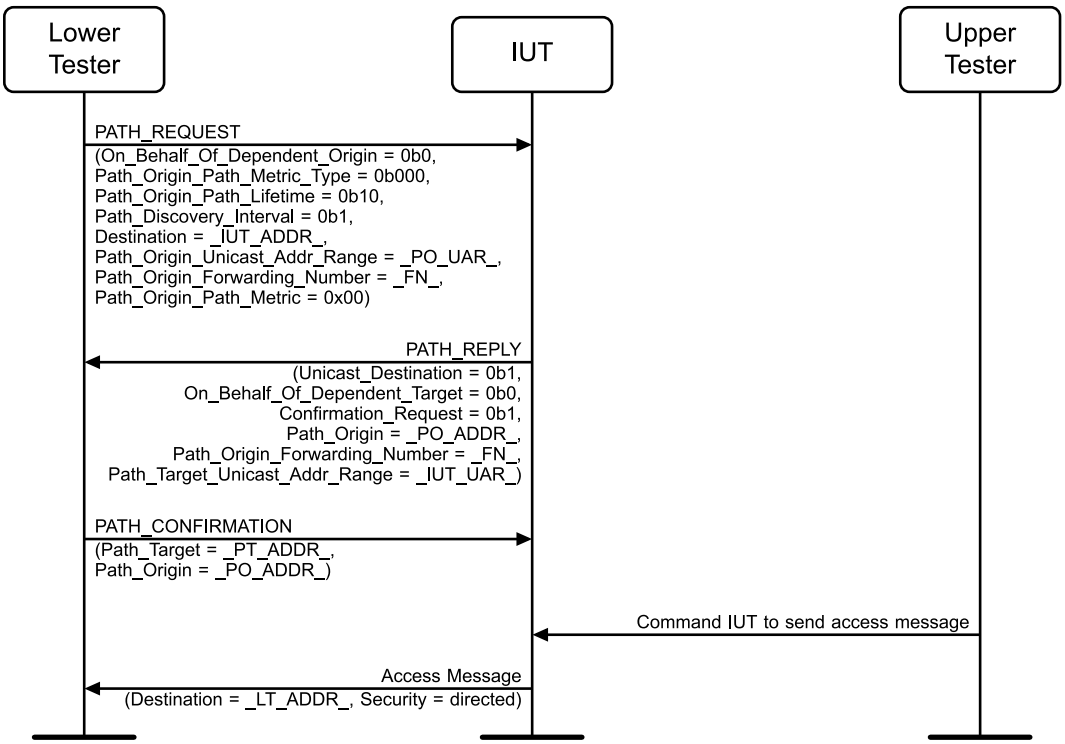


Figure 4.219: MESH/NODE/DF/INIT/BV-09-C MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Destination set to _IUT_ADDR_, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Origin_Path_Metric set to 0x00.
2. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b1, Path_Origin set to

_PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _IUT_UAR_.

3. The Lower Tester sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_.
4. The Upper Tester commands the IUT to send an access message to _PT_ADDR_ on the primary subnet.
5. The IUT sends an access message with valid values for TTL, DST, and Access PDU fields, and secured with directed security material.

- Expected Outcome

  Pass verdict

  The IUT does not send a PATH_REQUEST message after sending the access message to the Lower Tester.

  The IUT sends the access message secured with directed security credentials.

MESH/NODE/DF/INIT/BV-10-C [Multiple Lanes, PATH_REQUEST Uses Discovery Table Values]

- Test Purpose

  Verify that a Directed Forwarding node IUT uses values from the Discovery Table in a PATH_REQUEST message.

- Reference

  [10] 3.6.8.2.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured on the primary subnet with the Wanted Lanes state set to 0x02 and Lane Discovery Guard state set to 0b1 (10 seconds).

- Test Procedure

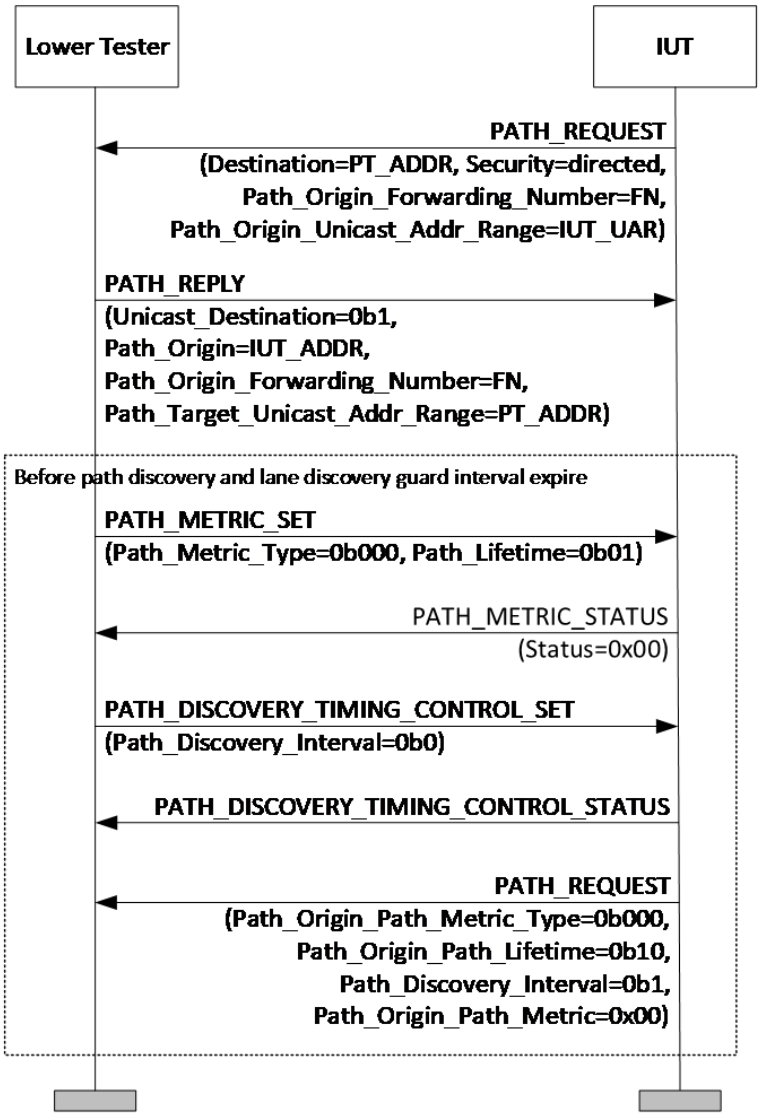


Figure 4.220: MESH/NODE/DF/INIT/BV-10-C MSC

1. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
2. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
3. Before the path discovery interval and lane discovery guard interval expire, the Lower Tester sends a PATH_METRIC_SET message to the IUT with Path_Metric_Type set to 0b000 and Path_Lifetime set to 0b01.

4. The IUT sends a PATH_METRIC_STATUS message to the Lower Tester with Status set to 0x00.
5. The Lower Tester sends a PATH_DISCOVERY_TIMING_CONTROL_SET message to the IUT with Path_Discovery_Interval set to 0b0 and other fields set to valid values.
6. The IUT sends a PATH_DISCOVERY_TIMING_CONTROL_STATUS message to the Lower Tester with the fields equal to values sent in Step 5.
7. After the Path Discovery Interval Timer and the Lane Discovery Guard Interval timer expires, the IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x00.

- Expected Outcome

  Pass verdict

  In Step 7, the IUT sends a PATH_REQUEST message with the Path_Origin_Path_Metric_Type, Path_Origin_Path_Lifetime, and Path_Discovery_Interval fields set to the values sent in Step 1.

4.18.3 Directed Forwarding Discovery

This subsection verifies the Directed Forwarding Discovery procedure.

MESH/NODE/DF/DISC/BV-01-C [Receive PATH_REQUEST, IUT not Path Target]

- Test Purpose

  Verify that a Directed Relay node IUT can forward a received PATH_REQUEST message where the target destination is a unicast address not matching one of its elements.

- Reference

  [10] 3.6.8.2.2

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

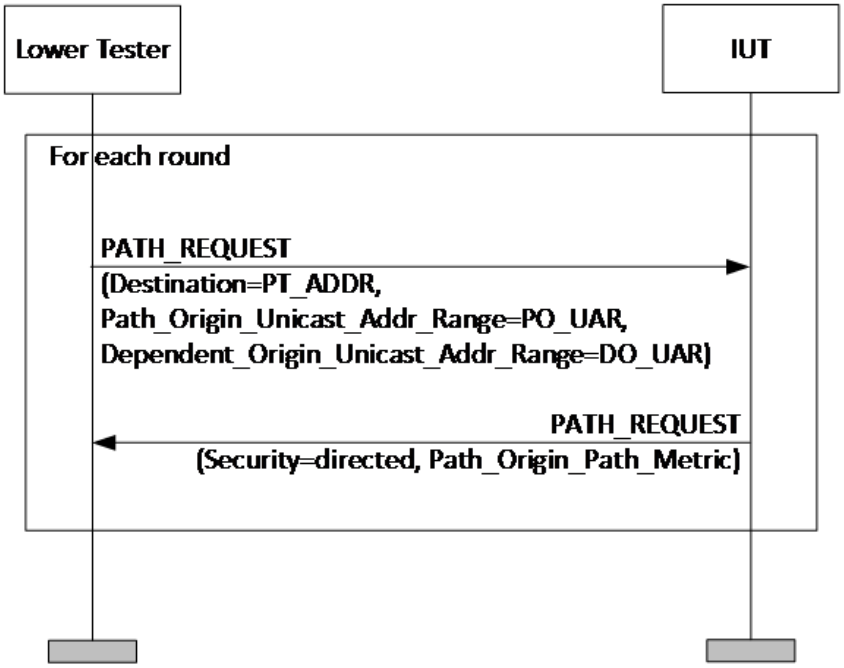


Figure 4.221: MESH/NODE/DF/DISC/BV-01-C MSC

For each round in Table 4.44 below:

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b1, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number and Path_Origin_Path_Metric set to the values from Table 4.44 Destination set to _PT_ADDR_, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, and Dependent_Origin_Unicast_Addr_Range set to _DO_UAR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to the value from Table 4.44 and other fields set to the values from Step 1.

   Table 4.44 specifies the rounds to be carried out. The IUT maintains its state between rounds.

Round	PATH_REQUEST from Lower Tester		PATH_REQUEST from IUT
	Path_Origin_Forwarding_Number	Path_Origin_Path_Metric	Path_Origin_Path_Metric
1	0x01	0x02	0x03
2	0x01	0x01	0x02
3	0x02	0x03	0x04

Table 4.44: Specification of rounds for MESH/NODE/DF/DISC/BV-01-C

- Expected Outcome

Pass verdict

The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0, after the Path_Request_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

- Notes

  All test steps are run within the path discovery interval.

MESH/NODE/DF/DISC/BV-02-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target]

- Test Purpose

  Verify that a Directed Forwarding node IUT receives a PATH_REQUEST message where the target destination is a unicast address matching one of its elements and replies with PATH_REPLY.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

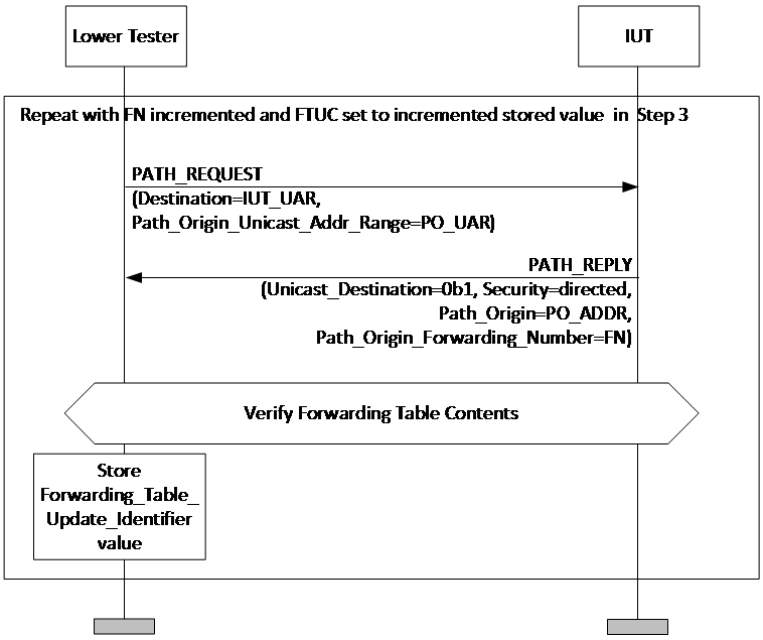


Figure 4.222: MESH/NODE/DF/DISC/BV-02-C MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b1, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1,

Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x00, Destination set to _IUT_ADDR_, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, and Dependent_Origin_Unicast_Addr_Range set to _DO_UAR_.

2. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _IUT_UAR_.
3. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.45 by running the test procedure from Section 4.18.1. The Lower Tester stores the Forwarding_Table_Update_Identifier value.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Dependent_Origin_List0	{_DO_UAR_}
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_IUT_UAR_

Table 4.45: Forwarding Table contents for MESH/NODE/DF/DISC/BV-02-C

4. Repeat Steps 1–3, with _FN_ set to an incremented value and Forwarding_Table_Update_Identifier set to a different value from the one in Step 3.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REPLY message with the expected fields and TTL set to 0, after the Path_Reply_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send any PATH_REQUEST messages.

MESH/NODE/DF/DISC/BV-03-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Path Target]

- Test Purpose

  Verify that an IUT acting as Directed Forwarding or a Directed Relay node receives a PATH_REQUEST message where the destination is a group or virtual address matching one of the addresses it subscribed to, replies with PATH_REPLY and forwards the PATH_REQUEST if the directed relay functionality is enabled.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is subscribed to _GR_ADDR_ and _VR_ADDR_.
- Test Procedure

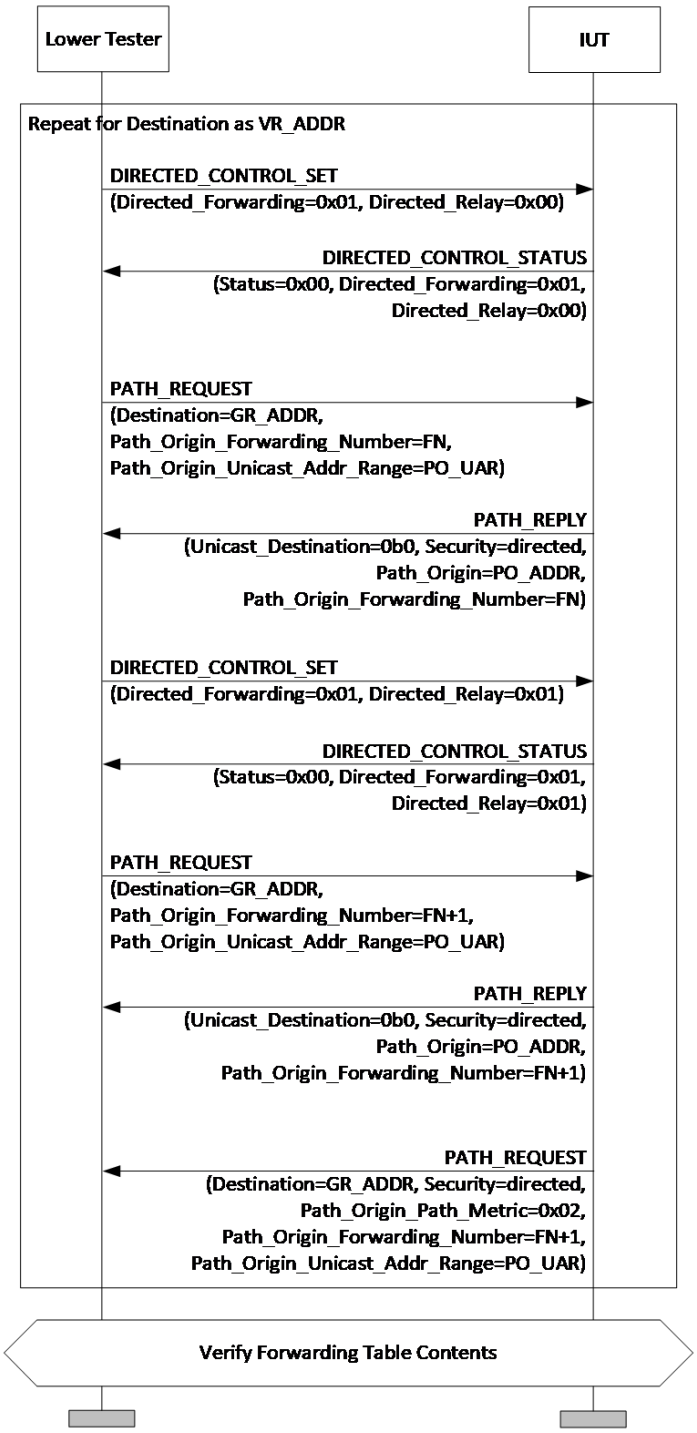


Figure 4.223: MESH/NODE/DF/DISC/BV-03-C MSC

1. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0xFF (Ignore).

2. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or unsupported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or unsupported), and Directed_Friend set to 0x00 or 0x02.
3. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _GR_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
4. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.
5. The IUT does not send any PATH_REQUEST message.
6. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x01 (Enabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0xFF (Ignore).
7. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x01 (Enabled), Directed_Proxy set to 0x00 or 0x02, Directed_Proxy_Use_Directed_Default set to 0x02, and Directed_Friend set to 0x00 or 0x02.
8. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_ + 1, Path_Origin_Path_Metric set to 0x01, Destination set to _GR_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
9. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number set to _FN_ + 1.
10. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and all other fields set to the same values as in Step 8.
11. Repeat Steps 1–10 replacing _GR_ADDR_ with _VR_ADDR_ and Path_Origin_Forwarding_Number set to _FN_ + 2 in Steps 3–4 and _FN_ + 3 in Steps 8–9.
12. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.46 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_ + 1
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Multicast_Destination0	_GR_ADDR_
Non_Fixed_Path #1	
Backward_Path_Validated1	0b0
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_FN_ + 3
Path_Origin_Unicast_Addr_Range1	_PO_UAR_
Bearer_Toward_Path_Origin1	0x0001
Multicast_Destination1	_VR_ADDR_

Table 4.46: Forwarding Table contents for MESH/NODE/DF/DISC/BV-03-C

- Expected Outcome

  Pass verdict

  The IUT sends the PATH_REPLY after the Path_Reply_Delay timeout has expired.

  If the IUT has the directed relay feature enabled, the IUT sends the PATH_REQUEST after the Path_Request_Delay timeout has expired.

  The IUT sends the PATH_REPLY and a PATH_REQUEST message with the expected fields and TTL set to 0.

  The PATH_REPLY and PATH_REQUEST messages are secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

4.18.3.1 Receive PATH_REQUEST for Dependent - Unicast Address, IUT as Path Target

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node receives a PATH_REQUEST message directed to a path target that is a unicast address matching one of the addresses of its dependent nodes and replies with PATH_REPLY.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.47 apply.
  - Lower Tester 2 exposes two elements.

- Test Procedure

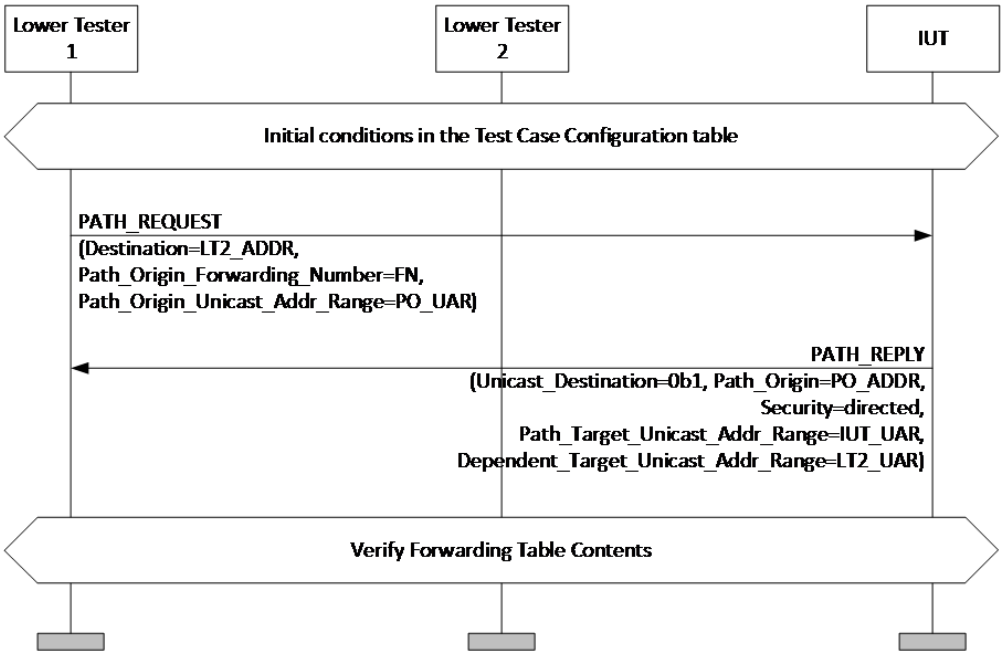


Figure 4.224: Receive PATH_REQUEST for Dependent - Unicast Address, IUT as Path Target MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x04, Destination set to _LT2_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b1, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, Path_Target_Unicast_Addr_Range set to _IUT_UAR_, and Dependent_Target_Unicast_Addr_Range set to the value from Table 4.47.
3. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.47 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated$_0$	0b0
Lane_Counter$_0$	0x01
Path_Origin_Forwarding_Number$_0$	_FN_
Path_Origin_Unicast_Addr_Range$_0$	_PO_UAR_
Bearer_Toward_Path_Origin$_0$	0x0001
Path_Target_Unicast_Addr_Range$_0$	_IUT_UAR_
Dependent_Target_Unicast_Addr_Range_List0	{Value from Table 4.48}

Table 4.47: Forwarding Table contents for Receive PATH_REQUEST for Dependent - Unicast Address, IUT as Path Target

- Test Case Configuration

TCID	Additional Initial Condition	Dependent Target Unicast Address Range
MESH/NODE/DF/DISC/BV-04-C [Receive PATH_REQUEST - Unicast Address, IUT as Friend Path Target]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.9.6.3.1.	_LT2_UAR_
MESH/NODE/DF/DISC/BV-05-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target for Proxy Client]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 adds _LT2_ADDR_ to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].	_LT2_ADDR_
MESH/NODE/DF/DISC/BV-06-C [Receive PATH_REQUEST - Unicast Address, IUT as Path Target for Directed Proxy Client]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13.	_LT2_UAR_

TCID	Additional Initial Condition	Dependent Target Unicast Address Range
MESH/NODE/DF/DISC/BV-12-C [Receive PATH_REQUEST - Unicast Address, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). The IUT is configured with the bridging table entry (0x01, 0, 1, _PO_ADDR_, _LT2_ADDR_) by running preamble 4.2.7.	_LTR_ADDR_

Table 4.48: Additional test conditions for Receive PATH_REQUEST for Dependent - Unicast Address, IUT as Path Target

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REPLY message with the expected fields and TTL set to 0, after the Path_Reply_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send any PATH_REQUEST messages.

  The IUT reports the Forwarding Table with the expected values.

4.18.3.2 Receive PATH_REQUEST for Dependent - Group/Virtual Address, IUT as Path Target

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node replies with a PATH_REPLY message and forwards the PATH_REQUEST message when receiving a PATH_REQUEST message where the path target is a group or virtual address matching one of the addresses its dependent node is subscribed to.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.50 apply.
  - Lower Tester 2 exposes two elements.

- Test Procedure

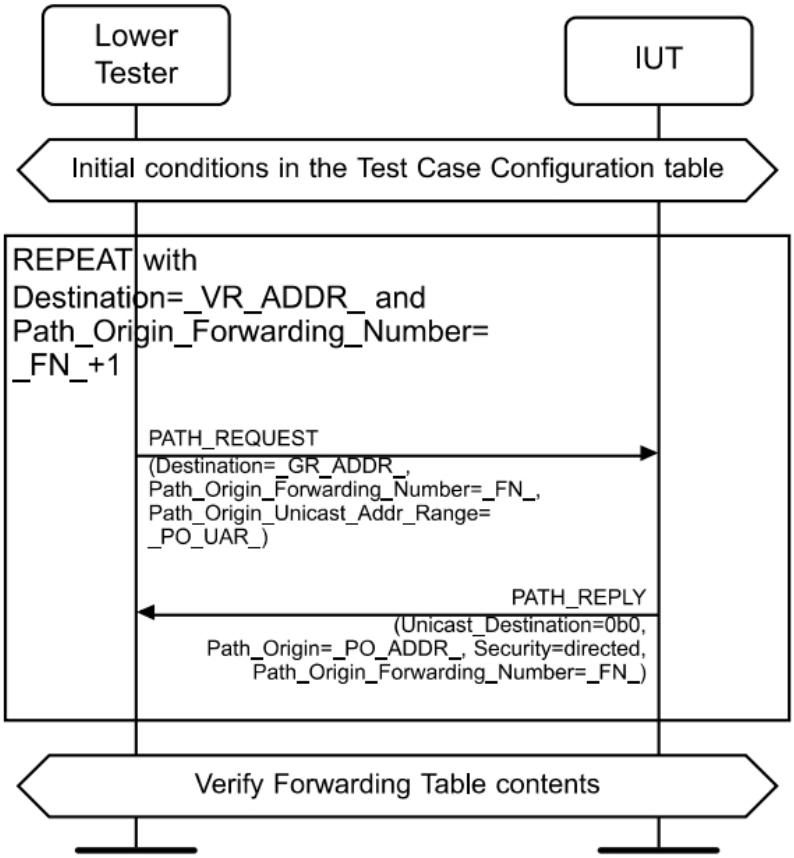


Figure 4.225: Test Procedure for Receive PATH_REQUEST for Dependent - Group/Virtual Address, IUT as Path Target MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x04, Destination set to _GR_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REPLY message to _LT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.
3. Repeat Steps 1 and 2 with Destination set to _VR_ADDR_ and Path_Origin_Forwarding_Number set to _FN_ + 1.
4. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.49 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Multicast_Destination0	_GR_ADDR_
Non_Fixed_Path #1	
Backward_Path_Validated1	0b0
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_FN_ + 1
Path_Origin_Unicast_Addr_Range1	_PO_UAR_
Bearer_Toward_Path_Origin1	0x0001
Multicast_Destination1	_VR_ADDR_

Table 4.49: Forwarding Table contents for Receive PATH_REQUEST for Dependent - Group/Virtual Address, IUT as Path Target

- Test Case Configuration

TCID	Additional Initial Condition
MESH/NODE/DF/DISC/BV-07-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Friend Path Target]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.9.6.3.1. Lower Tester 2 adds _GR_ADDR_ and _VR_ADDR_ to the Friend Subscription List on the IUT.
MESH/NODE/DF/DISC/BV-08-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Proxy Path Target]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13.
MESH/NODE/DF/DISC/BV-13-C [Receive PATH_REQUEST - Group/Virtual Address, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). The IUT is configured with the bridging table entry (0x01, 0, 1, _PO_ADDR_, _GR_ADDR_) by running preamble 4.2.7.

Table 4.50: Additional test conditions for Receive PATH_REQUEST for Dependent - Group/Virtual Address, IUT as Path Target

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REPLY and a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT sends the PATH_REPLY after the Path_Reply_Delay timeout has expired.

  The IUT reports the Forwarding Table with the expected values.

4.18.3.3 Ignore PATH_REQUEST, Directed Control Configurations

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a PATH_REQUEST message based on different configurations of the Directed Control state.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.51 apply.

- Test Procedure

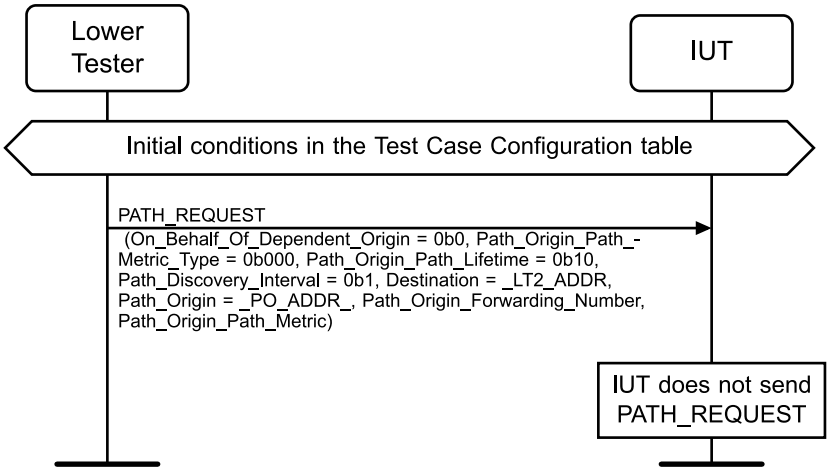


Figure 4.226: Ignore PATH_REQUEST, Directed Control Configurations MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Destination set to _LT2_ADDR_, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number and Path_Origin_Path_Metric set to random valid values.
2. The IUT does not send any PATH_REQUEST messages.

- Test Case Configuration

TCID	Additional Initial Condition
MESH/NODE/DF/DISC/BV-09-C [Ignore PATH_REQUEST – Directed Relay Disabled]	The directed relay functionality on the primary subnet is disabled on the IUT.
MESH/NODE/DF/DISC/BV-10-C [Ignore PATH_REQUEST – Directed Friend Enabled]	The directed friend functionality on the primary subnet is enabled on the IUT. The directed relay functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.8.5.3.1.
MESH/NODE/DF/DISC/BV-11-C [Ignore PATH_REQUEST – Directed Proxy Enabled]	The directed relay functionality on the primary subnet is enabled on the IUT. Directed Proxy and Directed Proxy Use Directed Default states on the primary subnet are enabled on the IUT. The IUT has performed a proxy connection to Lower Tester 2 through a connection-oriented channel.

Table 4.51: Additional test conditions for Ignore PATH_REQUEST, Directed Control Configurations]

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REQUEST messages.

MESH/NODE/DF/DISC/BV-14-C [Ignore PATH_REQUEST, Lower Forwarding Number]

- Test Purpose

  Verify that a Directed Relay node IUT does not forward an outdated PATH_REQUEST for which it is a path target.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality is enabled on the IUT for the primary subnet.
  - The IUT has established a path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10. The Lower Tester stores the _FN_ value.

- Test Procedure

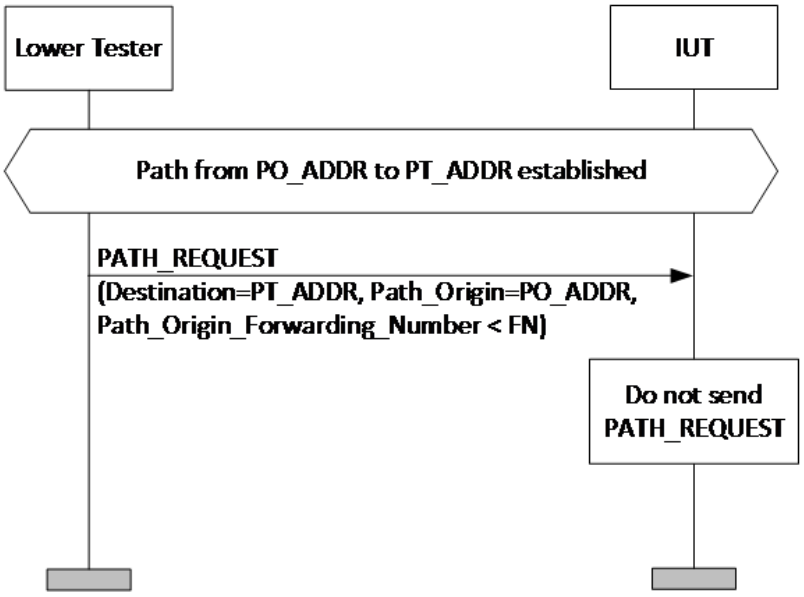


Figure 4.227: MESH/NODE/DF/DISC/BV-14-C MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with the On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Destination set to _PT_ADDR_, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to a value lower than _FN_, and Path_Origin_Path_Metric set to 0x01.
2. The IUT does not send any PATH_REQUEST messages.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REQUEST messages for duplicate and outdated messages.

MESH/NODE/DF/DISC/BV-15-C [Ignore PATH_REQUEST, Neighbor with Low RSSI]

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a PATH_REQUEST message received with a low RSSI from a neighbor node.

- Reference

  [10] 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality is enabled on the IUT.
  - The Default RSSI Threshold on the IUT is set to 127 dBm.

- Test Procedure

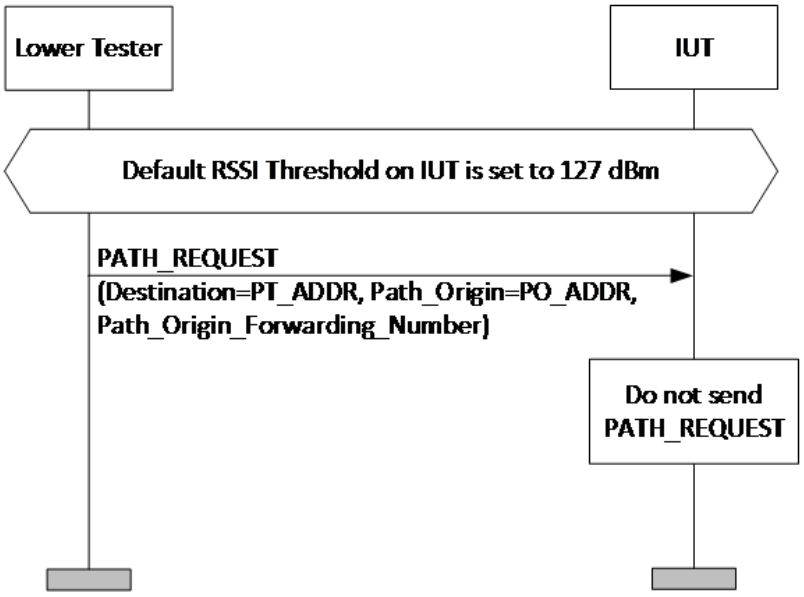


Figure 4.228: MESH/NODE/DF/DISC/BV-15-C MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Destination set to _IUT_ADDR_, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number and Path_Origin_Path_Metric set to random valid values.
2. The IUT does not send any PATH_REPLY messages.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REPLY messages.

MESH/NODE/DF/DISC/BV-16-C [Ignore PATH_REQUEST, Wrong Path Metric]

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a PATH_REQUEST message received with an unsupported Path Metric Type or a Path Metric that is not preferred.

- Reference

  [10] 3.6.8.2.2

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

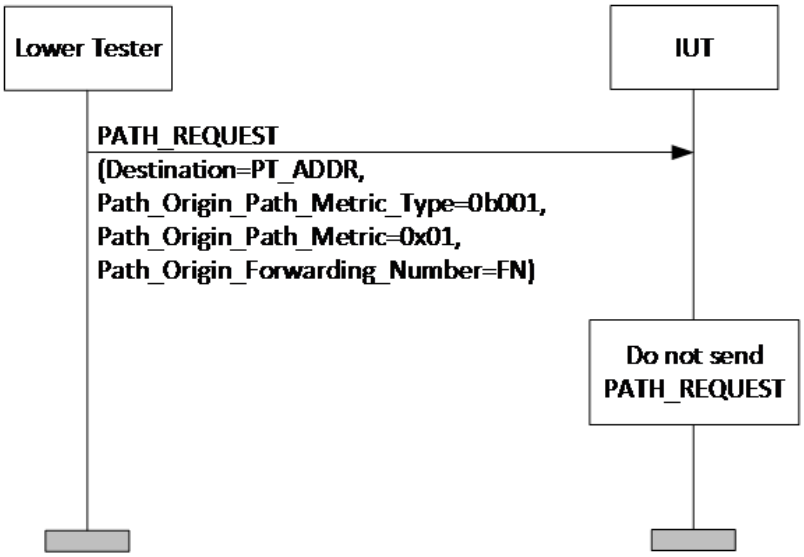


Figure 4.229: MESH/NODE/DF/DISC/BV-16-C MSC

1. The Lower Tester sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b001, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _IUT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT does not send any PATH_REPLY messages.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REPLY messages.

MESH/NODE/DF/DISC/BV-17-C [Multiple Lanes, IUT as Path Target]

- Test Purpose

  Verify that a Directed Forwarding node IUT can set up multiple lanes as a path target.

- Reference

  [10] 3.6.8.2.2

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT has established a path from _PO_ADDR_ by running preamble 4.2.11. The Lower Tester stores the Path_Origin_Forwarding_Number used.

- Test Procedure

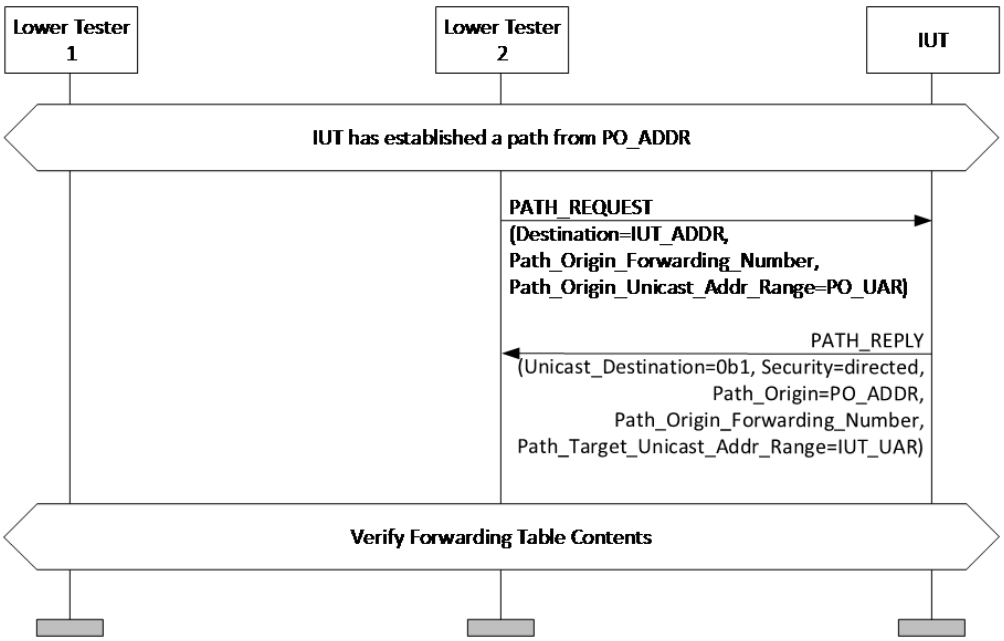


Figure 4.230: MESH/NODE/DF/DISC/BV-17-C MSC

1. After the path discovery and the lane discovery guard interval, Lower Tester 2 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to the stored value, Path_Origin_Path_Metric set to 0x04, Destination set to _IUT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REPLY message to _LT2_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to the stored value, and Path_Target_Unicast_Addr_Range set to _IUT_UAR_.
3. Lower Tester 2 verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.52 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x02
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_IUT_UAR_

Table 4.52: Forwarding Table contents for MESH/NODE/DF/DISC/BV-17-C

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REPLY message encrypted with directed security material with the expected fields and TTL set to 0, after the Path_Reply_Delay timeout has expired.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/DISC/BI-01-C [Ignore PATH_REQUEST – Invalid Parameters]

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a PATH_REQUEST with invalid parameters.

- Reference

  [10] 3.6.5.11, 3.6.8.2.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

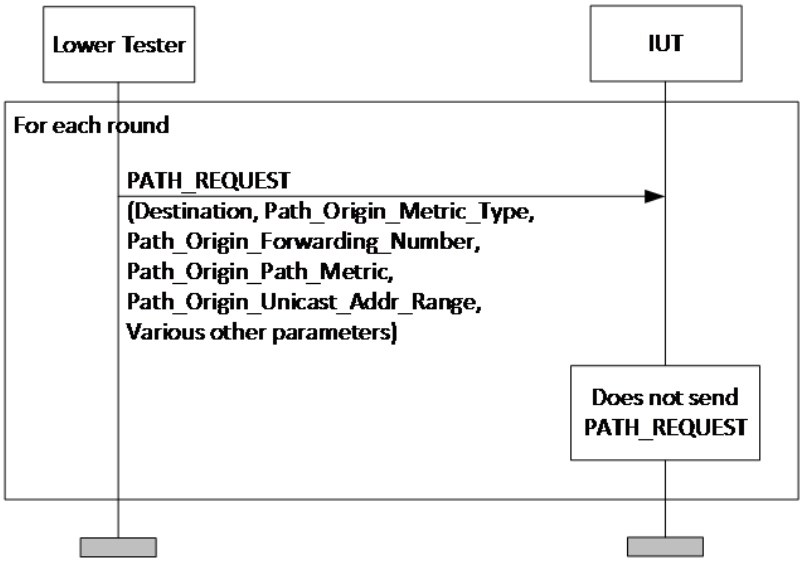


Figure 4.231: MESH/NODE/DF/DISC/BI-01-C MSC

For each round in Table 4.53 below:

1. The Lower Tester sends a PATH_REQUEST message with the following parameters, unless stated in Table 4.53: DST set to the all-directed-forwarding-nodes address, On_Behalf_Of_Dependent_Origin set to 0b1, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number and Path_Origin_Path_Metric set to valid values, Destination set to _PT_ADDR_, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, and Dependent_Origin_Unicast_Addr_Range set to random valid values.
2. The IUT does not send any PATH_REQUEST messages after the Path Request Delay.

Round	Additional Condition
1	Destination set to 0x0000
2	Prohibited bits set to 0b1
3	On_Behalf_Of_Dependent_Origin set to 0b1 and no Dependent_Origin_Unicast_Addr_Range field
4	Path_Origin_Unicast_Addr_Range set to 0xFFFF
5	Path_Origin_Unicast_Addr_Range set to 0xFFFF02
6	Path_Origin_Unicast_Addr_Range set to 0x800100
7	Dependent_Origin_Unicast_Addr_Range set to 0xFFFF
8	Dependent_Origin_Unicast_Addr_Range set to 0xFFFF02
9	Dependent_Origin_Unicast_Addr_Range set to 0x800100
10	DST set to _IUT_ADDR_
11	TTL set to 0x02
12	Message sent with managed flooding security material
13	Destination set to 0xFFFB (all-directed-forwarding-nodes fixed group address)

Round	Additional Condition
14	Destination set to 0xFFFF (all-nodes fixed group address)
15	Destination set to 0xFFFE (all-relays fixed group address)

Table 4.53: Specification of rounds for MESH/NODE/DF/DISC/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REQUEST messages.

4.18.4 Directed Forwarding Establishment

This subsection verifies the Directed Forwarding Establishment procedure.

MESH/NODE/DF/EST/BV-01-C [Receive PATH_REPLY, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT acting as Directed Relay node receives and forwards a PATH_REPLY message where the path origin and destination are unicast addresses do not match any of the elements on the IUT.

- Reference

  [10] 3.6.8.2.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

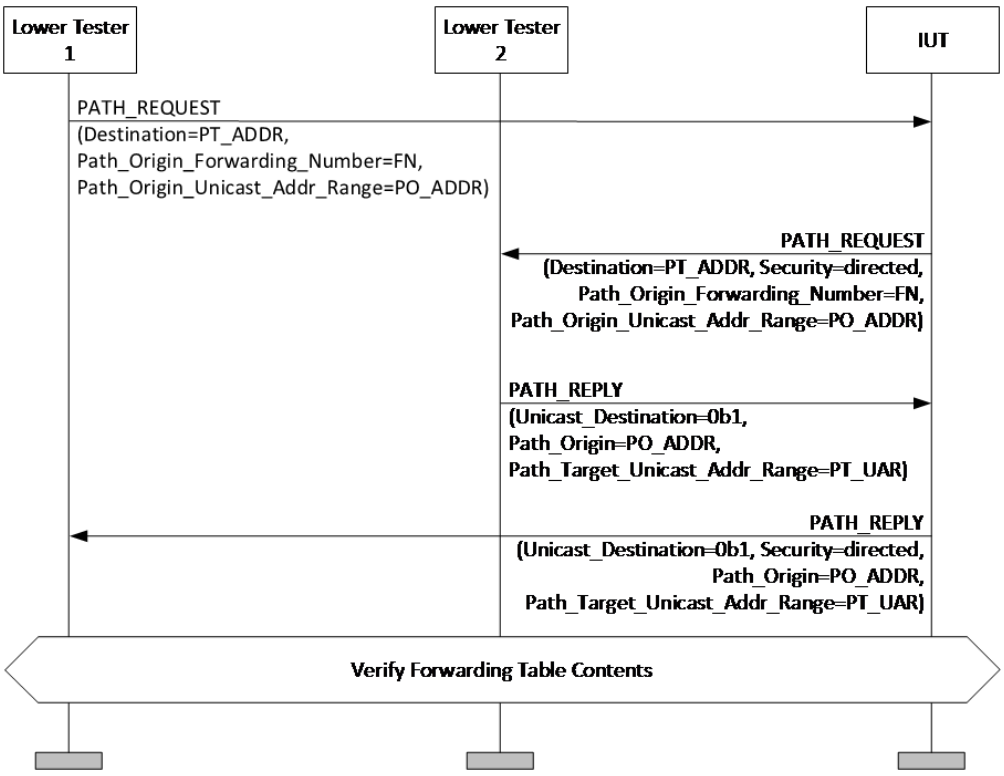


Figure 4.232: MESH/NODE/DF/EST/BV-01-C MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and all other fields set to the same value as in Step 1.
3. Lower Tester 2 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
4. The IUT sends a PATH_REPLY message to _LT1_ADDR_ with all fields set to the same value as in Step 3.
5. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.54 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001

Table 4.54: Forwarding Table contents for MESH/NODE/DF/EST/BV-01-C

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0 for the first received PATH_REQUEST message, after the Path_Request_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

  The IUT sends a PATH_REPLY message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/EST/BV-02-C [Ignore PATH_REPLY, Empty Discovery Table]

- Test Purpose

  Verify that an IUT acting as a Directed Relay node does not forward a PATH_REPLY message where the path origin and the Path_Origin_Forwarding_Number are not matching an entry in the Discovery Table.

- Reference

  [10] 3.6.8.2.3

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

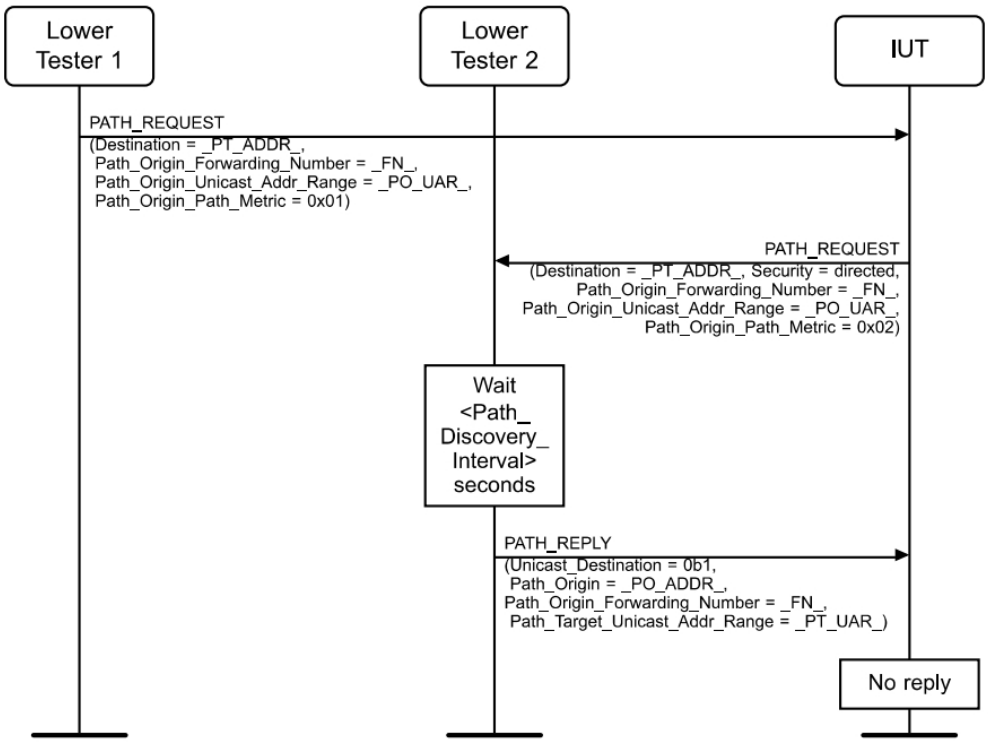


Figure 4.233: MESH/NODE/DF/EST/BV-02-C MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b0 (5s), Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and all other fields set to the same value as in Step 1.
3. After the path discovery interval, Lower Tester 2 sends a PATH_REPLY message to the IUT with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b1, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
4. The IUT does not send any PATH_REPLY message.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REPLY messages.

MESH/NODE/DF/EST/BV-03-C [Receive PATH_REPLY - Unicast Address, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT receives a PATH_REPLY as a response to a PATH_REQUEST

message where the path origin is a unicast address matching one of its elements. The IUT establishes a path.

- Reference

  [10] 3.6.8.2.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

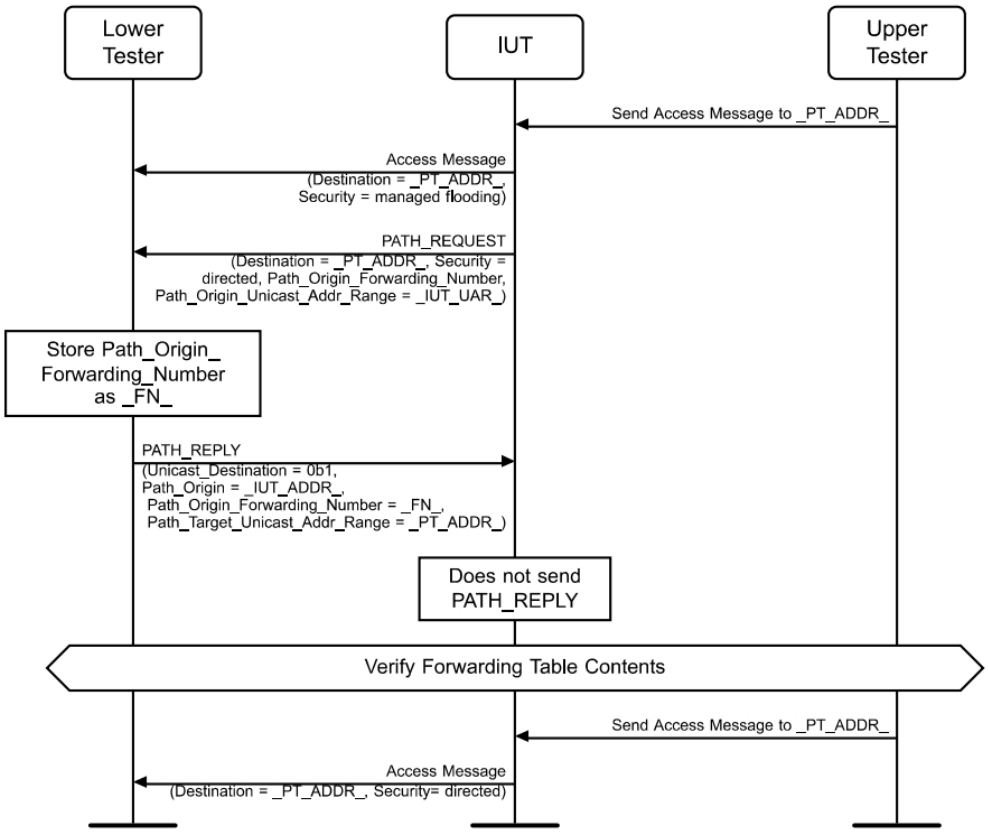


Figure 4.234: MESH/NODE/DF/EST/BV-03-C MSC

1. The Upper Tester commands the IUT to send an access message to _PT_ADDR_.
2. The IUT sends the access message to _PT_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and

Path_Target_Unicast_Addr_Range set to _PT_UAR_.

4. The IUT does not send any PATH_REPLY message.
5. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.55 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001

Table 4.55: Forwarding Table contents for MESH/NODE/DF/EST/BV-03-C MSC

6. Repeat Step 1.
7. The IUT sends the access message to _PT_ADDR_ and no PATH_REQUEST messages.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  In Step 2, the IUT sends the access message secured with managed flooding security credentials on the primary subnet.

  In Step 7, the IUT sends the access message secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

4.18.4.1 Receive PATH_REPLY for Dependent, IUT as Path Origin

- Test Purpose

  Verify that a Directed Forwarding node IUT receives a PATH_REPLY where the path origin is a unicast address matching one of the addresses of its dependent nodes and establishes a path.

- Reference

  [10] 3.6.8.2.3

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Additional initial conditions defined in Table 4.57 apply.

- Test Procedure

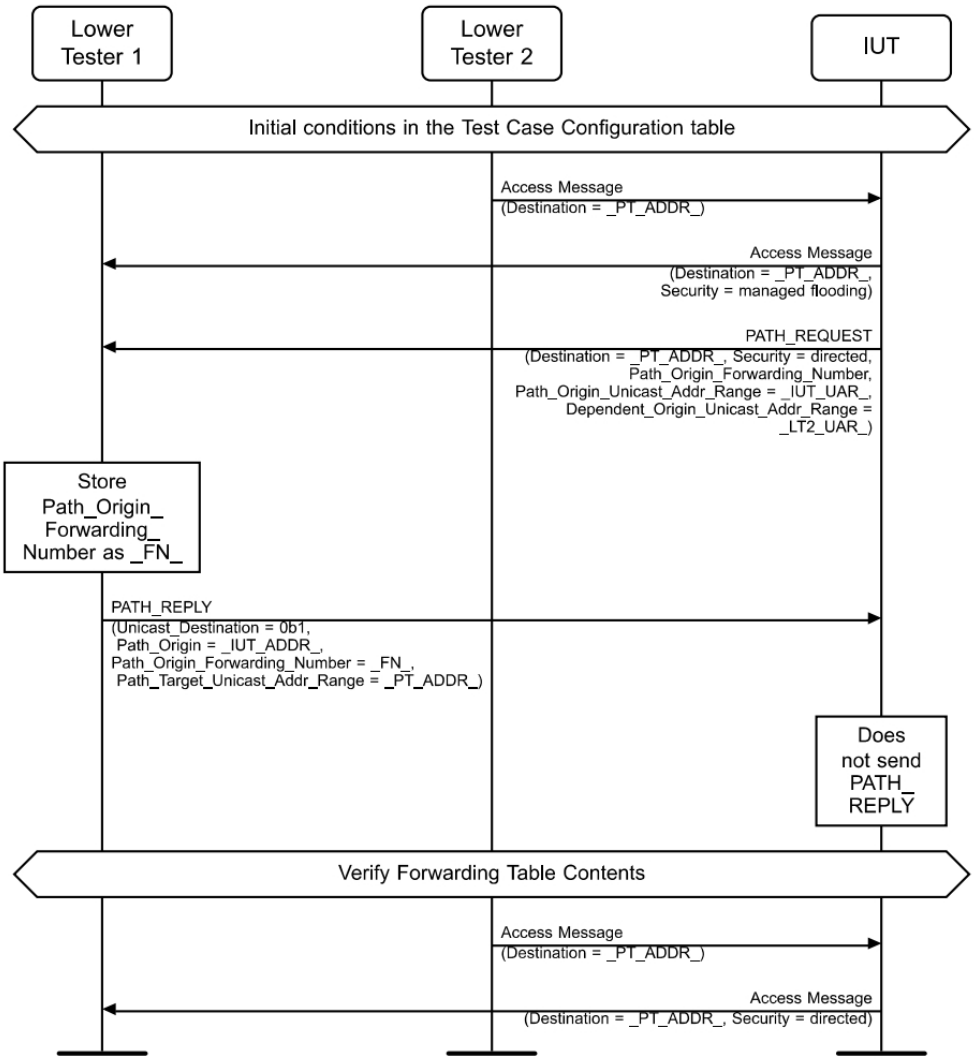


Figure 4.235: Receive PATH_REPLY for Dependent, IUT as Path Origin MSC

1. Lower Tester 2 sends an access message to _PT_ADDR_.
2. The IUT sends the access message to _PT_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b1, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, and Dependent_Origin_Unicast_Addr_Range set to _LT2_UAR_. Lower Tester 1 stores the Path_Origin_Forwarding_Number as _FN_.
3. Lower Tester 1 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.

4. The IUT does not send any PATH_REPLY message.
5. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.56 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Dependent_Origin_List0	{_LT2_UAR_}
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001

Table 4.56: Forwarding Table contents for Receive PATH_REPLY for Dependent, IUT as Path Origin

6. Repeat Step 1.
7. The IUT sends the access message to _PT_ADDR_ and no PATH_REQUEST messages.

- Test Case Configuration

TCID	Additional Initial Condition
MESH/NODE/DF/EST/BV-04-C [Receive PATH_REPLY, IUT as Friend Path Origin]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.8.5.3.1. Lower Tester 2 uses only friendship security credentials to send access messages.
MESH/NODE/DF/EST/BV-05-C [Receive PATH_REPLY, IUT as Proxy Path Origin]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13.
MESH/NODE/DF/EST/BV-09-C [Receive PATH_REPLY, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). Lower Tester 1 is only part of the first subnet (index 0), and Lower Tester 2 is only part of the second subnet (index 1). Directed forwarding is enabled on the IUT on the first subnet (index 0). The IUT is configured with the bridging table entry (0x01, 1, 0, _LT2_ADDR_, _PT_ADDR_) by running preamble 4.2.7.

Table 4.57: Additional test conditions for Receive PATH_REPLY for Dependent, IUT as Path Origin

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send any PATH_REPLY messages.

  In Step 2, the IUT sends the access message secured with managed flooding security credentials on the primary subnet.

  In Step 7, the IUT sends the access message secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/EST/BV-06-C [Receive PATH_REPLY - Group/Virtual Address, IUT as Directed Relay]

- Test Purpose

  Verify that a Directed Relay node IUT receives and forwards a PATH_REPLY message corresponding to a PATH_REQUEST with the path target that is a group or virtual address.

- Reference

  [10] 3.6.8.2.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

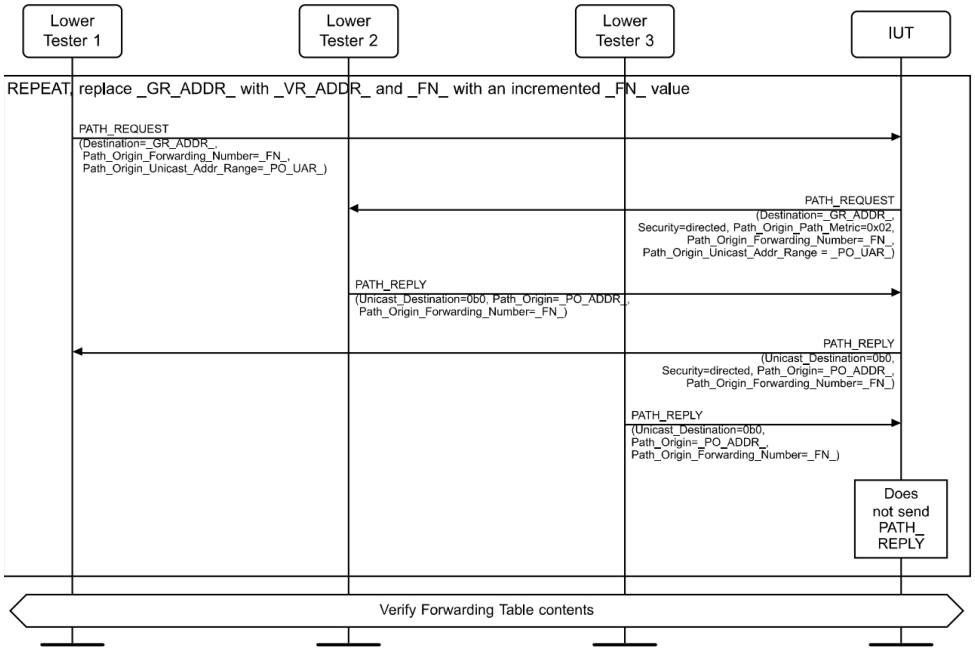


Figure 4.236: MESH/NODE/DF/EST/BV-06-C MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _GR_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UDR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and all other fields set to the same values as in Step 1.
3. Lower Tester 2 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.
4. The IUT sends a PATH_REPLY message to _LT1_ADDR_ with all fields set to the same values as in Step 3.
5. Lower Tester 3 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.
6. The IUT does not send a PATH_REPLY message.
7. Repeat Steps 1–6 by replacing _GR_ADDR_ with _VR_ADDR_ and an incremented _FN_ value.
8. The Lower Tester verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.58 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Multicast_Destination0	_GR_ADDR_
Bearer_Toward_Path_Target0	0x0001
Non_Fixed_Path #1	
Backward_Path_Validated1	0b0
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_FN_
Path_Origin_Unicast_Addr_Range1	_PO_UAR_
Bearer_Toward_Path_Origin1	0x0001
Multicast_Destination1	_VR_ADDR_
Bearer_Toward_Path_Target1	0x0001

Table 4.58: Forwarding Table contents for MESH/NODE/DF/EST/BV-06-C

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0 for the first received PATH_REQUEST message, after the Path_Request_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

  The IUT sends a PATH_REPLY message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT reports a Forwarding Table with the expected values.

MESH/NODE/DF/EST/BV-07-C [Receive PATH_REPLY – Group/Virtual Address, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT receives a PATH_REPLY as a response to a PATH_REQUEST message for a group or virtual address and establishes the path.

- Reference

[10] 3.6.8.2.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

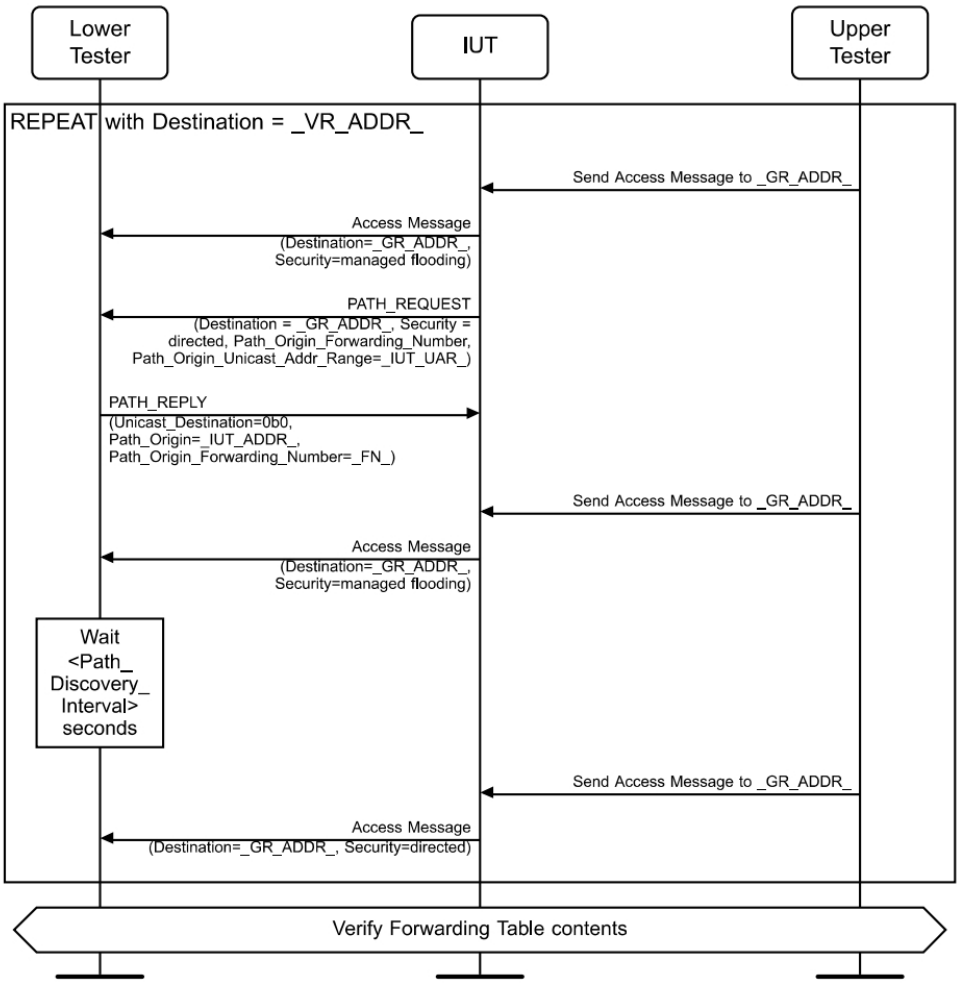


Figure 4.237: MESH/NODE/DF/EST/BV-07-C MSC

1. The Upper Tester commands the IUT to send an access message to _GR_ADDR_.
2. The IUT sends the access message to _GR_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _GR_ADDR_ , and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.

4. The IUT does not send any PATH_REPLY message.
5. Repeat Step 1.
6. The IUT sends the access message to _GR_ADDR_ and no PATH_REQUEST messages.
7. After the path discovery interval, repeat Step 1.
8. The IUT sends the access message to _GR_ADDR_ and no PATH_REQUEST messages.
9. Repeat Steps 1–8 with Destination set to _VR_ADDR_.
10. The Lower Tester verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.59 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Multicast_Destination0	_GR_ADDR_
Bearer_Toward_Path_Target0	0x0001
Non_Fixed_Path #1	
Backward_Path_Validated1	0b0
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_FN_
Path_Origin_Unicast_Addr_Range1	_IUT_UAR_
Multicast_Destination1	_VR_ADDR_
Bearer_Toward_Path_Target1	0x0001

Table 4.59: Forwarding Table contents for MESH/NODE/DF/EST/BV-07-C

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send a PATH_REPLY message.

  In Steps 2 and 6, the IUT sends the access message secured with managed flooding security credentials on the primary subnet.

  In Step 8, the IUT sends the access message secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/EST/BV-08-C [Multiple Lanes, IUT as Directed Relay]

- Test Purpose

  Verify that a Directed Relay IUT node can set up multiple lanes.

- Reference

  [10] 3.6.8.2.2

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10. The Lower Tester stores the Path_Origin_Forwarding_Number used.

- Test Procedure

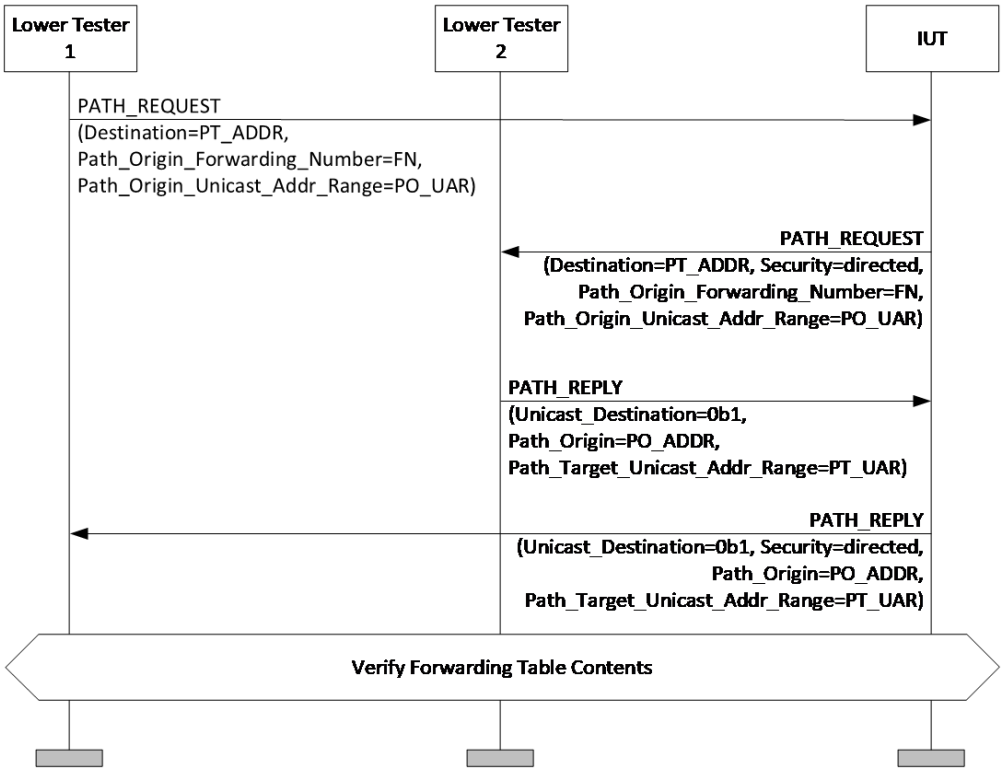


Figure 4.238: MESH/NODE/DF/EST/BV-08-C MSC

1. After the path discovery and the lane discovery guard interval, Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.

2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x03 and all other fields set to the same value as in Step 1.
3. Lower Tester 2 sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
4. The IUT sends a PATH_REPLY message to _LT1_ADDR_ with all fields set to the same value as in Step 3.
5. The Lower Tester verifies that the Forwarding Tpable contents on the primary subnet matches the values from Table 4.60 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x02
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001

Table 4.60: Forwarding Table contents for MESH/NODE/DF/EST/BV-08-C

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST message with the expected fields and TTL set to 0 for the first received PATH_REQUEST message, after the Path_Request_Delay timeout has expired. The message is secured with directed security credentials on the primary subnet.

  The IUT sends a PATH_REPLY message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/EST/BI-01-C [Ignore PATH_REPLY – Invalid Parameters]

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a PATH_REPLY with invalid parameters.

- Reference

  [10] 3.6.5.12, 3.6.8.2.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
- Test Procedure

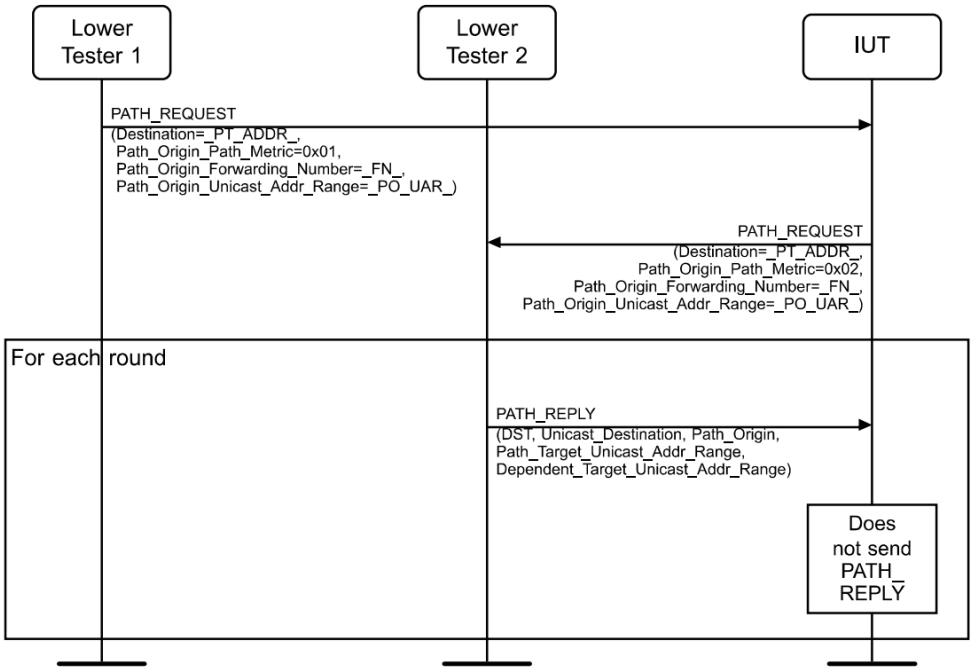


Figure 4.239: MESH/NODE/DF/EST/BI-01-C MSC

1. Lower Tester 1 sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to _FN_, Path_Origin_Path_Metric set to 0x01, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _PO_UAR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with Path_Origin_Path_Metric set to 0x02 and all other fields set to the values in Step 1.
3. For each round in Table 4.61, repeat Steps 4 and 5.
4. Lower Tester 2 sends a PATH_REPLY message with the following parameters, unless stated in Table 4.61: DST set to _IUT_ADDR_, Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b1, Confirmation_Request set to 0b1, Path_Origin set to _PO_ADDR_, Path_Origin_Forwarding_Number set to the same value as in Step 1, Path_Target_Unicast_Addr_Range set to _PT_UAR_, and Dependent_Target_Unicast_Addr_Range set to _DT_UAR_.
5. The IUT does not send any PATH_REPLY messages.

Round	Additional Condition
1	Path_Target set to 0x0000
2	Path_Origin set to a group address
3	Prohibited bits set to 0b1
4	Path_Target_Unicast_Addr_Range set to 0xFFFF
5	Path_Target_Unicast_Addr_Range set to 0xFFFF02
6	Path_Target_Unicast_Addr_Range set to 0x800100
7	Dependent_Target_Unicast_Addr_Range set to 0xFFFF
8	Dependent_Target_Unicast_Addr_Range set to 0xFFFF02
9	Dependent_Target_Unicast_Addr_Range set to 0x800100
10	DST set to an all-forwarding-nodes address
11	TTL set to 0x02
12	Message sent with managed flooding security material

Table 4.61: Specification of rounds for MESH/NODE/DF/EST/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REPLY messages.

4.18.5 Directed Forwarding Confirmation

This subsection verifies the Directed Forwarding Confirmation procedure.

MESH/NODE/DF/CONF/BV-01-C [Send PATH_CONFIRMATION, IUT as Path Origin]

- Test Purpose

  Verify that a Directed Forwarding node IUT sends a PATH_CONFIRMATION when it receives a PATH_REPLY message with the Confirmation_Request field set to 0b1 for an originated path discovery.

- Reference

  [10] 3.6.8.2.4

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Wanted Lanes state on the primary subnet is configured to 0x02 on the IUT.

- Test Procedure

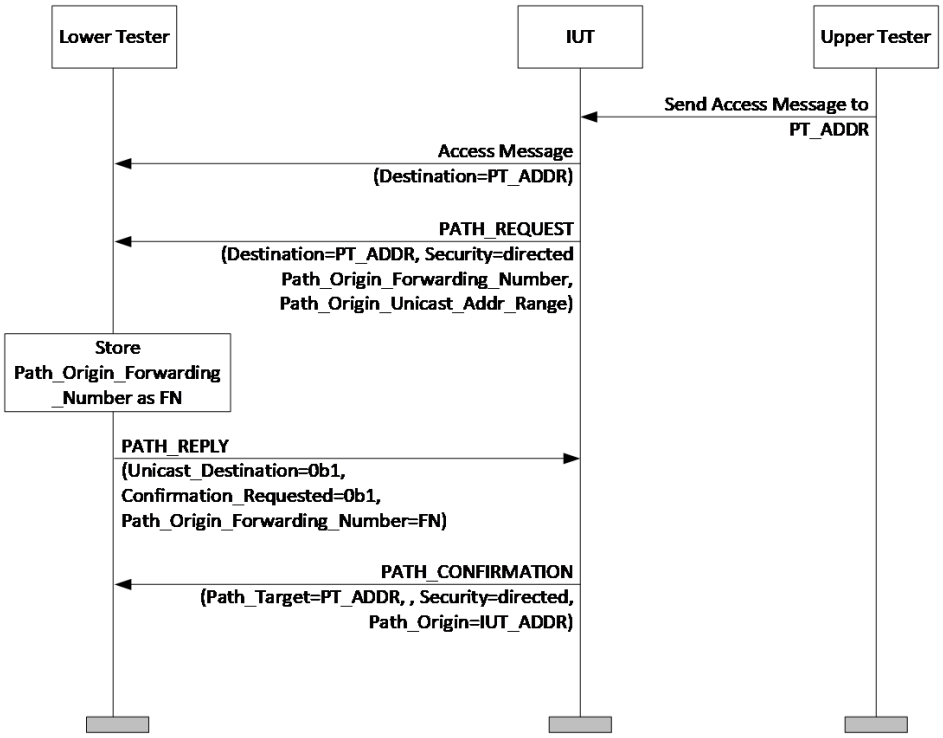


Figure 4.240: MESH/NODE/DF/CONF/BV-01-C MSC

1. The Upper Tester commands the IUT to send an access message to _PT_ADDR_.
2. The IUT sends the access message to _PT_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b1, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b1, Path_Origin set to _IUT_ADDR_, Path_Origin_Forwarding_Number set to _FN_, and Path_Target_Unicast_Addr_Range set to _PT_UAR_.
4. The IUT sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _IUT_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_CONFIRMATION message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

MESH/NODE/DF/CONF/BV-02-C [Receive PATH_CONFIRMATION, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT acting as Directed Relay node forwards a PATH_CONFIRMATION message where the path origin and destination are unicast addresses not matching one of its elements and the IUT has a matching path stored in the Forwarding Table with the Backward_Path_Validated field set to 0.

- Reference

  [10] 3.6.8.2.4

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10.

- Test Procedure

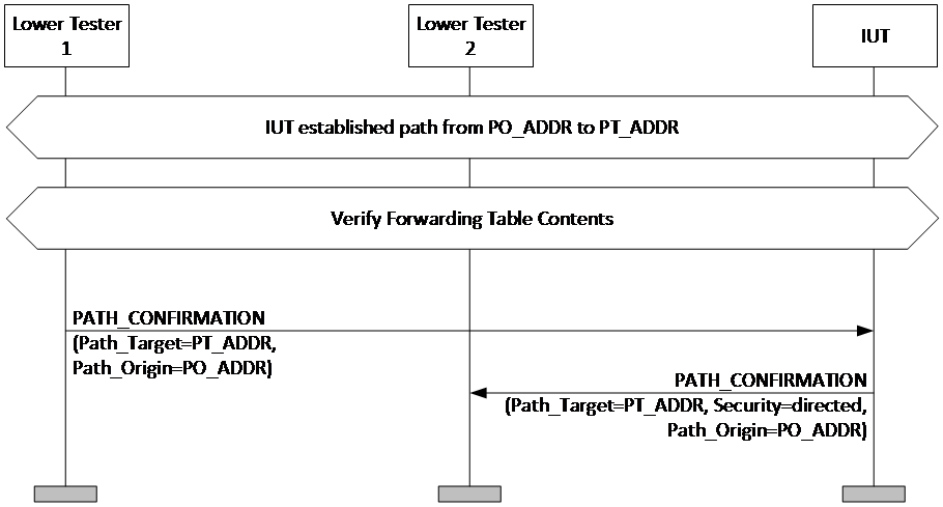


Figure 4.241: MESH/NODE/DF/CONF/BV-02-C MSC

1. Lower Tester 1 verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.62 by running the test procedure from Section 4.18.1. The Lower Tester stores the Forwarding_Table_Update_Identifier value.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001

Table 4.62: Forwarding Table contents for MESH/NODE/DF/CONF/BV-02-C

2. Lower Tester 1 sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_.
3. The IUT sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_.
4. Lower Tester 1 verifies that the Forwarding Table contents on the primary subnet match the values from Table 4.62 by running the test procedure from Section 4.18.1, with Backward_Path_Validated set to 0b1 and Forwarding_Table_Update_Identifier set to a different value than the stored value.

- Expected Outcome

  Pass verdict

  In Step 3, the IUT sends a PATH_CONFIRMATION message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send any additional PATH_CONFIRMATION messages.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/CONF/BV-03-C [Receive PATH_CONFIRMATION, IUT as Path Target]

- Test Purpose

  Verify that a Directed Forwarding node IUT processes a PATH_CONFIRMATION message and validates a matching path stored in the Forwarding Table where the path target is a unicast address matching one of its elements.

- Reference

  [10] 3.6.8.2.4

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- The Two Way Path state on the primary subnet is set to 0b1 on the IUT.
- The IUT has established a path from _PO_ADDR_ by running preamble 4.2.11. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.

- Test Procedure

  1. The Lower Tester verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.63 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b1
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_IUT_UAR_

Table 4.63: Forwarding Table contents for MESH/NODE/DF/CONF/BV-03-C

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_CONFIRMATION messages.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/CONF/BV-04-C [Ignore PATH_CONFIRMATION, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT acting as Directed Relay node ignores a PATH_CONFIRMATION message when it does not have a matching path stored in the Forwarding Table or the path was previously confirmed.

- Reference

  [10] 3.6.8.2.4

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10.

- Test Procedure

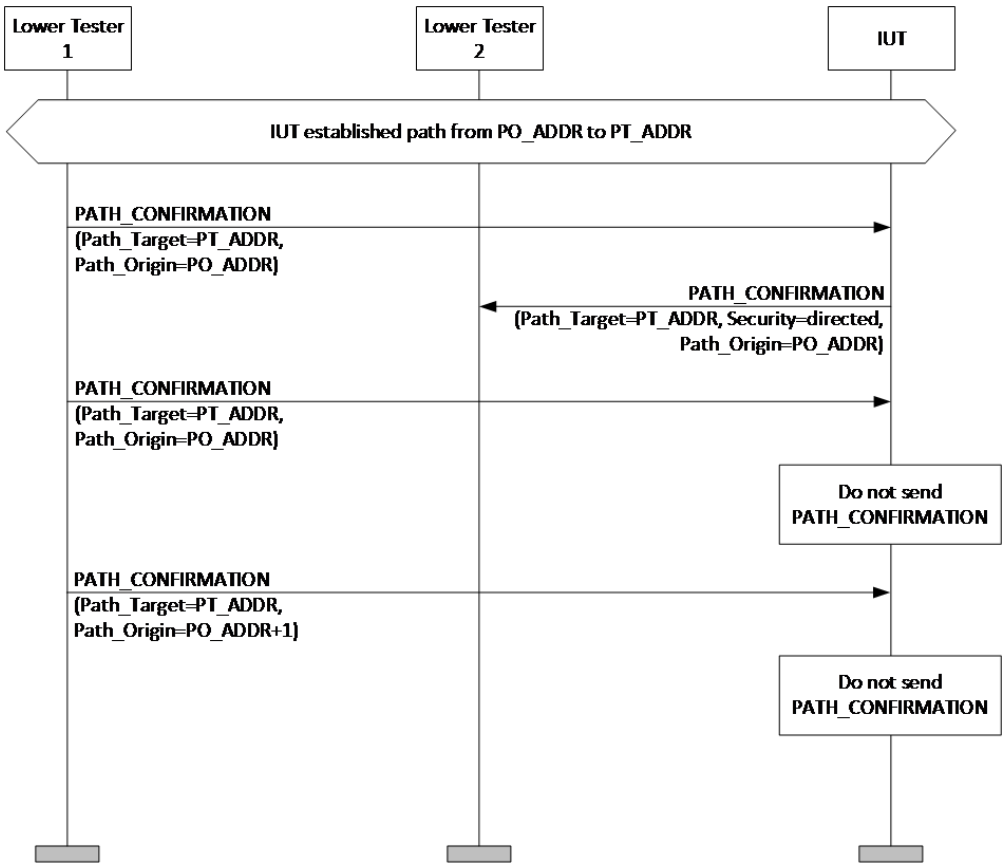


Figure 4.242: MESH/NODE/DF/CONF/BV-04-C MSC

1. Lower Tester 1 sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_.
2. The IUT sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_.
3. Repeat Step 1.
4. The IUT does not send any PATH_CONFIRMATION messages.
5. Lower Tester 1 sends a PATH_CONFIRMATION message to the all-directed-forwarding-nodes address with Path_Target set to _PT_ADDR_ and Path_Origin set to _PO_ADDR_ + 1.
6. The IUT does not send any PATH_CONFIRMATION messages.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_CONFIRMATION messages in Steps 4 and 6.

MESH/NODE/DF/CONF/BI-01-C [Ignore PATH_CONFIRMATION – Invalid Parameters]

- Test Purpose

  Verify that a Directed Relay node IUT ignores a PATH_CONFIRMATION with invalid parameters.

- Reference

[10] 3.6.5.13, 3.6.8.2.4

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10.
- Test Procedure

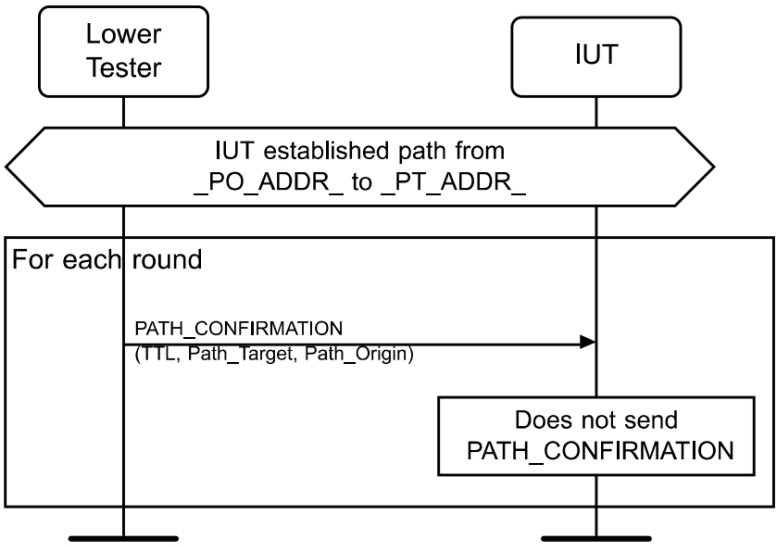


Figure 4.243: MESH/NODE/DF/CONF/BI-01-C MSC

For each round in Table 4.64 below:

1. The Lower Tester sends a PATH_CONFIRMATION message with the following parameters, unless stated in Table 4.64: DST set to the all-directed-forwarding-nodes address, TTL set to 0x02, Path_Target set to _PT_ADDR_, and Path_Origin set to _PO_ADDR_.
2. The IUT does not send any PATH_CONFIRMATION message.

Round	Additional Condition
1	DST set to a _IUT_ADDR_
2	TTL set to 0x02
3	Message sent with managed flooding security material

Table 4.64: Specification of rounds for MESH/NODE/DF/CONF/BI-01-C

- Expected Outcome

Pass verdict

The IUT does not send any PATH_CONFIRMATION messages.

4.18.6 Directed Forwarding Update

This subsection verifies the Directed Forwarding Update procedure.

4.18.6.1 Node Dependence

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node sends a DEPENDENT_NODE_UPDATE message when it establishes node dependence.

- Reference

  [10] 3.6.5.1.16, 3.6.8.2.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.66 apply.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9 and a two-way path from _PO_ADDR_ by running preamble 4.2.11. Lower Tester 1 stores the Path_Origin_Forwarding_Number values as _POFN1_ and _POFN2_, respectively.
  - Lower Tester 2 simulates two elements.

- Test Procedure

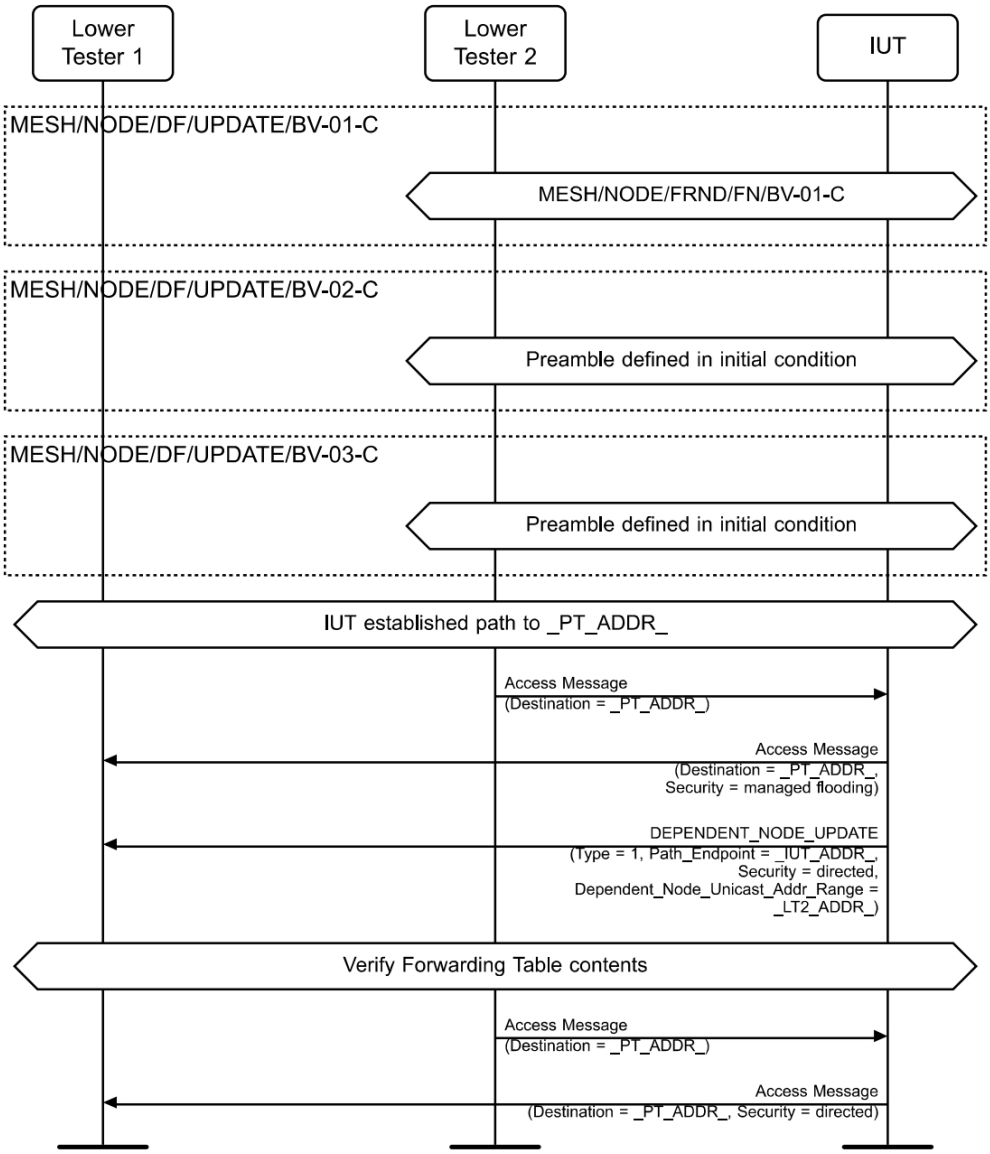


Figure 4.244: Node Dependence MSC

1. Lower Tester 2 sends an access message to _PT_ADDR_ with SRC set to _LT2_ADDR_.
2. The IUT sends the access message to _PT_ADDR_ and the DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 1, Path_Endpoint set to _IUT_ADDR_, and Dependent_Node_Unicast_Addr_Range set to the value from Table 4.66.
3. Lower Tester 1 verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.65 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_POFN1_
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Dependent_Origin_List0	{Value from Table 4.66}
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001
Non_Fixed_Path #1	
Backward_Path_Validated1	0b1
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_POFN2_
Path_Origin_Unicast_Addr_Range1	_PO_UAR_
Bearer_Toward_Path_Origin1	0x0001
Path_Target_Unicast_Addr_Range1	_IUT_UAR_
Dependent_Target_List1	{Value from Table 4.66}

Table 4.65: Forwarding Table contents for Node Dependence

4. Lower Tester 2 sends an access message to _PT_ADDR_.
5. The IUT sends the access message to _PT_ADDR_.

- Test Case Configuration

TCID	Additional Initial Condition	Dependent Unicast Address Range
MESH/NODE/DF/UPDATE/BV-01-C [Node Dependence, IUT as Directed Friend]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.8.5.3.1. Lower Tester 2 uses only friendship security credentials to send access messages.	_LT2_UAR_

TCID	Additional Initial Condition	Dependent Unicast Address Range
MESH/NODE/DF/UPDATE/BV-02-C [Node Dependence with Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 uses only managed flooding security credentials to send access messages.	_LT2_UAR_
MESH/NODE/DF/UPDATE/BV-03-C [Node Dependence with Directed Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	_LT2_UAR_
MESH/NODE/DF/UPDATE/BV-14-C [Node Dependence, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). The IUT is configured with the bridging table entries (0x01, 0, 1, _LT2_ADDR_, _PT_ADDR_) and (0x02, 1, 0, _PO_ADDR_, _LT2_ADDR_) by running preamble 4.2.7. Lower Tester 1 is only part of the first subnet (index 0), and Lower Tester 2 is only part of the second subnet (index 1). Directed forwarding is enabled on the IUT on the first subnet (index 0).	_LT2_ADDR_

Table 4.66: Additional test conditions for Node Dependence

- Expected Outcome

  Pass verdict

  The IUT sends a DEPENDENT_NODE_UPDATE message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  In Step 2, the IUT sends the access message secured with managed flooding security credentials.

  In Step 5, the IUT sends the access message secured with directed security credentials on the primary subnet.

  The IUT reports a Forwarding Table with the expected values.

4.18.6.2 Remove Node Dependence

- Test Purpose

  Verify that an IUT sends a DEPENDENT_NODE_UPDATE message when it loses a dependence relationship.

- Reference

  [10] 3.6.5.1.16, 3.6.8.2.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.68 apply.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9 and a two-way path from _PO_ADDR_ by running preamble 4.2.11. The Lower Tester stores the Path_Origin_Forwarding_Number values as _POFN1_ and _POFN2_, respectively.
  - Lower Tester 2 simulates two elements.

- Test Procedure

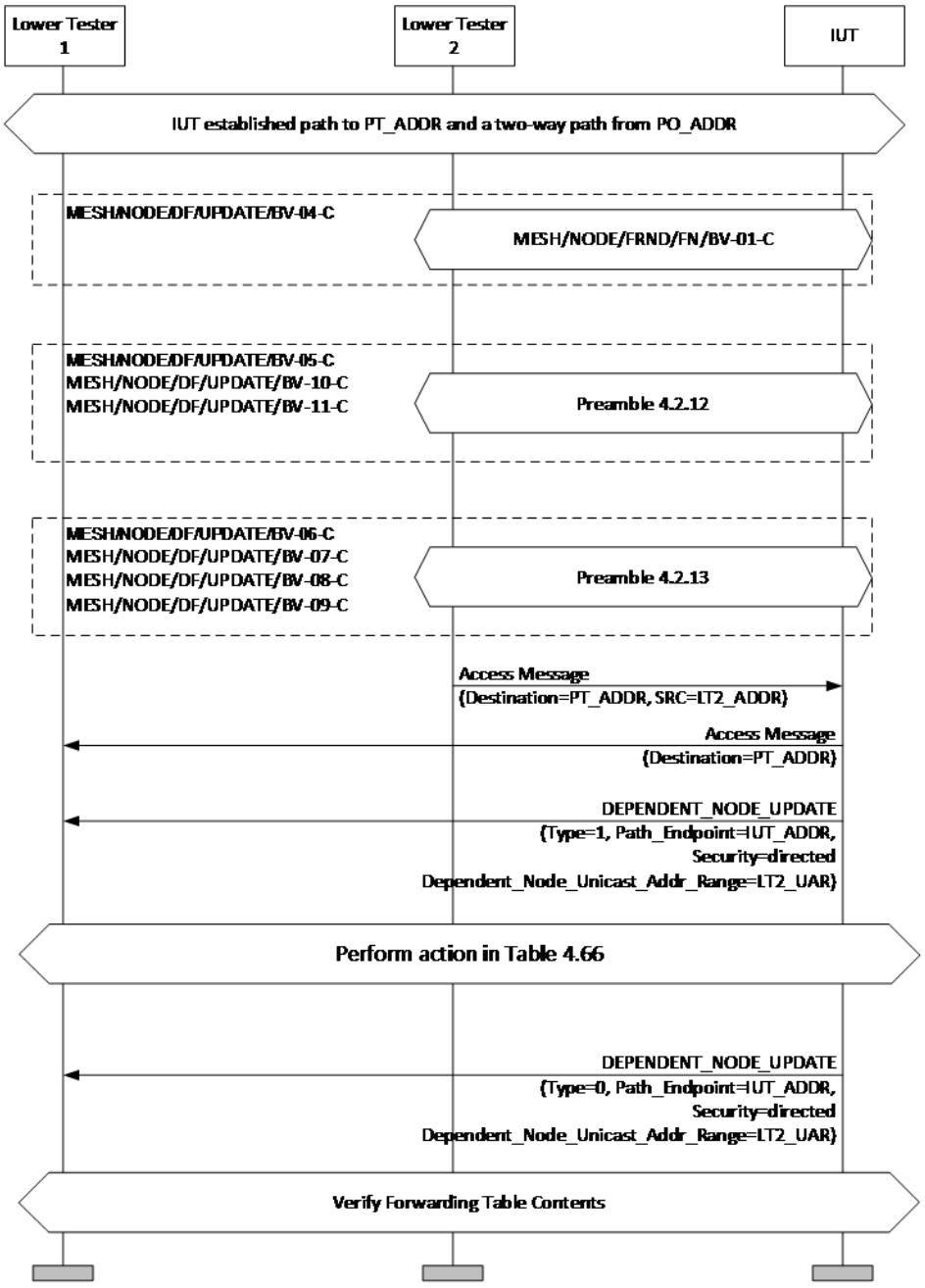


Figure 4.245: Remove Node Dependence MSC

1. Lower Tester 2 sends an access message to _PT_ADDR_ with SRC set to _LT2_ADDR_.
2. The IUT sends the access message to _PT_ADDR_ and the DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 1, Path_Endpoint set to _IUT_ADDR_, Dependent_Node_Unicast_Addr_Range set to _LT2_UAR_ if Lower Tester 2 is a Proxy Client, otherwise set to _LT2_UAR_.
3. Execute the instructions listed in Table 4.68.
4. The IUT sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 0, Path_Endpoint and Dependent_Node_Unicast_Addr_Range set to the

values from Step 2.

5. Lower Tester 1 verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.67 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0002
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_POFN1_
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Bearer_Toward_Path_Target0	0x0001
Non_Fixed_Path #1	
Backward_Path_Validated1	0b1
Lane_Counter1	0x01
Path_Origin_Forwarding_Number1	_POFN2_
Path_Origin_Unicast_Addr_Range1	_PO_UAR_
Bearer_Toward_Path_Origin1	0x0001
Path_Target_Unicast_Addr_Range1	_IUT_UAR_

Table 4.67: Forwarding Table contents for Remove Node Dependence

- Test Case Configuration

TCID	Additional Initial Condition	Step 3
MESH/NODE/DF/UPDATE/BV-04-C [Remove Node Dependence, IUT as Directed Friend]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT have previously established a friendship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node]. The Friendship security credentials are calculated as found in [3] Section 3.8.5.3.1. Lower Tester 2 uses only friendship security credentials to send access messages.	The directed friend functionality on the primary subnet is disabled on the IUT.

TCID	Additional Initial Condition	Step 3
MESH/NODE/DF/UPDATE/BV-05-C [Remove Node Dependence with Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 uses only managed flooding security credentials to send access messages.	The directed proxy functionality on the primary subnet is disabled on the IUT.
MESH/NODE/DF/UPDATE/BV-06-C [Remove Node Dependence with Directed Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	The directed proxy functionality on the primary subnet is disabled on the IUT.
MESH/NODE/DF/UPDATE/BV-07-C [Remove Node Dependence with Directed Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message to the IUT with NetKeyIndex field set to 0x0000, Use_Directed set to 0x00 (Disabled).
MESH/NODE/DF/UPDATE/BV-08-C [Remove Node Dependence with disconnected Directed Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	Lower Tester 2 disconnects.
MESH/NODE/DF/UPDATE/BV-09-C [Remove Node Dependence with Directed Proxy Client on filter type change, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13. Lower Tester 2 uses only managed flooding security credentials to send access messages.	Lower Tester 2 sends a Set Filter Type message to the IUT with FilterType field set to 0x01 (Reject List Filter) and receives a Filter Status message with FilterType set to 0x01 (Reject List) and ListSize set to 0x0000.

TCID	Additional Initial Condition	Step 3
MESH/NODE/DF/UPDATE/BV-10-C [Remove Node Dependence with disconnected Proxy Client, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 uses only managed flooding security credentials to send access messages.	Lower Tester 2 disconnects.
MESH/NODE/DF/UPDATE/BV-11-C [Remove Node Dependence with Proxy Client on filter type change, IUT as Directed Proxy]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12. Lower Tester 2 uses only managed flooding security credentials to send access messages.	Lower Tester 2 sends a Set Filter Type message to the IUT with FilterType field set to 0x01 (Reject List Filter) and receives a Filter Status message with FilterType set to 0x01 (Reject List) and ListSize set to 0x0000.
MESH/NODE/DF/UPDATE/BV-15-C [Remove Node Dependence, IUT as Subnet Bridge]	The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet). The IUT is configured with the bridging table entries (0x01, 0, 1, _LT2_ADDR_, _PT_ADDR_) and (0x02, 0, 1, _PO_ADDR_, _LT2_ADDR_) by running preamble 4.2.7. Lower Tester 1 is only part of the first subnet (index 0), and Lower Tester 2 is only part of the second subnet (index 1). Directed forwarding is enabled on the IUT on the first subnet (index 0).	Lower Tester 2 disables the subnet bridge functionality on the IUT.

Table 4.68: Additional test conditions for Remove Node Dependence

- Expected Outcome

  Pass verdict

  The IUT sends a DEPENDENT_NODE_UPDATE message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT sends the access messages secured with managed flooding security credentials.

The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/UPDATE/BV-12-C [Receive DEPENDENT_NODE_UPDATE, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT acting as Directed Relay node forwards a DEPENDENT_NODE_UPDATE message when there is a matching path stored in the Forwarding Table.

- Reference

  [10] 3.6.5.1.16, 3.6.8.2.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a two-way path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10. Lower Tester 1 stores the Path_Origin_Forwarding_Number as _FN_.
  - _DO_ADDR_ and _DT_ADDR_ are valid unicast addresses, not matching any other unicast addresses used.

- Test Procedure

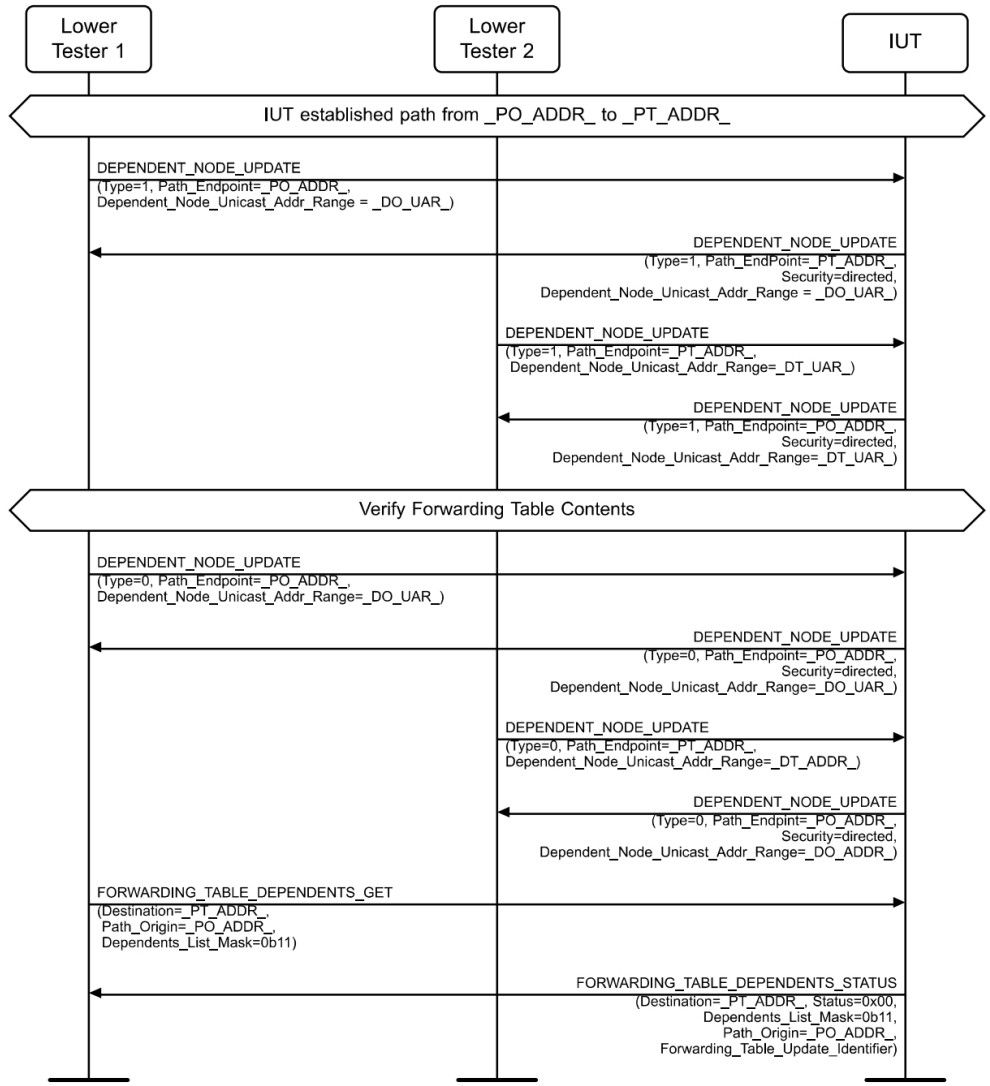


Figure 4.246: MESH/NODE/DF/UPDATE/BV-12-C MSC

1. Lower Tester 1 sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 1, Path_Endpoint set to _PO_ADDR_, and Dependent_Node_Unicast_Addr_Range set to _DO_UAR_.
2. The IUT sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with the same parameters from Step 1.
3. Lower Tester 2 sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 1, Path_Endpoint set to _PT_ADDR_, and Dependent_Node_Unicast_Addr_Range set to _DT_UAR_.
4. The IUT sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with the same parameters from Step 3.
5. Lower Tester 1 verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.69 by running the test procedure from Section 4.18.1.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b1
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	_FN_
Path_Origin_Unicast_Addr_Range0	_PO_UAR_
Dependent_Origin_List0	{_DO_ADDR_}
Bearer_Toward_Path_Origin0	0x0001
Path_Target_Unicast_Addr_Range0	_PT_UAR_
Dependent_Target_List0	{_DT_ADDR_}
Bearer_Toward_Path_Target0	0x0001

Table 4.69: Forwarding Table contents for MESH/NODE/DF/UDPATE/BV-12-C

6. Lower Tester 1 sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 0, Path_Endpoint set to _PO_ADDR_, and Dependent_Node_Unicast_Addr_Range set to _DO_ADDR_.
7. The IUT sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with the same parameters from Step 6.
8. Lower Tester 2 sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 0, Path_Endpoint set to _PT_ADDR_, and Dependent_Node_Unicast_Addr_Range set to _DT_ADDR_.
9. The IUT sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with the same parameters from Step 8.
10. Lower Tester 1 sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, and Destination set to _PT_ADDR_.
11. The IUT sends a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to a valid value, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x00, and Dependent_Target_Unicast_Addr_Range_List_Size set to 0x00.

- Expected Outcome

  Pass verdict

  The IUT sends a DEPENDENT_NODE_UPDATE message with the expected fields and TTL set to 0. The message is secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

  In Step 11, the IUT does not report any dependent nodes for the stored path from the Forwarding Table.

MESH/NODE/DF/UPDATE/BV-13-C [Ignore DEPENDENT_NODE_UPDATE]

- Test Purpose

  Verify that an IUT acting as Directed Relay node ignores a DEPENDENT_NODE_UPDATE message.

- Reference

  [10] 3.6.5.1.16, 3.6.8.2.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a two-way path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10.
  - _DT_ADDR_ is a valid unicast address, not matching any other unicast addresses used.

- Test Procedure

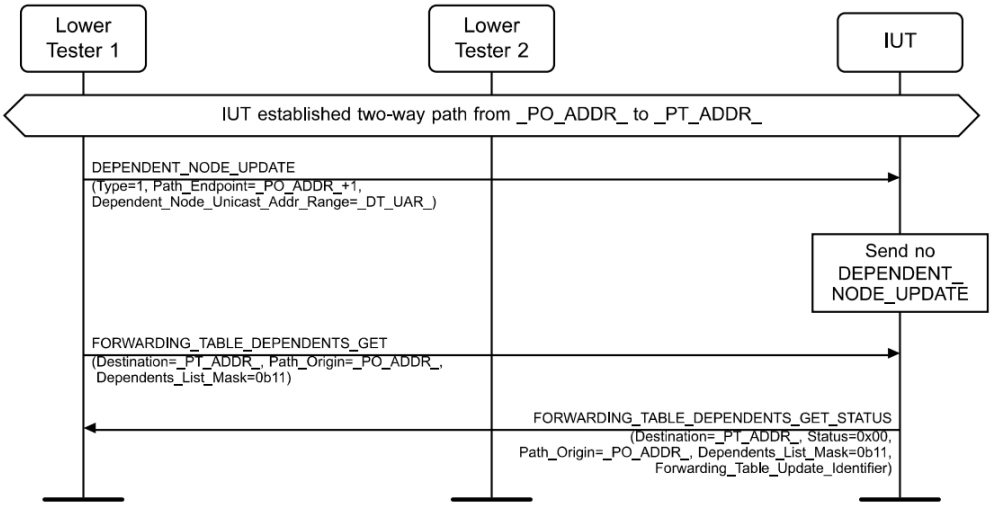


Figure 4.247: MESH/NODE/DF/UPDATE/BV-13-C MSC

1. Lower Tester 1 sends a DEPENDENT_NODE_UPDATE message to the all-directed-forwarding-nodes address with Type set to 1, Path_Endpoint set to _PO_ADDR_ + 1, and Dependent_Node_Unicast_Addr_Range set to _DT_UAR_.
2. Lower Tester 1 does not expect any DEPENDENT_NODE_UPDATE messages from the IUT.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Fixed_Path_Flag set to 0b0, Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, and Destination set to _PT_ADDR_.
4. The IUT sends a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Fixed_Path_Flag set to 0b0, Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to a valid value,

Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x00, and
Dependent_Target_Unicast_Addr_Range_List_Size set to 0x00.

- Expected Outcome

  Pass verdict

  The IUT does not send any DEPENDENT_NODE_UPDATE messages.

  The IUT reports no dependent nodes in the stored path from the Forwarding Table.

MESH/NODE/DF/UPDATE/BI-01-C [Receive DEPENDENT_NODE_UPDATE, Invalid Parameters]

- Test Purpose

  Verify that a Directed Forwarding node IUT ignores a DEPENDENT_NODE_UPDATE with invalid parameters.

- Reference

  [10] 3.6.5.1.16, 3.6.8.2.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.
  - The IUT has established a two-way path from _PO_ADDR_ to _PT_ADDR_ by running preamble 4.2.10.
  - _DO_ADDR_ is a valid unicast address, not matching any other unicast addresses used.

- Test Procedure

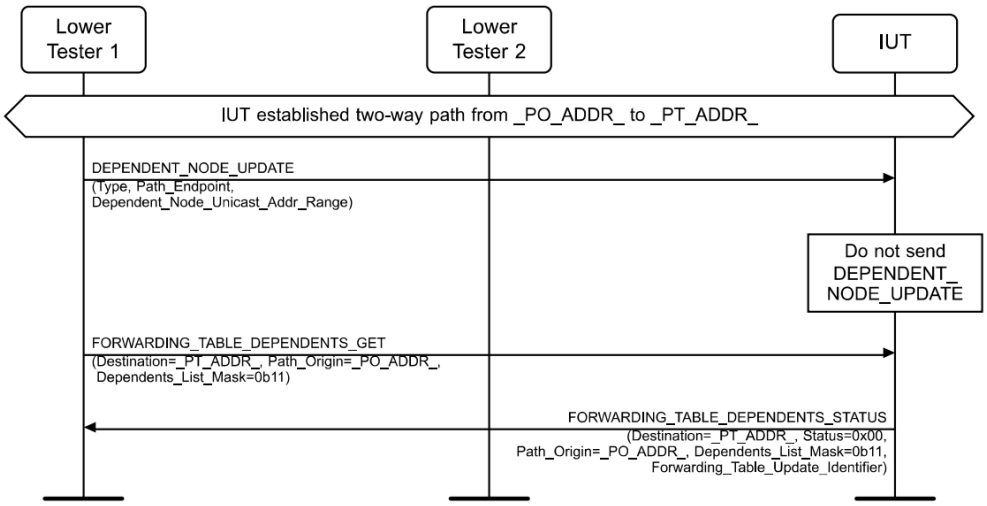


Figure 4.248: MESH/NODE/DF/UPDATE/BI-01-C MSC

For each round in Table 4.70:

1. Lower Tester 1 sends a DEPENDENT_NODE_UPDATE message with the following parameters, unless stated in Table 4.70: DST set to the all-directed-forwarding-nodes address, Type set to 1, Path_Endpoint set to _PO_ADDR_, and Dependent_Node_Unicast_Addr_Range set to _DO_UAR_.
2. Lower Tester 1 does not expect any DEPENDENT_NODE_UPDATE messages from the IUT.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Fixed_Path_Flag set to 0b0, Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, and Destination set to _PT_ADDR_.
4. The IUT sends a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11 (All lists), Fixed_Path_Flag set to 0b0, Start_Index set to 0x0000, Path_Origin set to the _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to a valid value, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x00, and Dependent_Target_Unicast_Addr_Range_List_Size set to 0x00.

Round	Additional Condition
1	Type set to 0b0
2	Dependent_Node_Unicast_Addr_Range set to 0xFFFF00
3	DST set to _IUT_ADDR_
4	TTL set to 0x02
5	Message sent with managed flooding security material

Table 4.70: Specification of rounds for MESH/NODE/DF/UPDATE/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any DEPENDENT_NODE_UPDATE messages.

The IUT does not report any dependent node addresses from the Forwarding Table.

4.18.7 Directed Forwarding Echo

This subsection verifies the Directed Forwarding Echo procedure.

MESH/NODE/DF/ECHO/BV-01-C [Path Echo, IUT as Path Origin]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node successfully completes a Directed Forwarding Echo procedure for an established path each Path Echo Request Lifetime interval.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Unicast Echo Interval state is set to 0x01. Multicast Echo Interval state is set to 0x00. The Path Echo Interval is calculated at 7.2 seconds.
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9.

- Test Procedure

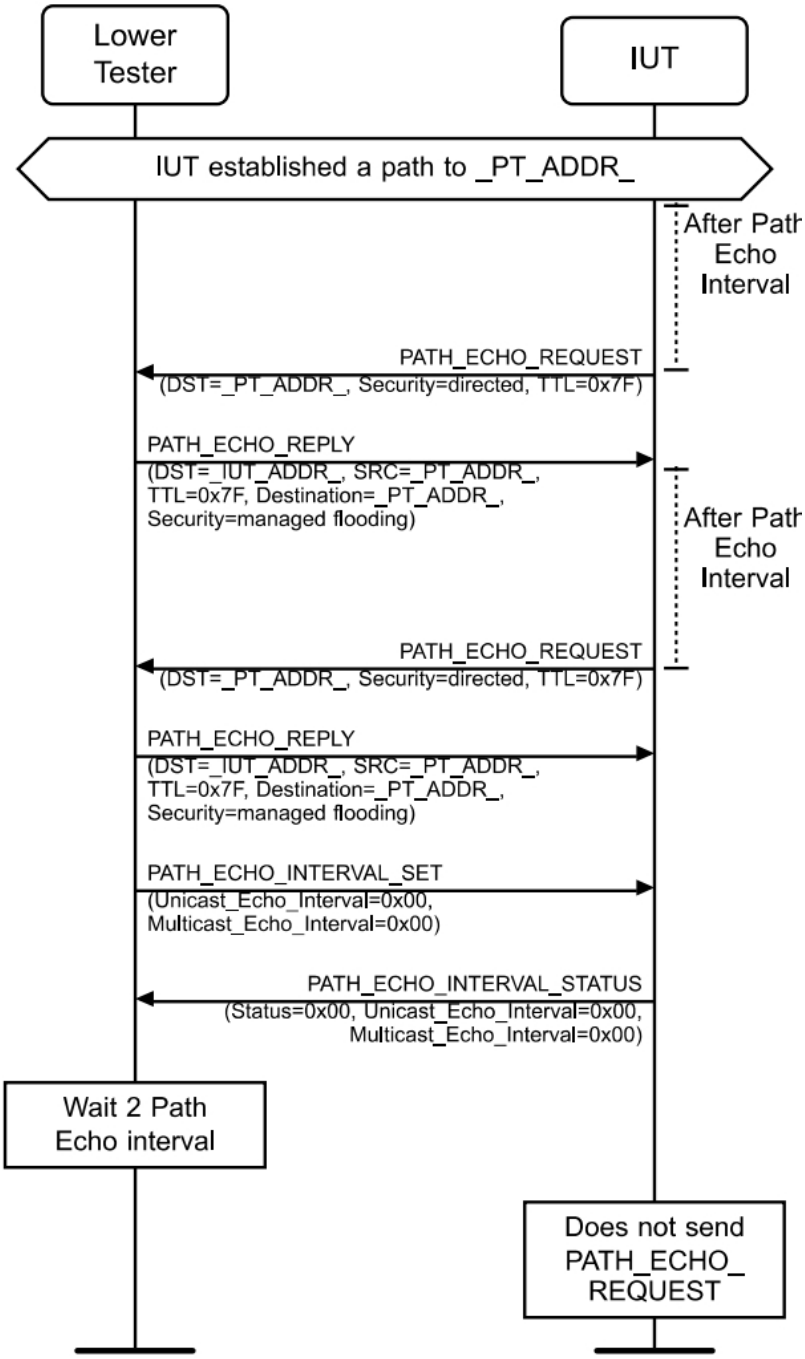


Figure 4.249: MESH/NODE/DF/ECHO/BV-01-C MSC

1. After the Path Echo interval, The IUT sends a PATH_ECHO_REQUEST message to _PT_ADDR_ with TTL set to 0x7F.
2. The Lower Tester sends a PATH_ECHO_REPLY message secured with managed flooding security credentials with SRC set to _PT_ADDR_, DST set to _IUT_ADDR_, TTL set to 0x7F, and Destination set to _PT_ADDR_.
3. The IUT does not send any PATH_ECHO_REPLY messages.
4. After another Path Echo interval, repeat Steps 1–3.

5. The Lower Tester sends a PATH_ECHO_INTERVAL_SET message with NetKeyIndex field set to 0x0000, Unicast_Echo_Interval field set to 0x00, and Multicast_Echo_Interval field set to 0x00.
6. The IUT responds with a PATH_ECHO_INTERVAL_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Unicast_Echo_Interval field set to 0x00, and Multicast_Echo_Interval field set to 0x00.
7. The Lower Tester waits for two Path Echo intervals and does not expect any PATH_ECHO_REQUEST messages from the IUT.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_ECHO_REQUEST message with the expected fields. The message is secured with directed security credentials on the primary subnet.

  The IUT does not send any PATH_ECHO_REQUEST messages after the Unicast Echo Interval state is set to 0x00.

  The IUT indicates the status of the path echo to the Upper Tester.

MESH/NODE/DF/ECHO/BV-02-C [Path Echo Failure, IUT as Path Origin]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node starts the Directed Forwarding Initialization procedure after failing a Directed Forwarding Echo procedure for an established path.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Path Monitoring test mode is activated on the IUT (via HW or SW interface). Unicast Echo Interval state is set to 0x01. Multicast Echo Interval state is set to 0x00. The Path Echo interval is calculated at 7.2 seconds.
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9.

- Test Procedure

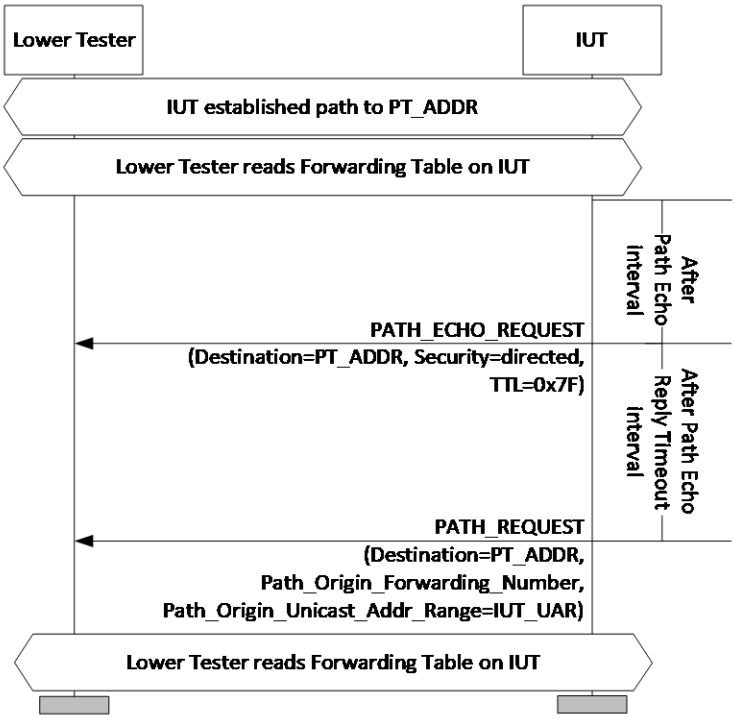


Figure 4.250: MESH/NODE/DF/ECHO/BV-02-C MSC

1. The Lower Tester reads the Forwarding Table on the IUT by executing the procedure described in Section 4.18.
2. After the Path Echo interval, the IUT sends a PATH_ECHO_REQUEST message to _PT_ADDR_ with TTL set to 0x7F.
3. After a Path Echo Reply Timeout interval, the IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
4. The IUT does not send any further PATH_ECHO_REQUEST messages.
5. Repeat Step 1 to confirm that the corresponding Forwarding Table entry has been deleted and that the Forwarding Table Update Identifier state has been changed.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_ECHO_REQUEST message with the expected fields, followed by a PATH_REQUEST message after the Path Echo Reply Timeout expires. The messages are secured with directed security credentials on the primary subnet.

  The IUT deletes the Forwarding Table entry corresponding to the failed Directed Forwarding Echo procedure and updates the Forwarding Table Update Identifier.

4.18.7.1 Path Echo, IUT as Path Target

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node replies with a PATH_ECHO_REPLY message when it receives a PATH_ECHO_REQUEST message where the path target is a unicast address matching one of its elements.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.71 apply.

- Test Procedure

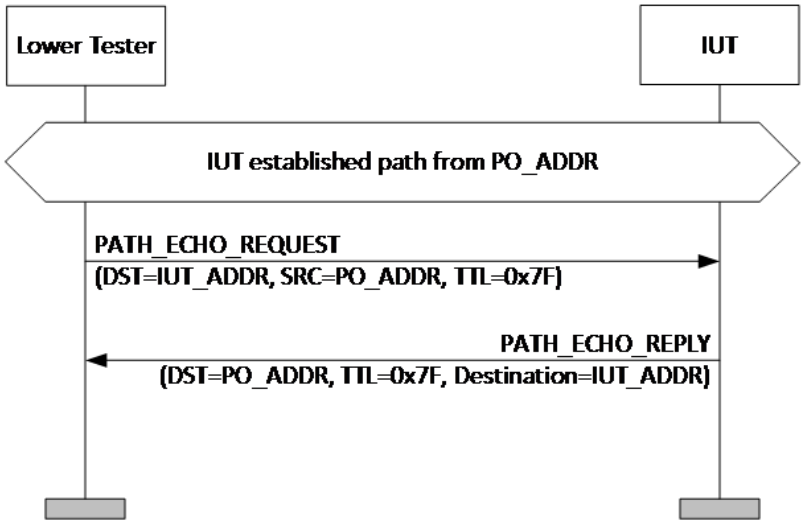


Figure 4.251: Path Echo, IUT as Path Target MSC

1. The Lower Tester sends a PATH_ECHO_REQUEST message with SRC set to _PO_ADDR_, DST set to _IUT_ADDR_, and TTL set to 0x7F.
2. The IUT sends a PATH_ECHO_REPLY message with DST set to _PO_ADDR_, TTL set to 0x7F, and Destination set to _IUT_ADDR_.

- Test Case Configuration

TCID	Additional Initial Condition	Additional Pass Verdict
MESH/NODE/DF/ECHO/BV-03-C [Path Echo, IUT as Path Target]	The Two Way Path state on the primary subnet is set to 0b0 on the IUT. The IUT has established a path from _PO_ADDR_ by running preamble 4.2.11.	The message is secured with managed flooding security credentials.
MESH/NODE/DF/ECHO/BV-04-C [Two-Way Path Echo, IUT as Path Target]	The Two Way Path state on the primary subnet is set to 0b1 on the IUT. The IUT has established a two-way path from _PO_ADDR_ by running preamble 4.2.11.	The message is secured with directed security credentials.

Table 4.71: Additional test conditions for Path Echo, IUT as Path Target

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_ECHO_REPLY message with the expected fields. The message is secured with the security credentials specified in Table 4.71 on the primary subnet.

  The IUT does not send any PATH_ECHO_REQUEST messages.

MESH/NODE/DF/ECHO/BV-05-C [Ignore Path Echo Messages]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node ignores a PATH_ECHO_REQUEST when no corresponding entry exists in the Forwarding Table and ignores a PATH_ECHO_REPLY message when there is no echo procedure ongoing, but a corresponding entry exists in the Forwarding Table.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT has established a path to _PT_ADDR_ by running preamble 4.2.9.
- Test Procedure

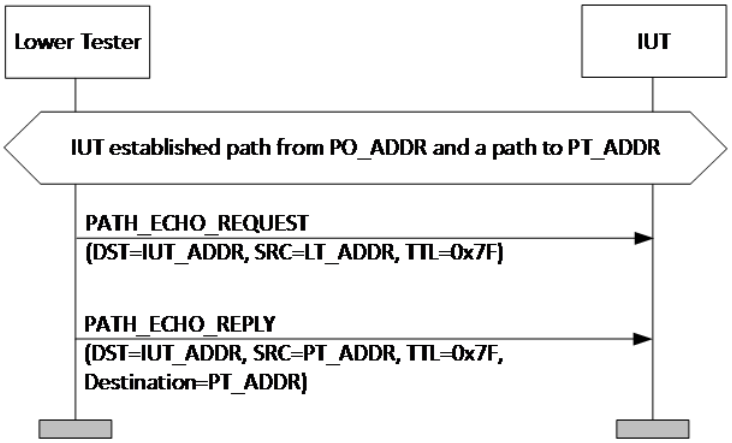


Figure 4.252: MESH/NODE/DF/ECHO/BV-05-C MSC

1. The Lower Tester sends a PATH_ECHO_REQUEST message with SRC set to _LT_ADDR_, DST set to _IUT_ADDR_, and TTL set to 0x7F.
2. The IUT does not send any PATH_ECHO_REPLY messages.
3. The Lower Tester sends a PATH_ECHO_REPLY message with SRC set to _PT_ADDR_, DST set to _IUT_ADDR_, TTL set to 0x7F, and Destination set to _PT_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_ECHO_REPLY messages.

MESH/NODE/DF/ECHO/BV-06-C [Path Echo for Group/Virtual Destination, IUT as Path Origin]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node successfully completes a Directed Forwarding Echo procedure for an established path towards a multicast destination.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Path Monitoring test mode is activated on the IUT (via HW or SW interface). Multicast Echo Interval state set to 0x01. The Path Echo Interval is calculated at 7.2 seconds.

- Test Procedure

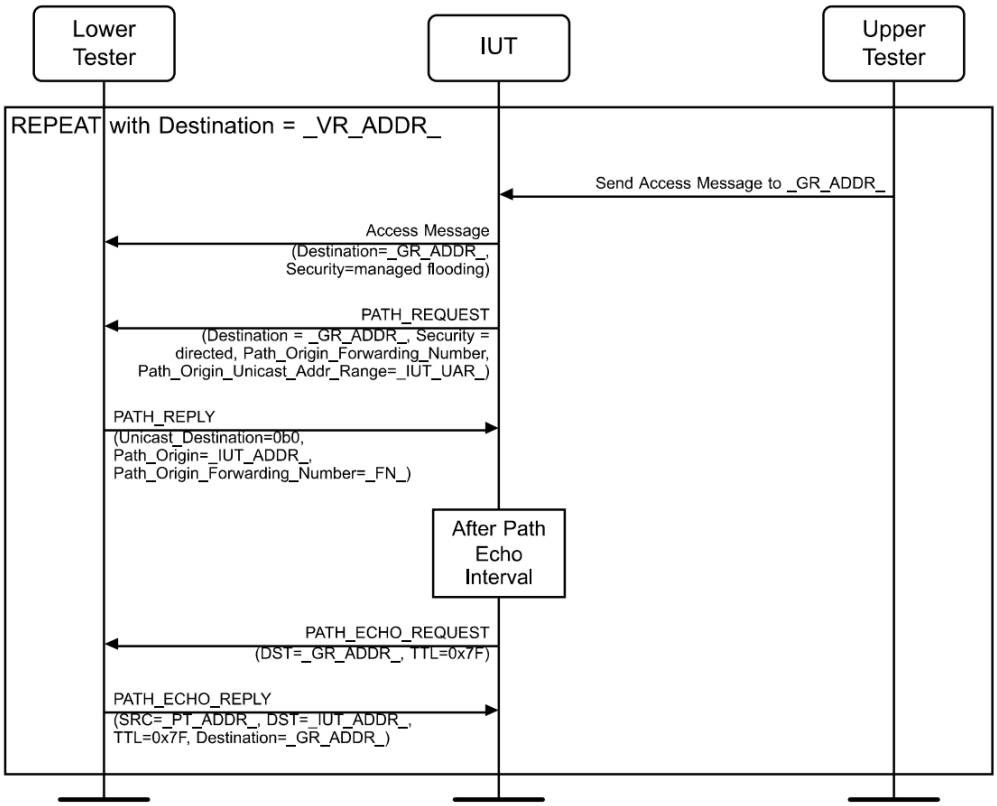


Figure 4.253: MESH/NODE/DF/ECHO/BV-06-C MSC

1. The Upper Tester commands the IUT to send an access message to _GR_ADDR_.
2. The IUT sends an access message to _GR_ADDR_ and a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _GR_ADDR_ , and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_. The Lower Tester stores the Path_Origin_Forwarding_Number as _FN_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target set to 0b0, Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, and Path_Origin_Forwarding_Number set to _FN_.
4. After the Path Echo interval, the IUT sends a PATH_ECHO_REQUEST message to _GR_ADDR_ with TTL set to 0x7F.
5. The Lower Tester sends a PATH_ECHO_REPLY message secured with managed flooding security credentials with SRC set to _PT_ADDR_, DST set to _IUT_ADDR_, TTL set to 0x7F, and Destination set to _GR_ADDR_.
6. Repeat Steps 1–5 replacing _GR_ADDR_ with _VR_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_ECHO_REQUEST message with the expected fields. The message is secured with

directed security credentials on the primary subnet.

The IUT does not send any PATH_ECHO_REPLY messages.

The IUT indicates the status of the path echo to the Upper Tester.

MESH/NODE/DF/ECHO/BI-01-C [Ignore Path Echo Messages – Invalid Parameters]

- Test Purpose

  Verify that an IUT ignores PATH_ECHO_REQUEST and PATH_ECHO_REPLY messages with invalid parameters.

- Reference

  [10] 3.6.5.14, 3.6.5.15, 3.6.8.2.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.
  - The IUT has established a two-way path from _PO_ADDR_ by running preamble 4.2.11.

- Test Procedure

  Repeat for each round in Table 4.72.

  1. The Lower Tester sends a PATH_ECHO_REQUEST message with the following parameters, unless stated in Table 4.72.
  2. : SRC set to _PO_ADDR_, DST set to _IUT_ADDR_, and TTL set to 0x7F.
  3. The IUT does not send a PATH_ECHO_REPLY message.

Round	Additional Condition
1	TTL set to 0x00
2	Message sent with managed flooding security material

Table 4.72: Specification of rounds for MESH/NODE/DF/ECHO/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REPLY messages.

4.18.7.2 Ignore Path Echo Reply - Incorrect Security Material

- Test Purpose

  Verify that an IUT ignores a PATH_ECHO_REPLY message sent with incorrect security material.

- Reference

[10] 3.6.5.15, 3.6.8.5.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Unicast Echo Interval is set to 0x01. The Multicast Echo Interval is set to 0x00.
  - Additional initial conditions defined in Table 4.73 apply.
- Test Case Configuration

TCID	Additional Initial Condition	Security Material
MESH/NODE/DF/ECHO/BI-02-C [Ignore Path Echo Reply – One-way Path – Incorrect Security Material]	The Two Way Path state on the primary subnet is set to 0b0 on the IUT. The IUT has established a path from _IUT_ADDR_ by running preamble 4.2.9.	Invalid directed security material
MESH/NODE/DF/ECHO/BI-03-C [Ignore Path Echo Reply – Two-way Path – Incorrect Security Material]	The Two Way Path state on the primary subnet is set to 0b1 on the IUT. The IUT has established a two-way path from _IUT_ADDR_ by running preamble 4.2.9.	Invalid managed flooding security material

Table 4.73: Additional test conditions for Ignore Path Echo Reply - Incorrect Security Material

- Test Procedure

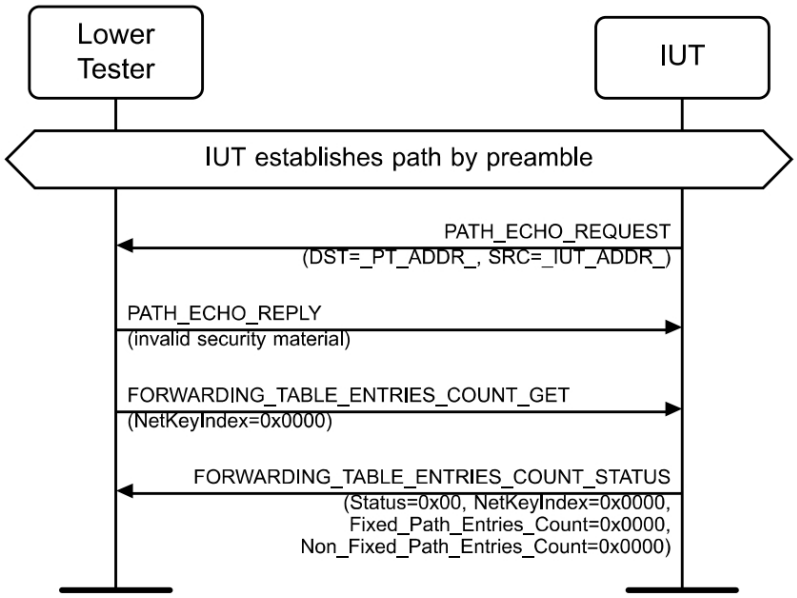


Figure 4.254: Test Procedure - Ignore Path Echo Reply - Incorrect Security Material MSC

1. The IUT sends a PATH_ECHO_REQUEST message to _PT_ADDR_.
2. The Lower Tester sends a PATH_ECHO_REPLY message secured with the security material specified in Table 4.73 with SRC set to _PT_ADDR_, DST set to _IUT_ADDR_, and Destination set to

_PT_ADDR_.

3. After the Path Echo Reply Timeout expires, the Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
4. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000.

- Expected Outcome

  Pass verdict

  The IUT ignores the PATH_ECHO_REPLY message. After the Path Echo Reply Timeout expires, the IUT deletes the corresponding Forwarding Table entry. The Forwarding_Table_Update_Identifier value is changed to a higher value.

MESH/NODE/DF/ECHO/BI-04-C [Ignore Path Echo Reply – Invalid Destination]

- Test Purpose

  Verify that an IUT ignores a PATH_ECHO_REPLY message sent with invalid Destination values.

- Reference

  [10] 3.6.5.15, 3.6.8.5.7

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Unicast Echo Interval is set to 0x01. The Multicast Echo Interval is set to 0x00.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.
- Test Procedure

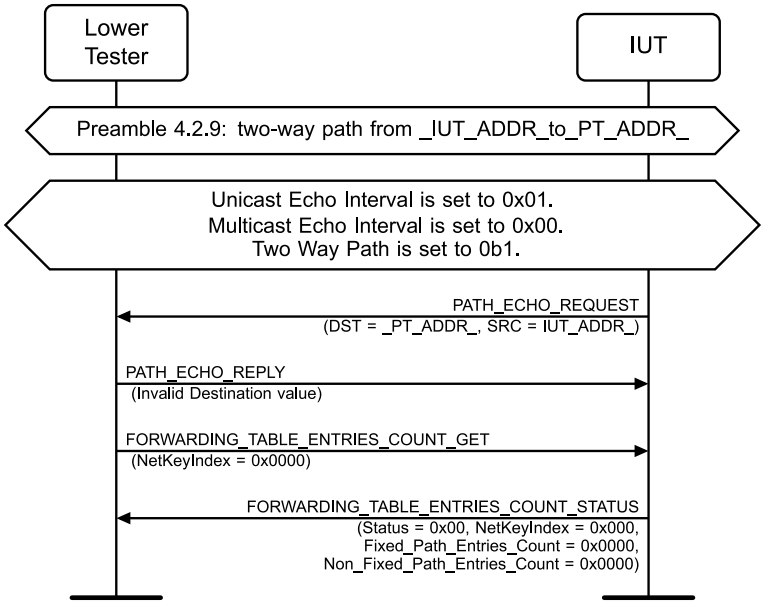


Figure 4.255: Test Procedure - Ignore Path Echo Reply - Invalid Destination MSC

Repeat for each round in Table 4.74.

1. Run preamble 4.2.9 to establish a two-way path on the IUT from _IUT_ADDR_ to _PT_ADDR_.
2. The IUT sends a PATH_ECHO_REQUEST message to _PT_ADDR_.
3. The Lower Tester sends a PATH_ECHO_REPLY message secured with valid security material specified in Table 4.74 with SRC set to _PT_ADDR_, DST set to _IUT_ADDR_, and Destination set to the value in Table 4.74.
4. After the Path Echo Reply Timeout expires, the Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
5. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000.

Round	Value
1	0x0000 (unassigned)
2	0xFFFB (all-directed-forwarding-nodes)
3	0xFFFE (all-relays)
4	0xFFFF (all-nodes)

Table 4.74: Rounds for Ignore Path Echo Reply - Invalid Destination

- Expected Outcome

Pass verdict

The IUT ignores the PATH_ECHO_REPLY message. After the Path Echo Reply Timeout expires, the IUT deletes the corresponding Forwarding Table entry. The Forwarding_Table_Update_Identifier value is

changed to a higher value.

4.18.8 Directed Forwarding Solicitation

This subsection verifies sending and receiving Directed Forwarding Solicitation messages.

MESH/NODE/DF/SOL/BV-01-C [Send PATH_REQUEST_SOLICITATION, IUT as Path Target]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node sends a PATH_REQUEST_SOLICITATION message when the Subscription List state changes.

- Reference

  [10] 3.6.5.17, 3.6.8.2.7

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT has established a path from _PO_ADDR_ by running preamble 4.2.11.
  - IXIT [6] provides ModelID that supports subscription mechanism.

- Test Procedure

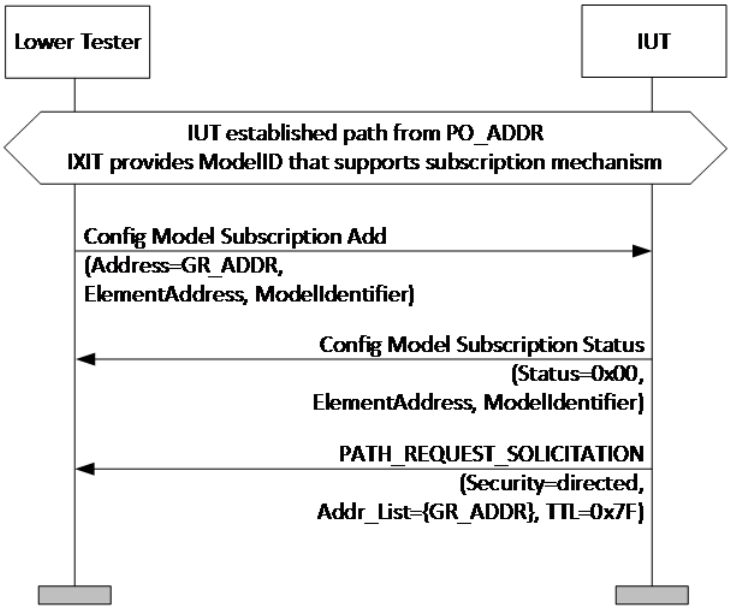


Figure 4.256: MESH/NODE/DF/SOL/BV-01-C MSC

1. The Lower Tester obtains and stores the ElementAddress, and the ModelIdentifier value of a valid model on the IUT from IXIT [6].

2. The Lower Tester sends a Config Model Subscription Add message to the IUT with the ElementAddress and ModelIdentifier values from Step 1 and the Address field set to _GR_ADDR_.
3. The IUT responds with a Config Model Subscription Status message with the Status field set to 0x00, the ElementAddress, and the Model Identifier fields set to the values in Step 2.
4. The IUT sends a PATH_REQUEST_SOLICITATION message to the all-directed-forwarding-nodes address with TTL set to 0x7F and Addr_List field set to {_GR_ADDR_}.

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST_SOLICITATION message with the expected fields. The message is secured with directed security credentials on the primary subnet.

4.18.8.1 Send PATH_REQUEST_SOLICITATION for Dependent

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node sends a PATH_REQUEST_SOLICITATION message when the Subscription List state changes on the dependent node.

- Reference

  [10] 3.6.5.17, 3.6.8.2.7

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.75 apply.

- Test Procedure

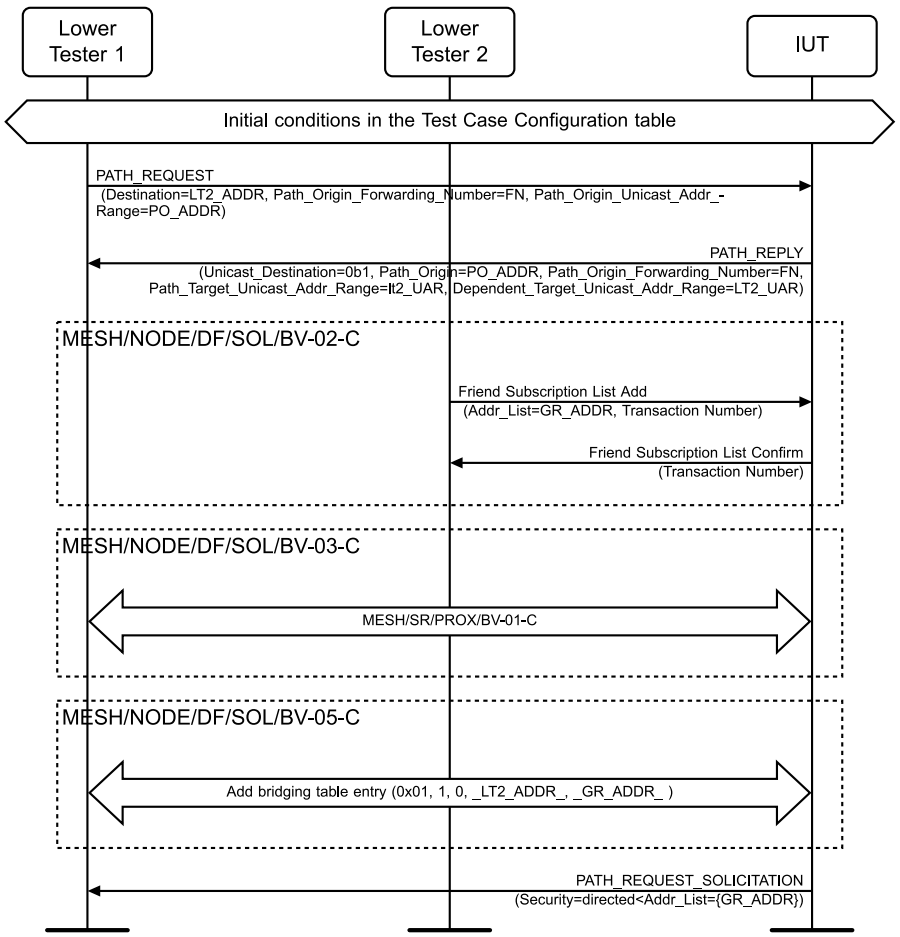


Figure 4.257: Send PATH_REQUEST_SOLICITATION for Dependent MSC

1. Lower Tester 1 sends a PATH_REQUEST to the IUT to establish a path to Lower Tester 2, and the IUT responds with a PATH_REPLY with correct parameters.
2. Execute the instructions listed in Table 4.75.
3. The IUT sends a PATH_REQUEST_SOLICITATION message to the all-directed-forwarding-nodes address with the Addr_List field set to {_GR_ADDR_}.

- Test Case Configuration

TCID	Additional Initial Condition	Step 3
MESH/NODE/DF/SOL/BV-02-C [Send PATH_REQUEST_SOLICITATION, IUT as Friend Path Target]	The directed friend functionality on the primary subnet is enabled on the IUT. Lower Tester 2 and the IUT establish a friend relationship through test MESH/NODE/FRND/FN/BV-01-C [Friendship Establishment Mode – Friend Node].	Lower Tester 2 sends a Friend Subscription List Add message to the IUT with the Addr_List field value set to _GR_ADDR_ and a random TransactionNumber field value. The IUT responds with a Friend Subscription List Confirm message with a TransactionNumber field equal to that of the value it sent in Step 3.
MESH/NODE/DF/SOL/BV-03-C [Send PATH_REQUEST_SOLICITATION, IUT as Proxy Path Target]	The IUT established a proxy connection with Lower Tester 2 as a Proxy client by running preamble 4.2.12.	Lower Tester 2 adds _GR_ADDR_ to the IUT's accept list by running test case MESH/SR/PROX/BV-01-C [Receive Accept List Filter Setup].
MESH/NODE/DF/SOL/BV-05-C [Send PATH_REQUEST_SOLICITATION, IUT as Subnet Bridge]	The subnet bridge functionality is enabled on the IUT. The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet).	Lower Tester 2 adds the bridging table entry (0x01, 1, 0, _LT2_ADDR_, _GR_ADDR_) on the IUT by running preamble 4.2.7.

Table 4.75: Additional test conditions for Send PATH_REQUEST_SOLICITATION for Dependent

- Expected Outcome

  Pass verdict

  The IUT sends a PATH_REQUEST_SOLICITATION message with the expected fields. The message is secured with directed security credentials on the primary subnet.

MESH/NODE/DF/SOL/BV-04-C [Receive PATH_REQUEST_SOLICITATION, IUT as Path Origin]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node receives a PATH_REQUEST_SOLICITATION message for a path where it acts as a path origin and initiates Directed Forwarding Initialization.

- Reference

  [10] 3.6.5.17, 3.6.8.2.7

- Initial Condition

- Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- The IUT has established a path to _PT1_ADDR_ by running preamble 4.2.9.

• Test Procedure

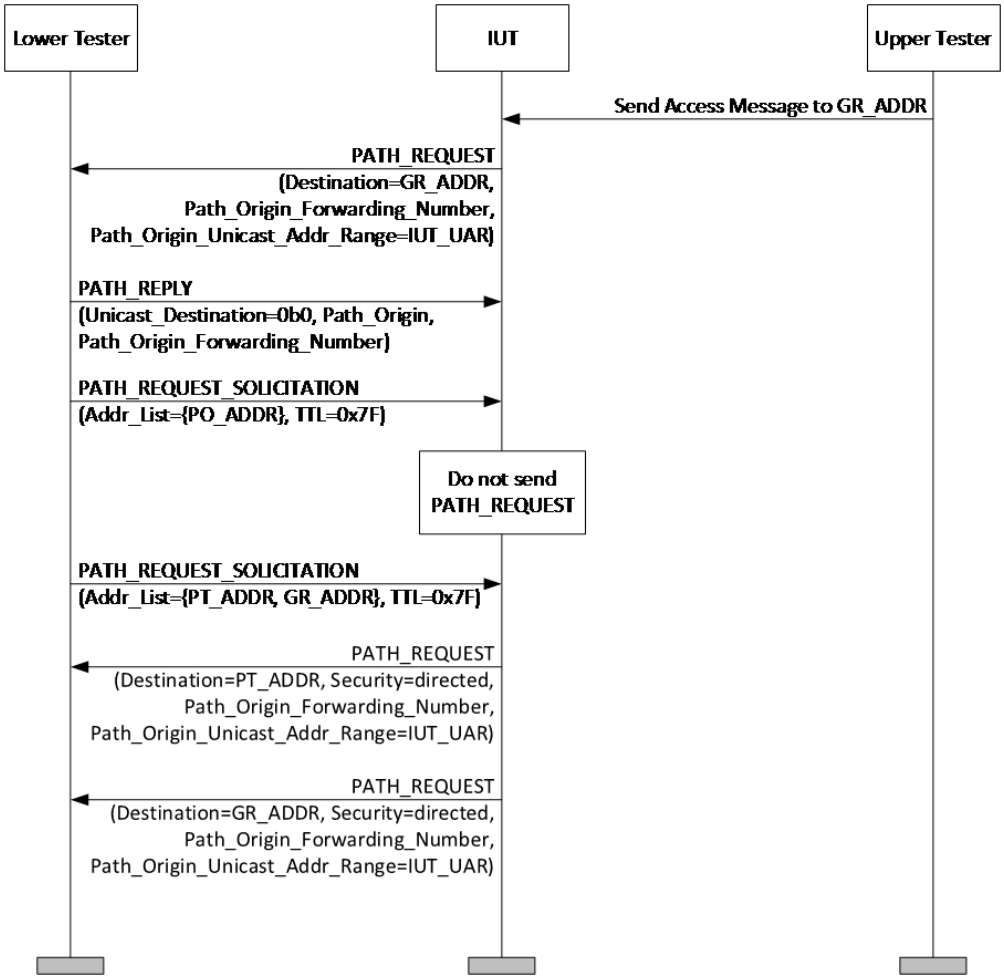


Figure 4.258: MESH/NODE/DF/SOL/BV-04-C MSC

1. The Upper Tester commands the IUT to send an access message to _GR1_ADDR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _GR1_ADDR_ , and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target and Confirmation_Request set to 0b0, Path_Origin set to _IUT_ADDR_, and Path_Origin_Forwarding_Number set to the same value as in Step 2.
4. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b0, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Multicast_Destination set to _GR2_ADDR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.

5. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _GR2_ADDR_.
6. The Lower Tester verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.76 by running the test procedure from Section 4.18.1. The Lower Tester stores the Forwarding_Table_Update_Identifier value.

Field	Value
Non_Fixed_Path_Entries_Count	0x0001
Fixed_Path_Entries_Count	0x0001
Non_Fixed_Path #0	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	Valid value
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Bearer_Toward_Path_Target0	0x0001
Multicast_Destination0	_GR1_UAR_
Non_Fixed_Path #1	
Backward_Path_Validated0	0b0
Lane_Counter0	0x01
Path_Origin_Forwarding_Number0	Valid value
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Bearer_Toward_Path_Target0	0x0001
Path_Origin_Target_Addr_Range0	_PT1_UAR_
Fixed_Path #0	
Backward_Path_Validated0	0b0
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Bearer_Toward_Path_Target0	0x0001
Multicast_Destination0	_GR2_UAR_

Table 4.76: Forwarding Table contents for MESH/NODE/DF/SOL/BV-04-C

7. The Lower Tester sends a PATH_REQUEST_SOLICITATION message to the all-directed-forwarding-nodes address with TTL set to 0x7F and Addr_List set to {_PT1_ADDR_, _PT2_ADDR_, _GR1_ADDR_, _GR2_ADDR_}.
8. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _GR1_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.

9. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _PT1_ADDR_, and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
10. The Lower Tester verifies that the Forwarding Table contents on the primary subnet matches the values from Table 4.77 by running the test procedure from Section 4.18.1. The Forwarding_Table_Update_Identifier value is set to a value other than the stored value from Step 6.

Field	Value
Fixed_Path_Entries_Count	0x0001
Fixed_Path #0	
Backward_Path_Validated0	0b0
Path_Origin_Unicast_Addr_Range0	_IUT_UAR_
Bearer_Toward_Path_Target0	0x0001
Multicast_Destination0	_GR2_UAR_

Table 4.77: Forwarding Table contents for MESH/NODE/DF/SOL/BV-04-C, after receiving the PATH_REQUEST_SOLICITATION message

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REQUEST_SOLICITATION messages.

  The IUT does not send any PATH_REQUEST messages for the following destinations: _PT2_ADDR_, _GR2_ADDR_.

  The IUT sends a PATH_REQUEST messages for the following destinations: _PT1_ADDR_, _GR1_ADDR_. The messages are secured with directed security credentials on the primary subnet.

  The IUT reports the Forwarding Table with the expected values.

MESH/NODE/DF/SOL/BI-01-C [Ignore PATH_REQUEST_SOLICITATION – Invalid Parameters]

- Test Purpose

  Verify that an IUT ignores a PATH_REQUEST_SOLICITATION with invalid parameters.

- Reference

  [10] 3.6.5.13, 3.6.8.2.7

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

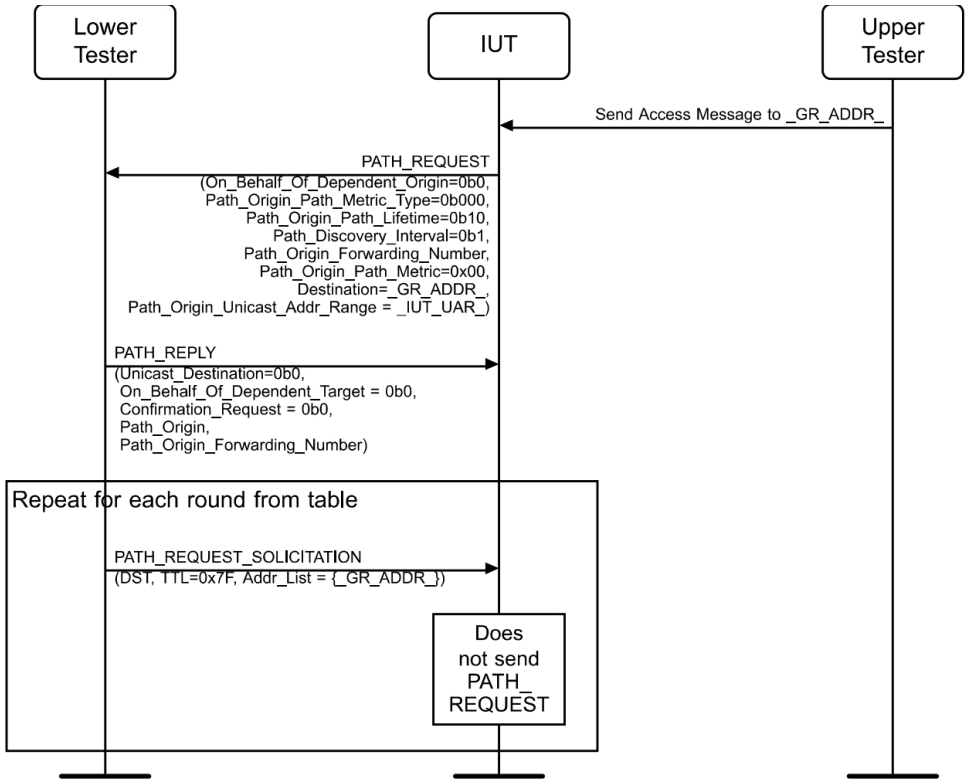


Figure 4.259: MESH/NODE/DF/SOL/BI-01-C MSC

1. The Upper Tester commands the IUT to send an access message to _GR_ADDR_.
2. The IUT sends a PATH_REQUEST message to the all-directed-forwarding-nodes address with On_Behalf_Of_Dependent_Origin set to 0b0, Path_Origin_Path_Metric_Type set to 0b000, Path_Origin_Path_Lifetime set to 0b10, Path_Discovery_Interval set to 0b1, Path_Origin_Forwarding_Number set to a random valid value, Path_Origin_Path_Metric set to 0x00, Destination set to _GR_ADDR_ , and Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
3. The Lower Tester sends a PATH_REPLY message to _IUT_ADDR_ with Unicast_Destination set to 0b0, On_Behalf_Of_Dependent_Target and Confirmation_Request set to 0b0, and Path_Origin and Path_Origin_Forwarding_Number set to the same value as in Step 2.
4. Repeat Steps 5 and 6 for each round in Table 4.78.
5. The Lower Tester sends a PATH_REQUEST_SOLICITATION message with the following parameters, unless stated in Table 4.78: DST set to the all-directed-forwarding-nodes address, TTL set to 0x7F, Addr_List set to {_GR_ADDR_}.
6. The IUT does not send any PATH_REQUEST messages.

Round	Additional Condition
1	DST set to a _IUT_ADDR_
2	TTL set to 0x00
3	Message sent with managed flooding security material
4	Addr_List set to 0x0FFFB (all-directed-forwarding-nodes)
5	Addr_List set to 0xFFFE (all-relays)
6	Addr_List set to 0xFFFF (all-nodes)

Table 4.78: Specification of rounds for MESH/NODE/DF/SOL/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any PATH_REQUEST messages.

4.18.9 Directed Forwarding Using Fixed Paths

This subsection verifies sending of access messages and forwarding of network messages using fixed paths.

MESH/NODE/DF/FIX/BV-01-C [Forward Network Message to Unicast Address, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT will forward a network message destined for a unicast address and secured using directed security material when at least one path is known for the destination.

- Reference

  [10] 3.4.6.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

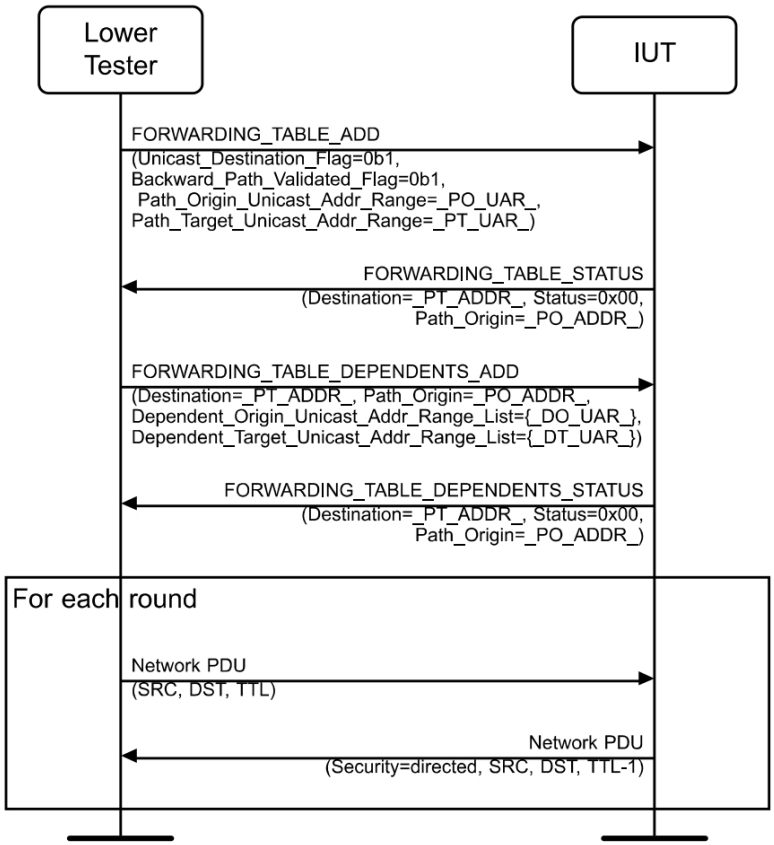


Figure 4.260: MESH/NODE/DF/FIX/BV-01-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO_UAR_}, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_UAR_}.
4. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
5. Repeat Steps 6 and 7 for each round in Table 4.79.
6. The Lower Tester sends a network PDU encrypted with the directed security material with SRC and DST set to the values from Table 4.79.

7. The IUT sends the network PDU received in Step 1 with the TTL field decremented by 1 and re-encrypted with the directed security material.

Round	SRC	DST
1	_PO_ADDR_	_PT_ADDR_
2	_PO_ADDR_ + 1	_PT_ADDR_ + 1
3	_DO_ADDR_	_DT_ADDR_
4	_DO_ADDR_ + 1	_DT_ADDR_ + 1
5	_PT_ADDR_	_PO_ADDR_

Table 4.79: Specification of rounds for MESH/NODE/DF/FIX/BV-01-C

- Expected Outcome

  Pass verdict

  The IUT forwards the message sent by the Lower Tester with the expected fields and re-encrypted with the directed security material.

MESH/NODE/DF/FIX/BV-02-C [Forward Network Message to Group/Virtual Address, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT will forward a network message destined for a group/virtual address, secured using directed security material and at least one path is known for the destination.

- Reference

  [10] 3.4.6.3

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

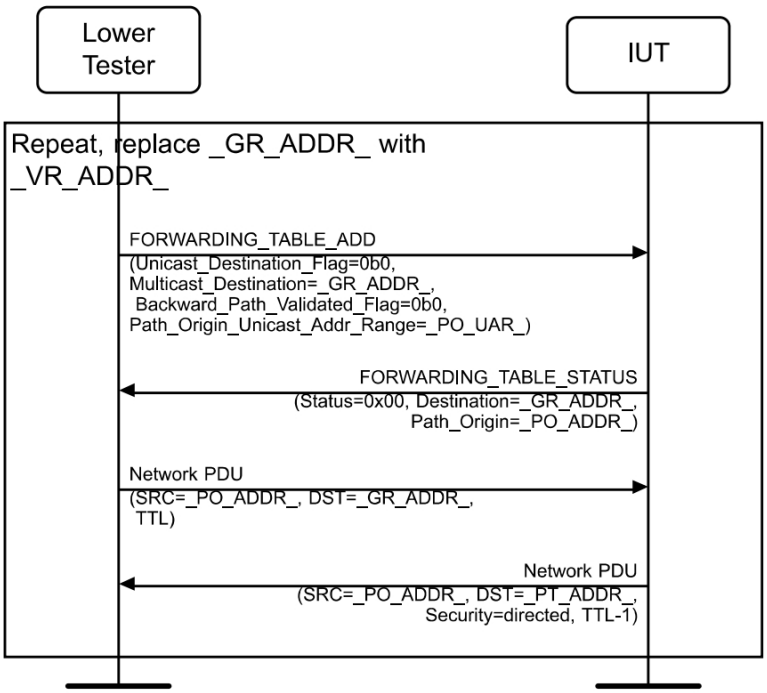


Figure 4.261: MESH/NODE/DF/FIX/BV-02-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b0, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Multicast_Destination set to _GR_ADDR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _GR_ADDR_.
3. The Lower Tester sends a network PDU with SRC set to _PO_ADDR_ and DST set to _GR_ADDR_, valid values for all other fields and encrypted with the directed security material.
4. The IUT sends the network PDU received in Step 1 with the TTL field decremented by 1 and re-encrypted with the directed security material.
5. Repeat Steps 1–4 replacing _GR_ADDR_ with _VR_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT forwards the message sent by the Lower Tester with the expected fields.

MESH/NODE/DF/FIX/BV-03-C [Ignore Network Message, IUT as Directed Relay]

- Test Purpose

  Verify that an IUT will discard a network message, secured using directed security material when no path is known for the destination or directed relay is disabled.

- Reference

  [10] 3.4.5.4

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The directed relay functionality on the primary subnet is enabled on the IUT.

- Test Procedure

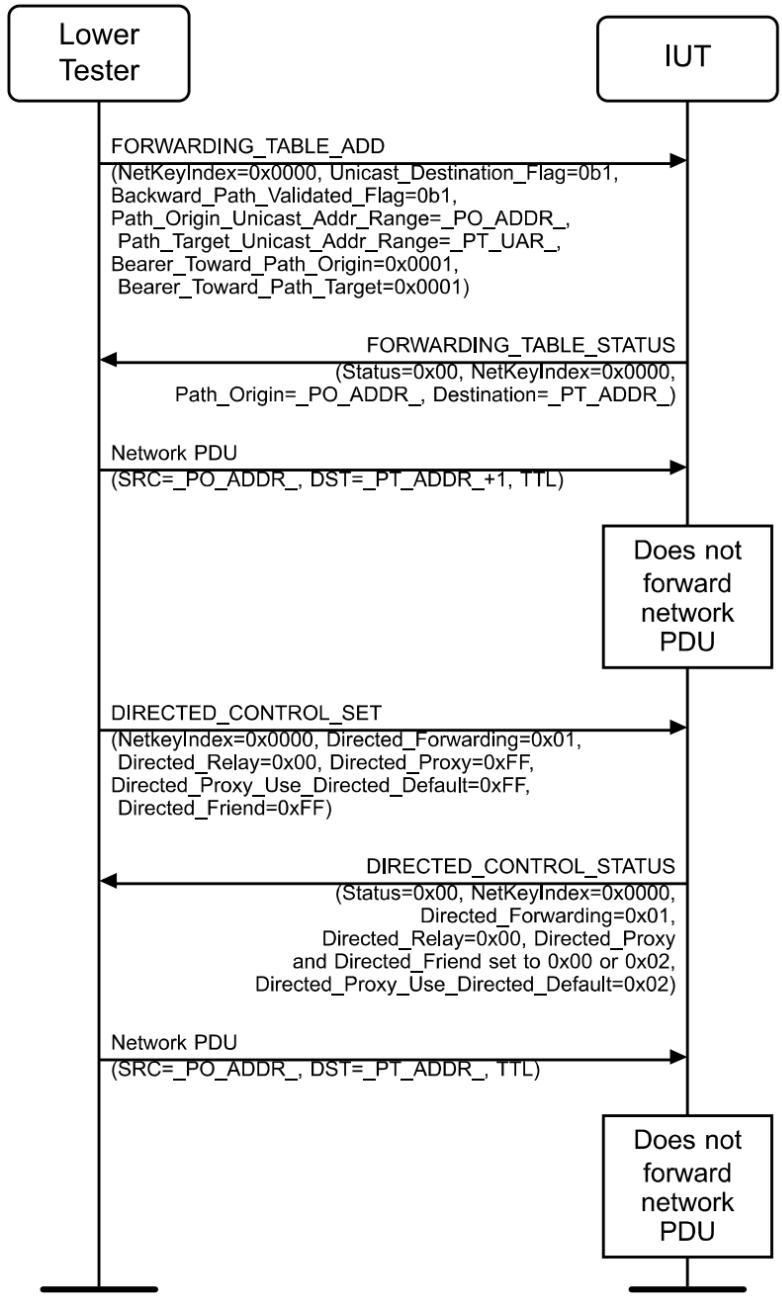


Figure 4.262: MESH/NODE/DF/FIX/BV-03-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_ADDR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a network PDU with SRC set to _PO_ADDR_ and DST set to _PT_ADDR_ + 1 with valid fields and encrypted with the directed security material.
4. The IUT does not forward the network PDU.
5. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0xFF (Ignore).
6. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy and Directed_Friend set to 0x00 or 0x02, and Directed_Proxy_Use_Directed_Default set to 0x02.
7. The Lower Tester sends a network PDU with SRC set to _PO_ADDR_ and DST set to _PT_ADDR_ with valid fields and encrypted with the directed security material.
8. The IUT does not forward the network PDU.

- Expected Outcome

  Pass verdict

  The IUT does not forward the message sent by the Lower Tester.

MESH/NODE/DF/FIX/BV-04-C [Transmit Unsolicited Access Message for Existing Path - Unicast Address]

- Test Purpose

  Verify that a Directed Forwarding node IUT can transmit an unsolicited access message destined for a unicast address, secured using directed security material when at least one path is known for the destination.

- Reference

  [10] 3.7.4.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

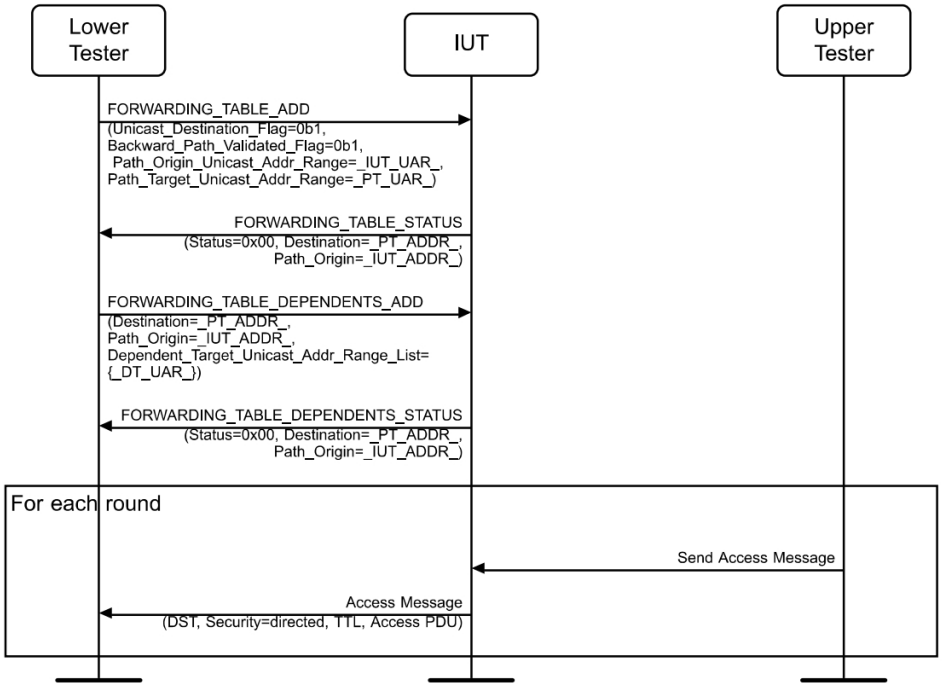


Figure 4.263: MESH/NODE/DF/FIX/BV-04-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _IUT_ADDR_, Destination set to _PT_ADDR_, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_UAR_}.
4. The IUT responds with a FORWARDING_TABLE_ DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
5. Repeat Steps 6 and 7 for each round in Table 4.80.
6. The Upper Tester commands the IUT to send an access message with DST set to the value in Table 4.80.
7. The IUT sends an access message with valid values for TTL, DST, and Access PDU fields, and to encrypt the message using the directed security material.

Round	DST
1	_PT_ADDR_
2	_PT_ADDR_ + 1
3	_DT_ADDR_
4	_DT_ADDR_ + 1

Table 4.80: Specification of rounds for MESH/NODE/DF/FIX/BV-04-C

- Expected Outcome

  Pass verdict

  The IUT sends an access message with the expected fields and encrypted with directed security material.

MESH/NODE/DF/FIX/BV-05-C [Transmit Unsolicited Access Message for Existing Path - Group/Virtual Address]

- Test Purpose

  Verify that a Directed Forwarding node IUT can transmit an unsolicited access message destined for a group/virtual address, secured using directed security material when at least one path is known for the destination.

- Reference

  [10] 3.7.4.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

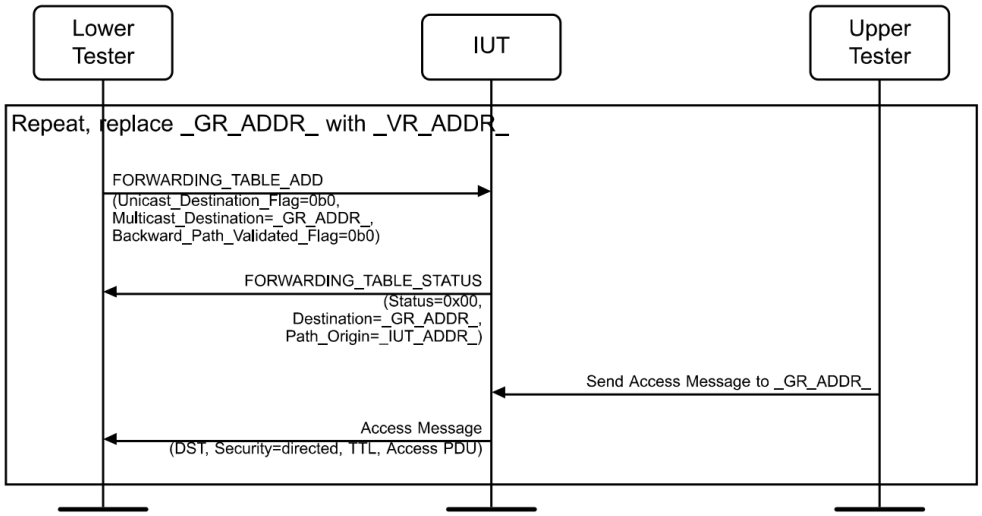


Figure 4.264: MESH/NODE/DF/FIX/BV-05-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b0, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Multicast_Destination set to _GR_ADDR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _GR_ADDR_.
3. The Upper Tester commands the IUT to send an access message to DST set to _GR_ADDR_.
4. The IUT sends an access message with valid values for TTL and Access PDU fields, DST set to a valid value other than all-nodes and all-relays fixed group addresses, and encrypts the message using the directed security material.
5. Repeat Steps 1–4 replacing _GR_ADDR_ with _VR_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT sends an access message with the expected fields and encrypted with directed security material.

MESH/NODE/DF/FIX/BV-06-C [Transmit Access Message Response for Existing Path - Unicast Address]

- Test Purpose

  Verify that a Directed Forwarding node IUT can transmit an access message response destined for a unicast address, secured with the correct security material based on whether the path is known for the destination.

- Reference

  [10] 3.7.4.1

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

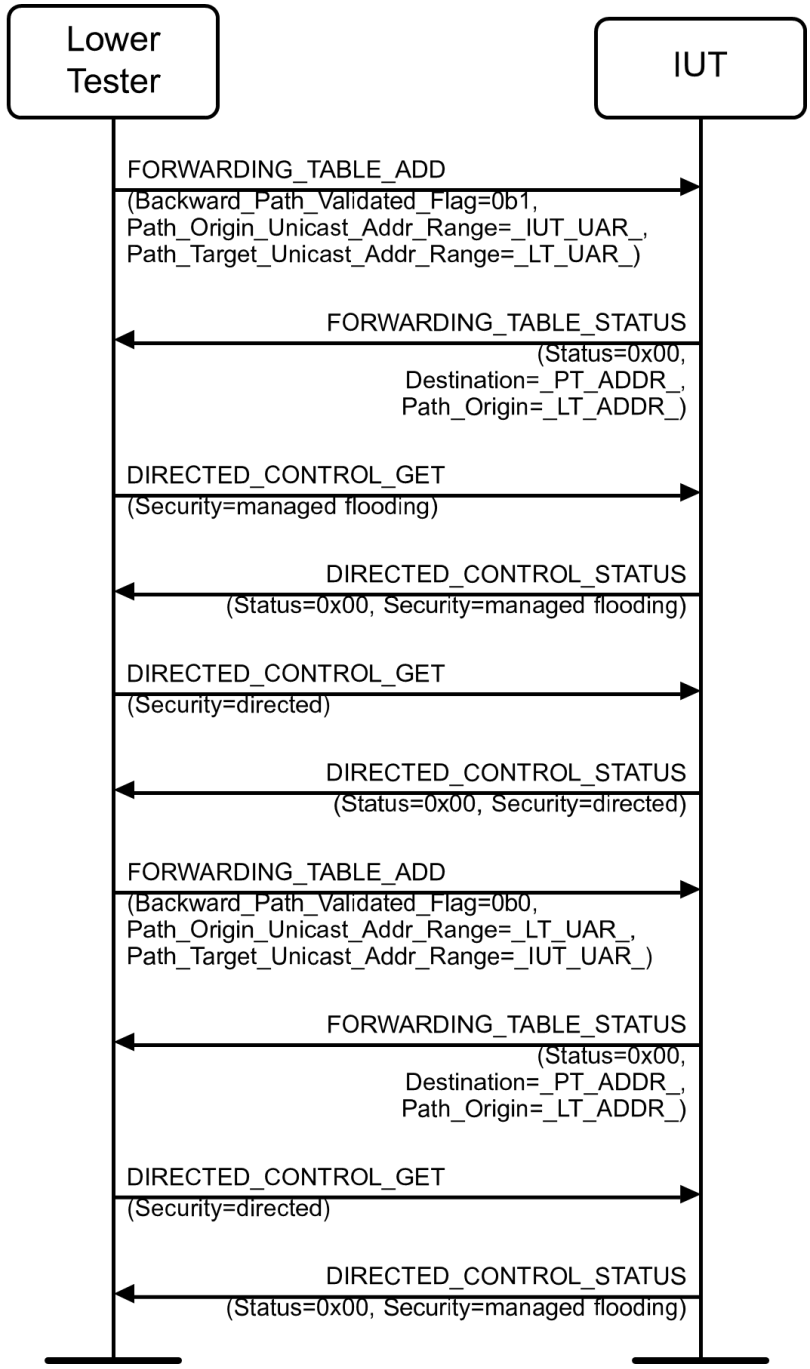


Figure 4.265: MESH/NODE/DF/FIX/BV-06-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to *LT_UAR*, Path_Target_Unicast_Addr_Range set to

_IUT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0000.

2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _LT_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a DIRECTED_CONTROL_GET message to the IUT secured with managed flooding security credentials with NetKeyIndex field set to 0x0000.
4. The IUT responds with a DIRECTED_CONTROL_STATUS message secured with managed flooding security credentials with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).
5. The Lower Tester sends a DIRECTED_CONTROL_GET message secured with directed security credentials with NetKeyIndex field set to 0x0000.
6. The IUT responds with a DIRECTED_CONTROL_STATUS message secured with directed security credentials with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).
7. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to *LT_UAR*, Path_Target_Unicast_Addr_Range set to _IUT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0000.
8. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _LT_ADDR_, and Destination set to _PT_ADDR_.
9. The Lower Tester sends a DIRECTED_CONTROL_GET message secured with directed security credentials with NetKeyIndex field set to 0x0000.
10. The IUT responds with a DIRECTED_CONTROL_STATUS message secured with managed flooding security credentials with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).

- Expected Outcome

  Pass verdict

  The IUT responds with an access message with the expected fields and encrypted with the expected security material.

MESH/NODE/DF/FIX/BV-07-C [Transmit Access Message for Existing Path Without PATH_REQUEST]

- Test Purpose

  Verify that an IUT acting as a Directed Forwarding node does not send a PATH_REQUEST message after transmitting an access message if there is an existing Forwarding Table entry for a fixed path.

- Reference

  [10] 3.7.4.1

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The Two Way Path state on the primary subnet is set to 0b1 on the IUT.

- Test Procedure

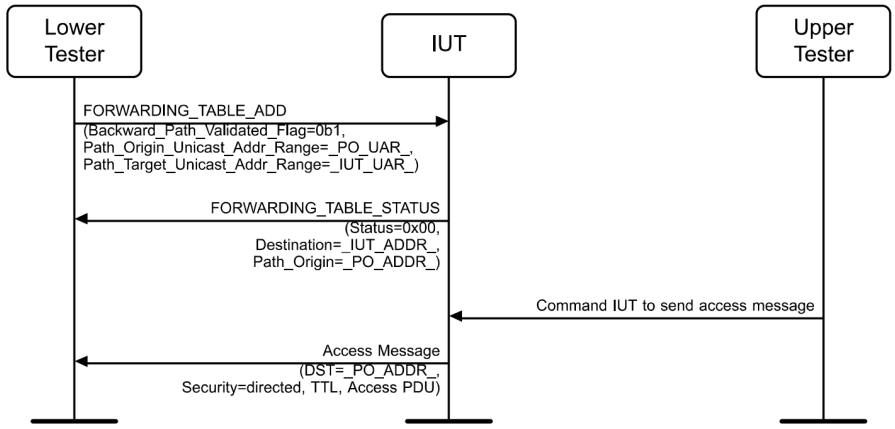


Figure 4.266: MESH/NODE/DF/FIX/BV-07-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _IUT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0000.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _IUT_ADDR_.
3. The Upper Tester commands the IUT to send an access message with DST set to _PO_ADDR_.
4. The IUT sends an access message with valid values for TTL, DST, and Access PDU fields, and secured with directed security material.

- Expected Outcome

  Pass verdict

The IUT does not send a PATH_REQUEST message after sending the access message to the Lower Tester.

The IUT sends the access message with the directed security material.

4.18.10 Directed Proxy

MESH/SR/DPROX/BV-01-C [Connection with Proxy Client]

- Test Purpose

  Verify that a Directed Proxy Server IUT can establish a connection with a Proxy Client and relay messages.

- Reference

  [10] 6.5, 6.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

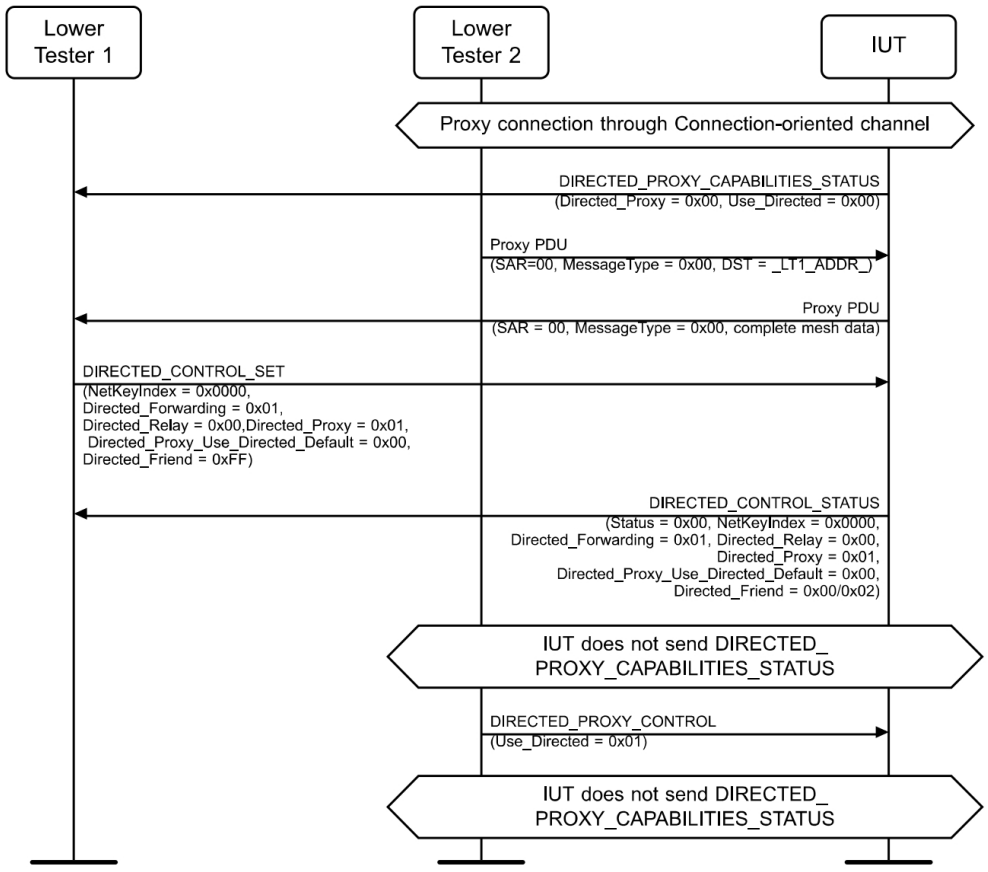


Figure 4.267: MESH/SR/DPROX/BV-01-C MSC

1. Lower Tester 2 performs a proxy connection to the IUT through a connection-oriented channel.

2. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
3. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 ("Complete higher-level message"), MessageType equal to 0x00 ("Network PDU"), and DST set to _LT1_ADDR_. The Proxy PDU does not require reassembly at the IUT.
4. The IUT sends a Proxy PDU with SAR field set to 00 ("Complete higher-level message"), Type equal to 0x00 ("Mesh Data"), and the complete mesh data to Lower Tester 1.
5. Lower Tester 1 sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled), and Directed_Friend set to 0xFF (Ignore).
6. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled), and Directed_Friend set to 0x00 or 0x02.
7. The IUT does not send a DIRECTED_PROXY_CAPABILITIES_STATUS message.
8. Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed set to 0x01 (Enabled) and Proxy_Client_Unicast_Addr_Range set to _LT2_UAR_.
9. The IUT does not respond with a DIRECTED_PROXY_CAPABILITIES_STATUS message.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields.

  In Steps 7 and 9, the IUT does not send a DIRECTED_PROXY_CAPABILITIES_STATUS message.

  The IUT relays the network message secured with managed flooding security material.

  Lower Tester 1 receives the exact data contained in the Proxy PDU send from Lower Tester 2.

MESH/SR/DPROX/BV-02-C [Connection with Directed Proxy Client]

- Test Purpose

  Verify that a Directed Proxy Server IUT can relay a message secured using the managed flooding security credentials when directed forwarding is not selected and does not initiate a path discovery.

- Reference

  [10] 6.5, 6.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

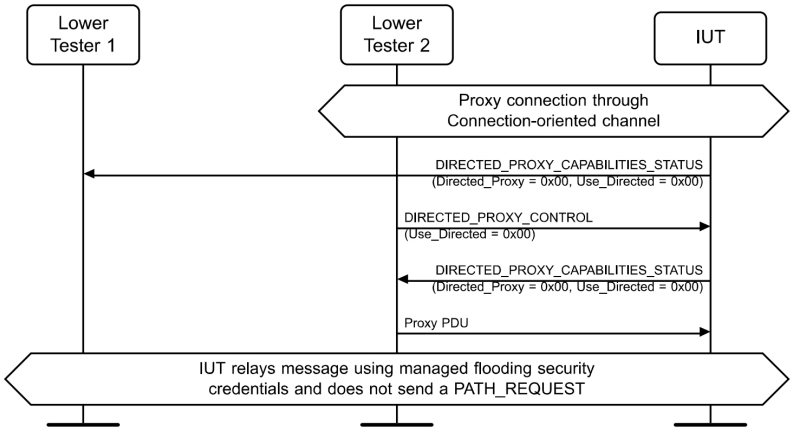


Figure 4.268: MESH/SR/DPROX/BV-02-C MSC

1. Lower Tester 2 performs a proxy connection to the IUT through a connection-oriented channel.
2. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
3. Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed set to 0x00 (Disabled).
4. The IUT responds with a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
5. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 ("Complete higher-level message"), MessageType equal to 0x00 ("Network PDU") and DST set to _LT1_ADDR_. The Proxy PDU does not require reassembly at the IUT.
6. The IUT relays the message using the managed flooding security credentials and does not send a PATH_REQUEST.

- Expected Outcome

  Pass verdict

  The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields and does not send a PATH_REQUEST.

MESH/SR/DPROX/BV-03-C [Connection with Reject List Client]

- Test Purpose

  Verify that a Directed Proxy Server IUT can establish a connection with a Reject List Proxy Client and relay messages.

- Reference

  [10] 6.5, 6.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured with Directed Proxy set to 0x01 (Enabled) and Directed Proxy Use Directed

Default set to 0x01 (Enabled) on the primary subnet.

- Test Procedure

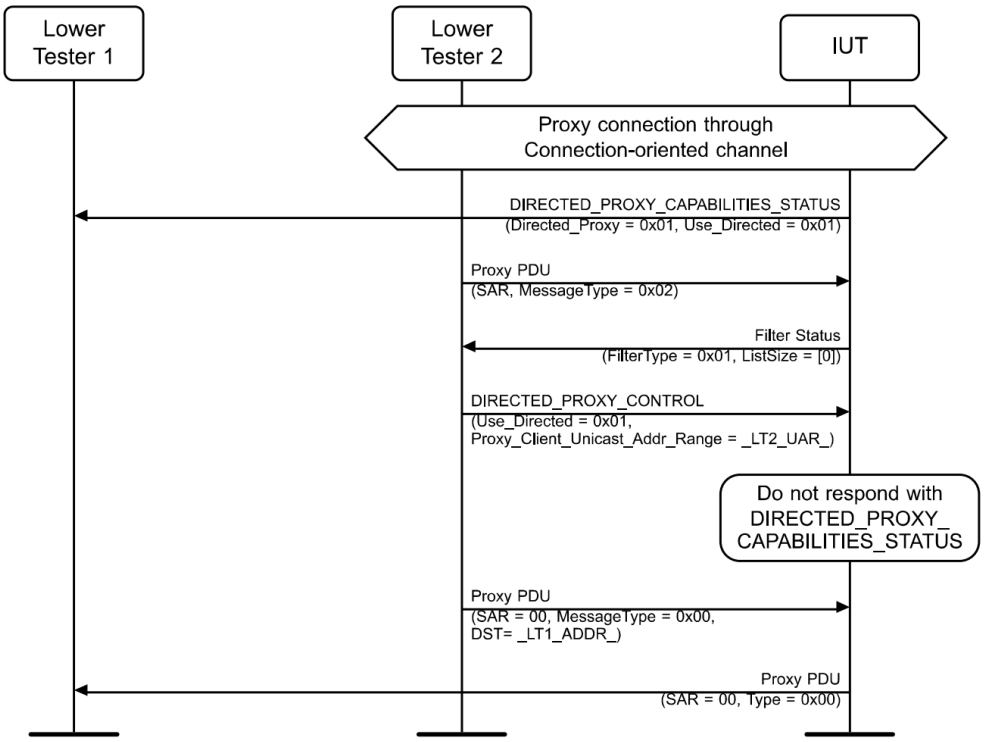


Figure 4.269: MESH/SR/DPROX/BV-03-C MSC

1. Lower Tester 2 performs a proxy connection to the IUT through a connection-oriented channel.
2. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x01 (Enabled).
3. Lower Tester 2 sends a Proxy PDU to the IUT with the SAR value of b00 (Data field contains a complete message) and a MessageType field set to 0x02 (Proxy Configuration Message). The Proxy Configuration Message has a CTL field value set to 1, a TTL field value set to 0, and a DST field value equal to the unassigned address. The Proxy Configuration OpCode field value will be set to 0x00 (Set Filter Type) and the FilterType field set to 0x01 (Reject List Filter).
4. The IUT responds with a Filter Status message with a FilterType value equal to 0x01 (Reject List) and a ListSize equal to [0].
5. Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed set to 0x01 (Enabled) and Proxy_Client_Unicast_Addr_Range set to _LT2_UAR_.
6. The IUT does not respond with a DIRECTED_PROXY_CAPABILITIES_STATUS message.
7. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 ("Complete higher-level message"), MessageType equal to 0x00 ("Network PDU"), and DST set to _LT1_ADDR_. The Proxy PDU does not require reassembly at the IUT.
8. The IUT sends a Proxy PDU with SAR field set to 00 ("Complete higher-level message"), Type equal to 0x00 ("Mesh Data"), and the complete mesh data to Lower Tester 1.

- Expected Outcome

Pass verdict

The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields.

The IUT relays the network message secured with managed flooding security material.

Lower Tester 1 receives the exact data contained in the Proxy PDU send from Lower Tester 2.

MESH/SR/DPROX/BV-04-C [Send DIRECTED_PROXY_CAPABILITIES_STATUS on Subnet Configuration]

- Test Purpose

  Verify that a Directed Proxy Server IUT in a connection with a Proxy client sends a DIRECTED_PROXY_CAPABILITIES_STATUS message when it is configured on a new subnet.

- Reference

  [10] 6.5, 6.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT supports multiple subnets, but it is configured only on the primary subnet.
  - The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13.

- Test Procedure

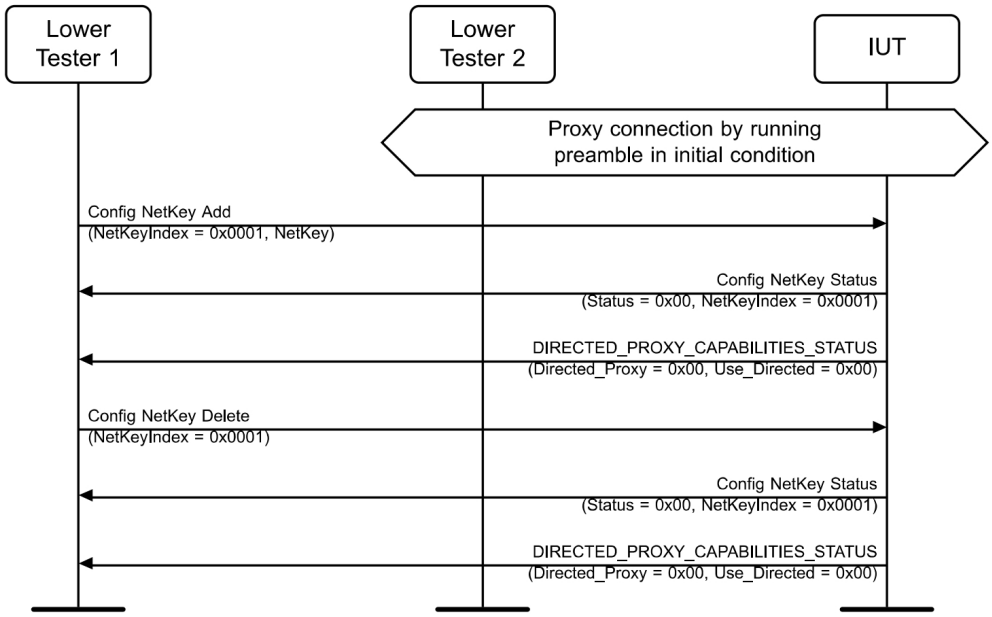


Figure 4.270: MESH/SR/DPROX/BV-04-C MSC

1. Lower Tester 1 sends a Config NetKey Add message with NetKeyIndex set to 0x0001 and a new NetKey value.
2. The IUT responds with a Config NetKey Status message with Status field set to 0x00 and the NetKeyIndex set to 0x0001.
3. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
4. Lower Tester 1 sends a Config NetKey Delete message with NetKeyIndex set to 0x0001.
5. The IUT responds with a Config NetKey Status message with Status field set to 0x00 and the NetKeyIndex set to 0x0001.
6. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).

- Expected Outcome

  Pass verdict

  In Steps 3 and 5, the IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields. The messages are encrypted with the NetKey from Step 1.

MESH/SR/DPROX/BV-05-C [Send DIRECTED_PROXY_CAPABILITIES_STATUS on Directed Proxy State Configuration]

- Test Purpose

  Verify that a Directed Proxy Server IUT in a connection with a Proxy client sends a DIRECTED_PROXY_CAPABILITIES_STATUS message when the Directed Proxy state is changed.

- Reference

[10] 6.5, 6.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT established a proxy connection with Lower Tester 2 as a Directed Proxy client by running preamble 4.2.13.
- Test Procedure

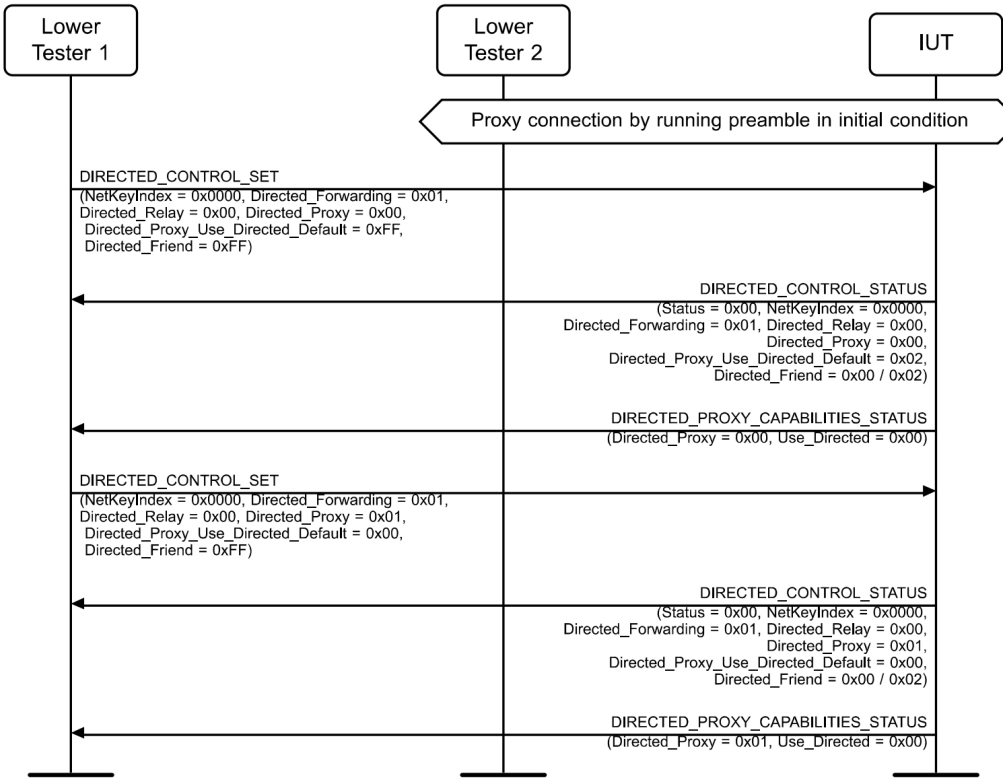


Figure 4.271: MESH/SR/DPROX/BV-05-C MSC

1. Lower Tester 1 sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x00 (Disabled), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0xFF (Ignore).
2. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x00 (Disabled), Directed_Proxy_Use_Directed_Default set to 0x02 (Not supported or disabled), and Directed_Friend set to 0x00 or 0x02.
3. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
4. Repeat Steps 1 and 2 with Directed_Proxy set to 0x01 (Enabled) and Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled).

5. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x00 (Disabled).

- Expected Outcome

  Pass verdict

  The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields.

MESH/SR/DPROX/BV-06-C [Multiple Subnet Configuration]

- Test Purpose

  Verify that a Directed Proxy Server IUT in a connection with a Proxy client disables the Use_Directed feature for the subnets that the Proxy client does not configure.

- Reference

  [10] 6.5, 6.6

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT supports multiple subnets.
  - The IUT is configured with Directed Proxy set to 0x01 (Enabled) and Directed Proxy Use Directed Default set to 0x01 (Enabled) on the primary subnet.

- Test Procedure

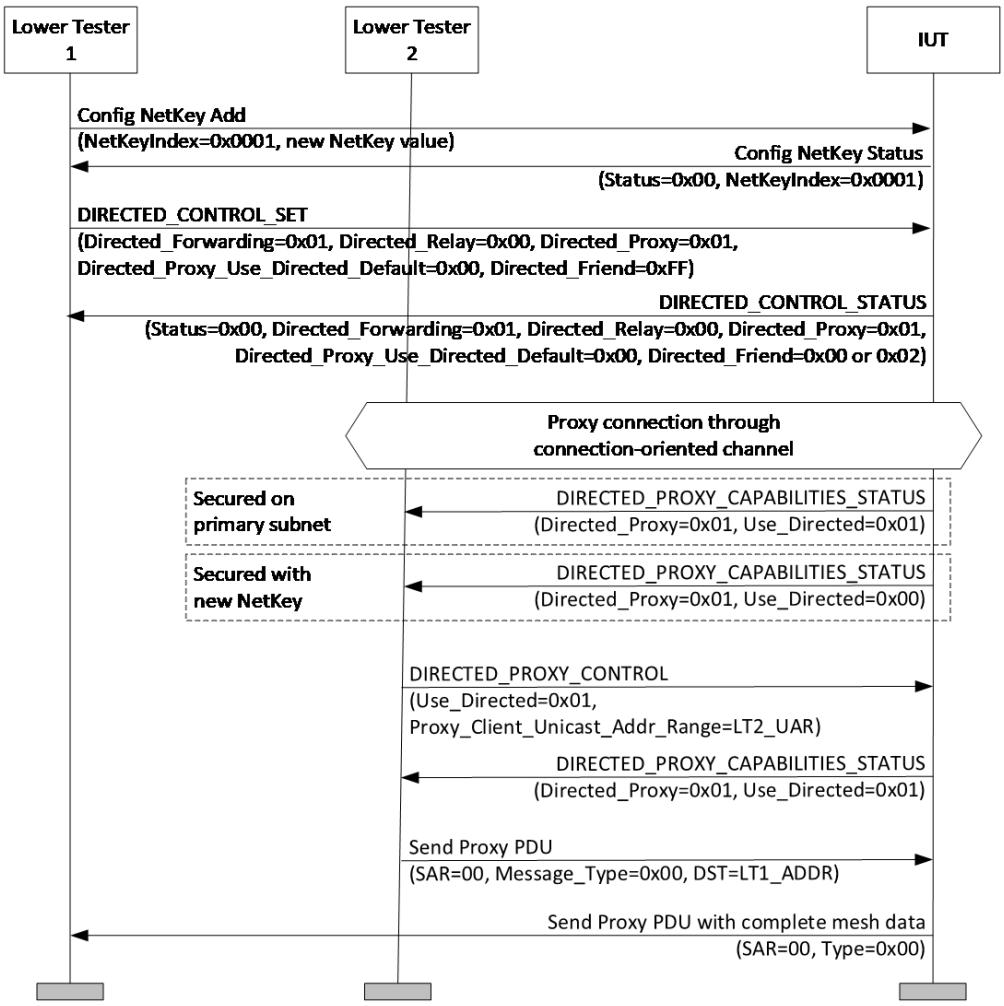


Figure 4.272: MESH/SR/DPROX/BV-06-C MSC

1. Lower Tester 1 sends a Config NetKey Add message with NetKeyIndex set to 0x0001 and a new NetKey value.
2. The IUT responds with a Config NetKey Status message with Status field set to 0x00 and the NetKeyIndex set to 0x0001.
3. Lower Tester 1 sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0001, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled), and Directed_Friend set to 0xFF (Ignore).
4. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success) and the NetKeyIndex set to 0x0001, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay fields set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled), and Directed_Friend set to 0x00 or 0x02.
5. Lower Tester 2 performs a proxy connection to the IUT through a connection-oriented channel.
6. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message secured on the primary subnet with Directed_Proxy set to 0x01 (Enabled), Use_Directed set to 0x01 (Enabled) and a DIRECTED_PROXY_CAPABILITIES_STATUS message secured with the NetKey from Step 1, with

Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x00 (Disabled).

7. Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message secured with the NetKey from Step 1 to the IUT with Use_Directed set to 0x01 (Enabled) and Proxy_Client_Unicast_Addr_Range set to _LT2_UAR_.
8. The IUT responds with a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x01 (Enabled).
9. Lower Tester 2 sends a single Proxy PDU to the IUT smaller than the negotiated proxy MTU with SAR field set to 00 ("Complete higher-level message"), MessageType equal to 0x00 ("Network PDU"), and DST set to _LT1_ADDR_. The Proxy PDU does not require reassembly at the IUT and is sent on the primary subnet.
10. The IUT sends an advertising PDU using the Mesh Message AD Type containing the complete mesh message.

- Expected Outcome

  Pass verdict

  The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with the expected fields. The message is encrypted with the network key added in Step 1.

  In Step 10, the IUT relays the network message secured with managed flooding security material, received on the primary subnet, and does not trigger the Directed Forwarding Initialization procedure.

  Lower Tester 1 receives the exact data contained in the Proxy PDU sent from Lower Tester 2.

MESH/SR/DPROX/BV-07-C [Ignore Unsupported Directed Proxy Messages]

- Test Purpose

  Verify that a Proxy Server IUT that does not support directed proxy functionality will ignore Directed Proxy messages.

- Reference

  [10] 6.6

- Initial Condition
  - The Lower Tester performed a proxy connection to the IUT through a connection-oriented channel.
  - The IUT does not support the directed proxy functionality.

- Test Procedure

  1. The Lower Tester sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed and Proxy_Client_Unicast_Addr_Range set to random valid values.
  2. The IUT ignores the message.

- Expected Outcome

  Pass verdict

  The IUT ignores the DIRECTED_PROXY_CONTROL message.

The IUT does not terminate the proxy connection with the Lower Tester.

MESH/SR/DPROX/BI-01-C [Ignore DIRECTED_PROXY_CONTROL Messages - Invalid Parameters]

- Test Purpose

  Verify that a Directed Proxy Server IUT can ignore DIRECTED_PROXY_CONTROL messages sent from a Directed Proxy Client with invalid parameters.

- Reference

  [10] 6.5, 6.6

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

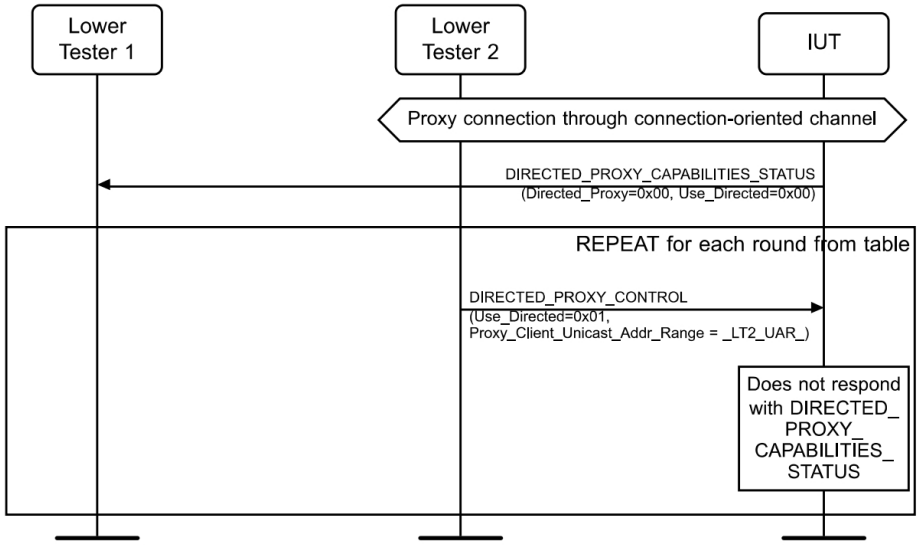


Figure 4.273: MESH/SR/DPROX/BI-01-C MSC

1. Lower Tester 2 performs a proxy connection to the IUT through a connection-oriented channel.
2. The IUT sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x00 (Disabled) and Use_Directed set to 0x00 (Disabled).
3. For each round in Table 4.81 below:
4. Lower Tester 2 sends a DIRECTED_PROXY_CONTROL message to the IUT with the following parameters, unless stated in Table 4.81: Use_Directed set to 0x01 (Enabled), Proxy_Client_Unicast_Addr_Range set to _LT2_UAR_.
5. The IUT does not respond with any DIRECTED_PROXY_CAPABILITIES_STATUS messages.

Round	Additional Condition
1	Use_Directed set to 0x02
2	Proxy_Client_Unicast_Addr_Range set to 0xFFFF
3	Proxy_Client_Unicast_Addr_Range set to 0xFFFF02
4	Proxy_Client_Unicast_Addr_Range set to 0x800100
5	Use_Directed set to 0x00

Table 4.81: Specification of rounds for MESH/SR/DROX/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not send any DIRECTED_PROXY_CAPABILITIES_STATUS messages.

MESH/CL/DPROX/BV-01-C [Connection with Directed Proxy Server]

- Test Purpose

  Verify that a Directed Proxy Client IUT can establish a connection with a Directed Proxy Server and control the proxy configuration.

- Reference

  [10] 6.5, 6.6

- Initial Condition
  - The IUT supports the Directed Proxy Client functionality.
- Test Procedure

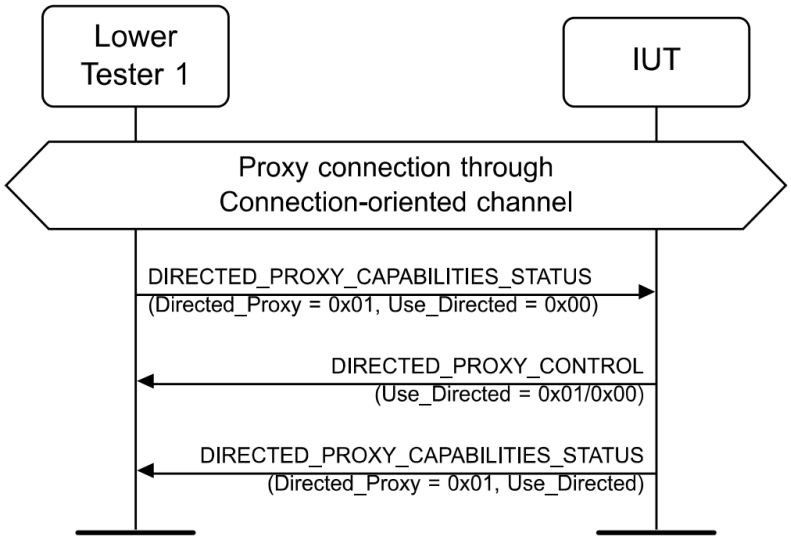


Figure 4.274: MESH/CL/DPROX/BV-01-C MSC

1. The IUT performs a proxy connection with the Lower Tester through a connection-oriented channel.
2. The Lower Tester sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to 0x00 (Disabled).
3. The IUT sends a DIRECTED_PROXY_CONTROL message to the IUT with Use_Directed set to 0x01 or 0x00. If Use_Directed is set to 0x01, Proxy_Client_Unicast_Addr_Range is present and is set to _IUT_UAR_.
4. The IUT responds with a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy set to 0x01 (Enabled) and Use_Directed set to the value in Step 3.

- Expected Outcome

  Pass verdict

  The IUT sends a DIRECTED_PROXY_CONTROL message with the expected fields.

MESH/CL/DPROX/BV-02-C [Ignore Unsupported Directed Proxy Messages]

- Test Purpose

  Verify that a Proxy Client IUT that does not support directed proxy client functionality will ignore Directed Proxy messages.

- Reference

  [10] 6.7

- Initial Condition
  - The Lower Tester performed a proxy connection to the IUT through a connection-oriented channel.
  - The IUT does not support the directed proxy client functionality.

- Test Procedure

1. The Lower Tester sends a DIRECTED_PROXY_CAPABILITIES_STATUS message with Directed_Proxy and Use_Directed fields set to random valid values.
2. The IUT ignores the message.

- Expected Outcome

  Pass verdict

  The IUT ignores the DIRECTED_PROXY_CAPABILITIES_STATUS message.

  The IUT does not terminate the proxy connection with the Lower Tester.

4.19 Directed Forwarding Configuration Model

4.19.1 Common Directed Forwarding Configuration Model Server Behavior

This section contains tests that verify common IUT functionality related to the Directed Forwarding Configuration Server model.

4.19.1.1 Common Subnet State Get/Set/Status Behavior

- Test Purpose

  This test procedure verifies common Get/Set/Status behavior for a Server Model on the IUT supporting the _STATE_ on each supported subnet.

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - If the IUT supports at least two subnets, it is configured on two subnets.
  - The IUT is configured with the default values for the _STATE_.

- Test Procedure

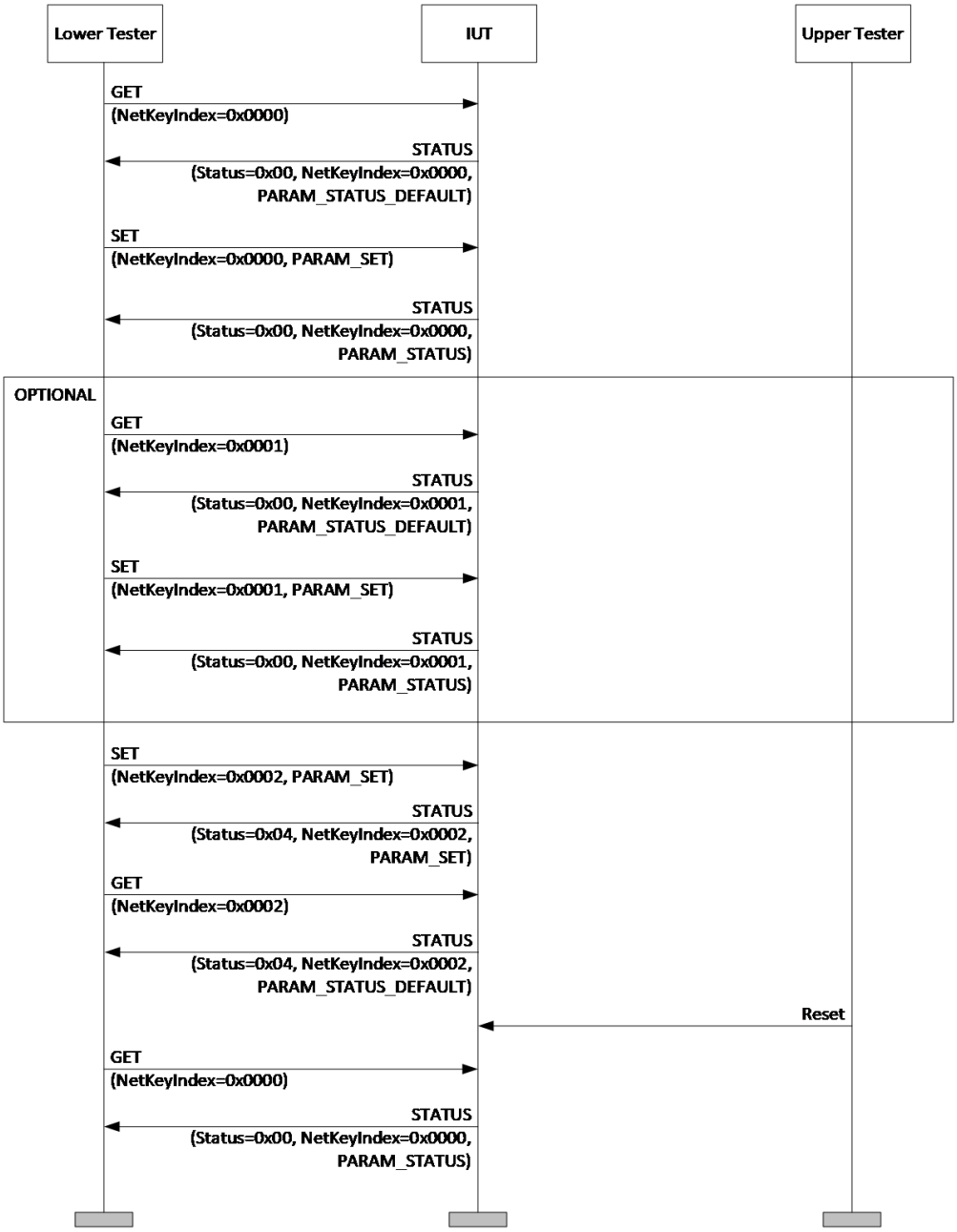


Figure 4.275: Test procedure for Common Subnet State Get/Set/Status Behavior on Directed Forwarding Configuration Server models MSC

1. The Lower Tester sends a _GET_ message with NetKeyIndex field set to 0x0000.
2. The IUT responds with a _STATUS_ message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000 and parameters defined by _PARAM_STATUS_DEFAULT_.
3. The Lower Tester sends a _SET_ message with NetKeyIndex field set to 0x0000 and parameters defined by _PARAM_SET_.
4. The IUT responds with a _STATUS_ message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000 and parameters defined by _PARAM_STATUS_.
5. If the IUT is configured on a second subnet, repeat Steps 1–4 with the NetKeyIndex field set to 0x0001. Otherwise, skip this step.

6. The Lower Tester sends a _SET_ message with NetKeyIndex field set to 0x0002 and parameters defined by _PARAM_SET_.
7. The IUT responds with a _STATUS_ message with Status set to 0x04 (Invalid NetKey Index) and other parameters set to the values from Step 6.
8. The Lower Tester sends a _GET_ message with NetKeyIndex field set to 0x0002.
9. The IUT responds with a _STATUS_ message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex set to 0x0002, and other parameters set to zeros.
10. The Upper Tester orders the IUT to reset itself.
11. The Lower Tester sends a _GET_ message with NetKeyIndex field set to 0x0000.
12. The IUT responds with a _STATUS_ message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000 and parameters defined by _PARAM_STATUS_.

- Expected Outcome

  Pass verdict

  The IUT responds with a _STATUS_ message with the expected fields. The messages are encrypted with the device key.

4.19.1.1.1 Directed Control State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Directed Control state
GET	DIRECTED_CONTROL_GET
STATUS	DIRECTED_CONTROL_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x02 if Directed Proxy not supported, otherwise 0x00 Directed_Proxy_Use_Directed_Default = 0x02 (Disabled or not supported) Directed_Friend = 0x02 if Directed Friend not supported, otherwise 0x00
SET	DIRECTED_CONTROL_SET
_PARAM_SET_	Directed_Forwarding = 0x01 Directed_Relay = 0x01 Directed_Proxy = 0xFF Directed_Proxy_Use_Directed_Default = 0xFF Directed_Friend = 0xFF

State/Message/Parameter	Value(s)
_PARAM_STATUS_	Directed_Forwarding = 0x01 Directed_Relay = 0x01 Directed_Proxy = 0x02 if Directed Proxy not supported, otherwise 0x00 Directed_Proxy_Use_Directed_Default = 0x02 (Disabled or not supported) Directed_Friend = 0x02 if Directed Friend not supported, otherwise 0x00

Table 4.82: Common Get/Set/Status Behavior values for the Directed Control state

4.19.1.1.2 Path Metric State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Path Metric state
GET	PATH_METRIC_GET
STATUS	PATH_METRIC_STATUS
_PARAM_STATUS_DEFAULT_	Path_Metric_Type = 0b000 Path_Lifetime = 0b10
SET	PATH_METRIC_SET
_PARAM_SET_, _PARAM_STATUS_	Path_Metric_Type = 0b000 Path_Lifetime = 0b11

Table 4.83: Common Get/Set/Status Behavior Values for the Path Metric state

4.19.1.1.3 Discovery Table Capabilities State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Discovery Table Capabilities state
GET	DISCOVERY_TABLE_CAPABILITIES_GET
STATUS	DISCOVERY_TABLE_CAPABILITIES_STATUS
_PARAM_STATUS_DEFAULT_	Max_Concurrent_Init = 0x02 Max_Discovery_Table_Entries_Count ≥ 0x02
SET	DISCOVERY_TABLE_CAPABILITIES_SET
_PARAM_SET_	Max_Concurrent_Init = 0x01
_PARAM_STATUS_	Max_Concurrent_Init = 0x01 Max_Discovery_Table_Entries_Count ≥ 0x02

Table 4.84: Common Get/Set/Status Behavior Values for the Discovery Table Capabilities state

4.19.1.1.4 Wanted Lanes State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Wanted Lanes state
GET	WANTED_LANES_GET
STATUS	WANTED_LANES_STATUS
_PARAM_STATUS_DEFAULT_	Wanted_Lanes = 0x01
SET	WANTED_LANES_SET
_PARAM_SET_, _PARAM_STATUS_	Wanted_Lanes = 0x04

Table 4.85: Common Get/Set/Status Behavior Values for the Wanted Lanes state

4.19.1.1.5 Two Way Path State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Two Way Path state
GET	TWO_WAY_PATH_GET
STATUS	TWO_WAY_PATH_STATUS
_PARAM_STATUS_DEFAULT_	Two_Way_Path = 0b1
SET	TWO_WAY_PATH_SET
_PARAM_SET_, _PARAM_STATUS_	Two_Way_Path = 0b0

Table 4.86: Common Get/Set/Status Behavior Values for the Two Way Path state

4.19.1.1.6 Path Echo Interval State Get/Set/Status, Change Both Intervals

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
STATUS	PATH_ECHO_INTERVAL_STATUS
_PARAM_STATUS_DEFAULT_	Unicast_Echo_Interval = 0x00 Multicast_Echo_Interval = 0x00
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_, _PARAM_STATUS_	Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0x01

Table 4.87: Common Get/Set/Status Behavior Values for the Path Echo Interval state, Change Both Intervals

4.19.1.1.7 Path Echo Interval State Get/Set/Status, Change Only Unicast

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
STATUS	PATH_ECHO_INTERVAL_STATUS
_PARAM_STATUS_DEFAULT_	Unicast_Echo_Interval = 0x00 Multicast_Echo_Interval = 0x00
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_	Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0xFF
_PARAM_STATUS_	Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0x00

Table 4.88: Common Get/Set/Status Behavior Values for the Path Echo Interval state, Change Only Unicast

4.19.1.1.8 Path Echo Interval State Get/Set/Status, Change Only Multicast

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
STATUS	PATH_ECHO_INTERVAL_STATUS
_PARAM_STATUS_DEFAULT_	Unicast_Echo_Interval = 0x00 Multicast_Echo_Interval = 0x00
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_	Unicast_Echo_Interval = 0xFF Multicast_Echo_Interval = 0x01
_PARAM_STATUS_	Unicast_Echo_Interval = 0x00 Multicast_Echo_Interval = 0x01

Table 4.89: Common Get/Set/Status Behavior Values for the Path Echo Interval state, Change Only Multicast

4.19.1.2 Common State Get/Set Behavior - Invalid Parameters

- Test Purpose

  Verify that a Directed Forwarding Server IUT ignores a message with invalid field values sent for the _STATE_.

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

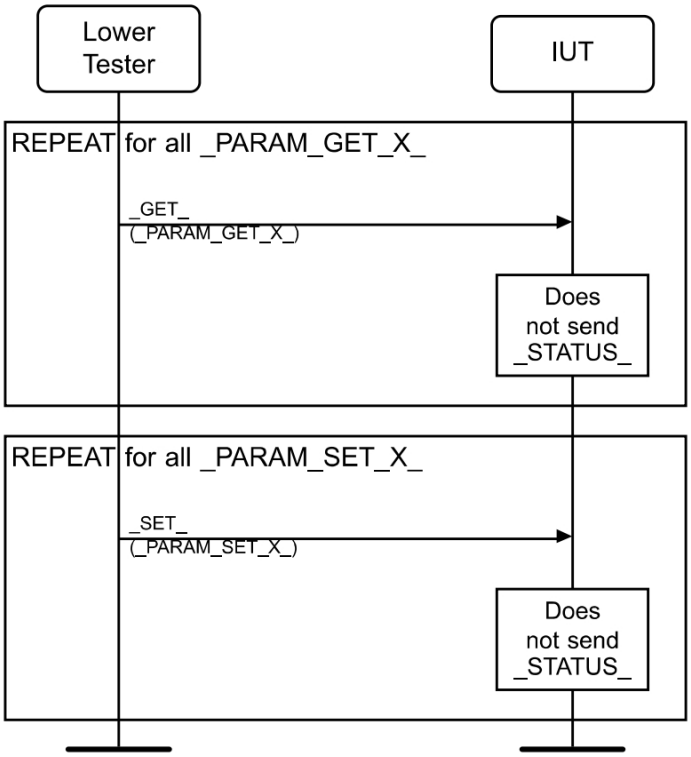


Figure 4.276: Test procedure for Common State Get/Set Behavior with Invalid Parameters on Directed Forwarding Configuration Server models MSC

1. The Lower Tester sends a _GET_ message with _PARAM_GET_0_.
2. The IUT does not respond with a _STATUS_ message.
3. Repeat Steps 1 and 2 for additional parameters defined in _PARAM_GET_X_.
4. The Lower Tester sends a _SET_ message with parameters set to as defined in _PARAM_SET_0_.
5. The IUT does not respond with a _STATUS_ message.
6. Repeat Steps 4 and 5 for additional parameters defined in _PARAM_SET_X_.

- Expected Outcome

  Pass verdict

  The IUT does not respond with a _STATUS_ message.

4.19.1.2.1 Directed Control State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Directed Control state
GET	DIRECTED_CONTROL_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	DIRECTED_CONTROL_STATUS
SET	DIRECTED_CONTROL_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0xFF Directed_Friend = 0x00
_PARAM_SET_1_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x01 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0xFF Directed_Friend = 0x00
_PARAM_SET_2_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x01 Directed_Proxy_Use_Directed_Default = 0x00 Directed_Friend = 0x00
_PARAM_SET_3_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0xFF Directed_Friend = 0x01
_PARAM_SET_4_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0x00 Directed_Friend = 0x00

State/Message/Parameter	Value(s)
_PARAM_SET_5_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x01 Directed_Relay = 0x00 Directed_Proxy = 0x01 Directed_Proxy_Use_Directed_Default = 0xFE Directed_Friend = 0x00

Table 4.90: Common Get/Set Behavior with invalid parameters for the Directed Control state

4.19.1.2.2 Path Metric State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Path Metric state
GET	PATH_METRIC_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	PATH_METRIC_STATUS
SET	PATH_METRIC_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Path_Metric_Type = 0b000 Path_Lifetime = 0b00
_PARAM_SET_1_	NetKeyIndex = 0x0000 Path_Metric_Type = 0b001 Path_Lifetime = 0b00
_PARAM_SET_2_	NetKeyIndex = 0x0000 Path_Metric_Type = 0b000 Path_Lifetime = 0b00 Prohibited = 0b111

Table 4.91: Common Get/Set Behavior with invalid parameters for the Path Metric state

4.19.1.2.3 Discovery Table Capabilities State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Discovery Table Capabilities state
GET	DISCOVERY_TABLE_CAPABILITIES_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	DISCOVERY_TABLE_CAPABILITIES_STATUS
SET	DISCOVERY_TABLE_CAPABILITIES_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Max_Concurrent_Init = 0x01
_PARAM_SET_1_	NetKeyIndex = 0x0000 Max_Concurrent_Init = 0x00

Table 4.92: Common Get/Set Behavior with invalid parameters for the Discovery Table Capabilities state

4.19.1.2.4 Wanted Lanes State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Wanted Lanes state
GET	WANTED_LANES_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	WANTED_LANES_STATUS
SET	WANTED_LANES_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Wanted_Lanes = 0x04
_PARAM_SET_1_	NetKeyIndex = 0x0000 Wanted_Lanes = 0x00

Table 4.93: Common Get/Set Behavior with invalid parameters for the Wanted Lanes state

4.19.1.2.5 Two Way Path State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Two Way Path state
GET	TWO_WAY_PATH_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	TWO_WAY_PATH_STATUS
SET	TWO_WAY_PATH_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Two_Way_Path = 0b0
_PARAM_SET_1_	NetKeyIndex = 0x0000 Two_Way_Path = 0b1 Prohibited=0x7F

Table 4.94: Common Get/Set Behavior with invalid parameters for the Two Way Path state

4.19.1.2.6 Path Echo Interval State Get/Set Behavior - Invalid Intervals

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	PATH_ECHO_INTERVAL_STATUS
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0x01
_PARAM_SET_1_	NetKeyIndex = 0x0000 Unicast_Echo_Interval = 0xFE Multicast_Echo_Interval = 0xFE

Table 4.95: Common Get/Set Behavior with invalid parameters for the Path Echo Interval state

4.19.1.2.7 Path Echo Interval State Get/Set Behavior - Invalid NetKeyIndex

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
_PARAM_GET_0_	NetKeyIndex = 0xFFFF
STATUS	PATH_ECHO_INTERVAL_STATUS
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_0_	NetKeyIndex = 0xFFFF Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0x01
_PARAM_SET_1_	NetKeyIndex = 0x0000 Unicast_Echo_Interval = 0xFF Multicast_Echo_Interval = 0xFF

Table 4.96: Common Get/Set Behavior with invalid parameters for the Path Echo Interval state

4.19.1.2.8 Directed Publish Policy State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Directed Publish Policy state
GET	DIRECTED_PUBLISH_POLICY_GET
_PARAM_GET_0_	Element_Addr = random valid group address Model_ID= valid supported Model ID
STATUS	DIRECTED_PUBLISH_POLICY_STATUS
SET	DIRECTED_PUBLISH_POLICY_SET
_PARAM_SET_0_	Directed_Publish_Policy = 0x00 Element_Addr = random valid group address Model_ID= valid supported Model ID

Table 4.97: Common Get/Set Behavior with invalid parameters for the Directed Publish Policy state

4.19.1.2.9 Path Discovery Timing Control State Get/Set Behavior - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Path Discovery Timing Control state
GET	PATH_DISCOVERY_TIMING_CONTROL_GET
STATUS	PATH_DISCOVERY_TIMING_CONTROL_STATUS
SET	PATH_DISCOVERY_TIMING_CONTROL_SET
_PARAM_SET_0_	Prohibited bits set to '1'

Table 4.98: Common Get/Set Behavior with invalid parameters for the Path Discovery Timing Control state

4.19.1.3 Common Node State Get/Set/Status Behavior

- Test Purpose

  This test procedure verifies common Get/Set/Status behavior for a Server Model on the IUT supporting the _STATE_ state.

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured with the default values for the _STATE_.

- Test Procedure

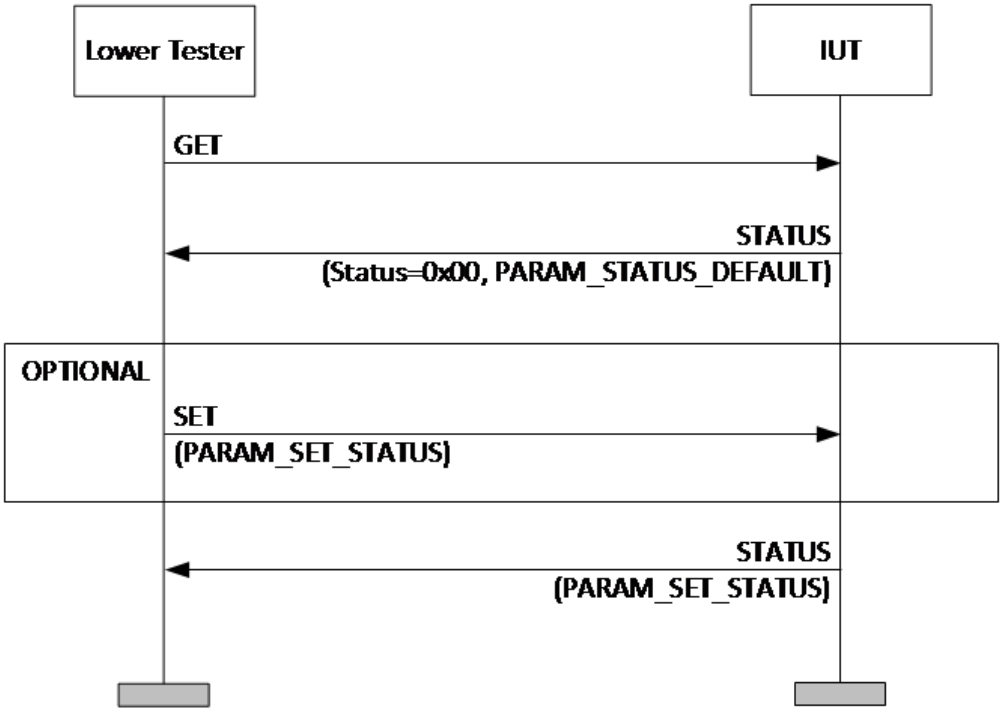


Figure 4.277: Test procedure for Common Node State Get/Set/Status Behavior on Directed Forwarding Configuration Server models MSC

1. The Lower Tester sends a _GET_ message.
2. The IUT responds with a _STATUS_ message with Status (if applicable) set to 0x00 (Success) and other parameters defined by _PARAM_STATUS_DEFAULT_.
3. If the _SET_ message is defined, the Lower Tester sends a _SET_ message parameters defined by _PARAM_SET_STATUS_, otherwise skip Step 4.
4. The IUT responds with a _STATUS_ message with Status set to 0x00 (Success) and other parameters defined by _PARAM_SET_STATUS_.

- Expected Outcome

  Pass verdict

  The IUT responds with a _STATUS_ message with the expected fields. The messages are encrypted with the device.

4.19.1.3.1 Directed Network Transmit State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Directed Network Transmit state
GET	DIRECTED_NETWORK_TRANSMIT_GET
STATUS	DIRECTED_NETWORK_TRANSMIT_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Network_Transmit_Count = 0b001 Directed_Network_Transmit_Interval_Steps = 0b01001
SET	DIRECTED_NETWORK_TRANSMIT_SET
_PARAM_SET_STATUS_	Directed_Network_Transmit_Count = 0b010 Directed_Network_Transmit_Interval_Steps = 0b10000

Table 4.99: Common Get/Set/Status Behavior Values for the Directed Network Transmit state

4.19.1.3.2 Directed Relay Retransmit State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Directed Relay Retransmit state
GET	DIRECTED_RELAY_RETRANSMIT_GET
STATUS	DIRECTED_RELAY_RETRANSMIT_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Relay_Retransmit_Count = 0b010 Directed_Relay_Retransmit_Interval_Steps = 0b01001
SET	DIRECTED_RELAY_RETRANSMIT_SET
_PARAM_SET_STATUS_	Directed_Relay_Retransmit_Count = 0b011 Directed_Relay_Retransmit_Interval_Steps = 0b10000

Table 4.100: Common Get/Set/Status Behavior Values for the Directed Relay Retransmit state

4.19.1.3.3 RSSI Threshold State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	RSSI Threshold state
GET	RSSI_THRESHOLD_GET
STATUS	RSSI_THRESHOLD_STATUS
_PARAM_STATUS_DEFAULT_	Default_RSSI_Threshold = any valid value RSSI_Margin = 0x14
SET	RSSI_THRESHOLD_SET
_PARAM_SET_STATUS_	Default_RSSI_Threshold = any valid value (Not included in _SET_ message) RSSI_Margin = 0x01

Table 4.101: Common Get/Set/Status Behavior Values for the RSSI Threshold state

4.19.1.3.4 Directed Paths State Get/Status

State/Message/Parameter	Value(s)
STATE	Directed Paths state
GET	DIRECTED_PATHS_GET
STATUS	DIRECTED_PATHS_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Node_Paths = any value greater or equal to 20 Directed_Relay_Paths = any value greater or equal to 20 Directed_Friend_Paths = any value greater or equal to 20 if the IUT supports directed friend functionality, otherwise 0x00. Directed_Proxy_Paths = any value greater or equal to 20 if the IUT supports directed proxy functionality, otherwise 0x00
SET	Not defined

Table 4.102: Common Get/Status Behavior Values for the Directed Paths state

4.19.1.3.5 Path Discovery Timing Control State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Path Discovery Timing Control state
GET	PATH_DISCOVERY_TIMING_CONTROL_GET
STATUS	PATH_DISCOVERY_TIMING_CONTROL_STATUS
_PARAM_STATUS_DEFAULT_	Path_Monitoring_Interval = 120 Path_Discovery_Retry_Interval = 300 Path_Discovery_Interval = 0b1 Lane_Discovery_Guard_Interval = 0b1
SET	PATH_DISCOVERY_TIMING_CONTROL_SET
_PARAM_SET_STATUS_	Path_Monitoring_Interval = 0x0000 Path_Discovery_Retry_Interval = 0x0000 Path_Discovery_Interval = 0b0 Lane_Discovery_Guard_Interval = 0b0

Table 4.103: Common Get/Set/Status Behavior Values for the Path Discovery Timing Control state

4.19.1.3.6 Directed Control Network Transmit State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Directed Control Network Transmit state
GET	DIRECTED_CONTROL_NETWORK_TRANSMIT_GET
STATUS	DIRECTED_CONTROL_NETWORK_TRANSMIT_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Control_Network_Transmit_Count = 0b001 Directed_Control_Network_Transmit_Interval_Steps = 0b01001
SET	DIRECTED_CONTROL_NETWORK_TRANSMIT_SET
_PARAM_SET_STATUS_	Directed_Control_Network_Transmit_Count = 0b010 Directed_Control_Network_Transmit_Interval_Steps = 0b10000

Table 4.104: Common Get/Set/Status Behavior Values for the Directed Control Network Transmit state

4.19.1.3.7 Directed Control Relay Retransmit State Get/Set/Status

State/Message/Parameter	Value(s)
STATE	Directed Control Relay Retransmit state
GET	DIRECTED_CONTROL_RELAY_RETRANSMIT_GET
STATUS	DIRECTED_CONTROL_RELAY_RETRANSMIT_STATUS
_PARAM_STATUS_DEFAULT_	Directed_Control_Relay_Retransmit_Count = 0b010 Directed_Control_Relay_Retransmit_Interval_Steps = 0b01001
SET	DIRECTED_CONTROL_RELAY_RETRANSMIT_SET
_PARAM_SET_STATUS_	Directed_Control_Relay_Retransmit_Count = 0b011 Directed_Control_Relay_Retransmit_Interval_Steps = 0b10000

Table 4.105: Common Get/Set/Status Behavior Values for the Directed Control Relay Retransmit state

4.19.2 Common Directed Forwarding Configuration Model Client Procedures

This section contains tests that verify common IUT functionality related to the Directed Forwarding Configuration Client model.

4.19.2.1 Common Get/Set/Status Procedure

- Test Purpose

  This test procedure verifies common Get/Set/Status behavior for a client model on the IUT supporting the _STATE_.

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

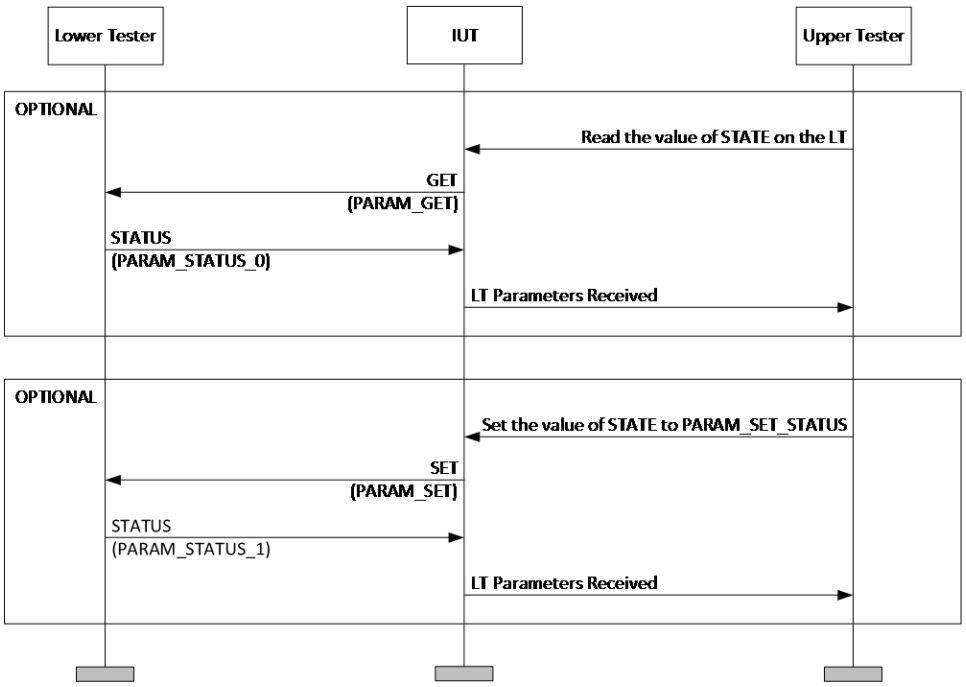


Figure 4.278: Test procedure for Common Get/Set/Status procedure on Directed Forwarding Configuration Client models MSC

1. If a _GET_ message is defined, the Upper Tester orders the IUT to read the value of the _STATE_ on the Lower Tester, otherwise skip Steps 2–4.
2. The IUT sends a _GET_ message with the parameters defined by _PARAM_GET_ if available.
3. The Lower Tester responds with a _STATUS_ message with parameters defined by _PARAM_STATUS_0_.
4. The IUT reports to the Upper Tester that it has received the parameters defined by _PARAM_STATUS_0_sent by the Lower Tester.
5. If a _SET_ message is defined, the Upper Tester orders the IUT to set the value of the _STATE_ on the Lower Tester to parameters defined by _PARAM_SET_STATUS_, otherwise skip Steps 6–8.
6. The IUT sends a _SET_ message with parameters defined by _PARAM_SET _.
7. The Lower Tester responds with a _STATUS_ message with parameters defined by _PARAM_STATUS_1_.
8. The IUT reports to the Upper Tester that it has received the parameters defined by _PARAM_STATUS_1_ sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the _GET_ or _SET_ messages with the expected fields. The messages are encrypted with the device key.

  The IUT reports the expected values sent by the Lower Tester.

4.19.2.1.1 Directed Control State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Control state
GET	DIRECTED_CONTROL_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	DIRECTED_CONTROL_STATUS
_PARAM_STATUS_0_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x00 Directed_Proxy = 0x00 Directed_Friend = 0x00
SET	DIRECTED_CONTROL_SET
_PARAM_SET_,	NetKeyIndex = 0x0000 Directed_Forwarding = 0x01 Directed_Relay = 0x01 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0xFF Directed_Friend = 0x00
_PARAM_STATUS_1_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x01 Directed_Relay = 0x01 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0x02 (Disabled or not supported) Directed_Friend = 0x00

Table 4.106: Common Get/Set/Status procedure values for the Directed Control state

4.19.2.1.2 Path Metric State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Path Metric state
GET	PATH_METRIC_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	PATH_METRIC_STATUS
_PARAM_STATUS_0_	NetKeyIndex = 0x0000 Path_Metric_Type = 0b000 Path_Lifetime = 0b10
SET	PATH_METRIC_SET
_PARAM_SET_, _PARAM_STATUS_1_	NetKeyIndex = 0x0000 Path_Metric_Type = 0b000 Path_Lifetime = 0b00

Table 4.107: Common Get/Set/Status procedure values for the Path Metric state

4.19.2.1.3 Discovery Table Capabilities State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Discovery Table Capabilities state
GET	DISCOVERY_TABLE_CAPABILITIES_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	DISCOVERY_TABLE_CAPABILITIES_STATUS
_PARAM_STATUS_0_	Status = 0x00 (Success) NetKeyIndex = 0x0000 Max_Concurrent_Init = 0x02 Max_Discovery_Table_Entries_Count = 0x02
SET	DISCOVERY_TABLE_CAPABILITIES_SET
_PARAM_SET_	NetKeyIndex = 0x0000 Max_Concurrent_Init = 0x01
_PARAM_STATUS_1_	Status = 0x00 (Success) NetKeyIndex = 0x0000 Max_Concurrent_Init = 0x01 Max_Discovery_Table_Entries_Count = 0x02

Table 4.108: Common Get/Set/Status procedure values for the Discovery Table Capabilities state

4.19.2.1.4 Wanted Lanes State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Wanted Lanes state
GET	WANTED_LANES_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	WANTED_LANES_STATUS
_PARAM_STATUS_0_	NetKeyIndex = 0x0000 Wanted_Lanes = 0x01
SET	WANTED_LANES_SET
_PARAM_SET_, _PARAM_STATUS_1_	NetKeyIndex = 0x0000 Wanted_Lanes = 0x04

Table 4.109: Common Get/Set/Status procedure values for the Wanted Lanes state

4.19.2.1.5 Two Way Path State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Two Way Path state
GET	TWO_WAY_PATH_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	TWO_WAY_PATH_STATUS
_PARAM_STATUS_0_	NetKeyIndex = 0x0000 Two_Way_Path = 0b1
SET	TWO_WAY_PATH_SET
_PARAM_SET_, _PARAM_STATUS_1_	NetKeyIndex = 0x0000 Two_Way_Path = 0b0

Table 4.110: Common Get/Set/Status procedure values for the Two Way Path state

4.19.2.1.6 Path Echo Interval State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	PATH_ECHO_INTERVAL_STATUS
_PARAM_STATUS_0_	NetKeyIndex = 0x0000 Unicast_Echo_Interval = 0xFF Multicast_Echo_Interval = 0xFF
SET	PATH_ECHO_INTERVAL_SET
_PARAM_SET_, _PARAM_STATUS_1_	NetKeyIndex = 0x0000 Unicast_Echo_Interval = 0x01 Multicast_Echo_Interval = 0x01

Table 4.111: Common Get/Set/Status procedure values for the Path Echo Interval state

4.19.2.1.7 Directed Network Transmit State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Network Transmit state
GET	DIRECTED_NETWORK_TRANSMIT_GET
STATUS	DIRECTED_NETWORK_TRANSMIT_STATUS
_PARAM_STATUS_0_	Directed_Network_Transmit_Count = 0b001 Directed_Network_Transmit_Interval_Steps = 0b01001
SET	DIRECTED_NETWORK_TRANSMIT_SET
_PARAM_SET_, _PARAM_STATUS_1_	Directed_Network_Transmit_Count = 0b010 Directed_Network_Transmit_Interval_Steps = 0b10000

Table 4.112: Common Get/Set/Status procedure values for the Directed Network Transmit state

4.19.2.1.8 Directed Relay Retransmit State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Relay Retransmit state
GET	DIRECTED_RELAY_RETRANSMIT_GET
STATUS	DIRECTED_RELAY_RETRANSMIT_STATUS
_PARAM_STATUS_0_	Directed_Relay_Retransmit_Count = 0b010 Directed_Relay_Retransmit_Interval_Steps = 0b01001
SET	DIRECTED_REALAY_TRANSMIT_SET
_PARAM_SET_, _PARAM_STATUS_1_	Directed_Relay_Retransmit_Count = 0b010 Directed_Relay_Retransmit_Interval_Steps = 0b10000

Table 4.113: Common Get/Set/Status procedure values for the Directed Relay Retransmit state

4.19.2.1.9 RSSI Threshold State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	RSSI Threshold state
GET	RSSI_THRESHOLD_GET
STATUS	RSSI_THRESHOLD_STATUS
_PARAM_STATUS_0_	Status = 0x00 Default_RSSI_Threshold = any valid value RSSI_Margin = {0x00-0x32}
SET	RSSI_THRESHOLD_SET
_PARAM_SET_	RSSI_Margin = {0x00-0x32}
_PARAM_STATUS_1_	Status = 0x00 Default_RSSI_Threshold = any valid value RSSI_Margin = {0x00-0x32}

Table 4.114: Common Get/Set/Status procedure values for the RSSI Threshold state

4.19.2.1.10 Directed Paths State Get/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Paths state
GET	DIRECTED_PATHS_GET
STATUS	DIRECTED_PATHS_STATUS
_PARAM_STATUS_0_	Directed_Node_Paths = random value >= 20 Directed_Relay_Paths = random value >= 20 Directed_Proxy_Paths = random value >= 20, Directed_Friend_Paths = random value >= 20

Table 4.115: Common Get/Status procedure values for the Directed Paths state

4.19.2.1.11 Directed Publish Policy Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Publish Policy state
GET	DIRECTED_PUBLISH_POLICY_GET
_PARAM_GET_	Element_Addr = _LT_ADDR_ Model_ID = random valid value
STATUS	DIRECTED_PUBLISH_POLICY_STATUS
_PARAM_STATUS_0_	Status = 0x00 Element_Addr = _LT_ADDR_ Directed_Publish_Policy = 0x00 Model_ID = random valid value
SET	DIRECTED_PUBLISH_POLICY_SET
_PARAM_SET _	Element_Addr = _LT_ADDR_ Directed_Publish_Policy = 0x01 Model_ID = random valid value
_PARAM_STATUS_1_	Status = 0x00 Element_Addr = _LT_ADDR_ Directed_Publish_Policy = 0x01 Model_ID = random valid value

Table 4.116: Common Get/Set/Status procedure values for the Directed Publish Policy state

4.19.2.1.12 Path Discovery Timing Control Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Path Discovery Timing Control state
GET	PATH_DISCOVERY_TIMING_CONTROL_GET
STATUS	PATH_DISCOVERY_TIMING_CONTROL_STATUS
_PARAM_STATUS_0_	Path_Monitoring_Interval = random valid value Path_Discovery_Retry_Interval = random valid value Path_Discovery_Interval = 0b0 or 0b1 Lane_Discovery_Guard_Interval = 0b0 or 0b1
SET	PATH_DISCOVERY_TIMING_CONTROL_SET
_PARAM_SET_, _PARAM_STATUS_1_	Path_Monitoring_Interval = random valid value Path_Discovery_Retry_Interval = random valid value Path_Discovery_Interval = 0b0 or 0b1 Lane_Discovery_Guard_Interval = 0b0 or 0b1

Table 4.117: Common Get/Set/Status procedure values for the Path Discovery Timing Control state

4.19.2.1.13 Directed Control Network Transmit State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Control Network Transmit state
GET	DIRECTED_CONTROL_NETWORK_TRANSMIT_GET
STATUS	DIRECTED_CONTROL_NETWORK_TRANSMIT_STATUS
_PARAM_STATUS_0_	Directed_Control_Network_Transmit_Count = 0b001 Directed_Control_Network_Transmit_Interval_Steps = 0b01001
SET	DIRECTED_CONTROL_NETWORK_TRANSMIT_SET
_PARAM_SET_, _PARAM_STATUS_1_	Directed_Control_Network_Transmit_Count = 0b010 Directed_Control_Network_Transmit_Interval_Steps = 0b10000

Table 4.118: Common Get/Set/Status procedure values for the Directed Control Network Transmit state

4.19.2.1.14 Directed Control Relay Retransmit State Get/Set/Status Procedure

State/Message/Parameter	Value(s)
STATE	Directed Control Relay Retransmit state
GET	DIRECTED_CONTROL_RELAY_RETRANSMIT_GET
STATUS	DIRECTED_CONTROL_RELAY_RETRANSMIT_STATUS
_PARAM_STATUS_0_	Directed_Control_Relay_Retransmit_Count = 0b010 Directed_Control_Relay_Retransmit_Interval_Steps = 0b01001
SET	DIRECTED_CONTROL_RELAY_TRANSMIT_SET
_PARAM_SET_, _PARAM_STATUS_1_	Directed_Control_Relay_Retransmit_Count = 0b010 Directed_Control_Relay_Retransmit_Interval_Steps = 0b10000

Table 4.119: Common Get/Set/Status procedure values for the Directed Control Relay Retransmit state

4.19.2.2 Common Get/Set/Status Procedure - Invalid Parameters

- Test Purpose

  This test procedure verifies the behavior of an IUT acting as a Directed Forwarding client when receiving a _STATUS_ message with invalid parameters.

- Initial Condition
  - The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure

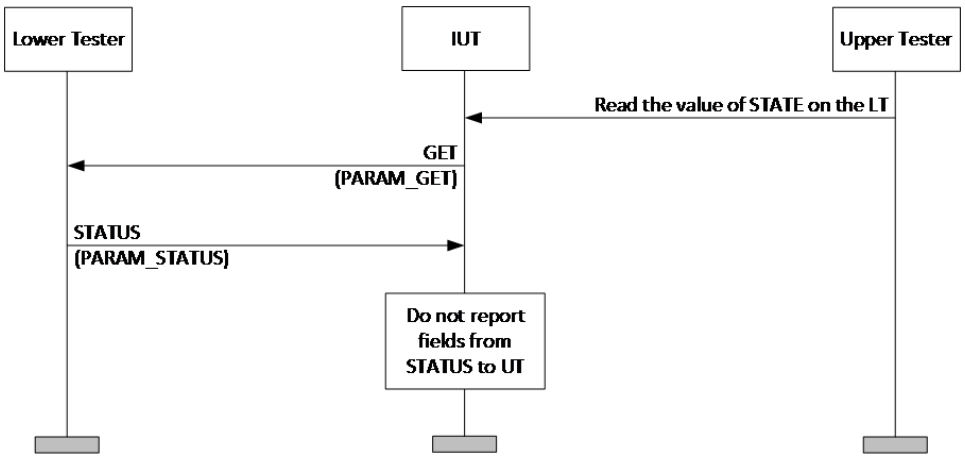


Figure 4.279: Test procedure for Common Receive Status Procedure on Directed Forwarding Configuration Client models MSC

1. The Upper Tester orders the IUT to read the value of the _STATE_ used on the primary subnet on the Lower Tester.
2. The IUT sends a _GET_ message with parameters defined by _ PARAM_ GET_.
3. The Lower Tester responds with a _STATUS_ message with parameters defined by _PARAM_STATUS_.
4. The IUT does not report to the Upper Tester any fields from the _STATUS_ message.

- Expected Outcome

  Pass verdict

  The IUT sends the _GET_ messages with the expected fields. The messages are encrypted with the device key.

  The IUT does not report the expected values sent by the Lower Tester.

4.19.2.2.1 Directed Control Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Directed Control state
GET	DIRECTED_CONTROL_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	DIRECTED_CONTROL_STATUS
_PARAM_STATUS_	NetKeyIndex = 0x0000 Directed_Forwarding = 0x00 Directed_Relay = 0x01 Directed_Proxy = 0x00 Directed_Proxy_Use_Directed_Default = 0x02 (Disabled or not supported) Directed_Friend = 0x00

Table 4.120: Common Status with Invalid Parameters procedure values for the Directed Control state

4.19.2.2.2 Path Metric Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Path Metric state
GET	PATH_METRIC_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	PATH_METRIC_STATUS
_PARAM_STATUS_	NetKeyIndex = 0x0000 Path_Metric_Type = 0b000 Path_Lifetime = 0b10 Prohibited = 0b111

Table 4.121: Common Status with Invalid Parameters procedure values for the Path Metric state

4.19.2.2.3 Discovery Table Capabilities Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Discovery Table Capabilities state
GET	DISCOVERY_TABLE_CAPABILITIES_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	DISCOVERY_TABLE_CAPABILITIES_STATUS
_PARAM_STATUS_	Status = 0x00 (Success) NetKeyIndex = 0x0000 Max_Concurrent_Init = 0x00 Max_Discovery_Table_Entries_Count = 0x02

Table 4.122: Common Status with Invalid Parameters procedure values for the Discovery Table Capabilities state

4.19.2.2.4 Wanted Lanes Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Wanted Lanes state
GET	WANTED_LANES_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	WANTED_LANES_STATUS
_PARAM_STATUS_	NetKeyIndex = 0x0000 Wanted_Lanes = 0x00

Table 4.123: Common Status with Invalid Parameters procedure values for the Wanted Lanes state

4.19.2.2.5 Two Way Path Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Two Way Path state
GET	TWO_WAY_PATH_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	TWO_WAY_PATH_STATUS
_PARAM_STATUS_	NetKeyIndex = 0x0000 Two_Way_Path = 0b1 Prohibited = 0b1111111

Table 4.124: Common Status with Invalid Parameters procedure values for the Two Way Path state

4.19.2.2.6 Path Echo Interval Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Path Echo Interval state
GET	PATH_ECHO_INTERVAL_GET
_PARAM_GET_	NetKeyIndex = 0x0000
STATUS	PATH_ECHO_INTERVAL_STATUS
_PARAM_STATUS_	NetKeyIndex = 0x0000 Unicast_Echo_Interval = 0xFE Multicast_Echo_Interval = 0xFE

Table 4.125: Common Status with Invalid Parameters procedure values for the Path Echo Interval state

4.19.2.2.7 RSSI Threshold Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	RSSI Threshold state
GET	RSSI_THRESHOLD_GET
STATUS	RSSI_THRESHOLD _STATUS
_PARAM_STATUS_	Status = 0x00 (Success) Default_RSSI_Threshold = 0x00 RSSI_Margin = 0xFF

Table 4.126: Common Status with Invalid Parameters procedure values for the RSSI Threshold state

4.19.2.2.8 Directed Paths Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Directed Paths state
GET	DIRECTED_PATHS_GET
STATUS	DIRECTED_PATHS _STATUS
_PARAM_STATUS_	Directed_Node_Paths = 0x00 Directed_Relay_Paths = 0x00 Directed_Proxy_Paths = 0x00 Directed_Friend_Paths = 0x00

Table 4.127: Common Status with Invalid Parameters procedure values for the Directed Paths state

4.19.2.2.9 Directed Publish Policy Status Procedure - Invalid Parameters

State/Message/Parameter	Value(s)
STATE	Directed Publish Policy state
GET	PUBLISH_POLICY_GET
_GET_PARAM_	Element_Addr = _LT_ADDR_ Model_ID = any supported client model identifier
STATUS	DIRECTED_PUBLISH_POLICY _STATUS
_PARAM_STATUS_	Status = 0x00 (Success) Directed_Publish_Policy = 0xFF Element_Addr = _LT_ADDR_ Model_ID = value from _GET_PARAM_

Table 4.128: Common Status with Invalid Parameters procedure values for the Directed Publish Policy state

4.19.3 Directed Control State

This section validates the Directed Control state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DFS/BV-01-C [Directed Control Get/Set/Status Behavior]

- Reference

  [10] 4.2.22, 4.4.5.6.1

Execute the common test case using the values from Section 4.19.1.1.1.

4.19.3.1 Directed Control Set Behavior - Bindings

- Test Purpose

  Verify the binding behavior between the Directed Proxy, Directed Friend, and Directed Forwarding states on an IUT supporting the Directed Forwarding Configuration Server model.

- Reference

[10] 4.2.22, 4.4.5.6.1

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Additional initial conditions defined in Table 4.129 apply.
- Test Procedure

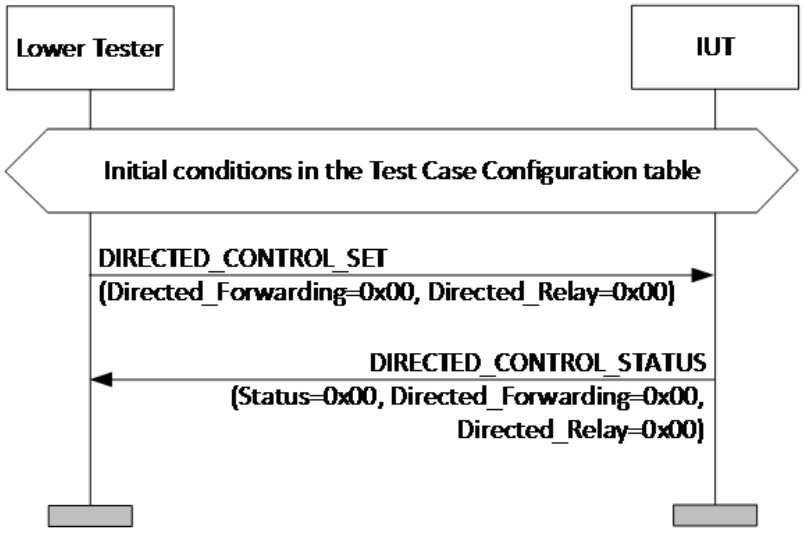


Figure 4.280: Directed Control Set Behavior - Bindings MSC

1. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x00 (Disabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and (Directed_Friend set to 0xFF (Ignore).
2. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x00 (Disabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and other fields set as described in Table 4.129.

- Test Case Configuration

TCID	Additional Initial Condition	Step 2
MESH/SR/DFM/DFS/BV-02-C [Directed Control Set Behavior – Binding with Directed Friend]	The directed friend functionality on the primary subnet is enabled on the IUT.	Directed_Proxy = 0x00 or 0x02 (if directed proxy functionality is not supported) Directed_Friend = 0x00
MESH/SR/DFM/DFS/BV-03-C [Directed Control Set Behavior – Binding with Directed Proxy]	Directed Proxy and Directed Proxy Use Directed Default states on the primary subnet are enabled on the IUT.	Directed_Proxy = 0x00 Directed_Friend = 0x00 or 0x02 (if directed friend functionality is not supported)

Table 4.129: Additional test conditions for Directed Control Set Behavior - Bindings

- Expected Outcome

  Pass verdict

  The IUT responds with a DIRECTED_CONTROL_STATUS message with the expected fields. The messages are encrypted with the device key.

MESH/SR/DFM/DFS/BV-04-C [Directed Control State Behavior – IUT as Low Power Node]

- Test Purpose

  Verify that an IUT acting as a Low Power Node that supports the Directed Forwarding Configuration Server model disables directed forwarding when establishing friendship with a Friend node and restores the old value when the friendship is terminated.

- Reference

  [10] 3.6.6.4.1, 4.2.22

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

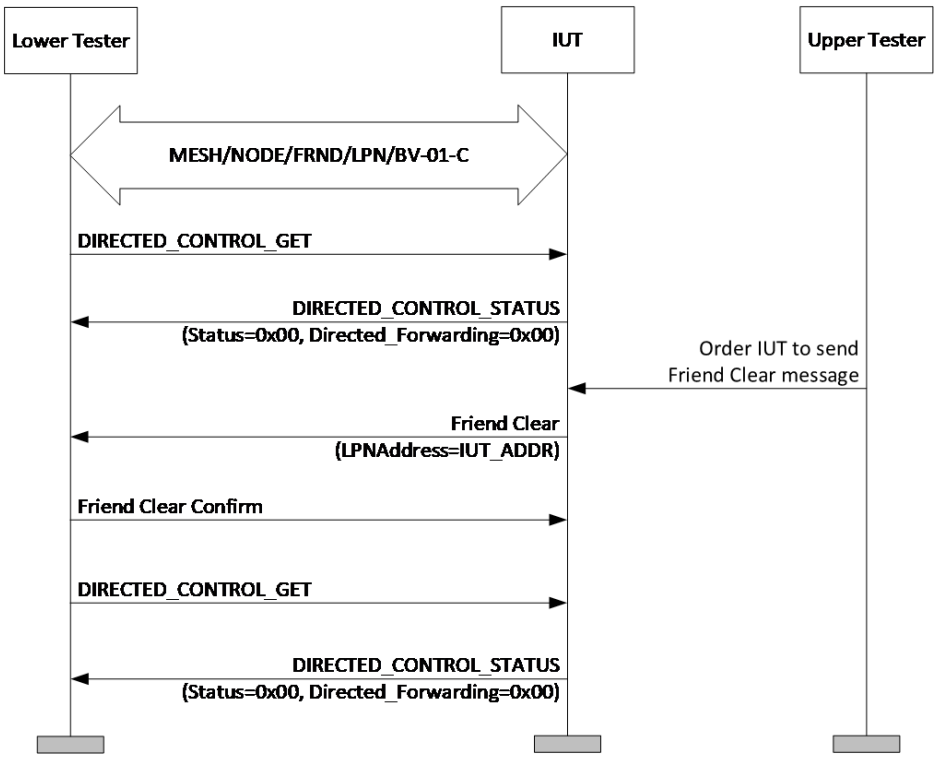


Figure 4.281: MESH/SR/DFM/DFS/BV-04-C MSC

1. The IUT establishes a friendship with the Lower Tester by running MESH/NODE/FRND/LPN/BV-01-C [Friendship Establishment Mode – Low Power Node].
2. The Lower Tester sends a DIRECTED_CONTROL_GET message with NetKeyIndex field set to 0x0000.
3. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x00 (Disabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x00 or 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).
4. The Upper Tester orders the IUT to send a Friend Clear message to the Lower Tester.
5. The IUT sends a Friend Clear message to the Lower Tester with the LPNAddress field set to _IUT_ADDR_.
6. The Lower Tester sends a Friend Clear Confirm message in response to the IUT.
7. The Lower Tester sends a DIRECTED_CONTROL_GET message with NetKeyIndex field set to 0x0000.
8. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).

- Expected Outcome

  Pass verdict

  The IUT responds with a DIRECTED_CONTROL_STATUS message with the expected fields. The messages are encrypted with the device key.

MESH/SR/DFM/DFS/BV-05-C [Directed Control State Behavior – Binding with GATT Proxy State]

- Test Purpose

  Verify that the binding behavior between the Directed Proxy and GATT Proxy states on an IUT supporting the Directed Forwarding Configuration Server model.

- Reference

  [10] 4.2.22, 4.4.5.6.1

- Initial Condition

  - Directed Proxy and Directed Proxy Use Directed Default states on the primary subnet are enabled on the IUT.

- Test Procedure

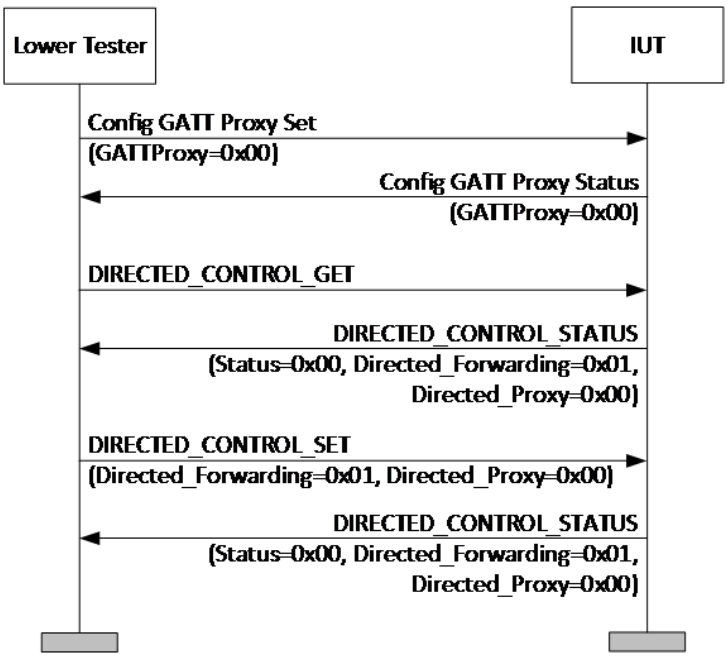


Figure 4.282: MESH/SR/DFM/DFS/BV-05-C [Directed Control State Behavior – Binding with GATT Proxy State] MSC

1. The Lower Tester sends a Config GATT Proxy Set message to the IUT with the GATTProxy field set to 0x00 (The Proxy feature is supported and disabled).
2. The IUT responds with a Config GATT Proxy Status message with the GATTProxy field set to 0x00 (The Proxy feature is supported and disabled).
3. The Lower Tester sends a DIRECTED_CONTROL_GET message with NetKeyIndex field set to 0x0000.
4. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 (Disabled), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).
5. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x01 (Enabled), Directed_Proxy_Use_Directed_Default set to 0x00 (Disable), and Directed_Friend set to 0xFF (Ignore).
6. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00, Directed_Proxy_Use_Directed_Default set to 0x00 (Disabled), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).

- Expected Outcome

  Pass verdict

The IUT responds with a DIRECTED_CONTROL_STATUS message with the expected fields. The messages are encrypted with the device key.

MESH/SR/DFM/DFS/BV-06-C [Directed Control State Behavior – Binding with Friend State]

- Test Purpose

  Verify the binding behavior between the Directed Proxy and Friend states on an IUT supporting the Directed Forwarding Configuration Server model.

- Reference

  [10] 4.2.22, 4.4.5.6.1

- Initial Condition
  - The directed friend functionality on the primary subnet is enabled on the IUT.

- Test Procedure

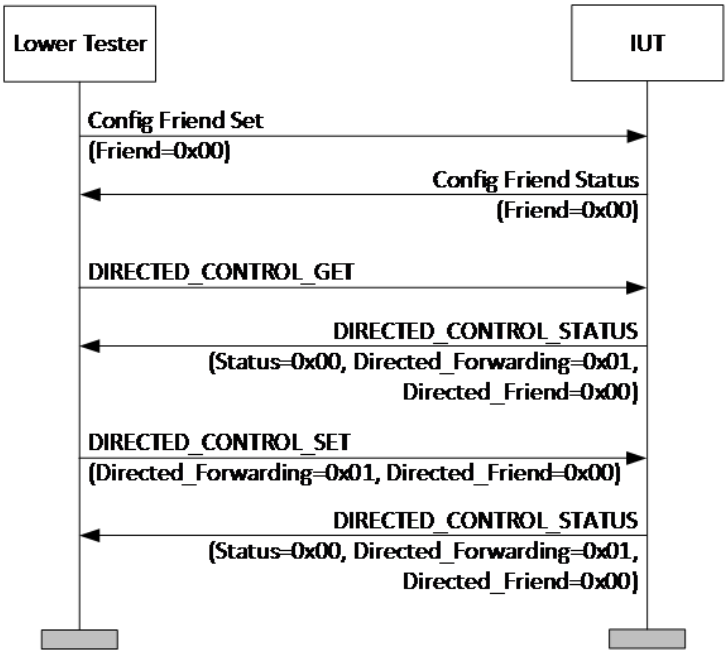


Figure 4.283: MESH/SR/DFM/DFS/BV-06-C [Directed Control State Behavior – Binding with Friend State] MSC

1. The Lower Tester sends a Config Friend Set message to the IUT with the Friend field set to 0x00 (The node supports Friend feature that is enabled).
2. The IUT responds with a Config Friend Status message with the Friend field set to 0x00 (The node supports Friend feature that is enabled).
3. The Lower Tester sends a DIRECTED_CONTROL_GET message with NetKeyIndex field set to 0x0000.

4. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 (Disabled).
5. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0x01 (Enabled).
6. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x01 (Enabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 (Enabled).

- Expected Outcome

  Pass verdict

  The IUT responds with a DIRECTED_CONTROL_STATUS message with the expected fields. The messages are encrypted with the device key.

MESH/SR/DFM/DFS/BI-01-C [Directed Forwarding State Get/Set Behavior – Invalid Parameters]

- Reference

  [10] 4.2.22, 4.4.5.6.1

Execute the common test case using the values from Section 4.19.1.2.1.

MESH/CL/DFM/DFS/BV-01-C [Send Directed Forwarding Messages Behavior]

- Reference

  [10] 4.2.22, 4.4.5.6.1

Execute the common test case using the values from Section 4.19.2.1.1 Send Directed Control Get/Set Behavior.

MESH/CL/DFM/DFS/BI-01-C [Directed Forwarding Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.22, 4.4.5.6.1

Execute the common test case using the values from Section 4.19.2.2.1.

4.19.4 Path Metric State

This section validates the Path Metric state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/PMS/BV-01-C [Path Metric State Get/Set/Status Behavior]

- Reference

  [10] 4.2.23, 4.4.5.6.2

Execute the common test case using the values from Section 4.19.1.1.2.

MESH/SR/DFM/PMS/BI-01-C [Path Metric State Get/Set Behavior – Invalid Parameters]

- Reference

  [10] 4.2.23, 4.4.5.6.2

Execute the common test case using the values from Section 4.19.1.2.2.

MESH/CL/DFM/PMS/BV-01-C [Path Metric Get/Set/Status Procedure]

- Reference

  [10] 4.2.23, 4.4.5.6.2

Execute the common test case using the values from Section 4.19.2.1.2.

MESH/CL/DFM/PMS/BI-01-C [Path Metric Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.23, 4.4.5.6.2

Execute the common test case using the values from Section 4.19.2.2.2.

MESH/CL/DFM/PMS/BI-02-C [Path Metric Status Behavior – RFU Bits]

- Reference

  [10] 4.2.23, 4.4.5.6.2

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

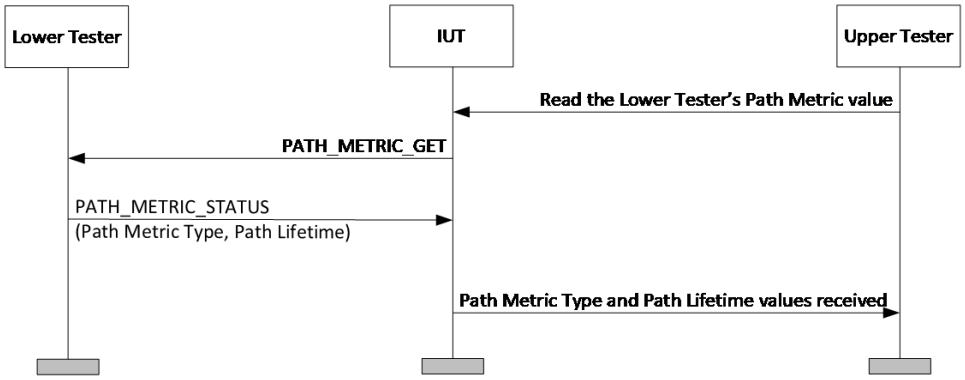


Figure 4.284: MESH/CL/DFM/PMS/BI-02-C

1. The Upper Tester orders the IUT to read the value of the Path Metric state used on the primary subnet on the Lower Tester.
2. The IUT sends a PATH_METRIC_GET message with NetKeyIndex field set to 0x0000.
3. The Lower Tester responds with a PATH_METRIC_STATUS message with NetKeyIndex field set to 0x0000, the Path Metric Type field set to a random RFU value, and the Path Lifetime field set to a random valid value.
4. The IUT reports to the Upper Tester that it has received the Path Metric Type and Path Lifetime values sent by the Lower Tester for the specified NetKeyIndex.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a PATH_METRIC_GET message.

  In Step 4, the IUT reports the expected values sent by the Lower Tester ignoring the RFU bits.

  The DF Path Metric messages are encrypted with the device key of the IUT.

4.19.5 Discovery Table Capabilities State

This section validates the Discovery Table Capabilities state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DTCS/BV-01-C [Discovery Table Capabilities Get/Set/Status Behavior]

- Reference

  [10] 4.2.24, 4.4.5.6.3

Execute the common test case using the values from Section 4.19.2.1.3.

MESH/SR/DFM/DTCS/BI-01-C [Discovery Table Capabilities State Get/Set Behavior – Invalid Parameters]

- Reference

[10] 4.2.24, 4.4.5.6.3

Execute the common test case using the values from Section 4.19.2.2.3.

MESH/CL/DFM/DTCS/BV-01-C [Discovery Table Capabilities Get/Set/Status Procedure]

- Reference

  [10] 4.2.24, 4.4.5.6.3

Execute the common test case using the values from Section 4.19.2.1.3.

MESH/CL/DFM/DTCS/BI-01-C [Discovery Table Capabilities Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.24, 4.4.5.6.3

Execute the common test case using the values from Section 4.19.2.2.3.

4.19.6 Forwarding Table State

This section validates the Forwarding Table state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/FTS/BV-01-C [Forwarding Table State Add Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_ADD and FORWARDING_TABLE_ENTRIES_COUNT_GET messages.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - If the IUT supports, it is configured on two subnets.

- Test Procedure

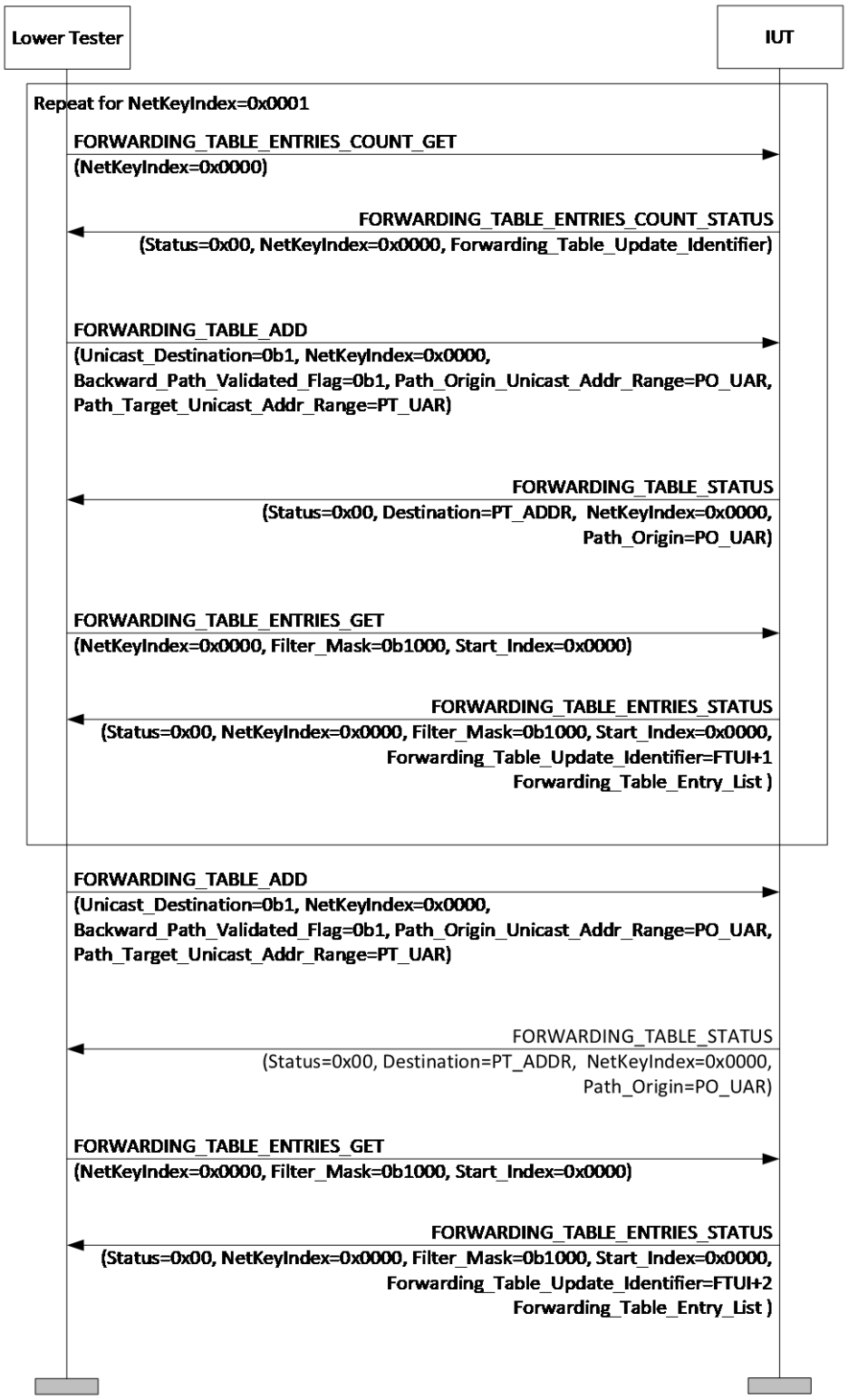


Figure 4.285: MESH/SR/DFM/FTS/BV-01-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.

2. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000. The Lower Tester stores the Forwarding_Table_Update_Identifier value.
3. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
4. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, Destination set to _PT_ADDR_.
5. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET with NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1, and Start_Index set to 0x0000.
6. The IUT responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to a different value than the stored value. Forwarding_Table_Entry_List fields contain the values from Table 4.130. The Lower Tester stores the new Forwarding_Table_Update_Identifier value.

Field	Value
Fixed Entry	
Forwarding_Table_Entry_Header	0x00DF
Path_Origin_Unicast_Addr_Range	_PO2_UAR_
Dependent_Origin_List_Size	0x01
Bearer_Toward_Path_Origin	0x0001
Path_Target_Unicast_Addr_Range	_PT2_UAR_
Bearer_Toward_Path_Target	0x0001

Table 4.130: Forwarding Table entry list contents for MESH/SR/DFM/FTS/BV-01-C, Step 6

7. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
8. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
9. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET with NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1, and Start_Index set to 0x0000.
10. The IUT responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to 0b1, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to a different value than the stored value.

Forwarding_Table_Entry_List fields contain the values from Table 4.131.

Field	Value
Fixed Entry	
Forwarding_Table_Entry_Header	0x001B
Path_Origin_Unicast_Addr_Range	_PO2_UAR_
Bearer_Toward_Path_Origin	0x0001
Path_Target_Unicast_Addr_Range	_PT2_UAR_
Bearer_Toward_Path_Target	0x0001

Table 4.131: Forwarding Table entry list contents for MESH/SR/DFM/FTS/BV-01-C, Step 10

11. If the IUT is configured on a second subnet, repeat Steps 1–6 with the NetKeyIndex field set to 0x0001. Otherwise, skip this step.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_STATUS and FORWARDING_TABLE_ENTRIES_COUNT_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-02-C [Forwarding Table State Delete Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to a FORWARDING_TABLE_DELETE message.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

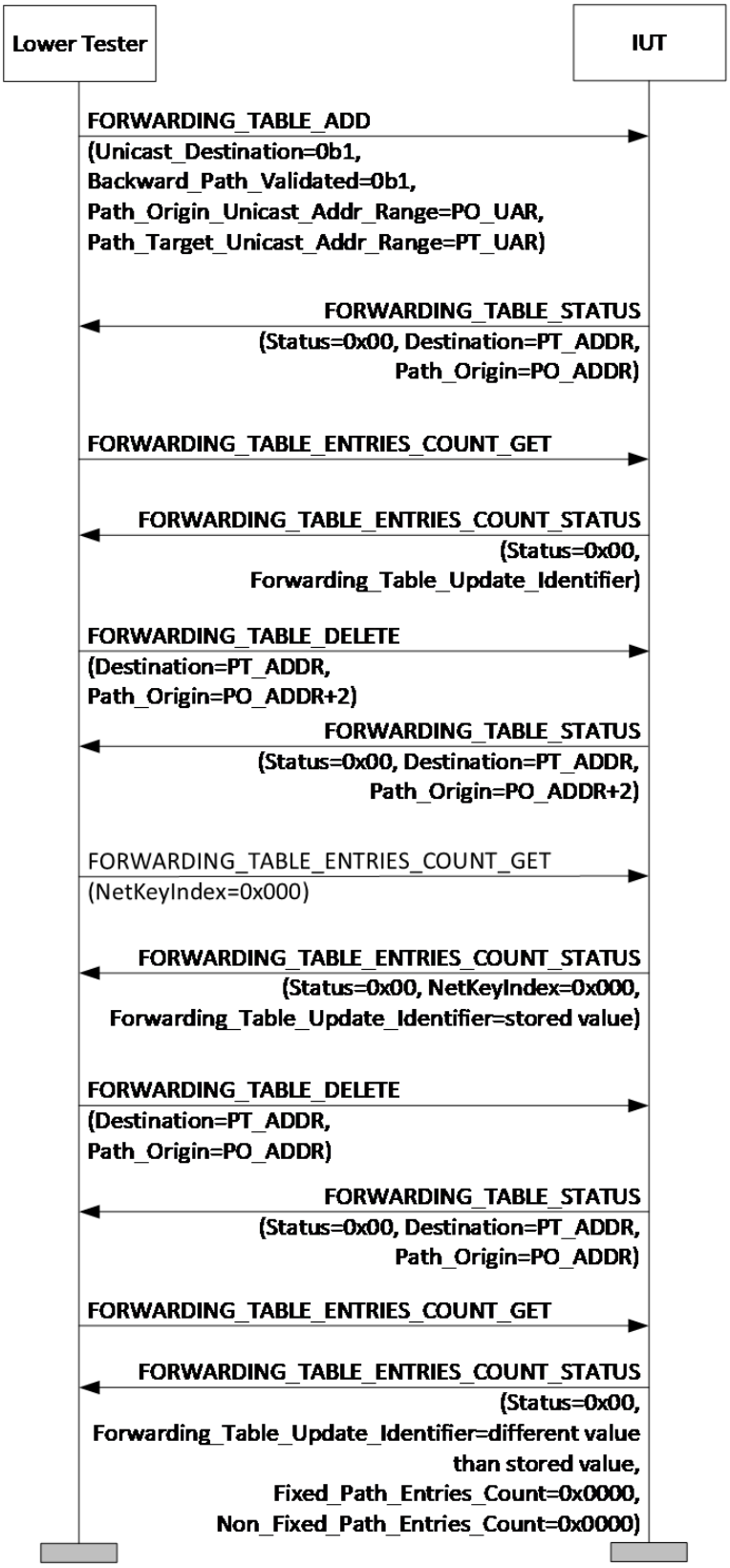


Figure 4.286: MESH/SR/DFM/FTS/BV-02-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
4. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0001, and Non_Fixed_Path_Entries_Count set to 0x0000. The Lower Tester stores the Forwarding_Table_Update_Identifier value.
5. The Lower Tester sends a FORWARDING_TABLE_DELETE message with NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_ + 2, and Destination set to _PT_ADDR_.
6. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_+ 2, and Destination set to _PT_ADDR_.
7. Repeat Steps 3 and 4 with Forwarding_Table_Update_Identifier set to the stored value.
8. The Lower Tester sends a FORWARDING_TABLE_DELETE message with NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
9. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
10. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
11. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a different value than the stored value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_STATUS and FORWARDING_TABLE_ENTRIES_COUNT_STATUS messages with valid values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-03-C [Forwarding Table State Dependents Add/Get Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_DEPENDENTS_ADD and FORWARDING_TABLE_DEPENDENTS_GET

messages.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure

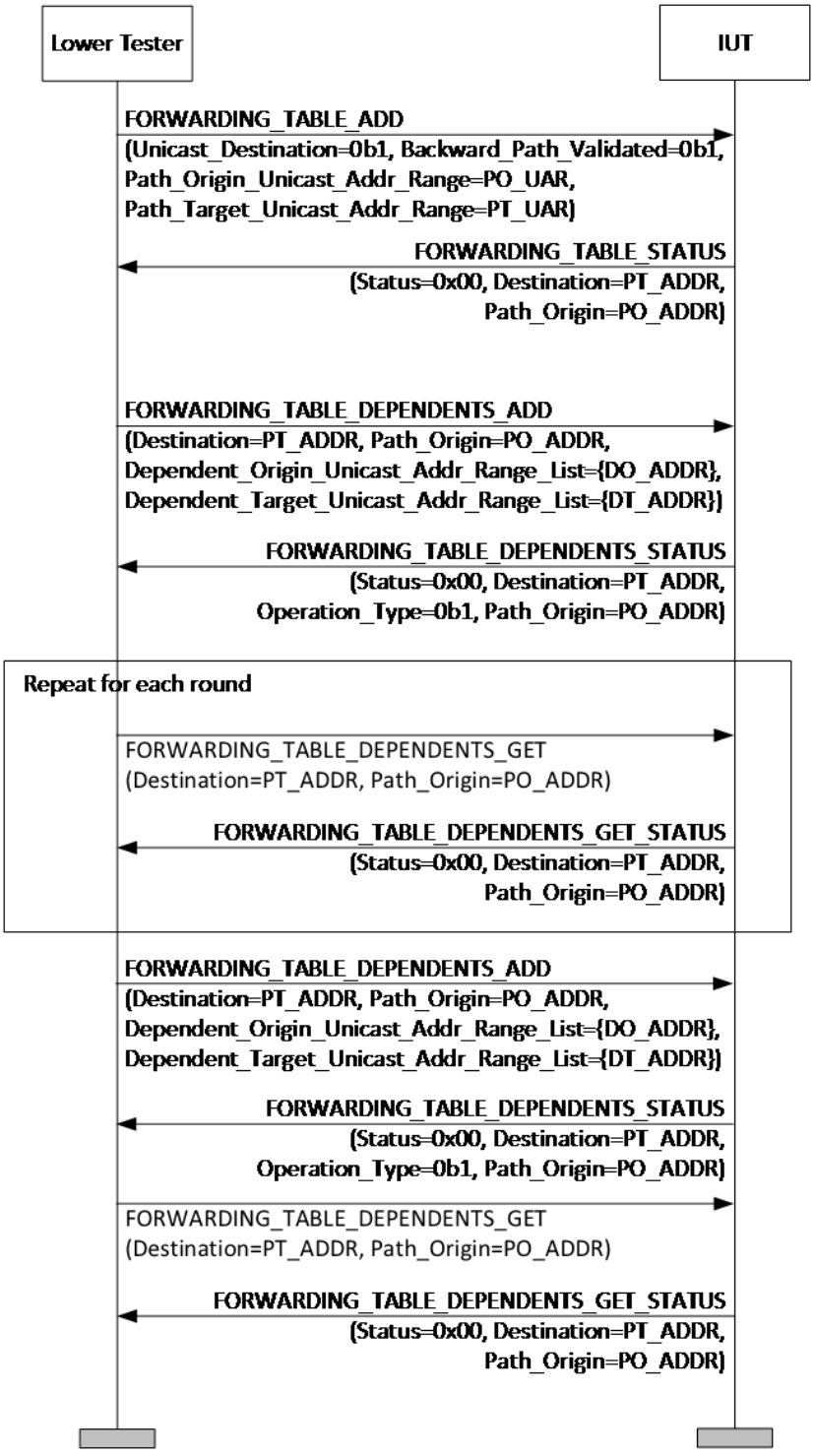


Figure 4.287: MESH/SR/DFM/FTS/BV-03-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to

_PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.

2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO_ADDR_}, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_ADDR_}.
4. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
5. Repeat Steps 6 and 7 for each round in Table 4.132.

Round	Additional Parameters
1	Dependents_List_Mask = 0b11 Start_Index = 0x0000 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x01 Dependent_Target_Unicast_Addr_Range_List_Size = 0x01 Dependent_Origin_Unicast_Addr_Range_List = {_DO_ADDR_} Dependent_Target_Unicast_Addr_Range_List = {_DT_ADDR_}
2	Dependents_List_Mask = 0b11 Start_Index = 0x0001 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x00 Dependent_Target_Unicast_Addr_Range_List_Size = 0x01 Dependent_Target_Unicast_Addr_Range_List = {_DT_ADDR_}
3	Dependents_List_Mask = 0b10 Start_Index = 0x0000 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x00 Dependent_Target_Unicast_Addr_Range_List_Size = 0x01 Dependent_Target_Unicast_Addr_Range_List = {_DT_ADDR_}
4	Dependents_List_Mask = 0b10 Start_Index = 0x0001 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x00 Dependent_Target_Unicast_Addr_Range_List_Size = 0x00
5	Dependents_List_Mask = 0b01 Start_Index = 0x0000 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x01 Dependent_Target_Unicast_Addr_Range_List_Size = 0x00 Dependent_Origin_Unicast_Addr_Range_List = {_DO_ADDR_}

Round	Additional Parameters
6	Dependents_List_Mask = 0b01 Start_Index = 0x0001 Dependent_Origin_Unicast_Addr_Range_List_Size = 0x00 Dependent_Target_Unicast_Addr_Range_List_Size = 0x00

Table 4.132: Specification of rounds for MESH/SR/DFM/FTS/BV-03-C

6. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Fixed_Path_Flag set to 0b1, Path_Origin set to _PO_ADDR_, Destination set to _PT_ADDR_, and Dependents_List_Mask and Start_Index set to the values from Table 4.132.
7. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Fixed_Path_Flag set to 0b1, Path_Origin set to _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to a valid value, and the other fields set to the values from Table 4.132.
8. Repeat Steps 3 and 4.
9. Repeat Steps 6–8 for round 1 from Table 4.132.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_DEPENDENTS_STATUS and FORWARDING_TABLE_DEPENDENTS_GET_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-04-C [Forwarding Table State Dependents Delete Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to a FORWARDING_TABLE_DEPENDENTS_DELETE message.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

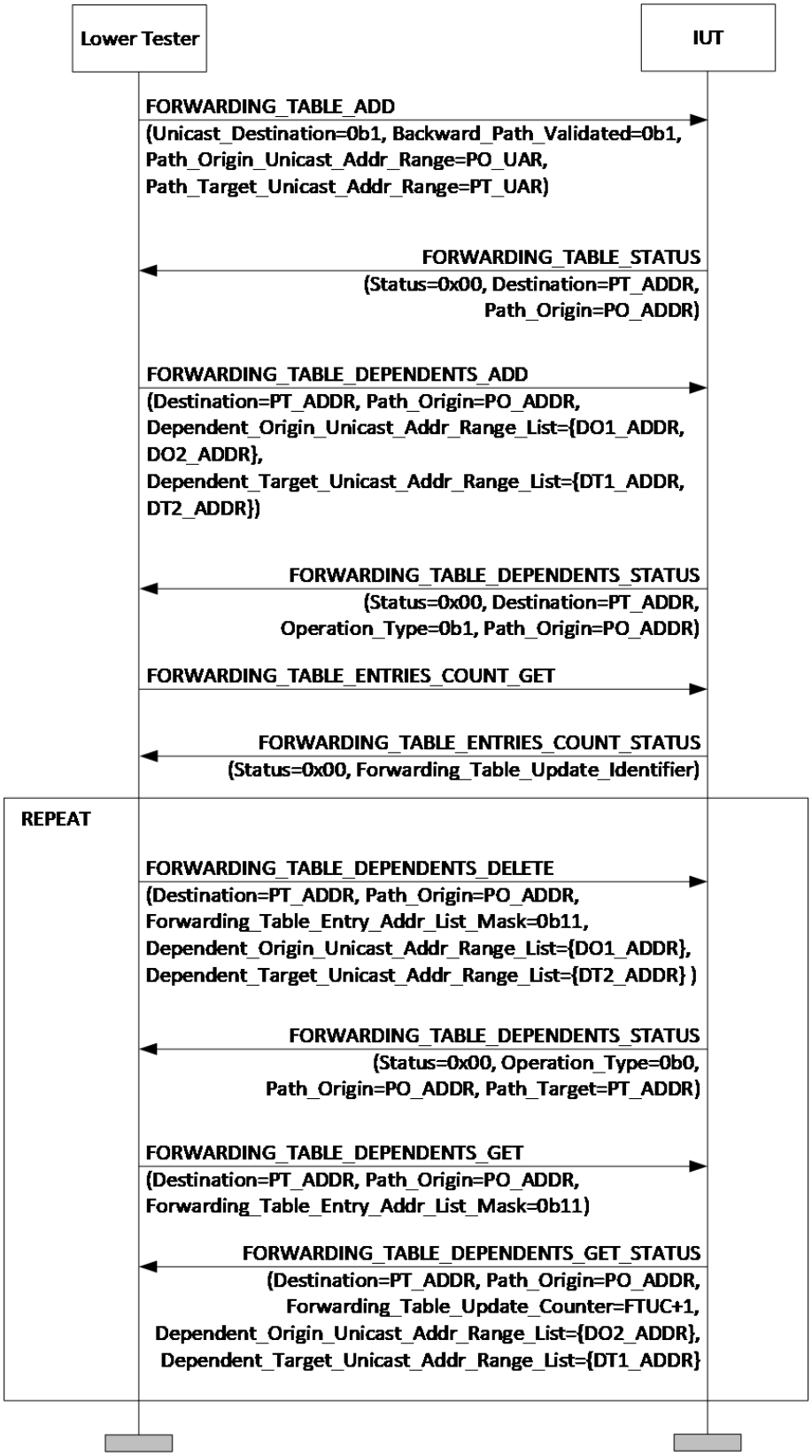


Figure 4.288: MESH/SR/DFM/FTS/BV-04-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x02, Dependent_Origin_Unicast_Addr_Range_List set to {_DO1_ADDR_, _DO2_ADDR_}, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x02, and Dependent_Target_Unicast_Addr_Range_List set to {_DT1_ADDR_, _DT2_ADDR_}.
4. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
5. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
6. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0001, and Non_Fixed_Path_Entries_Count set to 0x0000. The Lower Tester stores the Forwarding_Table_Update_Identifier value.
7. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_DELETE message with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_List_Size set to 0x01, Dependent_Origin_List set to {_DO1_ADDR_}, Dependent_Target_List_Size set to 0x01, and Dependent_Target_List set to {_DT2_ADDR_}.
8. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Path_Target set to _PT_ADDR_.
9. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, and Destination set to _PT_ADDR_.
10. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set to _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to a different value than the stored value, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO2_ADDR_}, and Dependent_Target_Unicast_Addr_Range_List set to {_DT1_ADDR_}.
11. Repeat Steps 7–10.

- Expected Outcome

Pass verdict

The IUT responds with FORWARDING_TABLE_DEPENDENTS_STATUS and FORWARDING_TABLE_DEPENDENTS_GET_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-05-C [Forwarding Table State Entries Get Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to a FORWARDING_TABLE_ENTRIES_GET message.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality is enabled on the IUT for the primary subnet.
  - The IUT has established a path from _PO1_ADDR_ to _PT1_ADDR_ by running preamble 4.2.10. Lower Tester 1 stores the Path_Origin_Forwarding_Number value as _FN_.

- Test Procedure

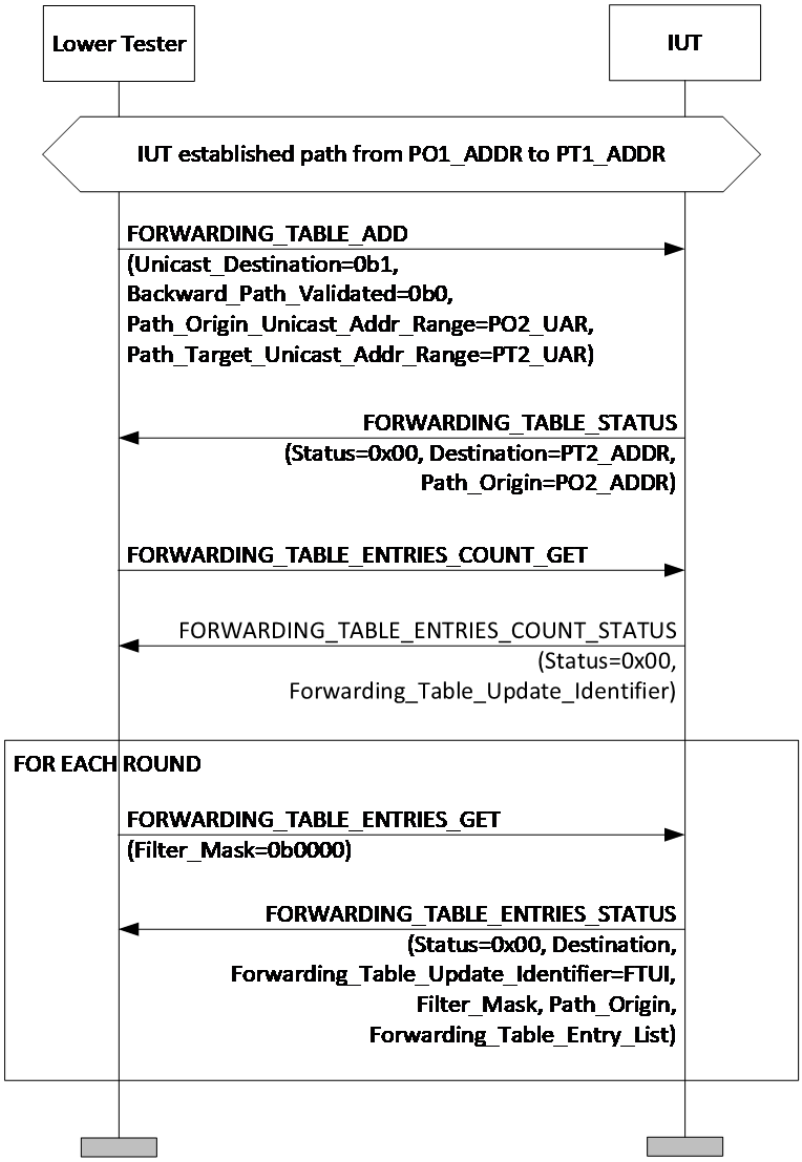


Figure 4.289: MESH/SR/DFM/FTS/BV-05-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _PO2_UAR_, Path_Target_Unicast_Addr_Range set to _PT2_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001. _PO2_UAR_ and _PT2_UAR_ are provided in the IXIT [6].
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO2_ADDR_, and Destination set to _PT2_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
4. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0001, and Non_Fixed_Path_Entries_Count set to

0x0001. The Lower Tester stores the Forwarding_Table_Update_Identifier value.

5. Repeat Steps 6 and 7 for each round in Table 4.133.
6. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET with NetKeyIndex field set to 0x0000, Filter_Mask set to the value in Table 4.133, and Start_Index set to 0x0000.
7. The IUT responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to the value in Table 4.133, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to stored value. If present, Path_Origin, Destination and Forwarding_Table_Entry_List fields contain the values from Table 4.134.

Round	Filter Mask	Additional Fields	Forwarding_Table_Entry_List (Table 4.134)
1	0b0011	Fields not present	Fixed Entry, Non-Fixed Entry
2	0b0001	Fields not present	Fixed Entry
3	0b0010	Fields not present	Non-Fixed Entry
4	0b0111	Path_Origin = _PO1_ADDR_	Non-Fixed Entry
5	0b1011	Destination = _PT2_ADDR_	Fixed Entry
6	0b1010	Destination = _PT2_ADDR_	Fields not present

Table 4.133: Specification of rounds for MESH/SR/DFM/FTS/BV-05-C

Field	Value
Non-Fixed Entry	
Forwarding_Table_Entry_Header	0x001A
Lane_Counter	0x01
Path_Remaining_Time	~ Path Lifetime
Path_Origin_Forwarding_Number	_FN_
Path_Origin_Unicast_Addr_Range	_PO1_UAR_
Bearer_Toward_Path_Origin	0x0001
Path_Target_Unicast_Addr_Range	_PT1_UAR_
Bearer_Toward_Path_Target	0x0001
Fixed Entry	
Forwarding_Table_Entry_Header	0x0013
Path_Origin_Unicast_Addr_Range	_PO2_UAR_
Bearer_Toward_Path_Origin	0x0001
Path_Target_Unicast_Addr_Range	_PT2_UAR_
Bearer_Toward_Path_Target	0x0001

Table 4.134: Forwarding Table entry list contents for MESH/SR/DFM/FTS/BV-05-C

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_ENTRIES_STATUS messages with expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-06-C [Forwarding Table State Add Behavior – Error Condition]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_ADD messages that are unsuccessfully processed.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - Directed relay functionality is enabled on the IUT for the primary subnet.
  - The IUT has established a path from _PO1_ADDR_ to _PT1_ADDR_ by running preamble 4.2.10.
  - The IUT is subscribed only to _GR_ADDR_.
- Test Procedure

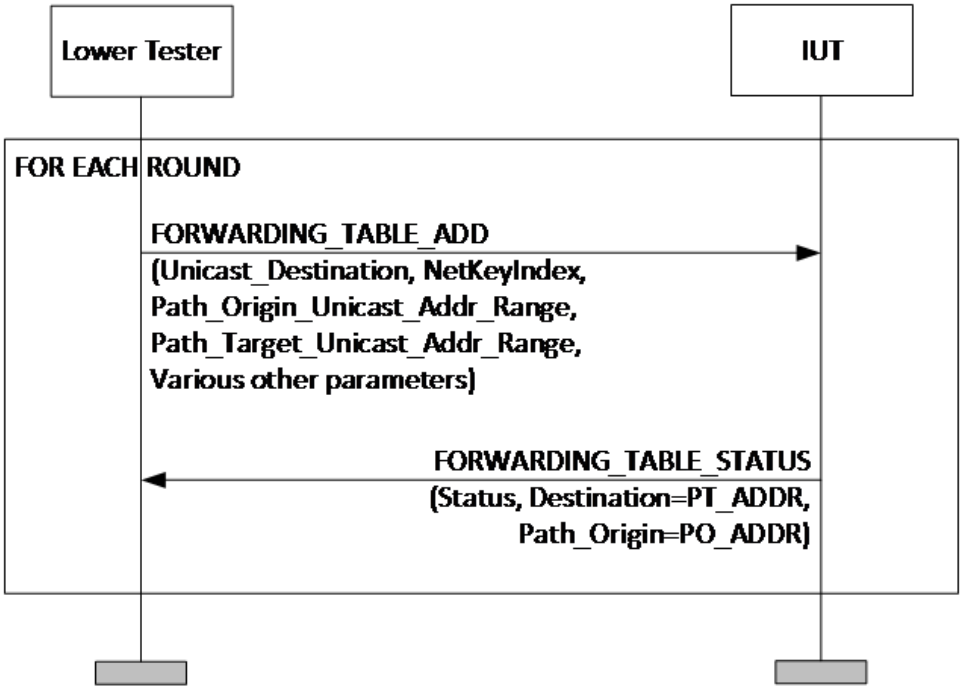


Figure 4.290: MESH/SR/DFM/FTS/BV-06-C MSC

For each round in Table 4.135:

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with the parameters from Table 4.135 if specified, otherwise with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001. If Unicast_Destination_Flag is set to 0b1, Path_Target_Unicast_Addr_Range is set to _PT_UAR_, otherwise Multicast_Destination is set to a random group or virtual address that the node is not subscribed to.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to Status Code from Table 4.135, NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
   Table 4.135 specifies the rounds to be carried out. The IUT maintains its state between rounds. If the IUT does not have secondary elements, then skip round 2. If the IUT supports all bearers, skip rounds 7 and 8.

Round	Field	Status Code
1	NetKeyIndex = 0x0002	0x04 (Invalid NetKey Index)
2	Path_Origin_Unicast_Addr_Range.RangeStart = _IUT_ADDR_, Bearer_Toward_Path_Target = 0x0000	0x15 (Invalid Bearer)
3	Path_Target_Unicast_Addr_Range.RangeStart = _IUT_ADDR_, Bearer_Toward_Path_Origin = 0x0000	0x15 (Invalid Bearer)
4	Path_Origin_Unicast_Addr_Range.RangeStart = _IUT_ADDR_, Bearer_Toward_Path_Origin ≠ 0x0000	0x15 (Invalid Bearer)
5	Path_Target_Unicast_Addr_Range.RangeStart = _IUT_ADDR_, Bearer_Toward_Path_Target ≠ 0x0000	0x15 (Invalid Bearer)
6	Bearer_Toward_Path_Origin = 0x0000	0x15 (Invalid Bearer)
7	Bearer_Toward_Path_Target = 0x0000	0x15 (Invalid Bearer)
8	Bearer_Toward_Path_Origin = 0x0100	0x15 (Invalid Bearer)
9	Bearer_Toward_Path_Target = 0x0100	0x15 (Invalid Bearer)
10	Unicast_Destination_Flag = 0b0 Multicast_Destination = _GR_ADDR_ Bearer_Toward_Path_Origin = 0x0000 Backward_Path_Validated_Flag = 0b0	0x15 (Invalid Bearer)
11	Unicast_Destination_Flag = 0b0 Bearer_Toward_Path_Origin = 0x0000	0x15 (Invalid Bearer)
12	Unicast_Destination_Flag = 0b0 Bearer_Toward_Path_Target = 0x0000	0x15 (Invalid Bearer)

Table 4.135: Specification of rounds for MESH/SR/DFM/FTS/BV-06-C

- Expected Outcome

  Pass verdict

  The IUT responds with a FORWARDING_TABLE_STATUS message with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-07-C [Forwarding Table State Delete Behavior – Error Condition]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_DELETE messages that are unsuccessfully processed.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure
  1. The Lower Tester sends a FORWARDING_TABLE_DELETE message with NetKeyIndex set to 0x0002, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
  2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex set to 0x0002, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
- Expected Outcome

  Pass verdict

  The IUT responds with a FORWARDING_TABLE_STATUS message with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-08-C [Forwarding Table State Dependents Add/Delete Behavior – Error Condition]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_DEPENDENTS_ADD and FORWARDING_TABLE_DEPENDENTS_DELETE messages that are unsuccessfully processed.
- Reference

  [10] 4.2.26, 4.4.5.6.5
- Initial Condition
  - The directed relay functionality on the primary subnet is enabled on the IUT.
- Test Procedure

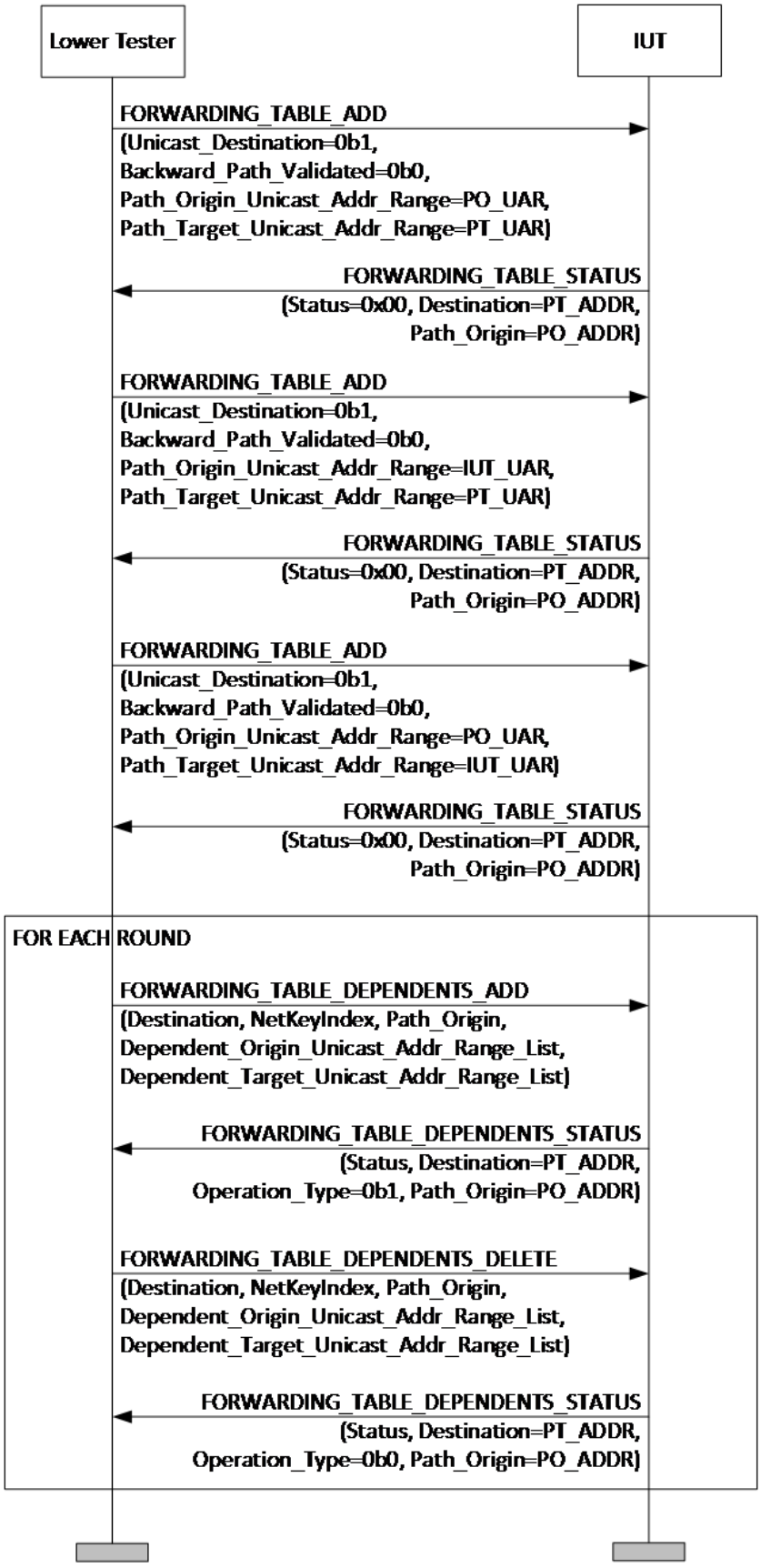


Figure 4.291: MESH/SR/DFM/FTS/BV-08-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. Repeat Steps 1 and 2 with Path_Origin_Unicast_Addr_Range set to _IUT_UAR_.
4. Repeat Steps 1 and 2 with Path_Target_Unicast_Addr_Range set to _IUT_UAR_.
5. Repeat Steps 6 and 7 for each round in Table 4.136.
6. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with the parameters from Table 4.136 if specified, otherwise with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO_UAR_}, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_UAR_}. If either of Directed Proxy or Directed friend are supported, then skip rounds 2 and 3.
7. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to the Status Code from Table 4.136, NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.

Round	Field	Status Code
1	NetKeyIndex = 0x0002	0x04 (Invalid NetKey Index)
2	Path_Origin = _IUT_ADDR_	0x0A (Feature Not Supported)
3	Destination = _IUT_ADDR_	0x0A (Feature Not Supported)
4	Path_Origin ≠ _PO_ADDR_ and Destination ≠ _PT_ADDR_	0x12 (Invalid Path Entry)

Table 4.136: Specification of rounds for Steps 6 and 7 of MESH/SR/DFM/FTS/BV-08-C

8. Repeat Steps 9 and 10 for each round in Table 4.137.
9. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_DELETE message with the parameters from Table 4.137 if specified, otherwise with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_List_Size set to 0x01, Dependent_Origin_List set to {_DO_ADDR_}, Dependent_Target_List_Size set to 0x01, and Dependent_Target_List set to {_DT_ADDR_}.
10. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to Status Code from Table 4.137, NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.

Round	Field	Status Code
1	NetKeyIndex = 0x0002	0x04 (Invalid NetKey Index)
2	Path_Origin ≠ _PO_ADDR_ and Destination ≠ _PT_ADDR_	0x12 (Invalid Path Entry)

Table 4.137: Specification of rounds for Steps 9 and 10 of MESH/SR/DFM/FTS/BV-08-C

- Expected Outcome

  Pass verdict

  The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-09-C [Forwarding Table State Get Behavior – Error Condition]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_ENTRIES_COUNT_GET, FORWARDING_TABLE_ENTRIES_GET, and FORWARDING_TABLE_DEPENDENTS_GET messages that are unsuccessfully processed.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

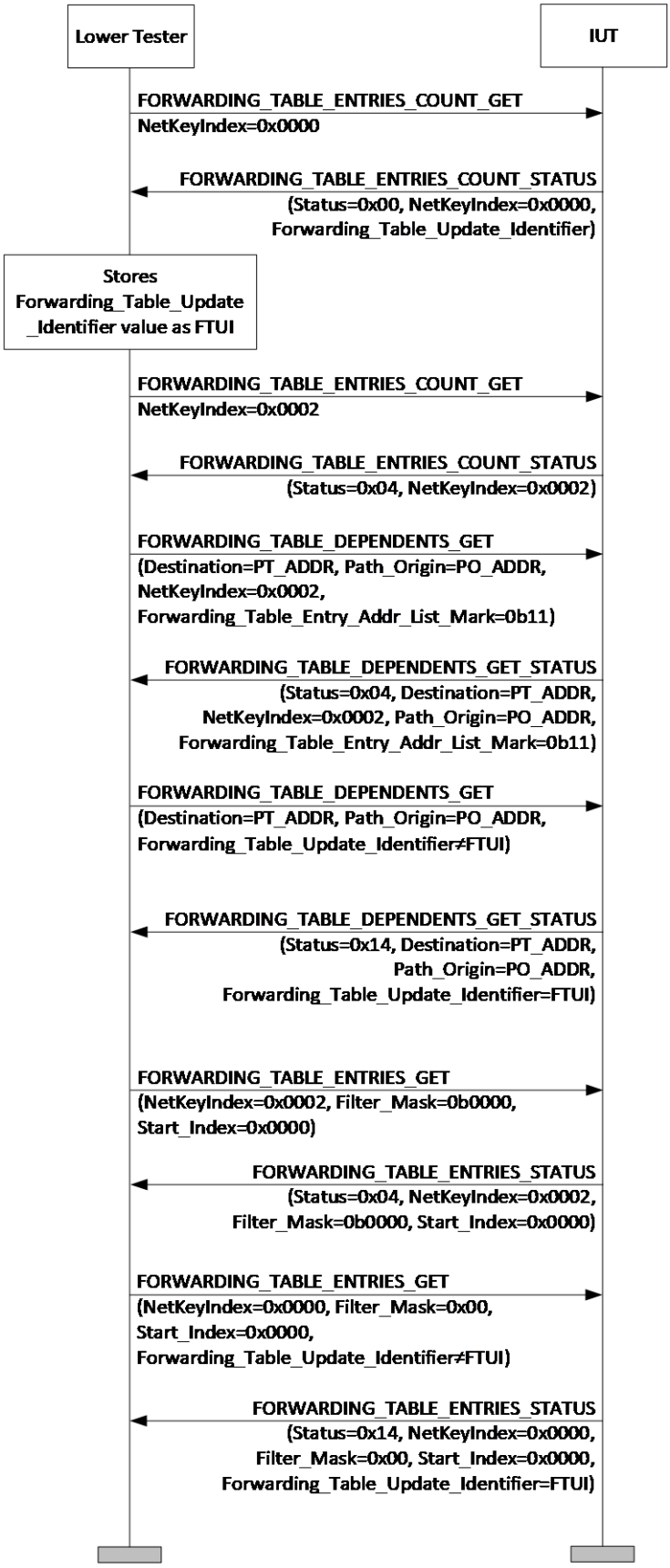


Figure 4.292: MESH/SR/DFM/FTS/BV-09-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
2. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success) and NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0000, and Non_Fixed_Path_Entries_Count set to 0x0000. The Lower Tester stores the Forwarding_Table_Update_Identifier value.
3. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0002.
4. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x04 (Invalid NetKey Index) and NetKeyIndex field set to 0x0002.
5. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0002, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, and Destination set to _PT_ADDR_.
6. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex field set to 0x0002, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, and Destination set to _PT_ADDR_.
7. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, and Forwarding_Table_Update_Identifier value set to a different value than the stored value.
8. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x14 (Obsolete Information), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR, and Forwarding_Table_Update_Identifier set to the stored value.
9. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET message with NetKeyIndex field set to 0x0002, Filter_Mask set to 0b11, and Start_Index set to 0x0000.
10. The IUT responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex field set to 0x0002, Filter_Mask set to 0b11, and Start_Index set to 0x0000.
11. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET message with NetKeyIndex field set to 0x0000, Filter_Mask set to 0x11, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier value set to a different value than the stored value.
12. The IUT responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status set to 0x14 (Obsolete Information), NetKeyIndex field set to 0x0000, Filter_Mask set to 0x11, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to the stored value.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_ENTRIES_COUNT_STATUS, FORWARDING_TABLE_DEPENDENTS_GET_STATUS, and FORWARDING_TABLE_ENTRIES_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-10-C [Forwarding Table Add Behavior – Insufficient Resources]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model can respond to FORWARDING_TABLE_ADD messages with the "insufficient resources" status.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The maximum number of paths that the IUT can store is described in the IXIT [6] as TSPX_Max_Number_Of_Paths.
  - The maximum number of paths that the IUT can store for a path is described in the IXIT [6] as TSPX_Max_Number_Of_Dependent_Nodes_Per_Path.

- Test Procedure

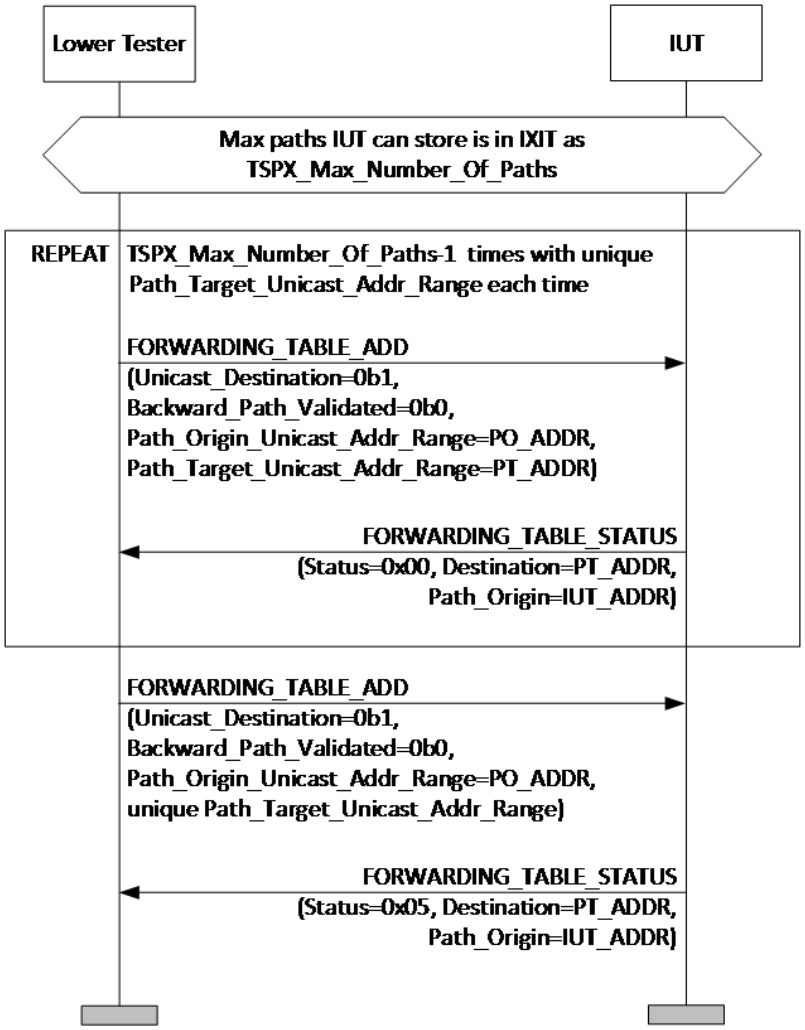


Figure 4.293: MESH/SR/DFM/FTS/BV-10-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
3. Repeat Steps 1 and 2 for (TSPX_Max_Number_Of_Paths - 1) times, with unique Path_Target_Unicast_Addr_Range values for each round.
4. Repeat Step 1 with a unique Path_Target_Unicast_Addr_Range value.
5. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x05 (Insufficient Resources), NetKeyIndex field set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
6. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _IUT_UAR_, Destination set to _PT_ADDR_,

Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x00, Dependent_Target_Unicast_Addr_Range_List_Size set to a valid value smaller than TSPX_Max_Number_Of_Dependent_Nodes_Per_Path, and Dependent_Target_Unicast_Addr_Range_List set to a list of unique random unicast addresses.

7. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
8. Repeat Steps 6 and 7 until the IUT has stored TSPX_Max_Number_Of_Dependent_Nodes_Per_Path - 1 dependent nodes for the path.
9. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex field set to 0x0000, Path_Origin set _IUT_UAR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x00, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to a list of one unique random unicast address.
10. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with 0x05 (Insufficient Resources), NetKeyIndex set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_STATUS and FORWARDING_TABLE_DEPENDENTS_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-11-C [Forwarding Table State – Power-up Behavior]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model restores the Forwarding Table state at power-up.

- Reference

  [10] 4.2.26, 4.4.5.5, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT has established a path to _PT1_ADDR_ by running preamble 4.2.9.

- Test Procedure

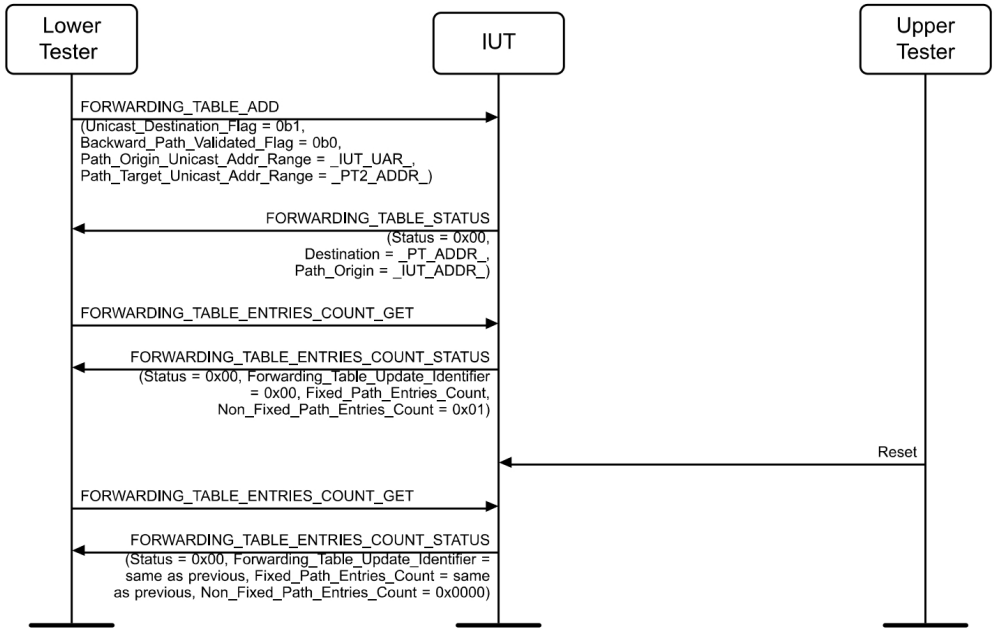


Figure 4.294: MESH/SR/DFM/FTS/BV-11-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Path_Target_Unicast_Addr_Range set to _PT2_ADDR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.
2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
4. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to 0x0000, Fixed_Path_Entries_Count set to a valid value, and Non_Fixed_Path_Entries_Count set to 0x0001.
5. The Upper Tester orders the IUT to reset itself.
6. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
7. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to the same value as in Step 4, Fixed_Path_Entries_Count set to the value from Step 4, and Non_Fixed_Path_Entries_Count set to 0x0000.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_STATUS messages with the expected values. The

messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BV-12-C [Forwarding Table State – Binding with Directed Forwarding]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model removes non-fixed paths from the Forwarding Table state when the Directed Forwarding state is set to 0x00.

- Reference

  [10] 4.2.26, 4.4.5.6.5, 4.4.5.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT has established a path to _PT1_ADDR_ by running preamble 4.2.9.
- Test Procedure

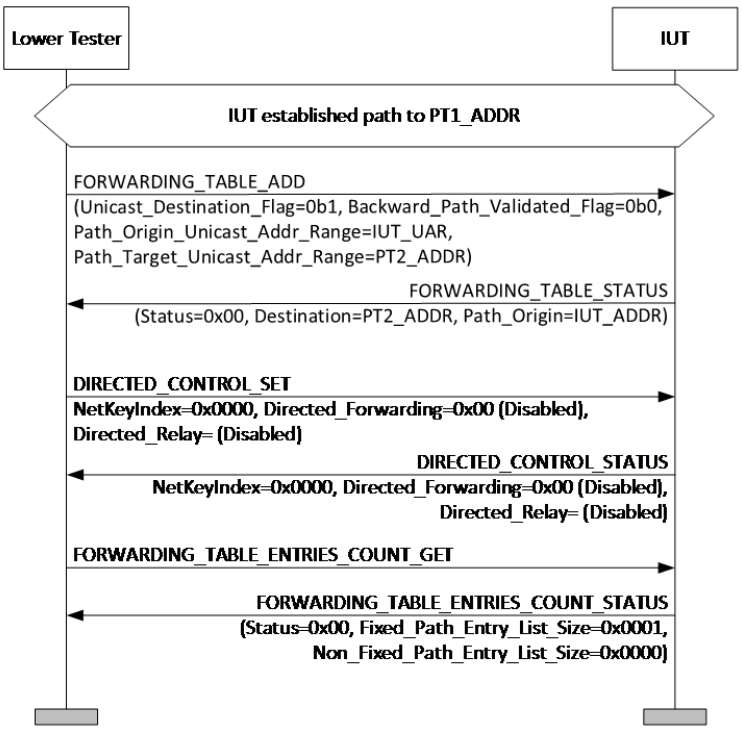


Figure 4.295: MESH/SR/DFM/FTS/BV-12-C MSC

1. The Lower Tester sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b0, Path_Origin_Unicast_Addr_Range set to _IUT_UAR_, Path_Target_Unicast_Addr_Range set to _PT2_ADDR_, Bearer_Toward_Path_Origin set to 0x0000, and Bearer_Toward_Path_Target set to 0x0001.

2. The IUT responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _IUT_ADDR_, and Destination set to _PT2_ADDR_.
3. The Lower Tester sends a DIRECTED_CONTROL_SET message with NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x00 (Disabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0xFF (Ignore), Directed_Proxy_Use_Directed_Default set to 0xFF (Ignore), and Directed_Friend set to 0xFF (Ignore).
4. The IUT responds with a DIRECTED_CONTROL_STATUS message with Status set to 0x00 (Success), the NetKeyIndex field set to 0x0000, Directed_Forwarding set to 0x00 (Disabled), Directed_Relay set to 0x00 (Disabled), Directed_Proxy set to 0x00 or 0x02 (Disabled or not supported), Directed_Proxy_Use_Directed_Default set to 0x02 (Disabled or not supported), and Directed_Friend set to 0x00 or 0x02 (Disabled or not supported).
5. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
6. The IUT responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0001, and Non_Fixed_Path_Entries_Count set to 0x0000.

- Expected Outcome

  Pass verdict

  The IUT responds with FORWARDING_TABLE_STATUS messages with the expected values. The messages are encrypted with the device key of the IUT.

MESH/SR/DFM/FTS/BI-01-C [Forwarding Table State Add Behavior – Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model ignores a FORWARDING_TABLE_ADD message with invalid parameters.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

  For each round in Table 4.138:

  1. The Lower Tester sends a FORWARDING_TABLE_ADD message with the parameters from Table 4.138 if specified, otherwise with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to

0x0001, and Bearer_Toward_Path_Target set to 0x0001.

2. The IUT does not send any FORWARDING_TABLE_STATUS messages.

Round	Condition
1	Path_Origin_Unicast_Addr_Range = 0xFFFF02
2	Path_Target_Unicast_Addr_Range = 0x800001
3	Prohibited bits set to 0b1
4	Path_Origin_Unicast_Addr_Range and Path_Target_Unicast_Addr_Range have a derived address in common
5	Unicast_Destination_Flag = 0b0, Backward_Path_Validated_Flag = 0b0, Multicast_Destination = 0xFFFB (all-directed-forwarding-nodes fixed group address)
6	Unicast_Destination_Flag = 0b0, Backward_Path_Validated_Flag = 0b0, Multicast_Destination = 0xFFFF (all-nodes fixed group address)
7	Unicast_Destination_Flag = 0b0, Backward_Path_Validated_Flag = 0b0, Multicast_Destination = 0xFFFE (all-relays fixed group address)

Table 4.138: Specification of rounds for MESH/SR/DFM/FTS/BI-01-C

- Expected Outcome

  Pass verdict

  The IUT does not respond with a FORWARDING_TABLE_STATUS message.

MESH/SR/DFM/FTS/BI-02-C [Forwarding Table State Delete Behavior – Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model ignores a FORWARDING_TABLE_DELETE message with invalid parameters.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

  For each round in Table 4.139:

  1. The Lower Tester sends a FORWARDING_TABLE_DELETE message with the parameters from Table 4.139 if specified, otherwise with NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
  2. The IUT does not send any FORWARDING_TABLE_STATUS messages.

Round	Condition
1	NetKeyIndex = 0xFFFF
2	Path_Origin set to a group address
3	Destination = Path_Origin
4	Destination = 0x0000
5	Destination = 0xFFFB (all-directed-forwarding-nodes-fixed group address)
6	Destination = 0xFFFF (all-nodes fixed group address)
7	Destination = 0xFFFE (all-relays fixed group address)

Table 4.139: Specification of rounds for MESH/SR/DFM/FTS/BI-02-C

- Expected Outcome

  Pass verdict

  The IUT does not respond with a FORWARDING_TABLE_STATUS message.

MESH/SR/DFM/FTS/BI-03-C [Forwarding Table State Dependent Add/Delete Behavior – Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model ignores a FORWARDING_TABLE_DEPENDENTS_ADD or a FORWARDING_TABLE_DEPENDENTS_DELETE message with invalid parameters.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.

- Test Procedure

  For each round in Table 4.140:

  1. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with the parameters from Table 4.140 if specified, otherwise with NetKeyIndex field set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO_UAR_}, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_UAR_}.
  2. The IUT does not send any FORWARDING_TABLE_DEPENDENTS_STATUS messages.
  3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_DELETE message with the parameters from Table 4.140 if specified, otherwise with NetKeyIndex field set to 0x0000, Path_Origin

set _PO_ADDR_, Destination set to _PT_ADDR_, Dependent_Origin_List_Size set to 0x01, Dependent_Origin_List set to {_DO_ADDR_}, Dependent_Target_List_Size set to 0x01, and Dependent_Target_List set to {_DT_ADDR_}.

4. The IUT does not send any FORWARDING_TABLE_DEPENDENTS_STATUS messages.

Round	Condition
1	Path_Origin set to a group address
2	Destination set to 0x0000
3	Dependent_Origin_Unicast_Addr_Range_List_Size = 0x00 and Dependent_Target_Unicast_Addr_Range_List_Size = 0x00
4	Dependent_Origin_Unicast_Addr_Range_List or Dependent_Origin_List = {random group address}
5	Dependent_Target_Unicast_Addr_Range_List or Dependent_Target_List = {random group address }
6	Path_Origin is set to Path_Target
7	Dependent_Origin_Unicast_Addr_Range_List or Dependent_Origin_List = {_PO_ADDR_}
8	Dependent_Taget_Unicast_Addr_Range_List or Dependent_Target_List = {_PO_ADDR_}
9	Dependent_Origin_Unicast_Addr_Range_List or Dependent_Origin_List has a derived address in common with Dependent_Target_Unicast_Addr_Range_List or Dependent_Target_List
10	Destination = 0xFFFB (all-directed-forwarding-nodes fixed group address)
11	Destination = 0xFFFF (all-nodes fixed group address)
12	Destination = 0xFFFE (all-relays fixed group address)

Table 4.140: Specification of rounds for MESH/SR/DFM/FTS/BI-03-C

- Expected Outcome

  Pass verdict

  The IUT does not respond with a FORWARDING_TABLE_DEPENDENTS_STATUS message.

MESH/SR/DFM/FTS/BI-04-C [Forwarding Table State Get Behavior – Invalid Parameters]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Server model ignores a FORWARDING_TABLE_ENTRIES_COUNT_GET, a FORWARDING_TABLE_ENTRIES_GET, or a FORWARDING_TABLE_DEPENDENTS_GET message with invalid parameters.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure
  1. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0xFFFF.
  2. The IUT does not send a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message.
     Execute Steps 3 and 4 only for rounds 1-4 and 6-8.
  3. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_GET message with the parameters from
  4. Table 4.141 if specified, otherwise with NetKeyIndex field set to 0x000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, and Destination set to _PT_ADDR_. The IUT does not send a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message.
     Execute Steps 5 and 6 only for rounds 1, 2, and 5-8.
  5. The Lower Tester sends a FORWARDING_TABLE_ENTRIES_GET message with the parameters from Table 4.141 if specified, otherwise with NetKeyIndex field set to 0x000, Filter_Mask set to 0b1111, Start_Index set to 0x0000, Path_Origin set _PO_ADDR_, Destination set to _PT_ADDR_.
  6. The IUT does not send a FORWARDING_TABLE_ENTRIES_STATUS message.

Round	Condition
1	Path_Origin set to a group address
2	Destination set to 0x0000
3	Prohibited bits set to 0b1
4	Dependents_List_Mask = 0b00
5	Filter_Mask = 0b0000
6	Destination = 0xFFFB (all-directed-forwarding-nodes fixed group address)
7	Destination = 0xFFFF (all-nodes fixed group address)
8	Destination = 0xFFFE (all-relays fixed group address)

Table 4.141: Specification of rounds for MESH/SR/DFM/FTS/BI-04-C

- Expected Outcome

  Pass verdict

  The IUT does not respond with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS, a FORWARDING_TABLE_DEPENDENTS_STATUS, or a FORWARDING_TABLE_ENTRIES_STATUS message.

MESH/CL/DFM/FTS/BV-01-C [Forwarding Table State Add Procedure]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Client model can add a fixed path

to the Directed Forwarding Configuration Server model.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

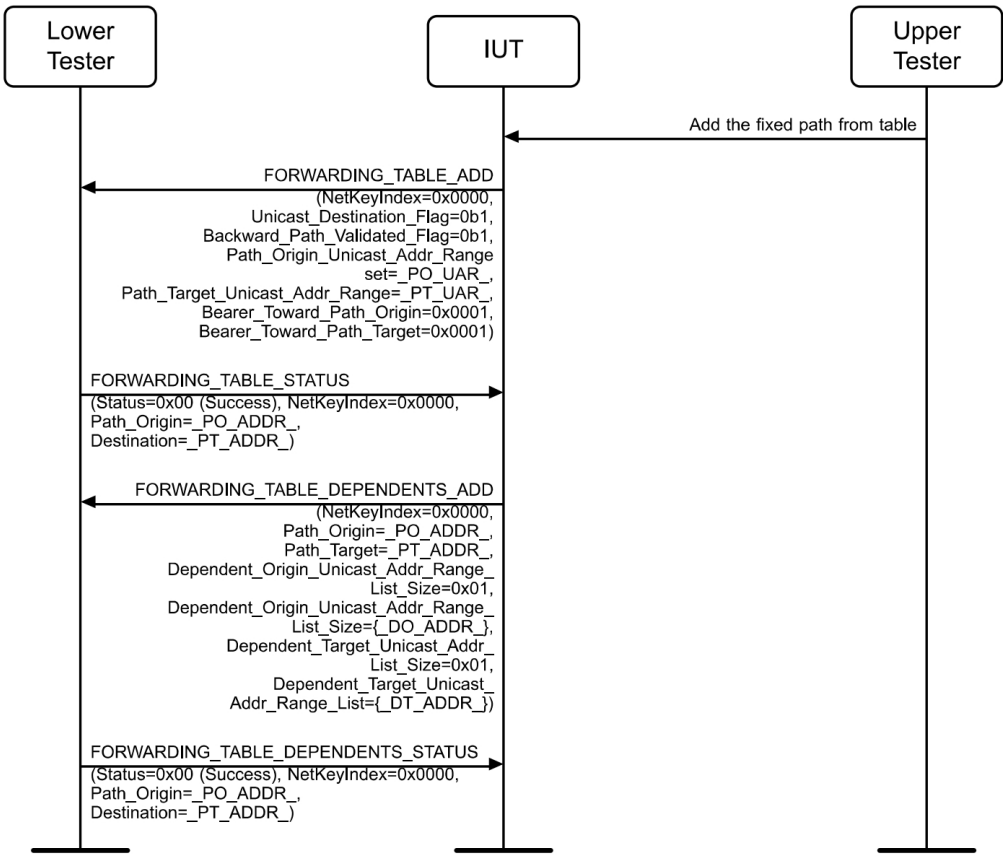


Figure 4.296: MESH/CL/DFM/FTS/BV-01-C MSC

1. The Upper Tester orders the IUT to add the fixed path from Table 4.142 on the Lower Tester on the primary subnet.
2. The IUT sends a FORWARDING_TABLE_ADD message with NetKeyIndex field set to 0x0000, Unicast_Destination_Flag set to 0b1, Backward_Path_Validated_Flag set to 0b1, Path_Origin_Unicast_Addr_Range set _PO_UAR_, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Bearer_Toward_Path_Origin set to 0x0001, and Bearer_Toward_Path_Target set to 0x0001.
3. The Lower Tester responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
4. The Lower Tester sends a FORWARDING_TABLE_DEPENDENTS_ADD message with NetKeyIndex

field set to 0x0000, Path_Origin set _PO_ADDR_, Path_Target set to _PT_ADDR_, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to {_DO_ADDR_}, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, and Dependent_Target_Unicast_Addr_Range_List set to {_DT_ADDR_}.

5. The IUT responds with a FORWARDING_TABLE_DEPENDENTS_STATUS message with Status set to 0x00 (Success), NetKeyIndex set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.

Field	Value
Backward_Path_Validated	0b1
Path_Origin_Unicast_Addr_Range	_PO_UAR_
Path_Dst_Unicast_Addr_Range	_PT_UAR_
Bearer_Toward_Path_Origin	0x0001
Bearer_Toward_Path_Dst	0x0001
Dependent_Origin_Unicast_Addr_Range_List	{_DO_UAR_}
Dependent_Target_Unicast_Addr_Range_List	{_DT_UAR_}

Table 4.142: Fixed path contents for MESH/CL/DFM/FTS/BV-01-C

- Expected Outcome

  Pass verdict

  The IUT sends a FORWARDING_TABLE_ADD and a FORWARDING_TABLE_DEPENDENTS_ADD message with valid fields. The messages are encrypted with the device key of the IUT.

MESH/CL/DFM/FTS/BV-02-C [Forwarding Table State Get Procedure]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Client model can send FORWARDING_TABLE_ENTRIES_COUNT_GET messages to a Directed Forwarding Configuration Server model.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition

  - The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

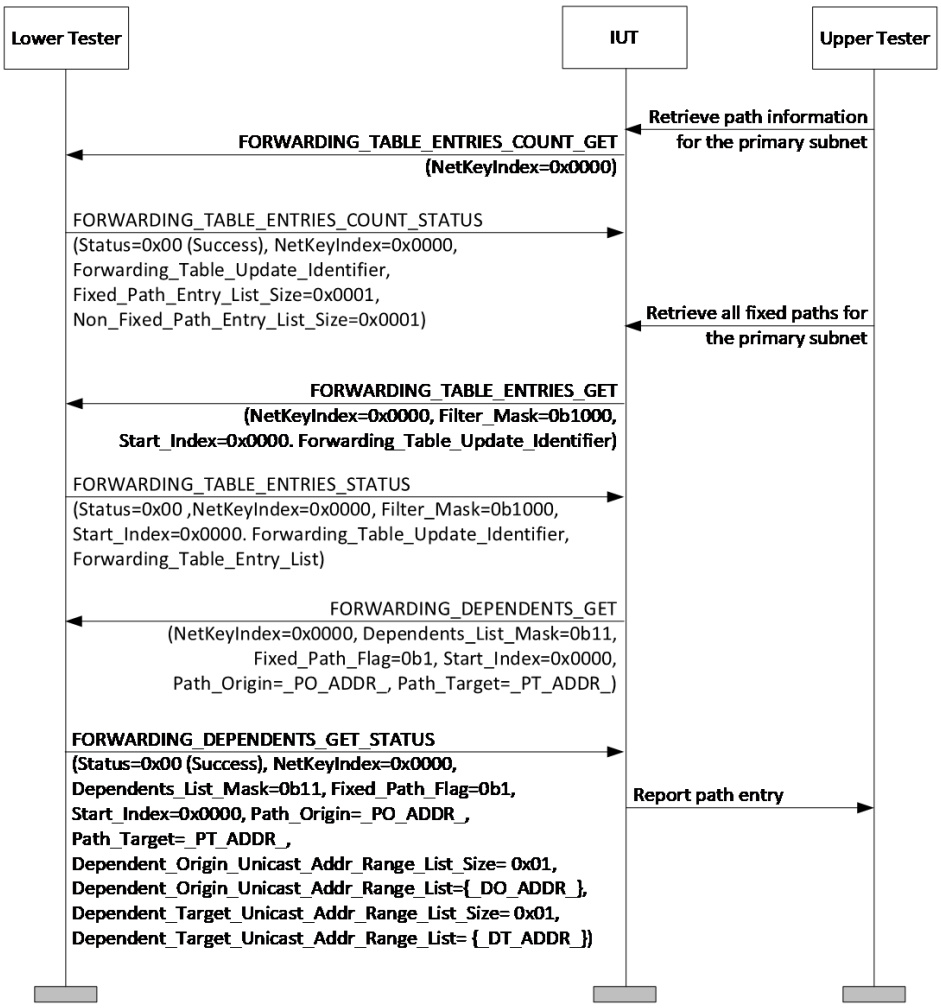


Figure 4.297: MESH/CL/DFM/FTS/BV-02-C MSC

1. The Upper Tester orders the IUT to request information about all paths on the Lower Tester for the primary subnet.
2. The IUT sends a FORWARDING_TABLE_ENTRIES_COUNT_GET message with NetKeyIndex field set to 0x0000.
3. The Lower Tester responds with a FORWARDING_TABLE_ENTRIES_COUNT_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Forwarding_Table_Update_Identifier set to a valid value, Fixed_Path_Entries_Count set to 0x0001, and Non_Fixed_Path_Entries_Count set to 0x0001. The Lower Tester stores the Forwarding_Table_Update_Identifier value.
4. The Upper Tester orders the IUT to request information about the non-fixed paths on the Lower Tester for the primary subnet.
5. The IUT sends a FORWARDING_TABLE_ENTRIES_GET message with NetKeyIndex field set to 0x0000, Filter_Mask set to 0b10, and Start_Index set to 0x0000. Forwarding_Table_Update_Identifier field may be present and set to the stored value.
6. The Lower Tester responds with a FORWARDING_TABLE_ENTRIES_STATUS message with Status

set to 0x00 (Success), NetKeyIndex field set to 0x0000, Filter_Mask set to 0b10, Start_Index set to 0x0000, and Forwarding_Table_Update_Identifier set to the stored value. Forwarding_Table_Entry_List contains one entry with Forwarding_Table_Entry_Header set to 0xE350, Path_Origin_Unicast_Addr_Range set to _PO_UAR_, Dependent_Origin_List_Size set to 0x01, Bearer_Toward_Path_Origin set to 0x0001, Path_Target_Unicast_Addr_Range set to _PT_UAR_, Dependent_Target_List_Size set to 0x01, and Bearer_Toward_Path_Target set to 0x0001. Forwarding_Table_Update_Identifier field may be present and set to the stored value.

7. The IUT sends a FORWARDING_TABLE_DEPENDENTS_GET message with NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_. Forwarding_Table_Update_Identifier field may be present and set to the stored value.
8. The Lower Tester responds with a FORWARDING_TABLE_DEPENDENTS_GET_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Dependents_List_Mask set to 0b11, Fixed_Path_Flag set to 0b1, Start_Index set to 0x0000, Path_Origin set to _PO_ADDR_, Destination set to _PT_ADDR_, Forwarding_Table_Update_Identifier set to the stored value, Dependent_Origin_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Target_Unicast_Addr_Range_List_Size set to 0x01, Dependent_Origin_Unicast_Addr_Range_List set to _DO_ADDR_, and Dependent_Target_Unicast_Addr_Range_List set to _DT_ADDR_.
9. The IUT reports to the Upper Tester that it has received the path entry at index 0x0000 with the fields set to the values in Steps 6 and 8.

- Expected Outcome

  Pass verdict

  The IUT sends a FORWARDING_TABLE_ENTRIES_COUNT_GET, a FORWARDING_TABLE_ENTRIES_GET, and a FORWARDING_TABLE_DEPENDENTS_GET message with the expected field values. The messages are encrypted with the device key of the IUT.

  The IUT reports the expected values to the Upper Tester.

MESH/CL/DFM/FTS/BV-03-C [Forwarding Table Delete Procedure]

- Test Purpose

  Verify that an IUT node supporting the Directed Forwarding Configuration Client model can delete a path stored on a Directed Forwarding Configuration Server model.

- Reference

  [10] 4.2.26, 4.4.5.6.5

- Initial Condition
  - The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- Test Procedure

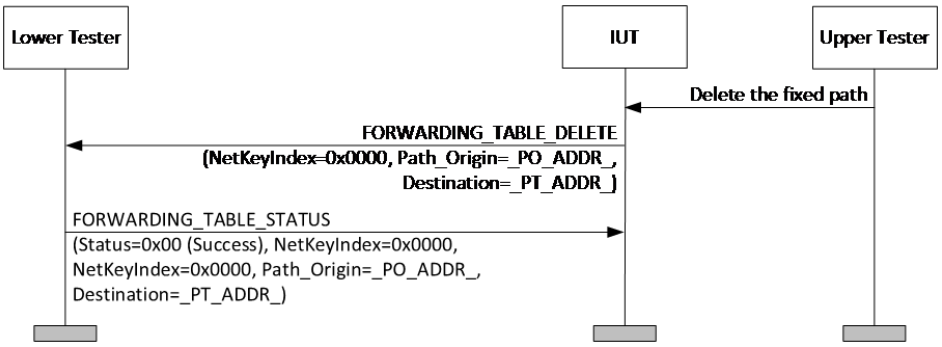


Figure 4.298: MESH/CL/DFM/FTS/BV-03-C MSC

1. The Upper Tester orders the IUT to delete the fixed path entry from _PO_ADDR_ to _PT_ADDR_ on the primary subnet stored on the Lower Tester.
2. The IUT sends a FORWARDING_TABLE_DELETE message with NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_.
3. The Lower Tester responds with a FORWARDING_TABLE_STATUS message with Status set to 0x00 (Success), NetKeyIndex field set to 0x0000, Path_Origin set to _PO_ADDR_, and Destination set to _PT_ADDR_

- Expected Outcome

  Pass verdict

  The IUT sends a FORWARDING_TABLE_DELETE message with valid fields. The messages are encrypted with the device key of the IUT.

4.19.7 Wanted Lanes State

This section validates the Wanted Lanes state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/WLS/BV-01-C [Wanted Lanes State Get/Set/Status Behavior]

- Reference

  [10] 4.2.27, 4.4.5.6.6

Execute the common test case using the values from Section 4.19.1.1.4.

MESH/SR/DFM/WLS/BI-01-C [Wanted Lanes Get/Set Behavior – Invalid Parameters]

- Reference

  [10] 4.2.27, 4.4.5.6.6

Execute the common test case using the values from Section 4.19.1.2.4.

MESH/CL/DFM/WLS/BV-01-C [Send Wanted Lanes Get/Set/Status Procedure]

- Reference

  [10] 4.2.27, 4.4.5.6.6

Execute the common test case using the values from Section 4.19.1.2.4.

MESH/CL/DFM/WLS/BI-01-C [Wanted Lanes Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.27, 4.4.5.6.6

Execute the common test case using the values from Section 4.19.2.2.4.

4.19.8 Two Way Path State

This section validates the Two Way Path state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/TWPS/BV-01-C [Two Way Path State Get/Set Behavior]

- Reference

  [10] 4.2.28, 4.4.5.6.7

Execute the common test case using the values from Section 4.19.1.1.5, Two Way Path State Get/Set/Status Behavior.

MESH/SR/DFM/TWPS/BI-01-C [Two Way Path State Get/Set Messages – Invalid Parameters]

- Reference

  [10] 4.2.28, 4.4.5.6.7

Execute the common test case using the values from Section 4.19.1.2.5.

MESH/CL/DFM/TWPS/BV-01-C [Two Way Path Get/Set/Status Procedure]

- Reference

  [10] 4.2.28, 4.4.5.6.7

Execute the common test case using the values from Section 4.19.2.1.5.

MESH/CL/DFM/TWPS/BI-01-C [Two Way Path Status Procedure – Invalid Parameters]

- Reference

[10] 4.2.28, 4.4.5.6.7

Execute the common test case using the values from Section 4.19.2.2.5.

4.19.9 Path Echo Interval State

This section validates the Path Echo Interval state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/PEIPS/BV-01-C [Path Echo Interval State Get/Set/Status Behavior, Change Both Intervals]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.1.1.6.

MESH/SR/DFM/PEIPS/BV-02-C [Path Echo Interval State Get/Set/Status Behavior, Change Only Unicast]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.1.1.7.

MESH/SR/DFM/PEIPS/BV-03-C [Path Echo Interval State Get/Set/Status Behavior, Change Only Multicast]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.1.1.8.

MESH/SR/DFM/PEIPS/BI-01-C [Path Echo Interval State Get/Set Behavior – Invalid Parameters, Invalid Intervals]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.1.2.6.

MESH/SR/DFM/PEIPS/BI-02-C [Path Echo Interval State Get/Set Behavior – Invalid Parameters, Invalid NetKeyIndex]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.1.2.7.

MESH/CL/DFM/PEIPS/BV-01-C [Path Echo Interval Get/Set/Status Procedure]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.2.1.6.

MESH/CL/DFM/PEIPS/BI-01-C [Path Echo Interval Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.29, 4.4.5.6.8

Execute the common test case using the values from Section 4.19.2.2.6.

4.19.10 Directed Network Transmit State

This section contains test for validating the Directed Network Transmit state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DNTS/BV-01-C [Directed Network Transmit State Get/Set Behavior]

- Reference

  [10] 4.2.30, 4.4.5.6.9

Execute the common test case using the values from Section 4.19.1.3.1, Directed Network Transmit State Get/Set/Status Behavior.

MESH/CL/DFM/DNTS/BV-01-C [Send Directed Network Transmit Get/Set Behavior]

- Reference

  [10] 4.2.30, 4.4.5.6.9

Execute the common test case using the values from Section 4.19.1.2.7.

4.19.11 Directed Relay Retransmit State

This section validates the Directed Relay Retransmit state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DRTS/BV-01-C [Directed Relay Retransmit State Get/Set Behavior]

- Reference

  [10] 4.2.31, 4.4.5.6.10

Execute the common test case using the values from Section 4.19.1.3.2.

MESH/CL/DFM/DRTS/BV-01-C [Send Directed Relay Retransmit Get/Set Behavior]

- Reference

  [10] 4.2.31, 4.4.5.6.10

Execute the common test case using the values from Section 4.19.2.1.8.

4.19.12 RSSI Threshold State

This section validates the RSSI Threshold state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/RTS/BV-01-C [RSSI Threshold State Get/Set Behavior]

- Reference

  [10] 4.2.32, 4.4.5.6.11

Execute the common test case using the values from Section 4.19.1.3.3.

MESH/SR/DFM/RTS/BI-01-C [RSSI Threshold State Set Behavior – Invalid Parameters]

- Test Purpose

  Verify that a Directed Forwarding Server IUT ignores a message with invalid field values sent for the RSSI Threshold state.

- Reference

  [10] 4.2.32, 4.4.5.6.11

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
- Test Procedure
  1. The Lower Tester sends an RSSI_THRESHOLD_SET message with RSSI_Margin field set to a random value greater than 0x32.
  2. The IUT does not respond with an RSSI_THRESHOLD_STATUS message.
- Expected Outcome

  Pass verdict

The IUT does not respond with an RSSI_THRESHOLD_STATUS message.

MESH/CL/DFM/RTS/BV-01-C [Send RSSI Threshold Get/Set Behavior]

- Reference

  [10] 4.2.32, 4.4.5.6.11

Execute the common test case using the values from Section 4.19.2.1.9.

MESH/CL/DFM/RTS/BI-01-C [RSSI Threshold Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.32, 4.4.5.6.11

Execute the common test case using the values from Section 4.19.2.2.7.

4.19.13 Directed Paths State

This section validates the Directed Paths state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DPS/BV-01-C [Directed Paths State Get/Status Behavior]

- Reference

  [10] 4.2.33, 4.4.5.6.12

Execute the common test case using the values from Section 4.19.1.3.4.

MESH/CL/DFM/DPS/BV-01-C [Send Directed Paths Get Behavior]

- Reference

  [10] 4.2.33, 4.4.5.6.12

Execute the common test case using the values from Section 4.19.2.1.10.

4.19.14 Directed Publish Policy State

This section validates the Directed Publish Policy state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/PPS/BV-01-C [Directed Publish Policy State Get/Set Behavior]

- Test Purpose

  Verify the behavior of a Server Model on the IUT supporting the Directed Publish Policy state for Directed Publish Policy State Get/Set messages.

- Reference

  [10] 4.2.35, 4.4.5.6.14

- Initial Condition
  - Common initial test conditions for Directed Forwarding tests defined in Section 4.2.8 apply.
  - The IUT is configured for model publication on the ModelID provided in the IXIT [6].

- Test Procedure

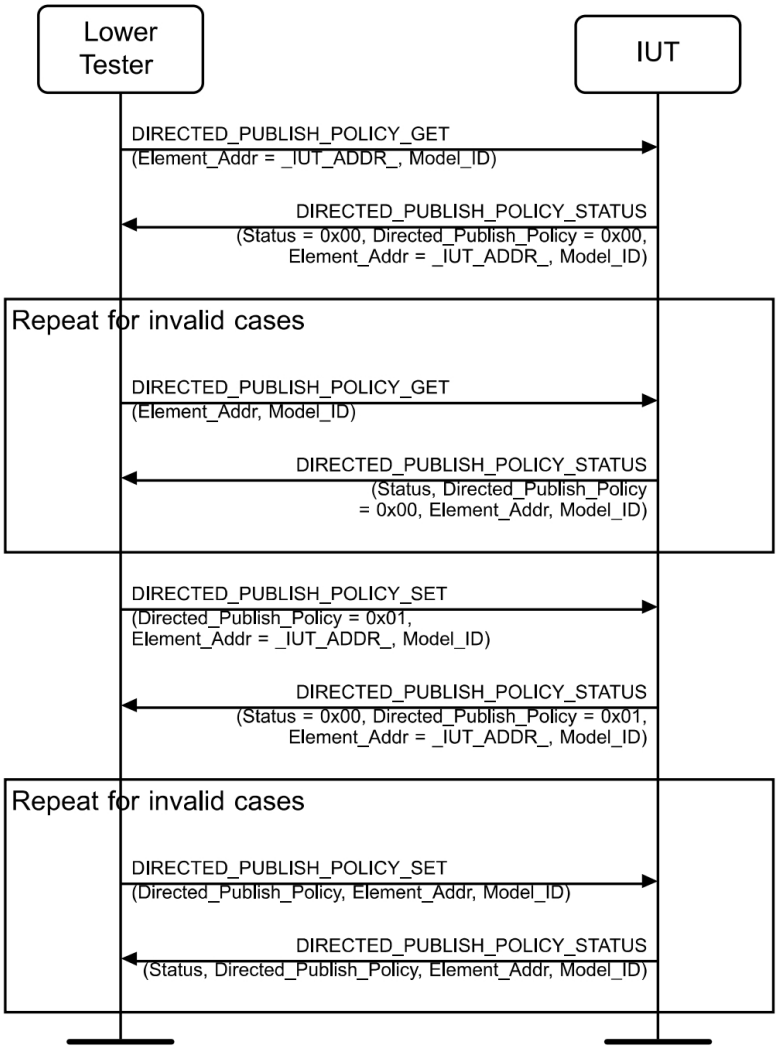


Figure 4.299: MESH/SR/DFM/PPS/BV-01-C MSC

1. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_GET message with Element_Addr set to _IUT_ADDR_ and Model_ID set to ModelID.
2. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x00 (Success), Directed_Publish_Policy set to 0x00, Element_Addr set to _IUT_ADDR_, and Model_ID set to ModelID.
3. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_GET message with Element_Addr set to a valid value other than any element address on the IUT and Model_ID set to ModelID.

4. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x01 (Invalid Address), Directed_Publish_Policy set to 0x00, and Element_Addr and Model_ID set to the values from Step 3.
5. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_GET message with Element_Addr set to _IUT_ADDR_, Model_ID set to a value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
6. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x02 (Invalid Model), Directed_Publish_Policy set to 0x00, and Element_Addr and Model_ID set to the values from Step 5.
7. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_SET message with Directed_Publish_Policy set to 0x01, Element_Addr set to _IUT_ADDR_, and Model_ID set to a ModelID.
8. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x00 (Success), Directed_Publish_Policy set to 0x01, Element_Addr set to _IUT_ADDR_, and Model_ID set to a ModelID.
9. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_SET message with Directed_Publish_Policy set to 0x01, Element_Addr set to a valid value other than any element address on the IUT, and Model_ID set to ModelID.
10. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x01 (Invalid Address), Directed_Publish_Policy set to 0x01, and Element_Addr and Model_ID set to the values from Step 9.
11. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_SET message with Directed_Publish_Policy set to 0x01, Element_Addr set to _IUT_ADDR_, and Model_ID set to a value within the range of the field but not one of the Model Identifiers listed in the IUT's IXIT [6] table.
12. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x02 (Invalid Model), Directed_Publish_Policy set to 0x01, and Element_Addr and Model_ID set to the values from Step 11.
13. The Lower Tester sends a DIRECTED_PUBLISH_POLICY_SET message with Directed_Publish_Policy set to 0x01, Element_Addr set to _IUT_ADDR_, and Model_ID set to a value within the range of the Model Identifiers listed in the IUT's IXIT [6] table, but not supporting publishing.
14. The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with Status set to 0x07 (Invalid Publish Parameters), Directed_Publish_Policy set to 0x00, and Element_Addr and Model_ID set to the values from Step 13.

- Expected Outcome

  Pass verdict

  The IUT responds with a DIRECTED_PUBLISH_POLICY_STATUS message with the expected fields. The messages are encrypted with the device key.

MESH/SR/DFM/PPS/BI-01-C [Directed Publish Policy State Get/Set Behavior – Invalid Parameters]

- Reference

[10] 4.2.35, 4.4.5.6.14

Execute the common test case using the values from Section 4.19.1.2.8.

MESH/CL/DFM/PPS/BV-01-C [Send Directed Publish Policy Get/Set Behavior]

- Reference

  [10] 4.2.35, 4.4.5.6.14

Execute the common test case using the values from Section 4.19.2.1.11.

MESH/CL/DFM/PPS/BI-01-C [Directed Publish Policy Status Procedure – Invalid Parameters]

- Reference

  [10] 4.2.35, 4.4.5.6.14

Execute the common test case using the values from Section 4.19.2.2.9.

4.19.15 Path Discovery Timing Control State

This section validates the Path Discovery Timing Control state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DTS/BV-01-C [Path Discovery Timing Control State Get/Set Behavior]

- Reference

  [10] 4.2.36, 4.4.5.6.15

Execute the common test case using the values from Section 4.19.1.3.5.

MESH/SR/DFM/DTS/BI-01-C [Path Discovery Timing Control State Get/Set Behavior – Invalid Parameters]

- Reference

  [10] 4.2.36, 4.4.5.6.15

Execute the common test case using the values from Section 4.19.1.2.9.

MESH/CL/DFM/DTS/BV-01-C [Path Discovery Timing Control State Get/Set Behavior]

- Reference

  [10] 4.2.36, 4.4.5.6.15

Execute the common test case using the values from Section 4.19.2.1.12.

4.19.16 Directed Control Network Transmit State

This section validates the Directed Control Network Transmit state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DCNTS/BV-01-C [Directed Control Network Transmit State Get/Set Behavior]

- Reference

  [10] 4.2.37, 4.4.5.6.16

Execute the common test case using the values from Section 4.19.1.3.6.

MESH/CL/DFM/DCNTS/BV-01-C [Send Directed Control Network Transmit Get/Set Behavior]

- Reference

  [10] 4.2.37, 4.4.5.6.16

Execute the common test case using the values from Section 4.19.2.1.13.

4.19.17 Directed Control Relay Retransmit State

This section validates the Directed Control Relay Retransmit state and its messages for both the Directed Forwarding Configuration Server model and the Directed Forwarding Configuration Client model.

MESH/SR/DFM/DCRTS/BV-01-C [Directed Control Relay Retransmit State Get/Set Behavior]

- Reference

  [10] 4.2.38, 4.4.5.6.17

Execute the common test case using the values from Section 4.19.1.3.7.

MESH/CL/DFM/DCRTS/BV-01-C [Send Directed Control Relay Retransmit Get/Set Behavior]

- Reference

  [10] 4.2.38, 4.4.5.6.17

Execute the common test case using the values from Section 4.19.2.1.14.

4.20 Subnet Bridge

This section contains tests for the Subnet Bridge functionality.

4.20.1 Network Layer Messages

This subsection verifies sending and receiving Network Layer messages on a Subnet Bridge IUT.

MESH/NODE/SBR/NET/BV-01-C [Receive Network Layer Message, DST as Unicast Address, IUT as Subnet Bridge]

- Test Purpose

  Verify that a Subnet Bridge IUT node will retransmit a Network Layer message to bridged subnets.

- Reference

  [10] 3.4.6.3

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - If supported, the Relay feature is disabled on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet).
  - The IUT is configured with the bridging table entries (0x01, 1, 0, _ADDR1_, _ADDR2_) and (0x02, 0, 1, _ADDR1_, _ADDR3_) by running preamble 4.2.7. _ADDR_ items are random unicast addresses.
  - The Lower Tester is configured to generate valid network PDUs with the CTL field set randomly to 0 or 1 on each new PDU, the SEQ field starting from 0 and increasing on each new PDU, and the TransportPDU field set randomly to a valid array of octets of valid length on each new PDU.

- Test Procedure

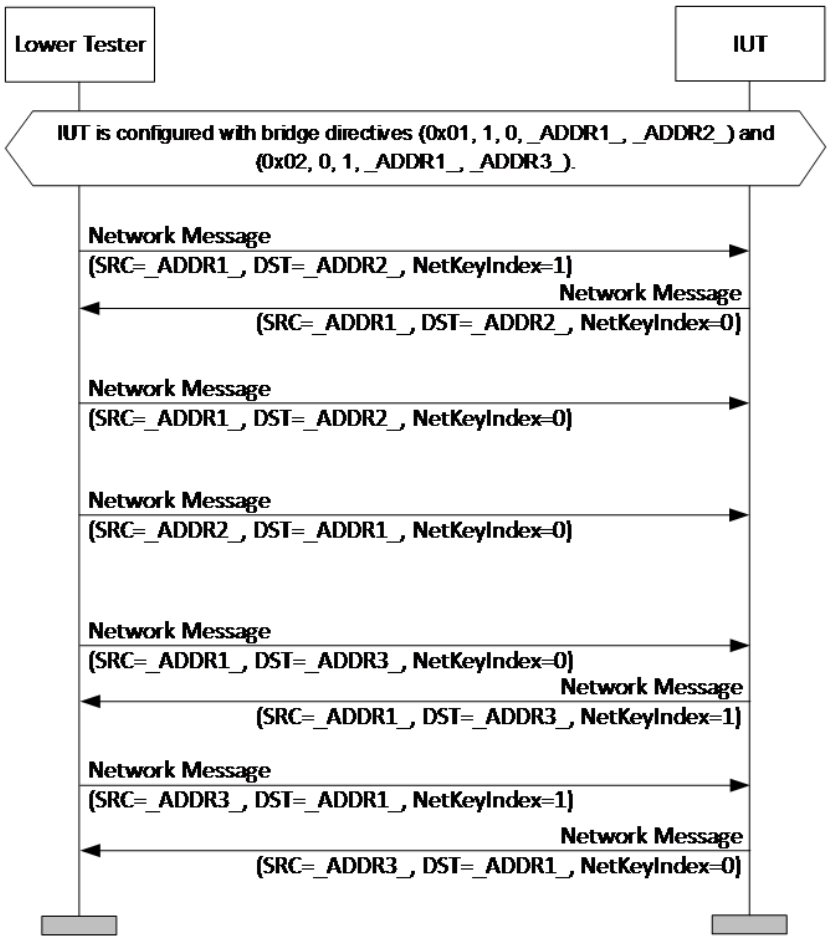


Figure 4.300: Test procedure for MESH/NODE/SBR/NET/BV-01-C [Receive Network Layer Message, DST as Unicast Address, IUT as Subnet Bridge] MSC

For each round, repeat Steps 1 and 2.

1. The Lower Tester sends a network message with the SRC and DST set to the values in Table 4.143, the TTL field set to a random valid value greater than or equal to 2. The message is encrypted with the network key on the subnet specified in Table 4.143.
2. If Table 4.143 specifies bridging, the IUT sends the network message received in Step 1 with the TTL field decremented by 1. The message is re-encrypted with the network key on the subnet specified in Table 4.143.

Round	Step 1	Step 2
1	SRC = _ADDR1_, DST = _ADDR2_, NetKey Index = 1	Bridge on NetKeyIndex = 0
2	SRC = _ADDR1_, DST = _ADDR2_, NetKey Index = 0	Do not bridge.
3	SRC = _ADDR2_, DST = _ADDR1_, NetKey Index = 0	Do not bridge.
4	SRC = _ADDR1_, DST = _ADDR3_, NetKey Index = 0	Bridge on NetKeyIndex = 1
5	SRC = _ADDR3_, DST = _ADDR1_, NetKey Index = 1	Bridge on NetKeyIndex = 0

Table 4.143: Rounds for MESH/NODE/SBR/NET/BV-01-C [Receive Network Layer Message, DST as Unicast Address, IUT as Subnet Bridge]

- Expected Outcome

  Pass verdict

  If required, the IUT bridges the network messages with the specified parameter values.

MESH/NODE/SBR/NET/BV-02-C [Ignore Replayed Network Layer Message]

- Test Purpose

  Verify that an IUT performs replay protection checks on a received Network Layer message before retransmitting it to the bridged subnets.

- Reference

  [10] 3.8.8

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - If supported, the Relay feature is disabled on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 (first subnet) and 1 (second subnet).
  - The IUT is configured with the bridging table entry (0x01, 0, 1, _ADDR1_, _ADDR2_) by running preamble 4.2.7. _ADDR_ items are random unicast addresses.

- The replay protection list on the IUT is empty.

• Test Procedure

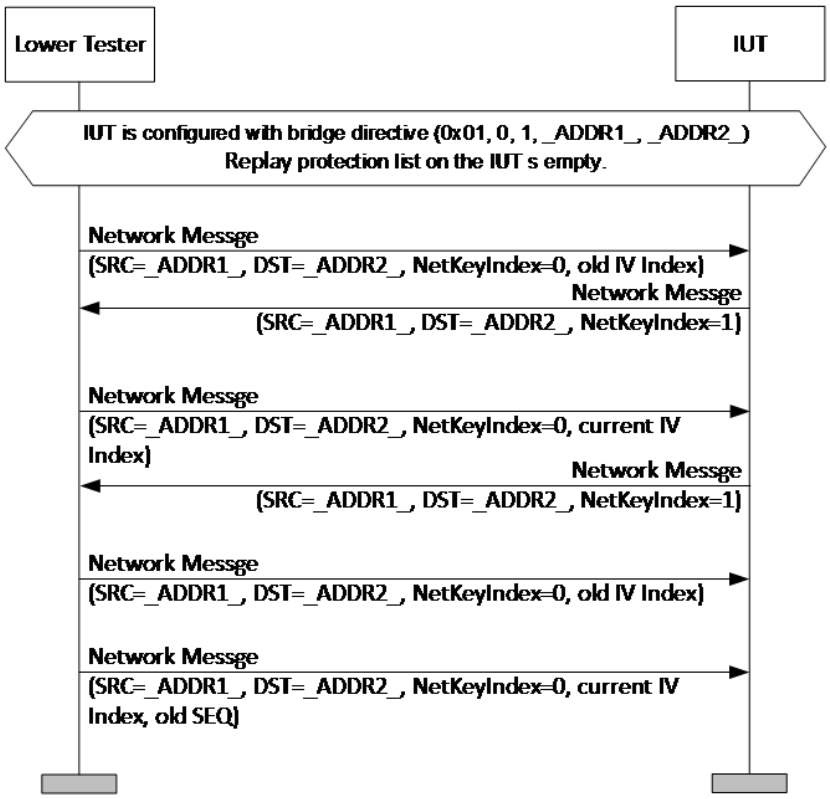


Figure 4.301: Test procedure for MESH/NODE/SBR/NET/BV-02-C [Ignore Replayed Network Layer Message] MSC

1. The Lower Tester sends a network message with the SRC set to _ADDR1_, DST set to _ADDR2_, and the TTL field set to a random valid value greater than or equal to 2. The message is encrypted on the first subnet and is using the previous IV index and a sequence number greater than 0.
2. The IUT sends the network message received in Step 1 with the TTL field decremented by 1. The message is re-encrypted with the network key on the second subnet.
3. The Lower Tester sends a network message with the SRC set to _ADDR1_, DST set to _ADDR2_, and the TTL field set to a random valid value greater than or equal to 2. The message is encrypted on the first subnet and is using the current IV index and a sequence number greater than the one used in Step 1.
4. The IUT sends the network message received in Step 3 with the TTL field decremented by 1. The message is re-encrypted with the network key on the second subnet.
5. The Lower Tester sends a network message with the SRC set to _ADDR1_, DST set to _ADDR2_, and the TTL field set to a random valid value greater than or equal to 2. The message is encrypted on the first subnet and is using the same IV index as the one used in Step 1 and a sequence number greater than the one used in Step 3.
6. The IUT does not send any network message.

7. The Lower Tester sends a network message with the SRC set to _ADDR1_, DST set to _ADDR2_, and the TTL field set to a random valid value greater than or equal to 2. The message is encrypted on the first subnet and is using the same IV index and a sequence number as the ones used in Step 3.
8. The IUT does not send any network message.

4.21 Bridge Configuration Models

4.21.1 Subnet Bridge State

This section validates the Subnet Bridge state and its messages for both the Bridge Configuration Server model and the Bridge Configuration Client model.

MESH/SR/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Behavior]

- Test Purpose

  Verify that a Bridge Configuration Server IUT can process SUBNET_BRIDGE_GET and SUBNET_BRIDGE_SET messages from a Bridge Configuration Client.

- Reference

  [10] 4.2.42, 4.9.2

- Initial Condition

  - The IUT is configured with the default values for the Subnet Bridge state.

- Test Procedure

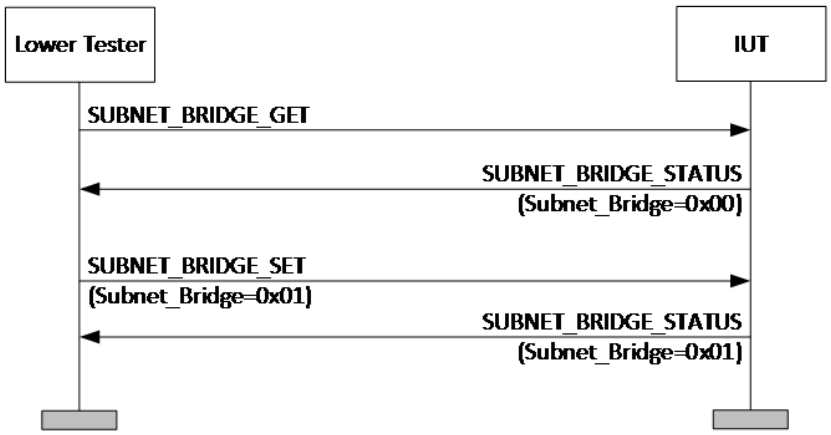


Figure 4.302: Test procedure for MESH/SR/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Behavior] MSC

1. The Lower Tester sends a SUBNET_BRIDGE_GET message.
2. The IUT responds with a SUBNET_BRIDGE_STATUS message with Subnet_Bridge field set to 0x00.
3. The Lower Tester sends a SUBNET_BRIDGE_SET message with Subnet_Bridge field set to 0x01.
4. The IUT responds with a SUBNET_BRIDGE_STATUS message with Subnet_Bridge field set to 0x01.

- Expected Outcome

  Pass verdict

  The IUT responds with SUBNET_BRIDGE_STATUS messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/SBS/BI-01-C [Subnet Bridge State Set Behavior – Invalid Parameters]

- Test Purpose

  Verify that a Bridge Configuration Server IUT ignores a SUBNET_BRIDGE_SET message with invalid parameters from a Bridge Configuration Client.

- Reference

  [10] 4.2.42, 4.3.11

- Test Procedure

  1. The Lower Tester sends a SUBNET_BRIDGE_SET message with Subnet_Bridge field set to a random prohibited value.
  2. The IUT does not respond with a SUBNET_BRIDGE_STATUS message.

- Expected Outcome

  Pass verdict

  The IUT does not respond with a SUBNET_BRIDGE_STATUS message.

MESH/CL/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Procedure]

- Test Purpose

  Verify that a Bridge Configuration Client IUT can configure the Subnet Bridge state on a Bridge Configuration Server.

- Reference

  [10] 4.2.42, 4.4.10

- Test Procedure

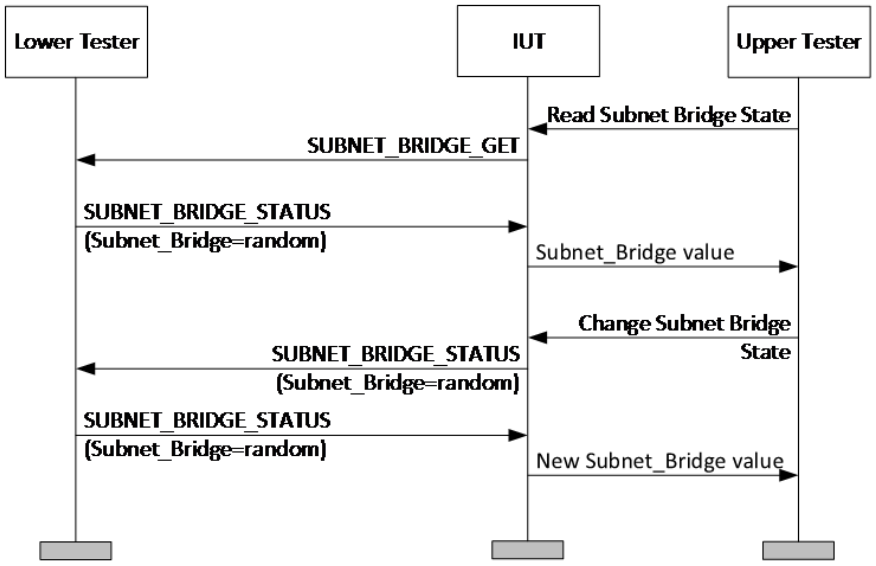


Figure 4.303: Test procedure for MESH/CL/BCM/SBS/BV-01-C [Subnet Bridge State Get/Set Procedure] MSC

1. The Upper Tester orders the IUT to read the value of the Subnet Bridge state on the Lower Tester.
2. The IUT sends a SUBNET_BRIDGE_GET message.
3. The Lower Tester responds with a SUBNET_BRIDGE_STATUS message with Subnet_Bridge set to a random valid value.
4. The IUT reports to the Upper Tester that it has received the parameters defined by SUBNET_BRIDGE_STATUS sent by the Lower Tester.
5. The Upper Tester orders the IUT to change the value of the Subnet Bridge state on the Lower Tester.
6. The IUT sends a SUBNET_BRIDGE_SET message with Subnet_Bridge set to a different value than the one from Step 3.
7. The Lower Tester responds with a SUBNET_BRIDGE_STATUS message with Subnet_Bridge set to the value from Step 6.
8. The IUT reports to the Upper Tester that it has received the parameters defined by SUBNET_BRIDGE_STATUS sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the SUBNET_BRIDGE_GET and SUBNET_BRIDGE_SET messages with the expected fields. The messages are encrypted with the device key.

  The IUT reports the expected values sent by the Lower Tester.

4.21.2 Bridging Table State

This section validates the Bridging Table state and its messages for both the Bridge Configuration Server model and the Bridge Configuration Client model.

MESH/SR/BCM/BDS/BV-01-C [Bridging Table State Add Behavior]

- Test Purpose

  Verify that a Bridge Server IUT can respond to a BRIDGING_TABLE_ADD message.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.

- Test Procedure

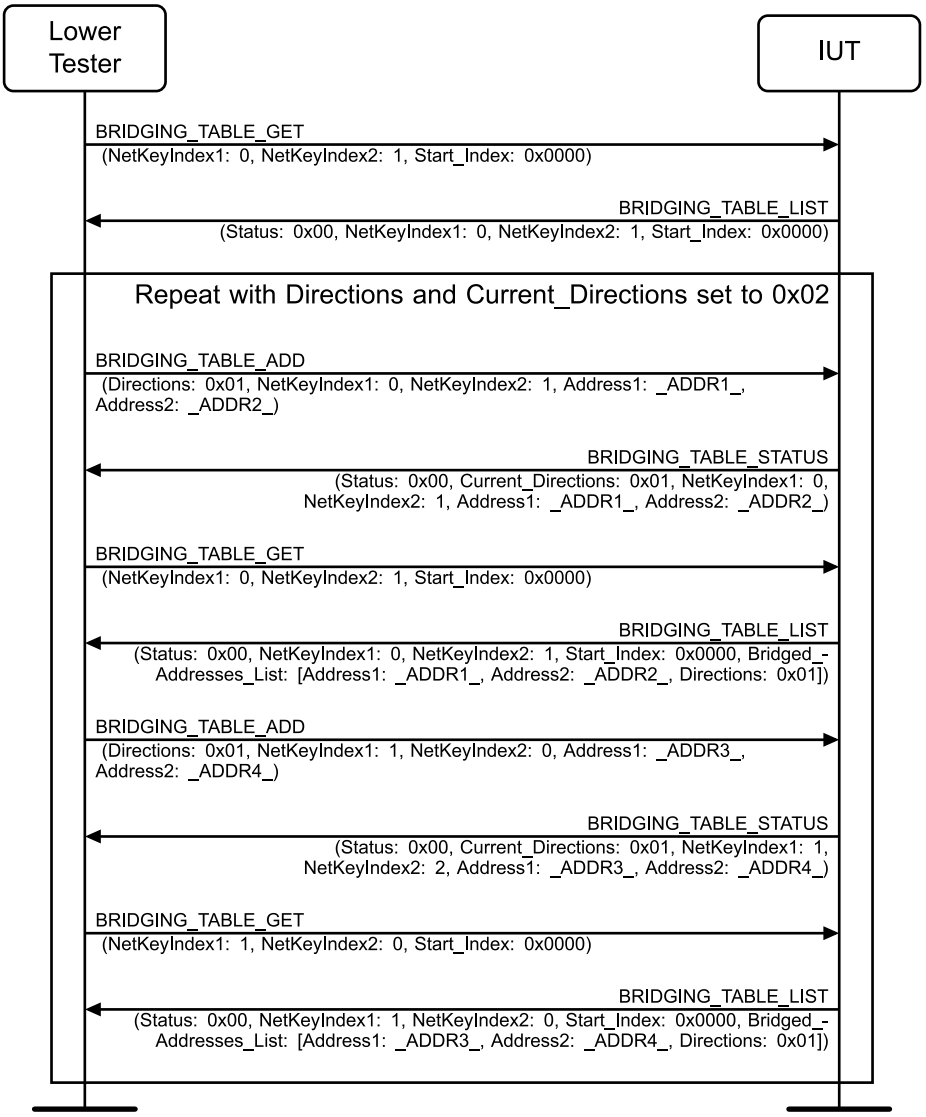


Figure 4.304: Test procedure for MESH/SR/BCM/BDS/BV-01-C [Bridging Table State Add Behavior] MSC

1. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
2. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and no Bridged_Addresses_List field.
3. The Lower Tester sends a BRIDGING_TABLE_ADD message with Directions set to 0x01, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
4. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x01, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
5. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.

6. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and Bridged_Addresses_List set to { (Address1 = _ADDR1_, Address2 = _ADDR2_, Directions = 0x01)}.
7. The Lower Tester sends a BRIDGING_TABLE_ADD message with Directions set to 0x01, NetKeyIndex1 set to 1, NetKeyIndex2 set to 0, Address1 set to _ADDR3_, and Address2 set to _ADDR4_.
8. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x01, NetKeyIndex1 set to 1, NetKeyIndex2 set to 0, Address1 set to _ADDR3_, and Address2 set to _ADDR4_.
9. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 1, NetKeyIndex2 set to 0, and Start_Index set to 0x0000.
10. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 1, NetKeyIndex2 set to 0, Start_Index set to 0x0000, and Bridged_Addresses_List set to { (Address1 = _ADDR3_, Address2 = _ADDR4_, Directions = 0x01)}.
11. Repeat Steps 3–10 with Current_Directions or Directions set to 0x02 in all the steps.

- Expected Outcome

  Pass verdict

  The IUT responds with BRIDGING_TABLE_STATUS and BRIDGING_TABLE_LIST messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/BDS/BV-02-C [Bridging Table State Remove Behavior]

- Test Purpose

  Verify that a Bridge Server IUT can respond to BRIDGING_TABLE_REMOVE messages.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.
  - The IUT is configured with the following bridging table entries by running preamble 4.2.7. (The _ADDR_ items are random unicast addresses.)
    - (0x01, 0, 1, _ADDR1_, _ADDR2_)
    - (0x02, 0, 1, _ADDR1_, _ADDR3_)
    - (0x01, 0, 1, _ADDR4_, _ADDR3_)
    - (0x02, 0, 1, _ADDR2_, _ADDR3_)

- Test Procedure

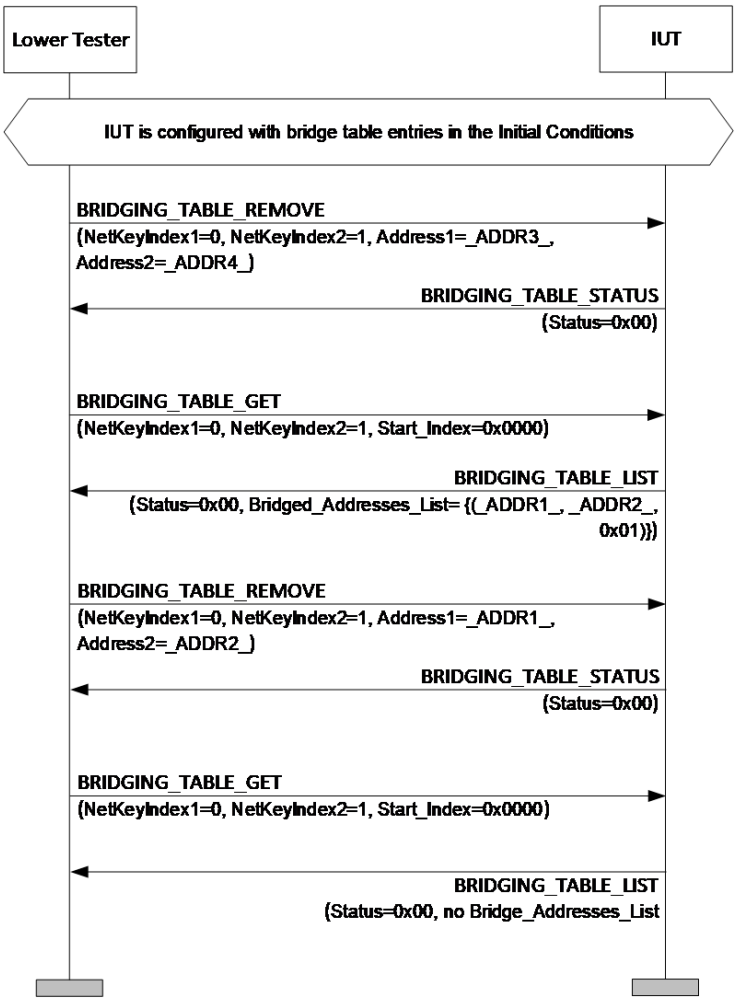


Figure 4.305: Test procedure for MESH/SR/BCM/BDS/BV-02-C [Bridging Table State Remove Behavior] MSC

1. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR3_.
2. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x00, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR3_.
3. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
4. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, Bridged_Addresses_List set to {(Address1 = _ADDR1_, Address2 = _ADDR2_, Directions = 0x01), (Address1 = _ADDR4_, Address2 = _ADDR3_, Directions = 0x01), (Address1 =_ADDR2_, Address2=_ADDR3_, Directions=0x02)}.
5. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.

6. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x00, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
7. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
8. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and Bridged_Addresses_List set to {(Address1 = _ADDR4_, Address2 = _ADDR3_, Directions = 0x01), (Address1 =_ADDR2_, Address2=_ADDR3_, Directions=0x02)}.
9. The Lower Tester adds all the deleted entries back to Bridging Table state, so that the Bridging Table state is restored as per initial conditions.
10. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to UNASSIGNED.
11. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x00, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to UNASSIGNED.
12. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
13. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and Bridged_Addresses_List set to { (Address1 = _ADDR4_, Address2 = _ADDR3_, Directions = 0x01), (Address1 =_ADDR2_, Address2=_ADDR3_, Directions=0x02)}.
14. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to UNASSIGNED, and Address2 set to _ADDR3_.
15. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x00, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to UNASSIGNED, and Address2 set to _ADDR3_.
16. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
17. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and an empty Bridged_Addresses_List field.

- Expected Outcome

  Pass verdict

  The IUT responds with BRIDGING_TABLE_STATUS and BRIDGING_TABLE_LIST messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/BDS/BV-03-C [Bridging Table State Get Behavior]

- Test Purpose

  Verify that a Bridge Server IUT can respond to BRIDGING_TABLE_GET messages.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition

  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.
  - The IUT is configured with the bridging table entries (0x02, 0, 1, _ADDR1_, _ADDR2_) and (0x01, 0, 1, _ADDR3_, _ADDR4_) by running preamble 4.2.7. _ADDR_ items are random unicast addresses.

- Test Procedure

  Repeat Steps 1 and 2 for each round in Table 4.144:

  1. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1, NetKeyIndex2, and Start_Index set to the values in Table 4.144.
  2. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1, NetKeyIndex2, and Start_Index set to the values in Step 1, and Bridged_Addresses_List set to the list in Table 4.144.

Round	Get	List of (Address1, Address2, Directions)
1	NetKeyIndex1 = 0, NetKeyIndex2 = 1, Start_Index = 0x0000	(_ADDR1_, _ADDR2_, 0x02), (_ADDR3_, _ADDR4_, 0x01)
2	NetKeyIndex1 = 0, NetKeyIndex2 = 1, Start_Index = 0x0001	List contains _ADDR3_, _ADDR4_, 0x01

Table 4.144: Rounds for MESH/SR/BCM/BDS/BV-03-C [Bridging Table State Get Behavior]

- Expected Outcome

  Pass verdict

  The IUT responds with BRIDGING_TABLE_LIST messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/BDS/BV-04-C [Bridging Table State Subnets Get Behavior]

- Test Purpose

  Verify that a Bridge Server IUT can respond to BRIDGED_SUBNETS_GET messages.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition

- The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
- The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.
- The IUT is configured with the bridging table entry (0x02, 0, 1, _ADDR1_, _ADDR2_) by running preamble 4.2.7. _ADDR_ items are random unicast addresses.

- Test Procedure

  Repeat steps for each round in Table 4.145:

  1. The Lower Tester sends a BRIDGED_SUBNETS_GET message with Filter, NetKeyIndex, and Start_Index set to the values in Table 4.145.
  2. The IUT responds with a BRIDGED_SUBNETS_LIST message with Filter, NetKeyIndex, and Start_Index set to the values in Step 1, and Bridged_Subnets_List set to the list in Table 4.145, if present.

Round	Get	List of (NetKeyIndex1, NetKeyIndex2)
1	Filter = 0b00, NetKeyIndex = 0, Start_Index = 0x0000	(NetKeyIndex1 = 0, NetKeyIndex2 = 1)
2	Filter = 0b01, NetKeyIndex = 0, Start_Index = 0x0000	(NetKeyIndex1 = 0, NetKeyIndex2 = 1)
3	Filter = 0b11, NetKeyIndex = 1, Start_Index = 0x0000	(NetKeyIndex1 = 0, NetKeyIndex2 = 1)
4	Filter = 0b11, NetKeyIndex = 1, Start_Index = 0x0001	Not present
5	Filter = 0b11, NetKeyIndex = 2, Start_Index = 0x0000	Not present
6	Filter = 0b10, NetKeyIndex = 0, Start_Index = 0x0000	Not present
7	Filter = 0b10, NetKeyIndex = 1, Start_Index = 0x0000	(NetKeyIndex1 = 0, NetKeyIndex2 = 1)

Table 4.145: Rounds for MESH/SR/BCM/BDS/BV-04-C [Bridging Table State Subnets Get Behavior]

- Expected Outcome

  Pass verdict

  The IUT responds with BRIDGED_SUBNETS_LIST messages with the expected field values. The

messages are encrypted with the device key.

MESH/SR/BCM/BDS/BV-05-C [Bridging Table State Behavior – Error Condition]

- Test Purpose

  Verify that a Bridge Server IUT can respond to Bridging Table messages that are unsuccessfully processed.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.
  - _ADDR_ items are random unicast addresses.

- Test Procedure

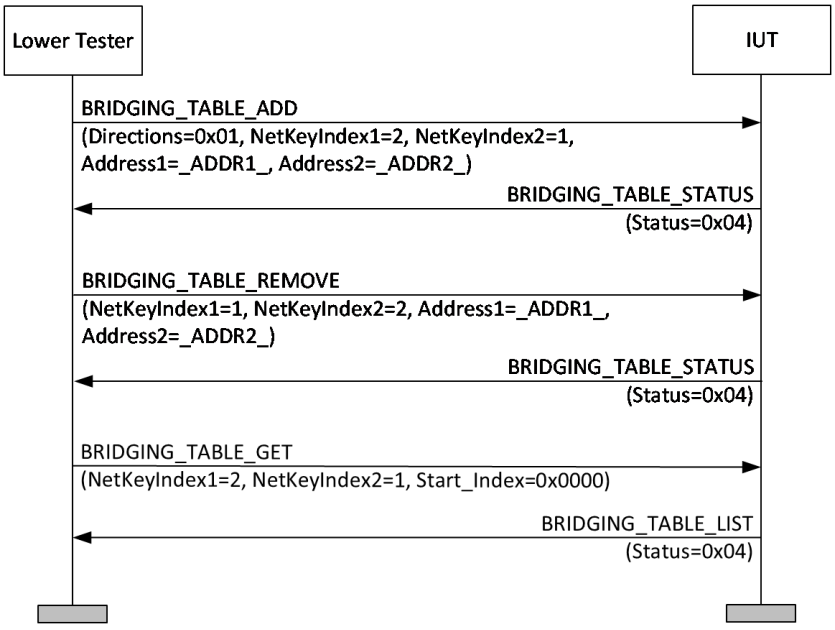


Figure 4.306: Test procedure for MESH/SR/BCM/BDS/BV-05-C [Bridging Table State Behavior – Error Condition] MSC

1. The Lower Tester sends a BRIDGING_TABLE_ADD message with Directions set to 0x01, NetKeyIndex1 set to 2, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
2. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x04 (Invalid NetKey Index), Current_Directions set to 0x01, NetKeyIndex1 set to 2, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.

3. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 2, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
4. The IUT responds with a BRIDGING_TABLE_STATUS message with Status set to 0x04 (Invalid NetKey Index), Current_Directions set to 0x00, NetKeyIndex1 set to 0, NetKeyIndex2 set to 2, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
5. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 2, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
6. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex1 set to 2, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.

- Expected Outcome

  Pass verdict

  The IUT responds with BRIDGING_TABLE_STATUS and BRIDGING_TABLE_LIST messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/BDS/BV-06-C [Bridging Table State – Binding with NetKey List State]

- Test Purpose

  Verify the binding between the Bridging Table state on a Bridge Server model and the NetKey List state on a Configuration Server model on the IUT.

- Reference

  [10] 4.2.42, 4.4.9

- Initial Condition
  - The subnet bridge functionality is enabled, and the Bridging Table state is empty on the IUT.
  - The IUT is configured in two subnets associated with the NetKey Index values of 0 and 1.
  - The IUT is configured with the bridging table entry (0x02, 0, 1, _ADDR1_, _ADDR2_) by running preamble 4.2.7. _ADDR_ items are random unicast addresses.

- Test Procedure

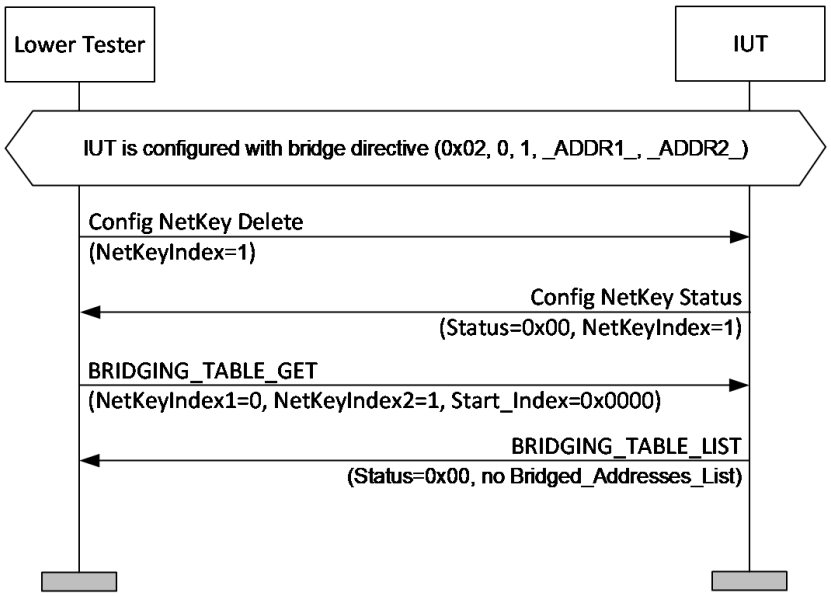


Figure 4.307: Test procedure for MESH/SR/BCM/BDS/BV-06-C [Bridging Table State – Binding with NetKey List State] MSC

1. The Lower Tester sends a Config NetKey Delete message to the IUT with NetKeyIndex field set to of 1.
2. The IUT sends a Config NetKey Status message in response from the IUT with Status set to 0x00 and the NetKeyIndex field set to 1.
3. The Lower Tester sends a BRIDGING_TABLE_GET message with NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, and Start_Index set to 0x0000.
4. The IUT responds with a BRIDGING_TABLE_LIST message with Status set to 0x04 (Invalid NetKey Index), NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Start_Index set to 0x0000, and no Bridged_Addresses_List field.

- Expected Outcome

  Pass verdict

  The IUT responds with Config NetKey Status and BRIDGING_TABLE_LIST messages with the expected field values. The messages are encrypted with the device key.

MESH/SR/BCM/BDS/BI-01-C [Bridging Table State Add Behavior – Invalid Parameters]

- Test Purpose

  Verify that a Bridge Server IUT ignores a Bridging Table Add message with invalid parameters.

- Reference

  [10] 4.2.42, 4.3.11

- Initial Condition
  - _ADDR_ items are random unicast addresses.
- Test Procedure

  Repeat Steps 1 and 2 for each row in Table 4.146:

  1. The Lower Tester sends a BRIDGING_TABLE_ADD message with the following parameters, unless stated in Table 4.146: Directions set to 0x01, NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
  2. The IUT does not respond with a BRIDGING_TABLE_STATUS message.

Round	Additional Condition
1	Directions field set to a random prohibited value
2	Address1 set to a random group address
3	Directions = 0x02, Address2 set to a random group address
4	Address2 = 0x0000
5	NetKeyIndex1 = NetKeyIndex2
6	Address1 = Address2
7	Directions = 0x01, Address2 = All Nodes Fixed Group Address
8	Address1 = 0x0000

Table 4.146: Rounds for MESH/SR/BCM/BDS/BI-01-C [Bridging Table State Add Behavior – Invalid Parameters]

- Expected Outcome

  Pass verdict

  The IUT does not respond with a BRIDGING_TABLE_STATUS message.

MESH/SR/BCM/BDS/BI-02-C [Bridging Table State Remove Behavior – Invalid Parameters]

- Test Purpose

  Verify that a Bridge Server IUT ignores a Bridging Table Remove message with invalid parameters.

- Reference

  [10] 4.2.42, 4.3.11

- Initial Condition
  - _ADDR_ items are random unicast addresses.
- Test Procedure

  Repeat Steps 1 and 2 for each row in Table 4.147:

1. The Lower Tester sends a BRIDGING_TABLE_REMOVE message with the following parameters, unless stated in Table 4.147: NetKeyIndex1 set to 0, NetKeyIndex2 set to 1, Address1 set to _ADDR1_, and Address2 set to _ADDR2_.
2. The IUT does not respond with a BRIDGING_TABLE_STATUS message.

Round	Additional Condition
1	Address1 set to a random group address
2	Address2 = all-nodes fixed group address
3	NetKeyIndex1 = NetKeyIndex2
4	Address1 = Address2 = _ADDR1_

Table 4.147: Rounds for MESH/SR/BCM/BDS/BI-02-C [Bridging Table State Remove Behavior – Invalid Parameters]

- Expected Outcome

  Pass verdict

  The IUT does not respond with a BRIDGING_TABLE_STATUS message.

MESH/SR/BCM/BDS/BI-03-C [Bridging Table State Subnets Get Behavior – Invalid Parameters]

- Test Purpose

  Verify that a Bridge Server IUT ignores a Bridge Subnets Get message with invalid parameters.

- Reference

  [10] 4.2.42, 4.3.11

- Initial Condition
  - The subnet bridge functionality is enabled on the IUT.
- Test Procedure
  1. The Lower Tester sends a BRIDGED_SUBNETS_GET message with Prohibited bits set, Filter, NetKeyIndex, and Start_Index set to valid values.
  2. The IUT does not respond with a BRIDGED_SUBNETS_LIST message.
- Expected Outcome

  Pass verdict

  The IUT does not respond with a BRIDGED_SUBNETS_LIST message.

MESH/CL/BCM/BDS/BV-01-C [Bridging Table State Procedure]

- Test Purpose

Verify that a Bridge Configuration Client IUT can configure and query the Bridging Table state on a Bridge Configuration Server.

- Reference

  [10] 4.2.42, 4.4.10

- Test Procedure

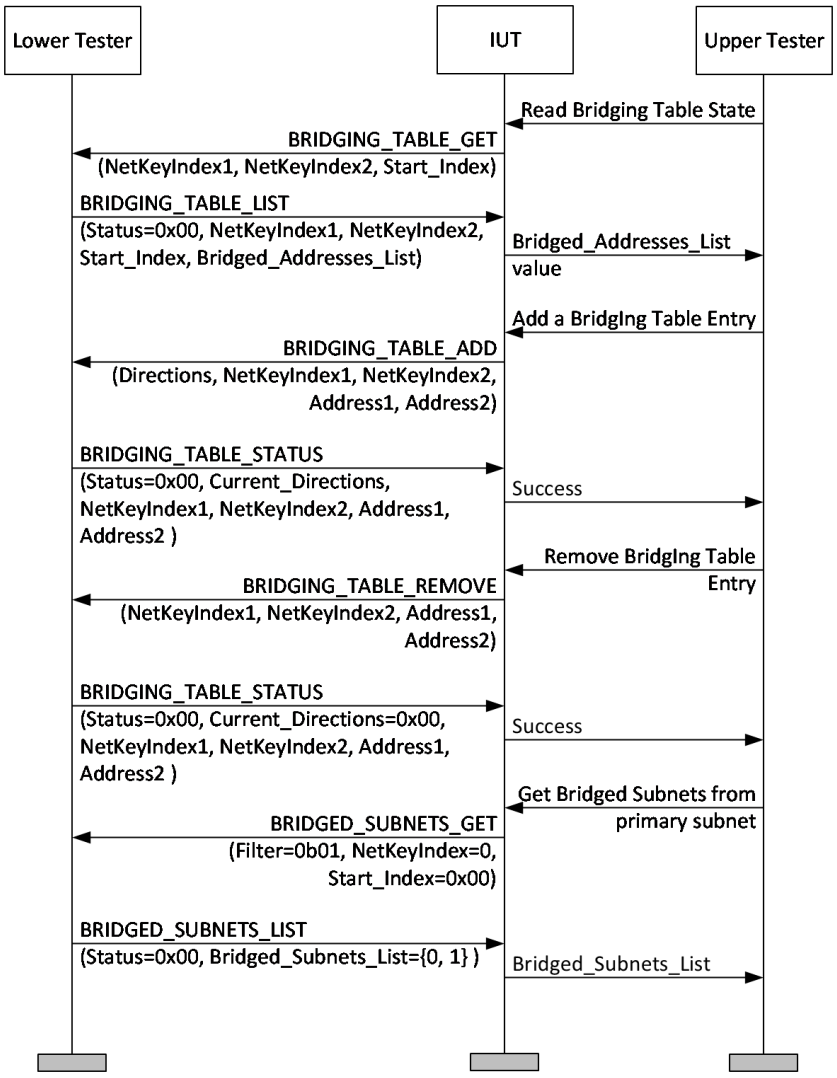


Figure 4.308: Test procedure for MESH/CL/BCM/BDS/BV-01-C [Bridging Table State Procedure] MSC

1. The Upper Tester orders the IUT to read the Bridging Table state entries on the Lower Tester.
2. The IUT sends a BRIDGING_TABLE_GET message with NetKeyIndex1, NetKeyIndex2, and Start_Index set to random valid values.
3. The Lower Tester responds with a BRIDGING_TABLE_LIST message with Status set to 0x00 (Success), NetKeyIndex1, NetKeyIndex2, Start_Index set to the values from Step 2, and the Bridged_Addresses_List field set to random valid values.

4. The IUT reports to the Upper Tester that it has received the parameters defined by BRIDGING_TABLE_LIST sent by the Lower Tester.
5. The Upper Tester orders the IUT to add a Bridging Table entry on the Lower Tester.
6. The IUT sends a BRIDGING_TABLE_ADD message with Directions, NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to random valid values.
7. The Lower Tester responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions, NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to the values from Step 6.
8. The IUT reports to the Upper Tester that it has received the parameters defined by BRIDGING_TABLE_STATUS sent by the Lower Tester.
9. The Upper Tester orders the IUT to remove the previously added Bridging Table entry on the Lower Tester.
10. The IUT sends a BRIDGING_TABLE_REMOVE message with NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to the values from Step 6.
11. The Lower Tester responds with a BRIDGING_TABLE_STATUS message with Status set to 0x00 (Success), Current_Directions set to 0x00, NetKeyIndex1, NetKeyIndex2, Address1, and Address2 set to the values from Step 6.
12. The IUT reports to the Upper Tester that it has received the parameters defined by BRIDGING_TABLE_STATUS sent by the Lower Tester.
13. The Upper Tester orders the IUT to get a pair of NetKey Indexes of bridged subnets with the NetKey Index of the first subnet matching 0, from the Lower Tester.
14. The IUT sends a BRIDGED_SUBNETS_GET message with Filter set to 0b01, Prohibited set to zeros, NetKeyIndex set to 0, and Start_Index set to 0x00.
15. The Lower Tester responds with a BRIDGED_SUBNETS_LIST message with NetKeyIndex set to 0, Start_Index set to 0x00, and Bridged_Subnets_List set to {0, 1}.
16. The IUT reports to the Upper Tester that it has received the parameters defined by BRIDGED_SUBNETS_LIST sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the specified messages with the expected field values. The messages are encrypted with the device key.

4.21.3 Bridging Table Size State

This section validates the Bridging Table Size state and its messages for both the Bridge Configuration Server model and the Bridge Configuration Client model.

MESH/SR/BCM/BTS/BV-01-C [Bridging Table Size Get Behavior]

- Test Purpose

  Verify that a Bridge Configuration Server IUT can respond to a BRIDGING_TABLE_SIZE_GET message.

- Reference

[10] 4.2.43, 4.4.9

- Test Procedure

  1. The Lower Tester sends a BRIDGING_TABLE_SIZE_GET message.
  2. The IUT responds with a BRIDGING_TABLE_SIZE_STATUS message with the Bridging_Table_Size field set to a valid value.

- Expected Outcome

  Pass verdict

  The IUT responds with a BRIDGING_TABLE_SIZE_STATUS message with the expected field value. The messages are encrypted with the device key.

MESH/CL/BCM/BTS/BV-01-C [Bridging Table Size State Procedure]

- Test Purpose

  Verify that a Bridge Configuration Client IUT can read the Bridging Table Size state on a Bridge Configuration Server.

- Reference

  [10] 4.2.43, 4.4.10

- Test Procedure

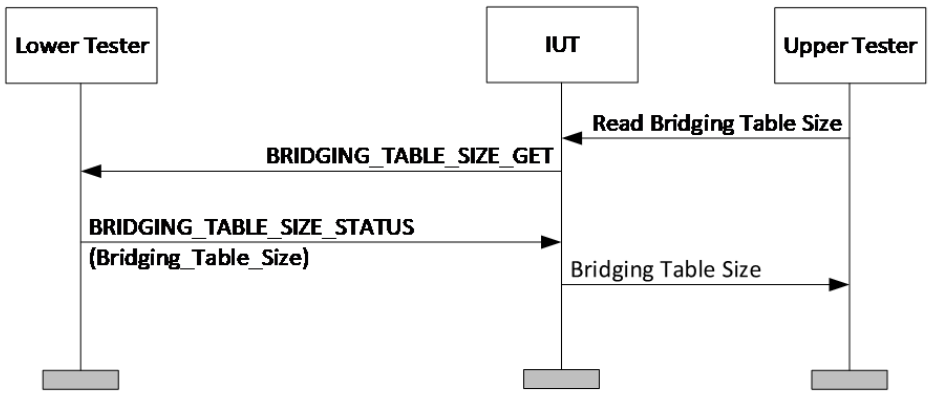


Figure 4.309: Test procedure for MESH/CL/BCM/BTS/BV-01-C [Bridging Table Size State Procedure] MSC

1. The Upper Tester orders the IUT to read the Bridging Table Size state on the Lower Tester.
2. The IUT sends a BRIDGING_TABLE_SIZE_GET message.
3. The Lower Tester responds with a BRIDGING_TABLE_SIZE_STATUS message with the Bridging_Table_Size field set to a random valid value.
4. The IUT reports to the Upper Tester that it has received the parameters defined by BRIDGING_TABLE_SIZE_STATUS sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the BRIDGING_TABLE_GET and is able to extract the Bridging_Table_Size from the received BRIDGING_TABLE_SIZE_STATUS message. The messages are encrypted with the device key.

4.22 Remote Provisioning

For all the Remote Provisioning tests in Sections 4.22.1, 4.22.2, and 4.22.3, the following generic requirements and Pass verdicts will be applied in addition to the ones specified individually for each test:

- In Server tests (MESH/SR/RPR), all remote provisioning messages are encrypted using the device key of the IUT.
- In Client tests (MESH/CL/RPR), all remote provisioning messages are encrypted using the device key of the Lower Tester.
- All remote provisioning messages are exchanged between the Lower Tester and the primary element of the IUT.

4.22.1 Scanning

MESH/SR/RPR/SCN/BV-01-C [Scan Any Device UUID]

- Test Purpose

  Verify that the IUT can scan and report unprovisioned devices with any Device UUID.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device.
  - If the IUT supports more than one network key, then a second network key has been configured on the IUT.
- Test Procedure

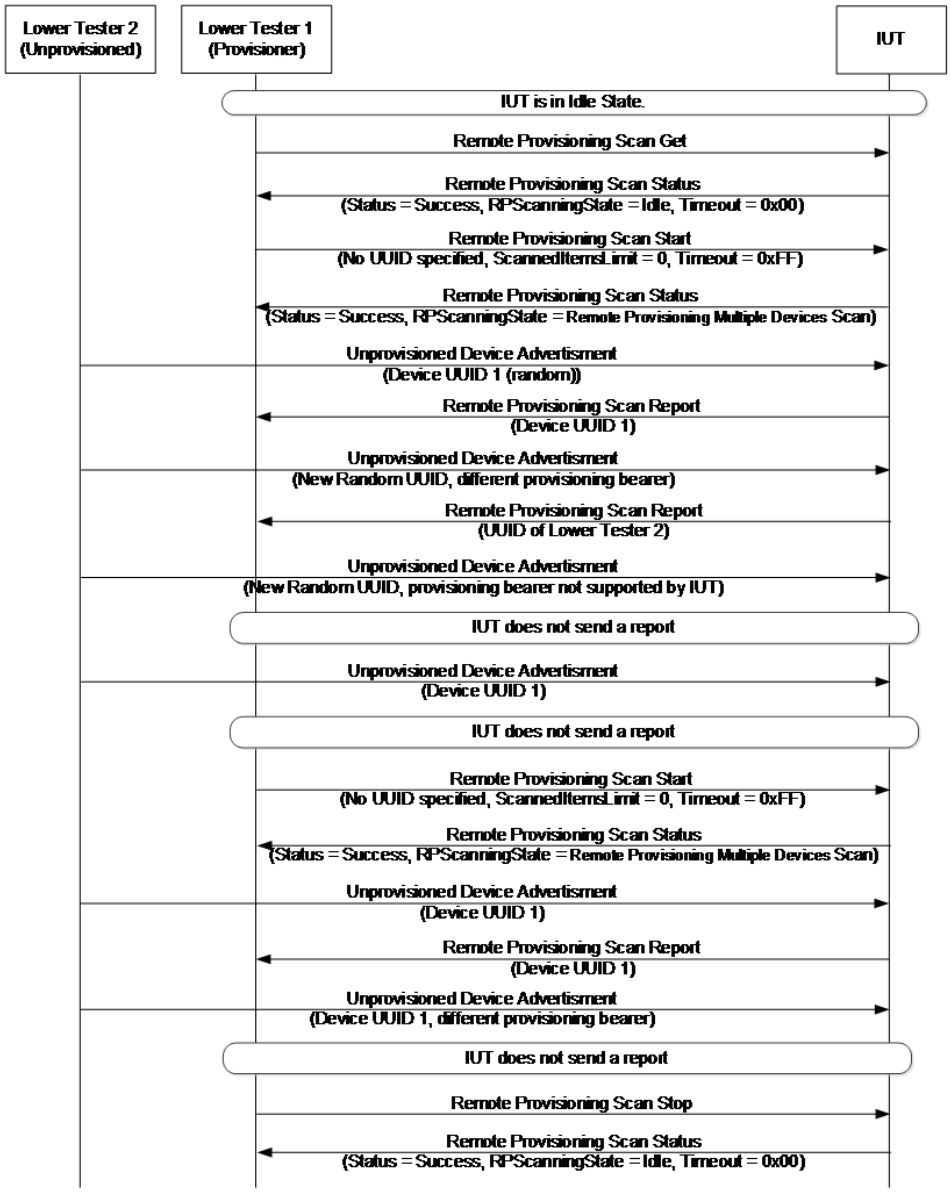


Figure 4.310: Test procedure for MESH/SR/RPR/SCN/BV-01-C MSC - Page 1 of 2

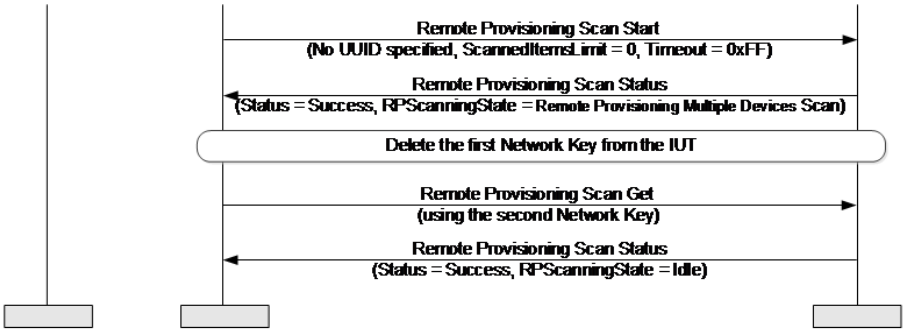


Figure 4.311: Test procedure for MESH/SR/RPR/SCN/BV-01-C MSC - Page 2 of 2

1. Lower Tester 1 sends a Remote Provisioning Scan Get message and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success, the RPScanningState field set to Idle, and the Timeout field set to 0.
2. Lower Tester 1 sends a Remote Provisioning Scan Start message, setting the ScannedItemsLimit field to 0, the Timeout field to 0xFF, and omitting the UUID field, and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success, the RPScanningState field set to Remote Provisioning Multiple Devices Scan, and the Timeout field set to 0xFF or 0xFE.
3. Lower Tester 2 begins sending unprovisioned device advertisements for a provisioning bearer supported by the IUT, using a randomly generated Device UUID.
4. The IUT sends a Remote Provisioning Scan Report message containing the UUID used by Lower Tester 2.
5. If the IUT supports more than one provisioning bearer, then repeat Steps 3 and 4, generating a new Device UUID and advertising for another provisioning bearer supported by the IUT (different than the one used in Step 3).
6. If the IUT does not support all provisioning bearers, then repeat Step 3, generating a new Device UUID and advertising for a provisioning bearer not supported by the IUT. If the IUT sends a Remote Provisioning Scan Report message containing the new Device UUID, then the test ends with a Fail verdict.
7. Repeat Step 3 using the same Device UUID generated in the first iteration of this step. If the IUT sends another Remote Provisioning Scan Report message containing the same Device UUID, then the test ends with a Fail verdict.
8. Repeat Steps 2–4 using the same Device UUID generated in the first iteration of Step 3.
9. If the IUT supports more than one provisioning bearer, then Lower Tester 2 begins advertising for another provisioning bearer supported by the IUT (different than the one used in Step 8), using the same Device UUID as in Step 8. If the IUT sends another Remote Provisioning Scan Report message containing the same Device UUID, then the test ends with a Fail verdict.
10. Lower Tester 1 sends a Remote Provisioning Scan Stop message to the IUT, and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success, the RPScanningState field set to Idle, and the Timeout field set to 0.
11. If the IUT supports more than one network key, repeat Step 2 and then delete the network key on the IUT that was used to start scanning. Then Lower Tester 1 sends a Remote Provisioning Scan Get message to the IUT using the second network key and the IUT responds with a Remote Provisioning

Scan Status message with the Status field set to Success and the RPScanningState field set to Idle.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the correct status codes and state values and sends scan reports when appropriate.

  The IUT does not send duplicate scan reports (containing the same Device UUID) during the same scanning session (before a scan restart is performed).

  The IUT does not send scan reports for unprovisioned devices advertising for a provisioning bearer not supported by the IUT.

  In each Remote Provisioning Scan Report message: the OOB field is set to the value of the OOB Information field from the corresponding unprovisioned device advertisement that is being reported; the URI Hash field is present only if the corresponding unprovisioned device advertisement contains a URI Hash field (in which case it has the same value), otherwise it is not present.

  The Remote Provisioning Scan Report messages are sent as segmented access messages.

MESH/SR/RPR/SCN/BV-02-C [Scan Specific Device UUID]

- Test Purpose

  Verify that the IUT can scan and report unprovisioned devices with a specified Device UUID.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device.
- Test Procedure

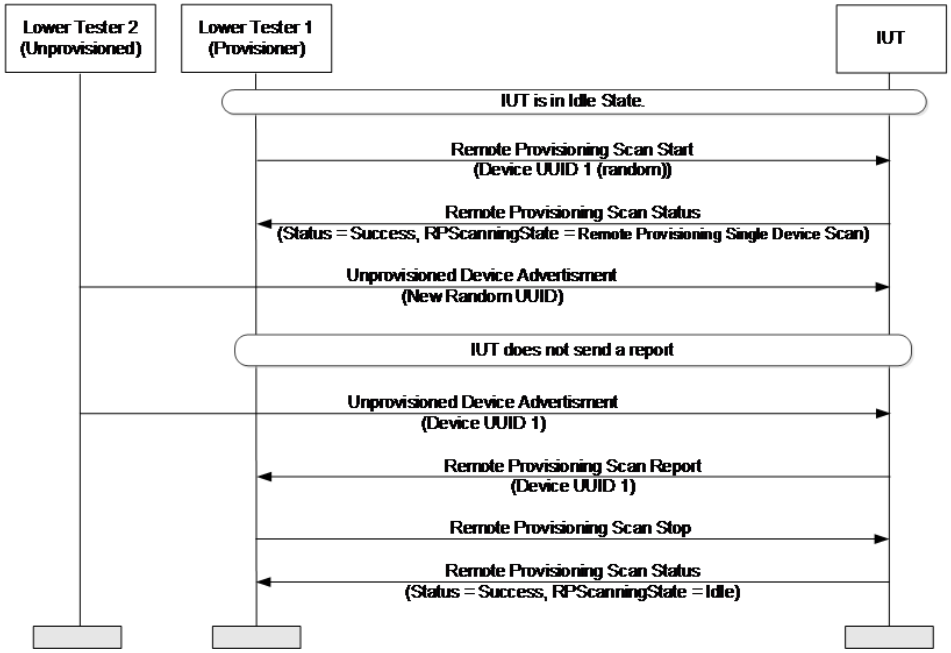


Figure 4.312: Test procedure for MESH/SR/RPR/SCN/BV-02-C MSC

1. Lower Tester 1 sends a Remote Provisioning Scan Start message, setting the UUID field to a randomly generated Device UUID, and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success and the RPScanningState field set to Remote Provisioning Single Device Scan.
2. Lower Tester 2 begins sending unprovisioned device advertisements for a provisioning bearer supported by the IUT, using a different UUID than the one generated in Step 1. The IUT ignores the second UUID and does not send any Remote Provisioning Scan Report messages.
3. Lower Tester 2 stops sending beacons or advertising, then restarts it using the same Device UUID generated in Step 1.
4. The IUT sends a Remote Provisioning Scan Report message containing the UUID used by Lower Tester 2.
5. Lower Tester 1 sends a Remote Provisioning Scan Stop message to the IUT and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success and the RPScanningState field set to Idle.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the correct status codes and state values and sends scan reports only when appropriate.

  In each Remote Provisioning Scan Report message, the OOB field is set to the value of the OOB Information field from the corresponding unprovisioned device advertisement that is being reported.

  The Remote Provisioning Scan Report message is sent as a segmented access message.

MESH/SR/RPR/SCN/BV-03-C [Process Consecutive Scan Requests]

- Test Purpose

  Verify that the IUT accepts redundant scan requests and that the IUT rejects scan requests from another client when scanning is in progress.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The IUT has been configured with two network keys.
  - The Lower Tester has access to the device key of the IUT.
  - If the IUT supports more than one network key, then a second network key has been configured on the IUT.

- Test Procedure

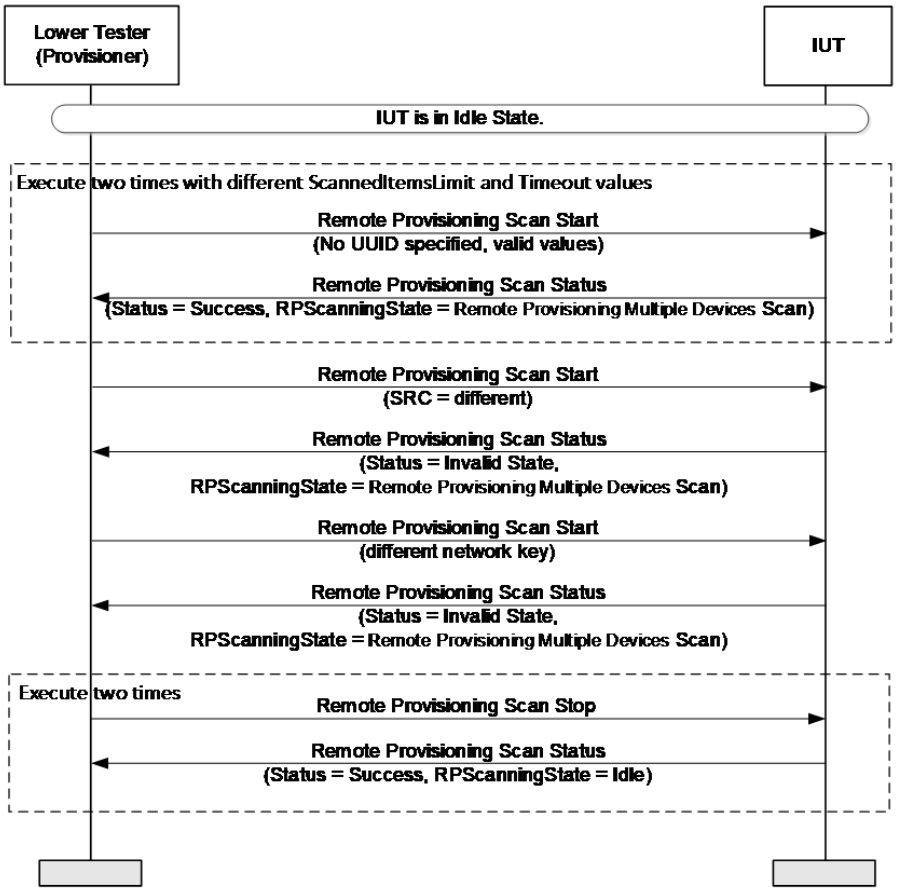


Figure 4.313: Test procedure for MESH/SR/RPR/SCN/BV-03-C MSC

1. The Lower Tester sends a Remote Provisioning Scan Start message, setting the ScannedItemsLimit field to a valid non-zero value, the Timeout field to a large value, and omitting the UUID field, and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success, the RPScanningState field set to Remote Provisioning Multiple Devices Scan, the ScannedItemsLimit field set to the same value as in the request message, and the Timeout value set to the same value as in the request message (or to a value that is lower by 1 or 2 seconds, to account for the processing time on the IUT).
2. Repeat Step 1 using different values for the ScannedItemsLimit and Timeout fields; the IUT reports the new values in the status message.
3. The Lower Tester resends the same message from Step 1 but using a different source unicast address and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Invalid State and the RPScanningState field set to Remote Provisioning Multiple Devices Scan.
4. If the IUT supports more than one network key, the Lower Tester resends the same message from Step 1 but using a different network key and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Invalid State and the RPScanningState field set to Remote Provisioning Multiple Devices Scan.
5. The Lower Tester sends a Remote Provisioning Scan Stop message to the IUT and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success and the

RPScanningState field set to Idle.

6. Repeat Step 5.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the correct status codes and state values, accepting duplicated requests and rejecting a scan start request with a different source address or network key when scanning is in progress.

4.22.1.1 Extended Scanning on Specific UUID

- Test Purpose

  Verify that the IUT can perform extended scanning of unprovisioned device advertisements.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition

  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

- Test Procedure

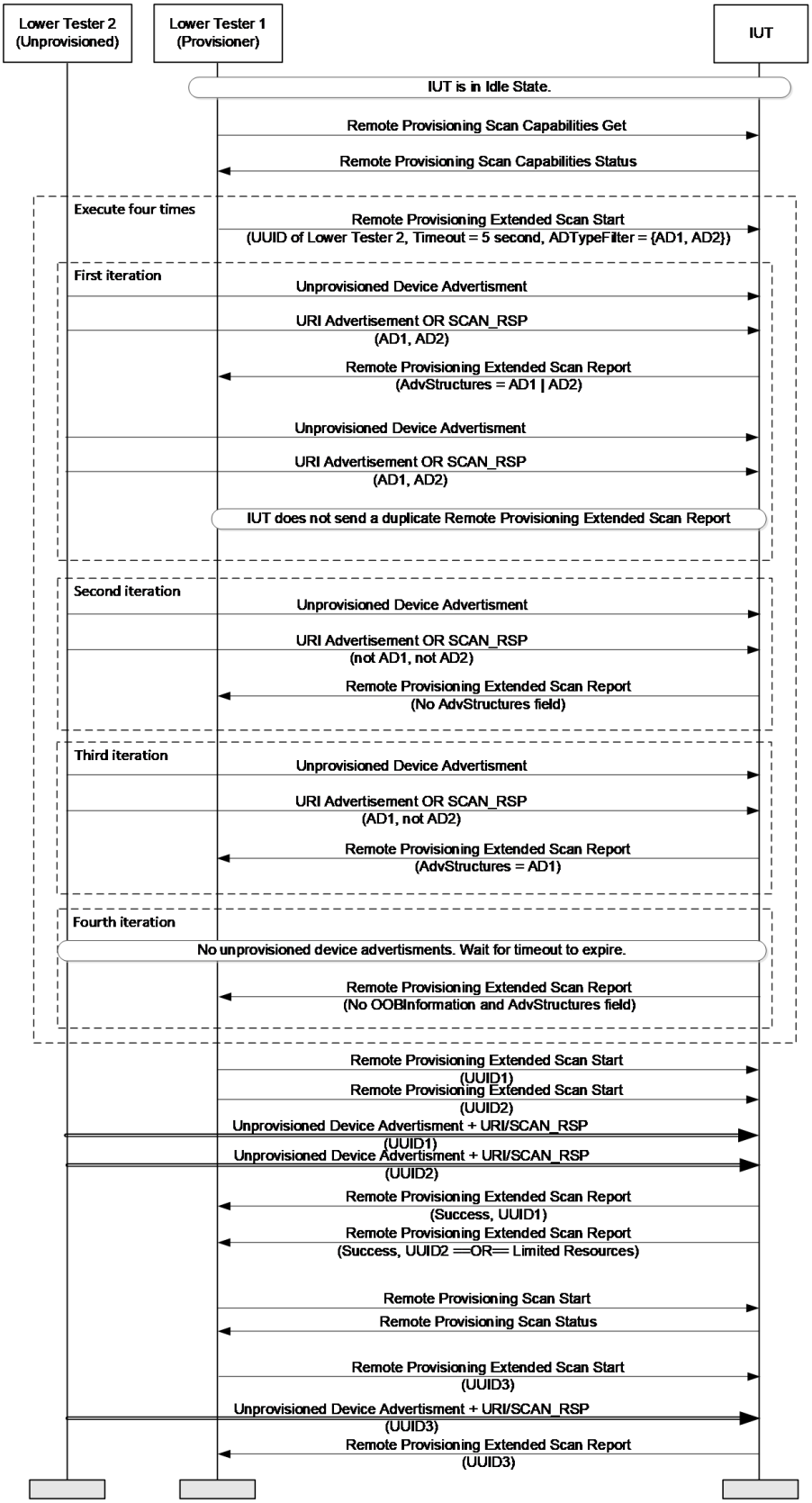


Figure 4.314: Test procedure - Extended Scanning on Specific UUID MSC

1. Lower Tester 1 sends a Remote Provisioning Scan Capabilities Get message and the IUT responds with a Remote Provisioning Scan Capabilities Status message with the MaxScannedItems field value in the interval 0x04-0xFF and the ActiveScan field set to the value specified in the Active Scan column in Table 4.148.
2. Lower Tester 1 sends a Remote Provisioning Extended Scan Start message, setting the UUID field to the Device UUID of Lower Tester 2, the Timeout field to 0x05, the ADTypeFilterCount field to 0x02, and the ADTypeFilter field to two random, valid AD Types.
3. Lower Tester 2 immediately begins either sending Unprovisioned Device Beacons periodically or advertising the Mesh Provisioning Service, according to the Unprovisioned Device Advertisements column in Table 4.148.
4. The IUT sends a Remote Provisioning Extended Scan Report message with the AdvStructures field containing the two requested AD structures sent by Lower Tester 2 in Step 3. The IUT sends the message at least 1 second before the 5-second timeout requested in Step 2 expires. If the IUT sends the message after the 5-second timeout expires, the test ends with a Fail verdict.
5. Repeat Step 3 for another 10 seconds, and if the IUT sends another Remote Provisioning Extended Scan Report message, then the test ends with a Fail verdict.
6. Repeat Steps 2 and 3, but in Step 3 send AD Types that are not contained in the ADTypeFilter field value from Step 2.
7. Within the timeout specified in Step 2, the IUT sends a Remote Provisioning Extended Scan Report message without the AdvStructures field. If the IUT sends the message after the 5-second timeout expires, the test ends with a Fail verdict.
8. Repeat Steps 2 and 3, but in Step 3 send only one AD Type that is contained in the ADTypeFilter field value from Step 2.
9. Within the timeout specified in Step 2, the IUT sends a Remote Provisioning Extended Scan Report message with the AdvStructures field containing the AD Type sent in Step 8 by Lower Tester 1 that matches the requested filter. If the IUT sends the message after the 5-second timeout expires, the test ends with a Fail verdict.
10. Repeat Step 2 and do not generate any unprovisioned device advertisements from Lower Tester 2. When the timeout expires, the IUT sends a Remote Provisioning Extended Scan Report message with the requested UUID and without the OOBInformation and AdvStructures field.
11. Lower Tester 1 sends two consecutive Remote Provisioning Extended Scan Start messages, similar to the message sent in Step 2, using two different UUIDs, then Lower Tester 2 begins sending unprovisioned device advertisements as instructed in Step 3, but interleaving two sets of advertisements, one for each UUID, alternating at an interval less than 1 second.
12. The IUT sends either:
    a. Two Remote Provisioning Extended Scan Report messages as described in Step 4, one for each UUID used by Lower Tester 2, or
    b. One Remote Provisioning Extended Scan Report message as described in Step 4 for one of the UUIDs used by Lower Tester 2, and one Remote Provisioning Extended Scan Report message with the Status field set to Limited Resources and the OOBInformation and AdvStructures fields omitted; the two messages can arrive in any order, or
    c. One Remote Provisioning Extended Scan Report message as described in Step 4 for one of the UUIDs used by Lower Tester 2, and a report to the Upper Tester that it cannot process the second Remote Provisioning Extended Scan Start message from Step 11 due to limited

resources.

13. Lower Tester 1 sends a Remote Provisioning Scan Start message, setting the ScannedItemsLimit field to 0, the Timeout field to 0xFF and omitting the UUID field, and the IUT responds with a Remote Provisioning Scan Status message with correct parameters.
14. Repeat Steps 2–4 to verify that the IUT can perform extended scanning while regular scanning is on.

- Test Case Configuration

TCID	Active Scan (Step 1)	Unprovisioned Device Advertisements (Step 3)
MESH/SR/RPR/SCN/BV-04-C [Extended Scanning of Unprovisioned Device Beacons]	ActiveScan set to 0x00 or 0x01	Lower Tester 2 is sending Unprovisioned Device Beacons, and the beacons contain the URI Hash field, and Lower Tester 2 also separately sends a second non-connectable advertisement with the URI AD Type together with both AD Types from the requested filter. The URI Hash from the beacon correctly identifies the second non-connectable advertisement.
MESH/SR/RPR/SCN/BV-05-C [Extended Scanning of Mesh Provisioning Service]	ActiveScan set to 0x01	Lower Tester 2 is sending connectable Mesh Provisioning Service advertisements, and the scan response contains both AD Types from the requested filter.

Table 4.148: Unprovisioned Device Advertisement Selection

- Expected Outcome

Pass verdict

The IUT sends the extended scan reports with the Status field set to Success and with correct values for the UUID and OOBInformation fields. The reports contain an AdvStructures field only when the AD Types from the unprovisioned device advertisements are matching the requested filter. The reports are sent immediately if the complete set of AD Types have been collected, and they are sent after the timeout expires if an incomplete or empty set of AD Types have been collected. When the timeout expires without any advertisements detected, the IUT sends the report containing only the UUID.

The IUT performs extended scanning even when regular scanning is active, and is able to perform two extended scanning procedures in parallel, unless it has limited resources, in which case it either sends a report with the Limited Resources error code, or it silently ignores the request and reports this to the Upper Tester.

The Remote Provisioning Extended Scan Report messages are sent as segmented access messages.

MESH/SR/RPR/SCN/BV-06-C [Extended Scanning on the Remote Provisioning Server]

- Test Purpose

Verify that the IUT can perform extended scanning on its own unprovisioned device information.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition

  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The Lower Tester has access to the device key of the IUT.
  - The Device UUID and unprovisioned device advertisements of the IUT are listed in the IXIT [6].

- Test Procedure

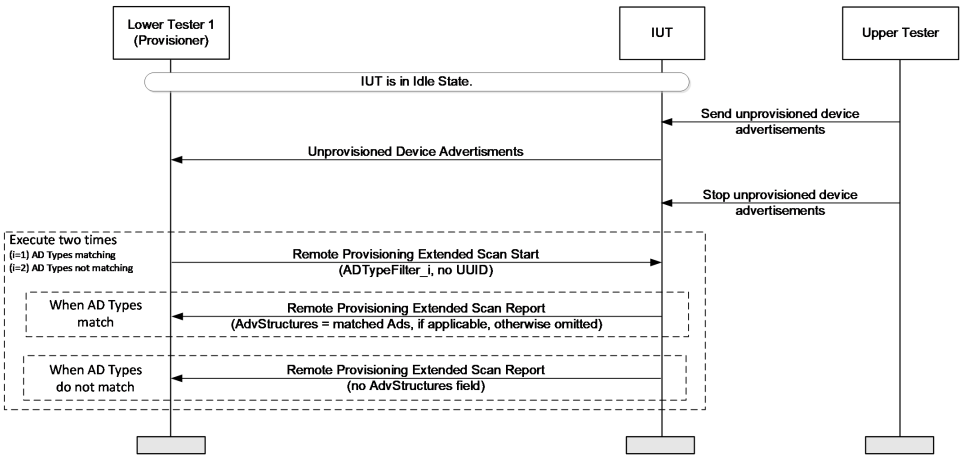


Figure 4.315: Test procedure for MESH/SR/RPR/SCN/BV-06-C MSC

1. The Upper Tester orders the IUT to send its unprovisioned device advertisements. The Lower Tester scans and saves the AD Types from these advertisements, then the Upper Tester orders the IUT to stop the unprovisioned device advertisements.
2. The Lower Tester sends a Remote Provisioning Extended Scan Start message, omitting the UUID field and setting all other fields to appropriate values.
3. The IUT sends a Remote Provisioning Extended Scan Report message with the AdvStructures field either omitted or containing the selected AD structures from the IUT's unprovisioned device advertisements.
4. Repeat Step 2 using an ADTypeFilter field value that does not correspond to any AD Type present in the IUT's unprovisioned device advertisement.
5. The IUT sends a Remote Provisioning Extended Scan Report message without the AdvStructures field.

- Expected Outcome

  Pass verdict

  The IUT sends the extended scan reports with the Status field set to Success and with correct values for

the UUID and OOBInformation fields. The reports may contain an AdvStructures field only when the AD Types from the unprovisioned device advertisements are matching the requested filter. The IUT sends the reports for its own advertisements when the UUID field is omitted.

MESH/SR/RPR/SCN/BV-07-C [Scan Items Limit and Timeout]

- Test Purpose

  Verify that the IUT limits the number of scanned devices and the scan time as requested by the Remote Provisioning Client.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device.

- Test Procedure

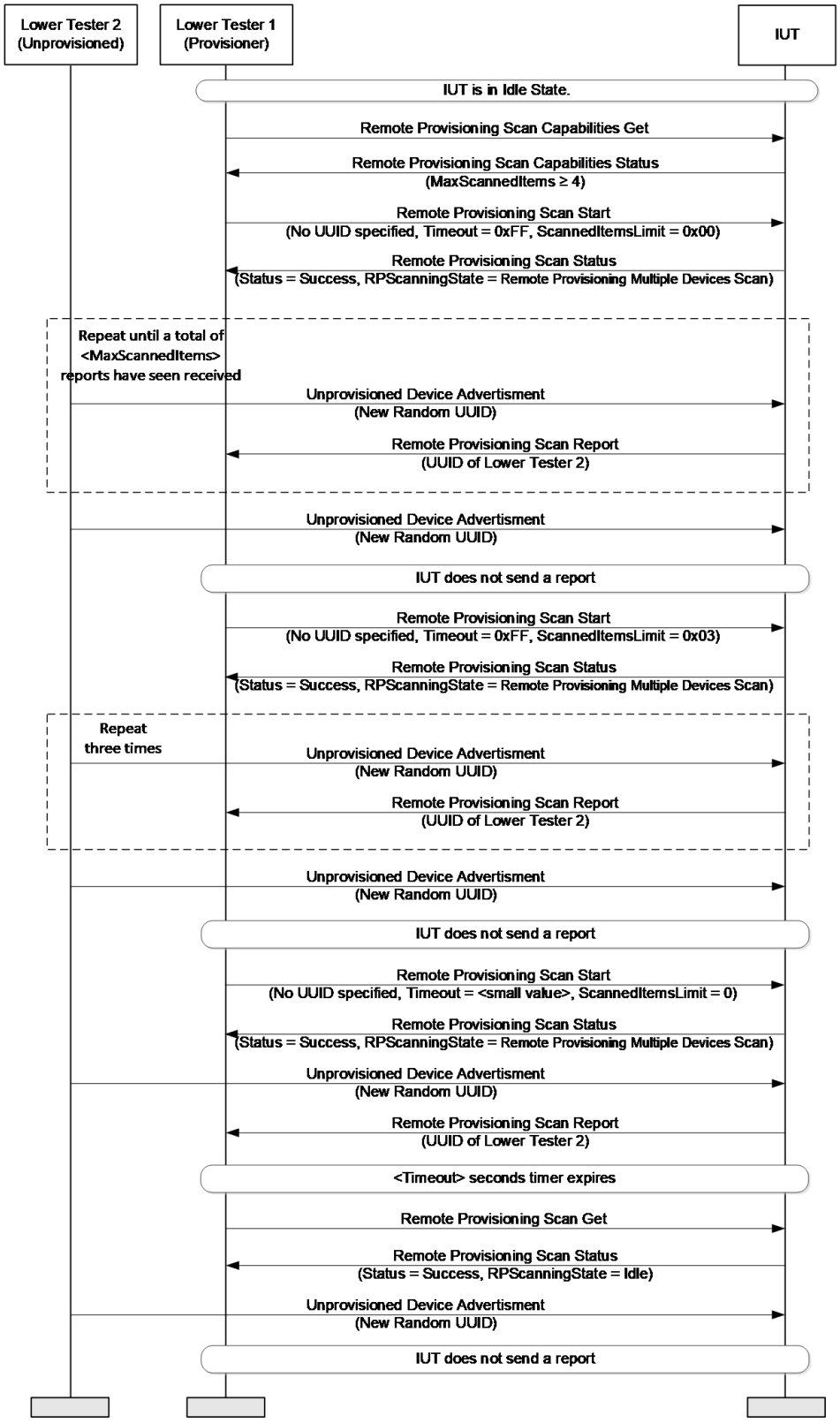


Figure 4.316: Test procedure for MESH/SR/RPR/SCN/BV-07-C MSC

1. Lower Tester 1 sends a Remote Provisioning Scan Capabilities Get message and the IUT responds with a Remote Provisioning Scan Capabilities Status message with the MaxScannedItems field set to a value greater than or equal to 0x04.
2. Lower Tester 1 sends a Remote Provisioning Scan Start message, setting the ScannedItemsLimit field to 0, the Timeout field to 0xFF and omitting the UUID field, and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success, the RPScanningState field set to Remote Provisioning Multiple Devices Scan, the Timeout field set in the range 0xFD-0xFF, and the ScannedItemsLimit field set to the value of the MaxScannedItems field of the Remote Provisioning Scan Capabilities Status message from Step 1.
3. Lower Tester 2 begins sending unprovisioned device advertisements for a provisioning bearer supported by the IUT, using a randomly generated Device UUID.
4. The IUT sends a Remote Provisioning Scan Report message containing the UUID used by Lower Tester 2.
5. Continue generating new advertisements as in Step 3 using new Device UUIDs until the number of Remote Provisioning Scan Report messages sent by the IUT is equal to the value of the MaxScannedItems field received in Step 1.
6. Repeat Step 3 using a new Device UUID. If the IUT sends a new Remote Provisioning Scan Report message, then the test ends with a Fail verdict.
7. Repeat Steps 2–6, but in Step 2 set the ScannedItemsLimit field to 0x03, and in Step 5 use this value instead of the MaxScannedItems field value (the IUT sends 3 reports).
8. Repeat Steps 2–4, but in Step 2 set the Timeout field to a reasonably small value.
9. After the timeout set in the previous step has expired, Lower Tester 1 sends a Remote Provisioning Scan Get message and the IUT responds with a Remote Provisioning Scan Status message with the Status field set to Success and the RPScanningState field set to Idle.
10. Repeat Step 3 using a new Device UUID and the IUT does not respond.

- Expected Outcome

  Pass verdict

  The IUT limits the number of reports and the scan time as requested by the Remote Provisioning Client.

  The Remote Provisioning Scan Report messages are sent as segmented access messages.

MESH/SR/RPR/SCN/BV-08-C [Extended Scanning with Bad Hash or Different BD_ADDR]

- Test Purpose

  Verify that the IUT checks the URI Hash and ignores the BD_ADDR when performing extended scanning of Unprovisioned Device Beacons.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition

- The IUT is in the Idle state.
- Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
- Lower Tester 1 has access to the device key of the IUT.
- Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

• Test Procedure

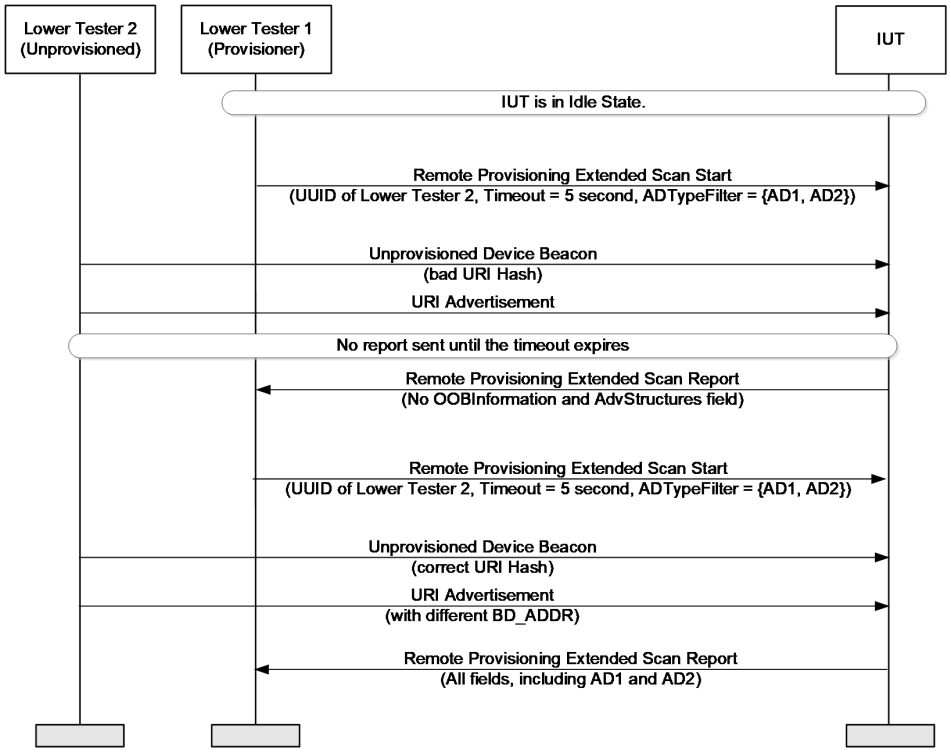


Figure 4.317: Test procedure for MESH/SR/RPR/SCN/BV-08-C MSC

1. Lower Tester 1 sends a Remote Provisioning Extended Scan Start message, setting the UUID field to the Device UUID of Lower Tester 2, the Timeout field to 0x05, the ADTypeFilterCount field to 0x02, and the ADTypeFilter field to two random, valid AD Types.
2. Lower Tester 2 immediately starts sending non-connectable advertisements containing Unprovisioned Device Beacons and no other AD Types, and the beacons contain the URI Hash field, and Lower Tester 2 also separately sends a second non-connectable advertisement with the URI AD Type together with both AD Types from the requested filter. The URI Hash from the beacon does not identify the second non-connectable advertisement (i.e., it is different than the correctly computed hash value).
3. When the timeout specified in Step 1 expires, the IUT sends a Remote Provisioning Extended Scan Report message with the requested UUID and with the OOBInformation field, but without the AdvStructures field.
4. Repeat Step 1.
5. Lower Tester 2 immediately starts sending Unprovisioned Device Beacons, and the beacons contain the URI Hash field. Lower Tester 2 also separately sends a second non-connectable advertisement

with the URI AD Type together with both AD Types from the requested filter. The URI Hash from the beacon identifies the second non-connectable advertisement. The BD_ADDR used when sending the Unprovisioned Device Beacons is different than the BD_ADDR used when sending the URI advertisement.

6. The IUT sends a Remote Provisioning Extended Scan Report message with the AdvStructures field containing the two requested AD structures sent by Lower Tester 2 in Step 2. The IUT sends this report at least 1 second before the timeout specified in Step 2 expires.
7. Repeat Step 1.
8. Lower Tester 2 starts sending Unprovisioned Device Beacons but with a different Device UUID.
9. When the timeout specified in Step 1 expires, the IUT sends a Remote Provisioning Extended Scan Report message with the requested UUID and without the OOBInformation and AdvStructures fields.

- Expected Outcome

  Pass verdict

  The IUT ignores the URI advertisements when the advertising data is not identified by the URI Hash value from the beacon.

  The IUT reports the AD structures from the URI advertisement when it is sent with a different BD_ADDR.

MESH/SR/RPR/SCN/BI-01-C [Ignore Invalid Messages]

- Test Purpose

  Verify that the IUT ignores Remote Provisioning Scan Start and Remote Provisioning Extended Scan Start messages with invalid parameters.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The Lower Tester has access to the device key of the IUT.

- Test Procedure

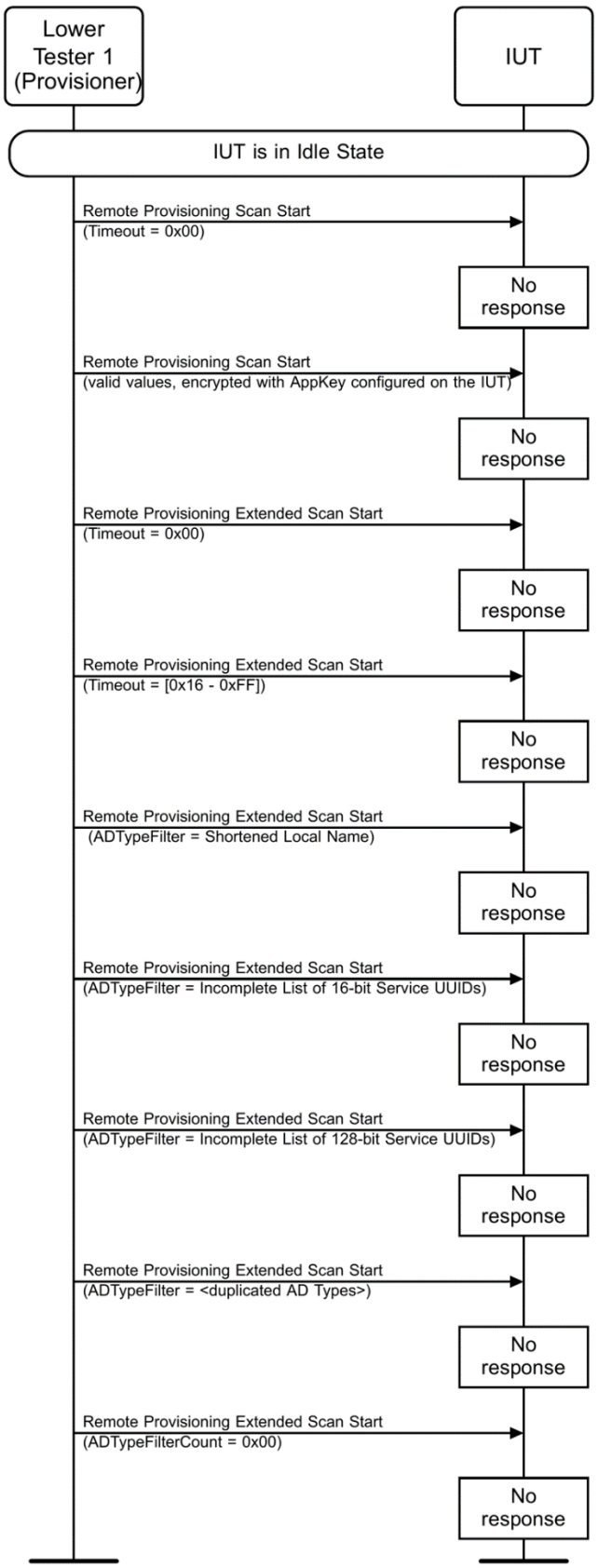


Figure 4.318: Test procedure for MESH/SR/RPR/SCN/BI-01-C MSC

1. The Lower Tester sends a Remote Provisioning Scan Start message with the Timeout field set to 0 and the IUT does not respond.
2. The Lower Tester sends a Remote Provisioning Scan Start message with valid parameters, but encrypted with an application key that has been configured on the IUT, and the IUT does not respond.
3. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the Timeout field set to 0 and the IUT does not respond.
4. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the Timeout field set to a random value in the range 0x16-0xFF and the IUT does not respond.
5. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the ADTypeFilter field containing only the Shortened Local Name AD Type and the IUT does not respond.
6. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the ADTypeFilter field containing only the Incomplete List of 16-bit Service UUIDs AD Type and the IUT does not respond.
7. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the ADTypeFilter field containing only the Incomplete List of 128-bit Service UUIDs AD Type and the IUT does not respond.
8. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the ADTypeFilter field containing duplicated valid AD Type values and the IUT does not respond.
9. The Lower Tester sends a Remote Provisioning Extended Scan Start message with the ADTypeFilterCount field set to 0 and the IUT does not respond.

- Expected Outcome

  Pass verdict

  The IUT ignores all invalid messages.

MESH/SR/RPR/SCN/BI-02-C [Reject Scan Requests with Invalid Parameters]

- Test Purpose

  Verify that the IUT accepts scan request messages only when all the appropriate conditions are met.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The Lower Tester has access to the device key of the IUT.

- Test Procedure

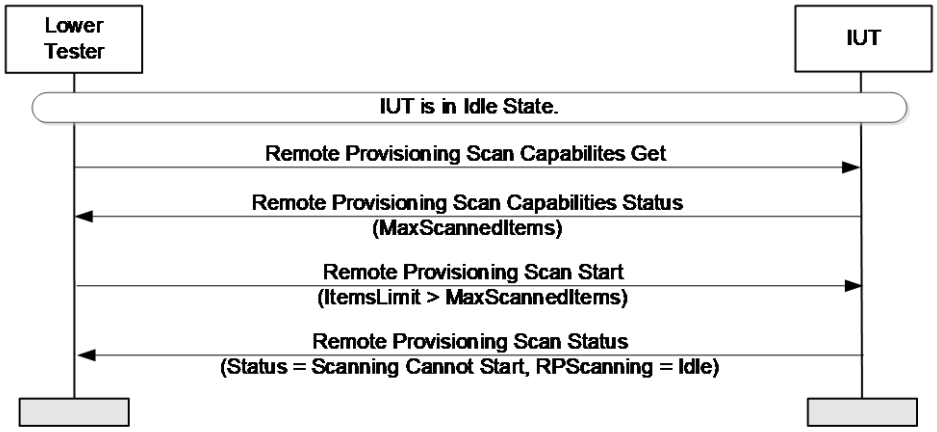


Figure 4.319: Test procedure for MESH/SR/RPR/SCN/BI-02-C MSC

1. The Lower Tester sends a Remote Provisioning Scan Capabilities Get message and the IUT responds with a Remote Provisioning Scan Capabilities Status message with the MaxScannedItems field value in the range 0x04-0xFF. The Lower Tester saves the value of the MaxScannedItems field for the next steps.
2. If the IUT's scanned items limit is equal to 0xFF, then skip this step. Otherwise, the Lower Tester sends a Remote Provisioning Scan Start message, setting the ItemsLimit field to a value greater than the one supported by the IUT. All other fields are set to valid values. The IUT responds with a Remote Provisioning Scan Status message with the Status field set to Scanning Cannot Start and the RPScanningState field set to Idle.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the correct status codes and state values, rejecting the request to start scanning when parameters do not meet the acceptance conditions.

MESH/CL/RPR/SCN/BV-01-C [Send Scan Commands and Process Scan Reports]

- Test Purpose

  Verify that the IUT can send scan commands and process scan reports.

- Reference

  [10] 4.3.4, 4.4.6

- Initial Condition
  - The Lower Tester acts as Remote Provisioning Server and shares previously established network security credentials with the IUT.
  - The IUT has access to the device key of the Lower Tester.

- Test Procedure

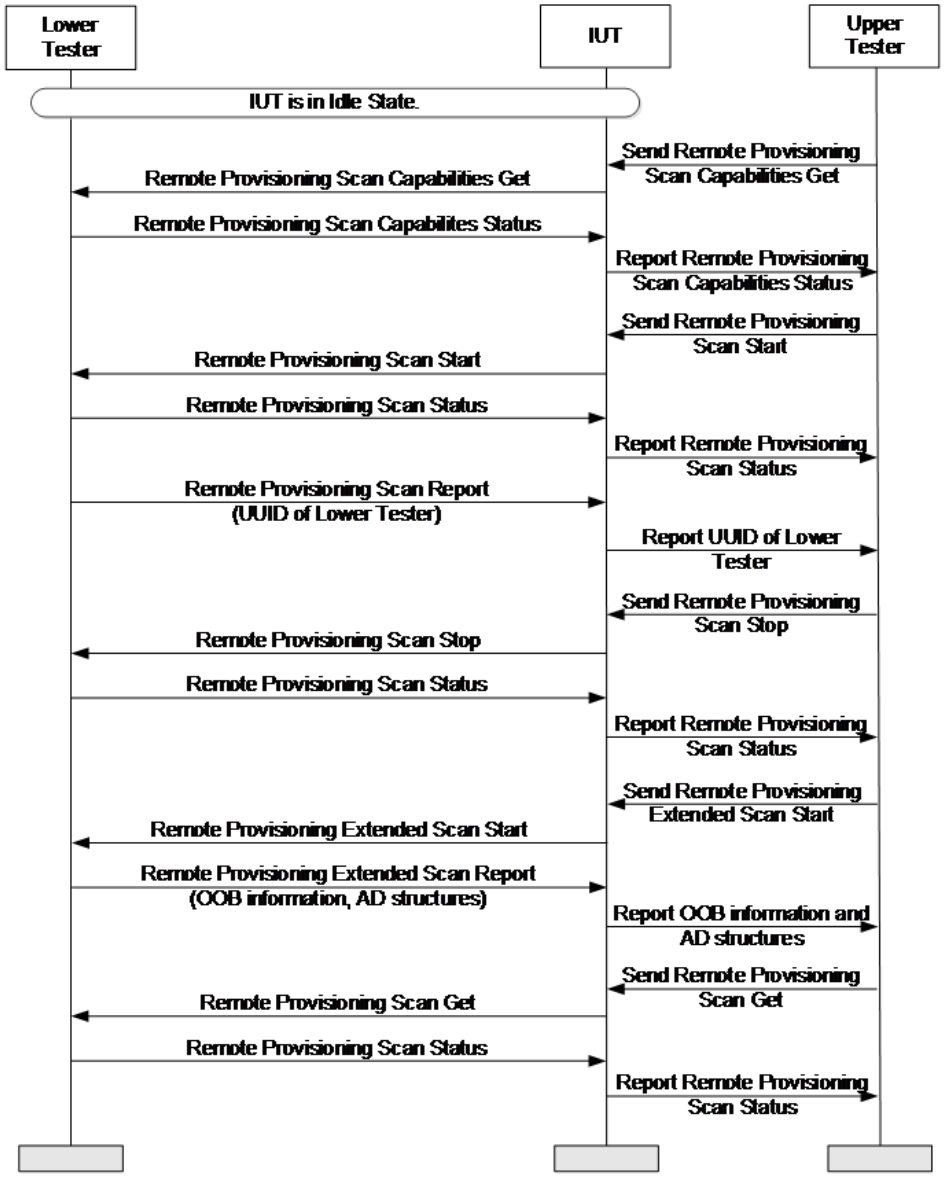


Figure 4.320: Test procedure for MESH/CL/RPR/SCN/BV-01-C MSC

1. The Upper Tester orders and the IUT sends a Remote Provisioning Scan Capabilities Get message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.
2. The Upper Tester orders and the IUT sends a Remote Provisioning Scan Start message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.
3. The Lower Tester sends a Remote Provisioning Scan Report message to the IUT and the IUT reports the Device UUID from the report.
4. The Upper Tester orders and the IUT sends a Remote Provisioning Scan Stop message. The Lower

Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.

5. The Upper Tester orders and the IUT sends a Remote Provisioning Extended Scan Start message. The Lower Tester responds with a Remote Provisioning Extended Scan Report message to the IUT and the IUT reports the OOB information and AD structures from the report.
6. The Upper Tester orders and the IUT sends a Remote Provisioning Scan Get message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the requested messages with correct format and parameter values and reports procedure completion and scanned devices to the Upper Tester.

4.22.2 Link Management

4.22.2.1 Open and Close the Link

- Test Purpose

  Verify that the IUT can process link commands to open and close the link and can report link status.

- Reference

  [10] 4.3.4, 4.4.5, 5.2.3

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

- Test Procedure

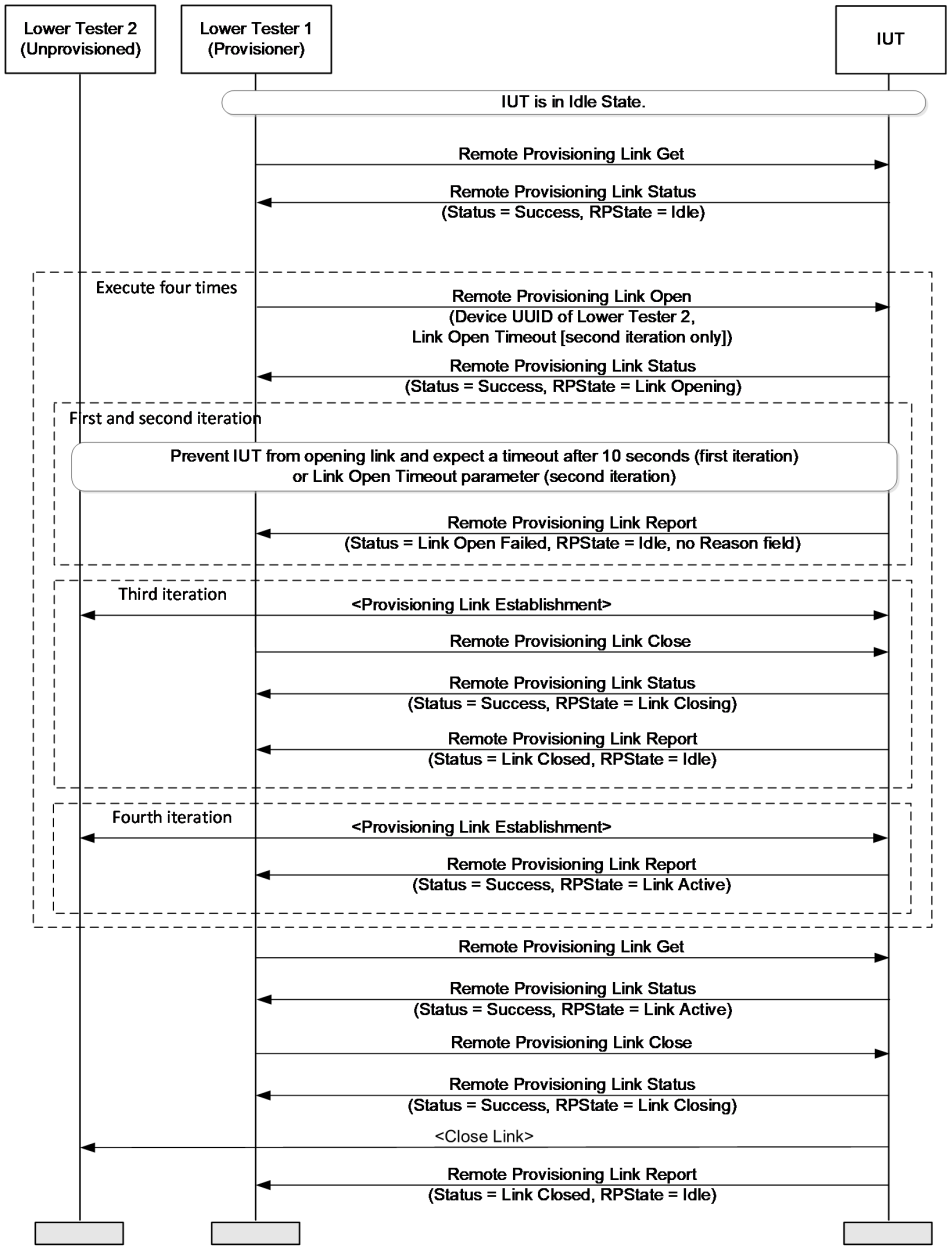


Figure 4.321: Test procedure - Open and Close the Link MSC

1. Lower Tester 1 sends a Remote Provisioning Link Get message to the IUT, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Idle.
2. Execute the instructions listed in the Step 2 column in Table 4.149.
3. Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the Device UUID used by Lower Tester 2 and the Link Open Timeout parameter omitted. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Opening.
4. Execute the instructions listed in the Steps 4 and 7 column in Table 4.149.

5. The IUT sends a Remote Provisioning Link Report message, after 10 seconds from responding in Step 3, with the Status field set to Link Open Failed, the RPState field set to Idle, and without the Reason field.
6. Repeat Steps 3–5, but in Step 3 use a valid value for the Link Open Timeout parameter (different than 10), and in Step 5 expect the IUT to send the report after the corresponding interval in seconds.
7. Repeat Step 3, then execute the instructions listed in the Steps 4 and 7 column in Table 4.149.
8. After 5 seconds have passed since the IUT responded in Step 6 (in the repetition of Step 3), Lower Tester 1 sends a Remote Provisioning Link Close message and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report message with the Status field set to Link Closed by Client and the RPState field set to Idle.
9. Repeat Step 3, then execute the instructions listed in the Step 9 column in Table 4.149.
10. Lower Tester 2 accepts the IUT's attempts to open a provisioning link, and the IUT sends a Remote Provisioning Link Report message with the Status field set to Success and the RPState field set to Link Active.
11. Lower Tester 1 sends a Remote Provisioning Link Get message and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active.
12. Lower Tester 1 sends a Remote Provisioning Link Close message and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing.
13. The IUT closes the provisioning link, then the IUT sends a Remote Provisioning Link Report message with the Status field set to Link Closed by Client, the RPState field set to Idle, and without the Reason field.

- Test Case Configuration

TCID	Step 2	Steps 4 and 7	Step 9
MESH/SR/RPR/LNK/BV-01-C [Open and Close the Link – PB-ADV]	Lower Tester 2 begins sending Unprovisioned Device Beacons.	Lower Tester 2 ignores the IUT's attempts to open a provisioning link.	No action.
MESH/SR/RPR/LNK/BV-02-C [Open and Close the Link – PB-GATT]	No action.	No action.	Lower Tester 2 begins sending connectable advertisements containing the Mesh Provisioning Service UUID.

Table 4.149: Additional test procedure steps for Open and Close the Link

- Note

  When executing Steps 7–8 and 10-11, it is possible that only the Remote Provisioning Link Report message is received. In that case, the Lower Tester will assume that the Remote Provisioning Link Status message was sent by the IUT but lost due to imperfect radio reception and will carry on with the procedure.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the appropriate state values and status codes and sends a link report when the link opens.
  The Remote Provisioning Link Report message is sent as a segmented access message.

4.22.2.2 Respond to Redundant Link Requests with Unprovisioned Device

- Test Purpose

  Verify that the IUT responds correctly to redundant link requests when provisioning an unprovisioned device.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT. The IUT is configured with two network keys.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.
  - If the IUT supports more than one network key, then a second network key has been configured on the IUT.

- Test Procedure

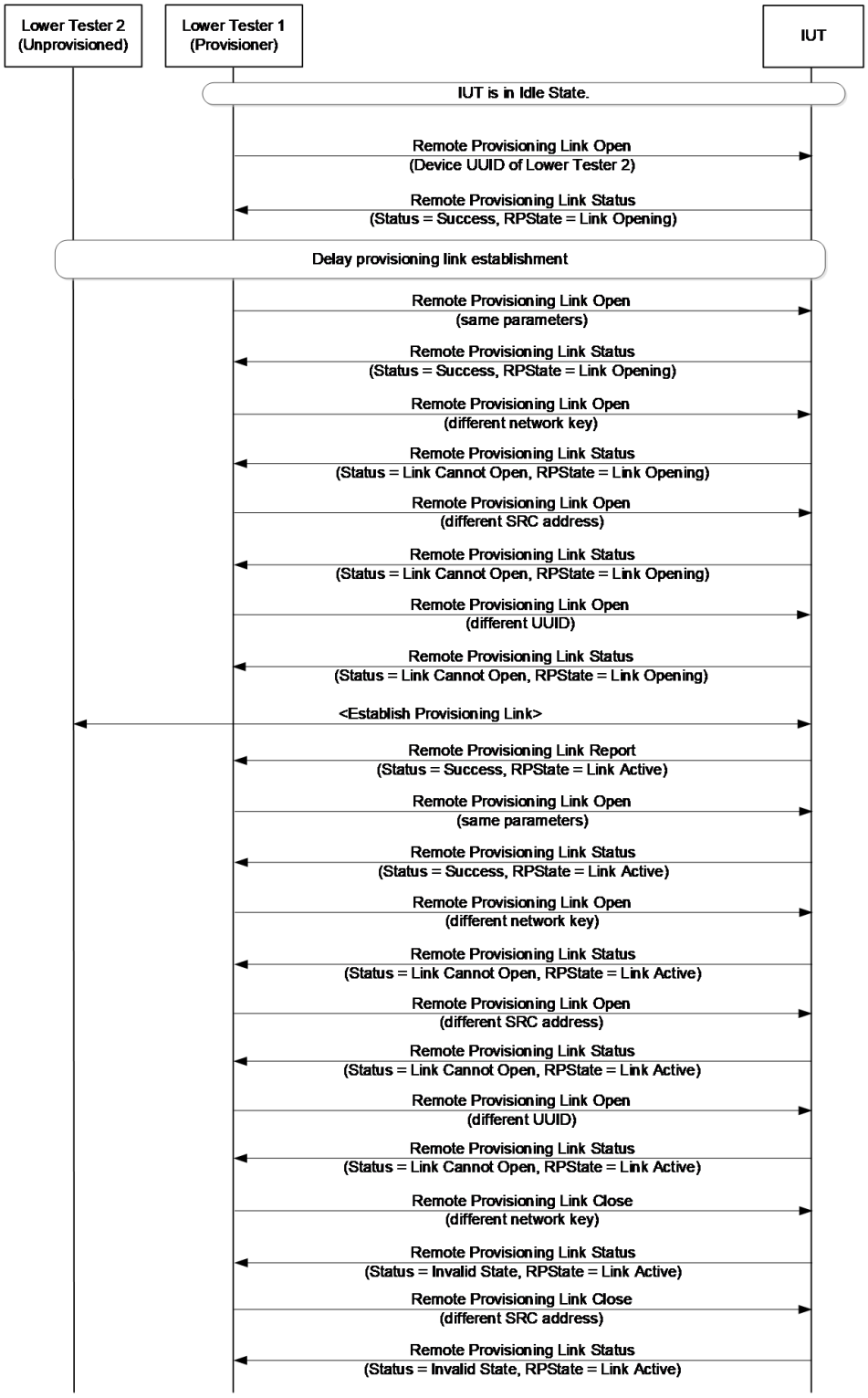


Figure 4.322: Test procedure - Respond to Redundant Link Requests with Unprovisioned Device MSC - Page 1 of 2

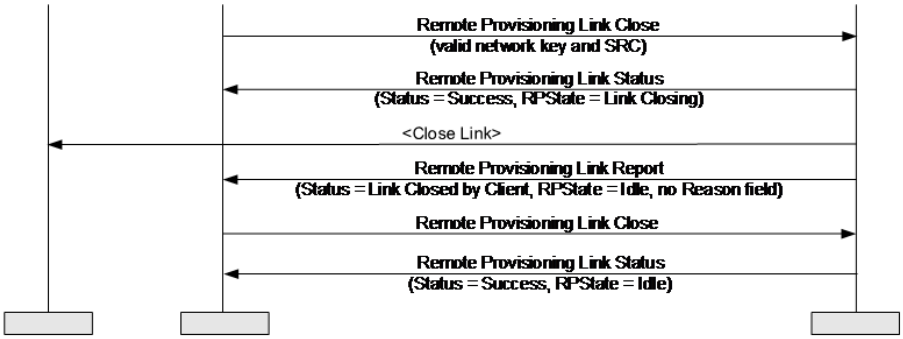


Figure 4.323: Test procedure - Respond to Redundant Link Requests with Unprovisioned Device MSC - Page 2 of 2

1. Execute the instructions listed in the Step 1 column in Table 4.150.
2. Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the Device UUID used by Lower Tester 2. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Opening.
3. Execute the instructions listed in the Step 3 column in Table 4.150.
4. Lower Tester 1 resends the message from Step 2 and the IUT sends the same response.
5. If the IUT supports more than one network key, Lower Tester 1 resends the message from Step 2, but encrypted with a different network key, and the IUT responds on the second network key with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Opening.
6. Lower Tester 1 resends the message from Step 2, but with a different source unicast address, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Opening.
7. Lower Tester 1 resends the message from Step 2, but with a different value in the UUID field, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Opening.
8. Execute the instructions listed in the Step 8 column in Table 4.150.
9. Lower Tester 2 accepts the IUT's attempts to open a provisioning link, and IUT sends a Remote Provisioning Link Report message with the Status field set to Success and the RPState field set to Link Active.
10. Lower Tester 1 resends the message from Step 2 and the IUT sends the same response but with the RPState field set to Link Active.
11. If the IUT supports more than one network key, Lower Tester 1 resends the message from Step 2, but encrypted with a different network key, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.
12. Lower Tester 1 resends the message from Step 2, but with a different source unicast address, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.
13. Lower Tester 1 resends the message from Step 2, but with a different value in the UUID field, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.

14. If the IUT supports more than one network key, Lower Tester 1 sends a Remote Provisioning Link Close message encrypted with a different network key, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Invalid State and the RPState field set to Link Active.
15. Lower Tester 1 sends a Remote Provisioning Link Close message with a different source unicast address, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Invalid State and the RPState field set to Link Active.
16. Lower Tester 1 sends a Remote Provisioning Link Close message (with the correct network key and source unicast address) and the IUT closes the provisioning link established with Lower Tester 2 (using the same Reason value as in the incoming message) and to respond with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report messages with the Status field set to Link Closed by Client, the RPState field set to Idle, and without the Reason field.
17. Lower Tester 1 resends the Remote Provisioning Link Close message from the previous step, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Idle.

- Test Case Configuration

TCID	Step 1	Step 3	Step 8
MESH/SR/RPR/LNK/BV-03-C [Respond to Redundant Link Requests with Unprovisioned Device – PB-ADV]	Lower Tester 2 begins sending Unprovisioned Device Beacons.	Lower Tester 2 ignores the IUT's attempts to open a provisioning link.	No action.
MESH/SR/RPR/LNK/BV-04-C [Respond to Redundant Link Requests with Unprovisioned Device – PB-GATT]	No action.	No action.	Lower Tester 2 begins sending connectable advertisements containing the Mesh Provisioning Service UUID.

Table 4.150: Additional test procedure steps for Respond to Redundant Link Requests with Unprovisioned Device

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the appropriate state values and status codes.

4.22.2.3 Handle Requests and Events in the Outbound Packet Transfer State

- Test Purpose

  Verify that the IUT correctly handles link requests and various events when in the Outbound Packet Transfer state.

- Reference

[10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.
- Test Procedure

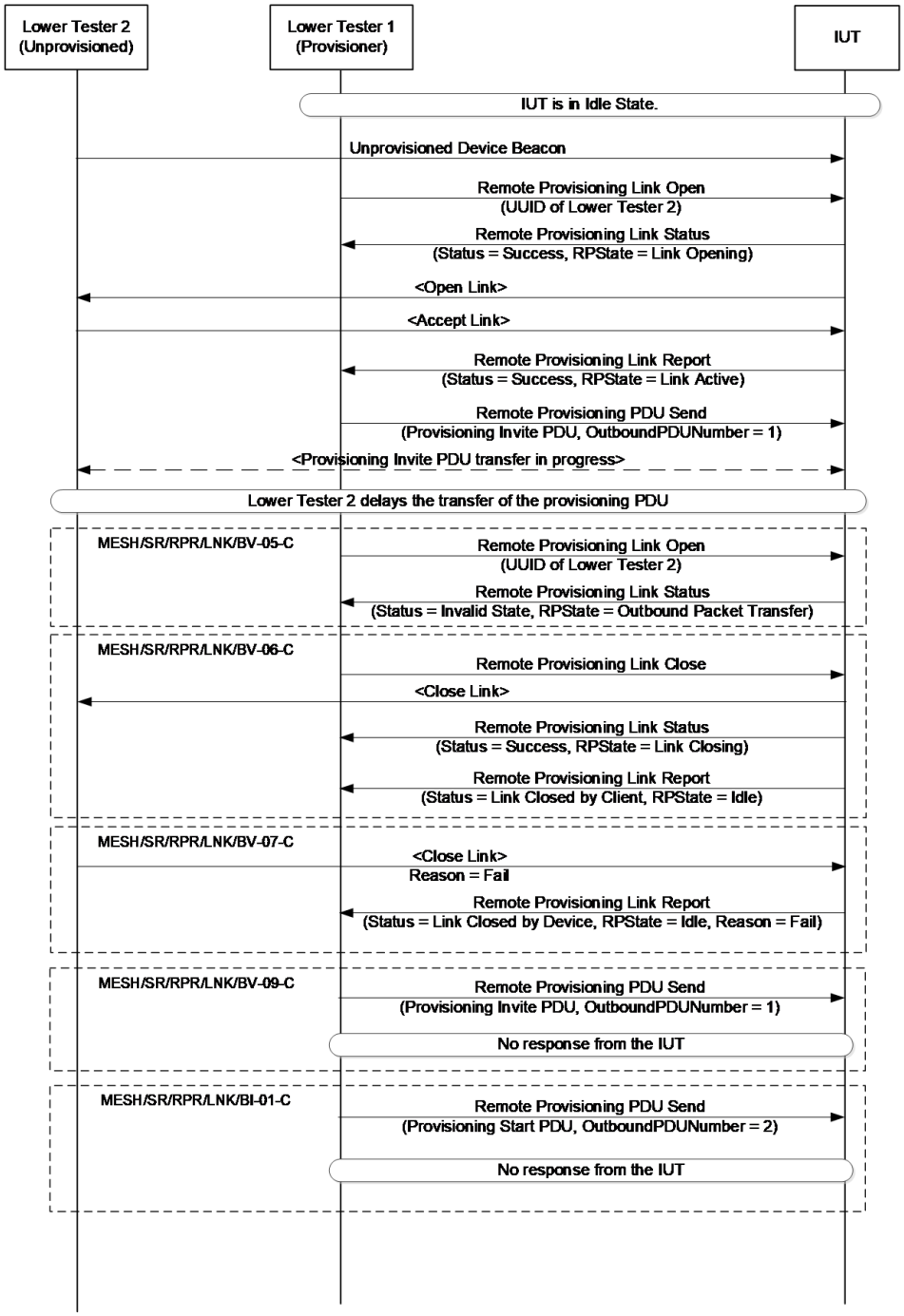


Figure 4.324: Test procedure - Handle Requests and Events in the Outbound Packet Transfer State MSC - Page 1 of 2

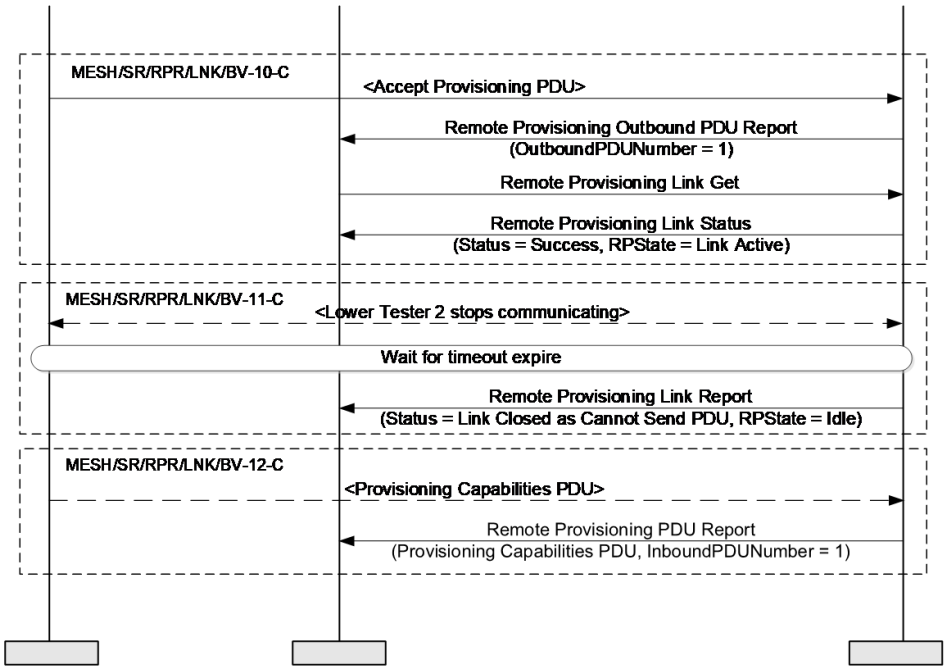


Figure 4.325: Test procedure - Handle Requests and Events in the Outbound Packet Transfer State MSC - Page 2 of 2

1. Bring the IUT in the Outbound Packet Transfer state by executing the following sequence:
    a. Bring the IUT into the Initial Condition.
    b. Lower Tester 2 begins sending Unprovisioned Device Beacons.
    c. Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the Device UUID used by Lower Tester 2. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Opening.
    d. Lower Tester 2 accepts the IUT's attempts to open a provisioning link, and IUT sends a Remote Provisioning Link Report message with the Status field set to Success and the RPState field set to Link Active.
    e. Lower Tester 1 sends a Remote Provisioning PDU Send message containing a Provisioning Invite PDU and with the OutboundPDUNumber field set to 1, and the IUT begins sending the PDU over the provisioning link. Lower Tester 2 delays the transfer of the provisioning PDU (by not sending Link Ack messages) to allow the next step to happen while the IUT is still in the Outbound Packet Transfer state.
2. Execute the instructions listed in the Step 2 column in Table 4.151.

- Test Case Configuration

TCID	Step 2
MESH/SR/RPR/LNK/BV-05-C [Outbound Packet Transfer State – Handle Link Open] Additional Reference: [10] 4.4.5.5.3.2	Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the Device UUID used by Lower Tester 2, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Invalid State and the RPState field set to Outbound Packet Transfer.
MESH/SR/RPR/LNK/BV-06-C [Outbound Packet Transfer State – Handle Link Close] Additional Reference: [10] 4.4.5.5.3.3	Lower Tester 1 sends a Remote Provisioning Link Close message and the IUT aborts the transfer of the provisioning PDU over the provisioning link, closes the provisioning link with the same Reason field value as in the incoming message, and responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report message with the Status field set to Link Closed by Client and the RPState field set to Idle.
MESH/SR/RPR/LNK/BV-07-C [Outbound Packet Transfer State – Handle Link Closed by Device] Additional Reference: [10] 4.4.5.5.3.5	Lower Tester 2 closes the provisioning link with the Reason field set to Fail, and IUT sends a Remote Provisioning Link Report message with the Status field set to Link Closed by Device, the RPState field set to Idle and the Reason field set to Fail.
MESH/SR/RPR/LNK/BV-09-C [Outbound Packet Transfer State – Handle Resending the Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	Lower Tester 1 resends the Remote Provisioning PDU Send message from Step 1e and the IUT does not send any response.
MESH/SR/RPR/LNK/BV-10-C [Outbound Packet Transfer State – Handle PDU Transfer Complete] Additional Reference: [10] 4.4.5.5.4.2	Lower Tester 2 accepts the provisioning PDU over the provisioning link, and IUT sends a Remote Provisioning Outbound PDU Report message with the OutboundPDUNumber field set to 1. Lower Tester 1 then sends a Remote Provisioning Link Get message and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active.
MESH/SR/RPR/LNK/BV-11-C [Outbound Packet Transfer State – Handle Provisioning Link Timeout] Additional Reference: [10] 4.4.5.5.4.2	Lower Tester 2 stops communicating over the provisioning link, and after the provisioning link timeout expires IUT sends a Remote Provisioning Link Report message with the Status field set to Link Closed as Cannot Send PDU and the RPState field set to Idle.
MESH/SR/RPR/LNK/BI-01-C [Outbound Packet Transfer State – Handle Unexpected Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	Lower Tester 1 sends a Remote Provisioning PDU Send message containing a Provisioning Start PDU and with the OutboundPDUNumber field set to 2, and the IUT does not send any response.

Table 4.151: Additional test procedure steps for Handle Requests and Events in the Outbound Packet Transfer State

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the appropriate state values and status codes or ignores the messages when appropriate.

4.22.2.4 Handle Requests and Events in the Link Active State with Unprovisioned Device

- Test Purpose

  Verify that the IUT correctly handles link requests and various events when in the Link Active state when provisioning an unprovisioned device.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT. The IUT is configured with two network keys.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

- Test Procedure

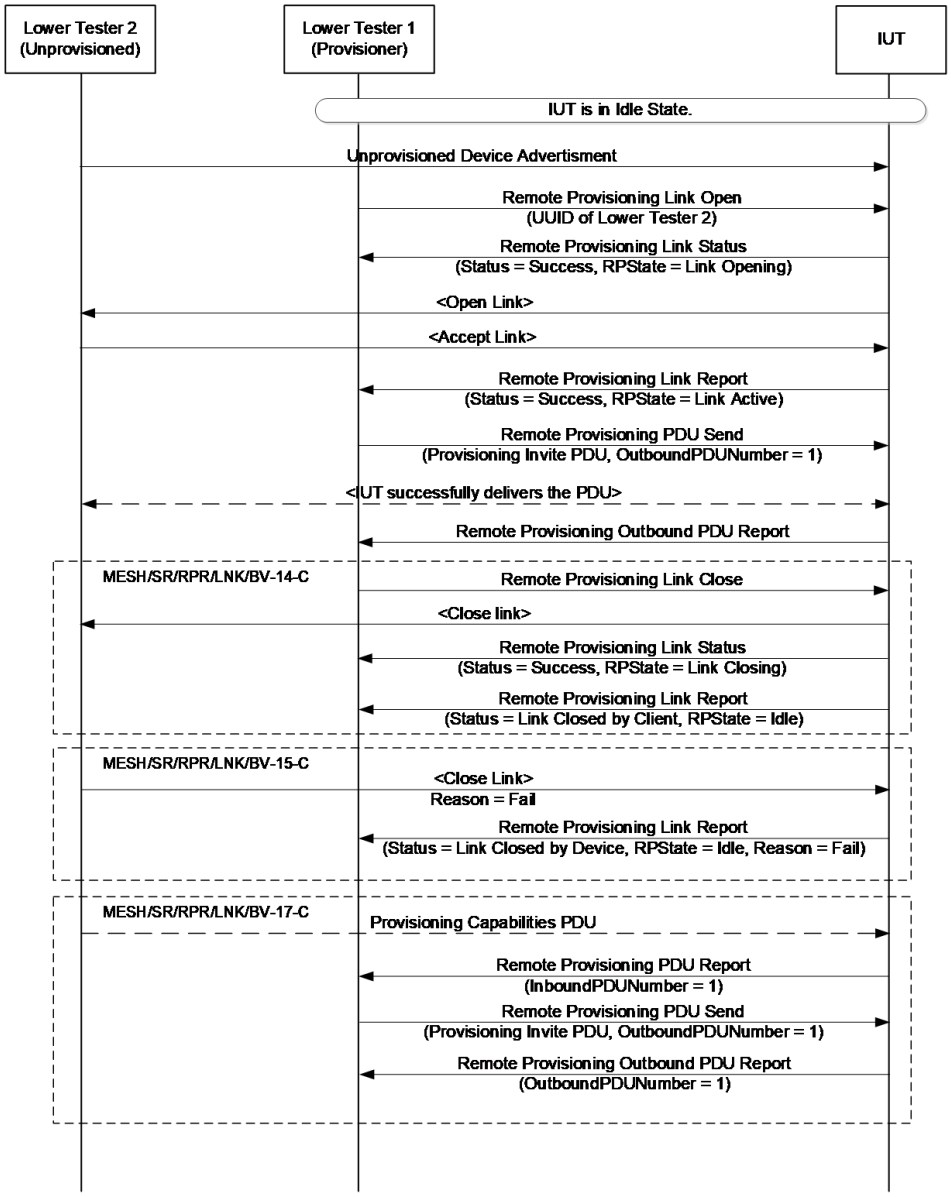


Figure 4.326: Test procedure - Handle Requests and Events in the Link Active State with Unprovisioned Device MSC - Page 1 of 2

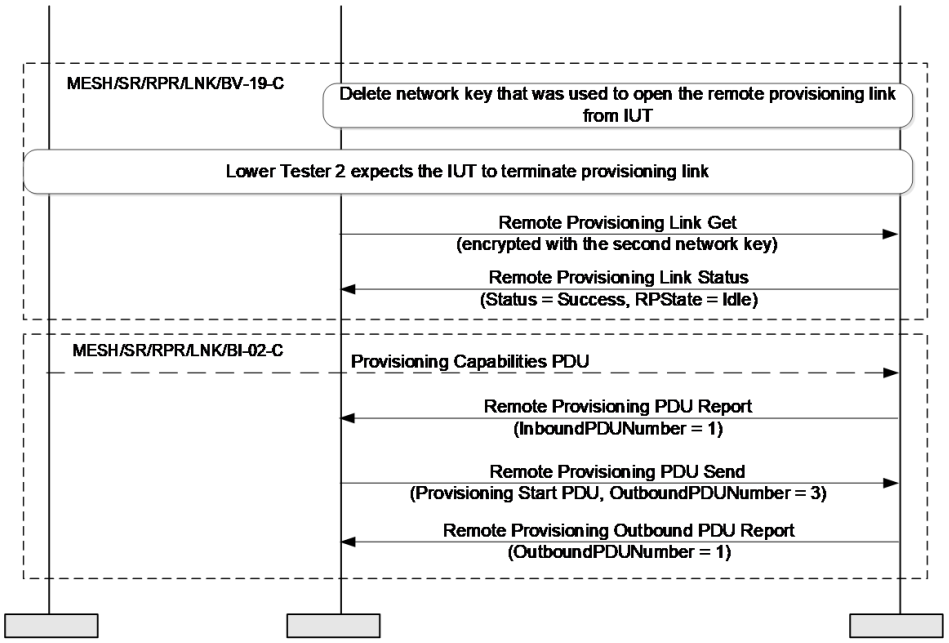


Figure 4.327: Test procedure - Handle Requests and Events in the Link Active State with Unprovisioned Device MSC - Page 2 of 2

1. Bring the IUT in the Link Active state by executing the following sequence:
    a. If the value in the Step 1a column in Table 4.152 is "PB-ADV", then Lower Tester 2 begins sending Unprovisioned Device Beacons; otherwise, if the value is "PB-GATT", then it begins sending connectable advertisements containing the Mesh Provisioning Service UUID.
    b. Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the Device UUID used by Lower Tester 2. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Opening.
    c. Lower Tester 2 accepts the IUT's attempts to open a provisioning link, and the IUT sends a Remote Provisioning Link Report message with the Status field set to Success and the RPState field set to Link Active.
    d. Lower Tester 1 sends a Remote Provisioning PDU Send message containing the Provisioning Invite PDU and with the OutboundPDUNumber field set to 1; the IUT successfully delivers the PDU over the provisioning link and sends a Remote Provisioning PDU Outbound Report to Lower Tester 1.
2. Execute the instructions listed in the Step 2 column in Table 4.152.

- Test Case Configuration

TCID	Step 1a	Step 2
MESH/SR/RPR/LNK/BV-13-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Link Close] Additional Reference: [10] 4.4.5.5.3.3	PB-ADV	Lower Tester 1 sends a Remote Provisioning Link Close message and the IUT closes the provisioning link with the same Reason field value as in the incoming message, and responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report message with the Status field set to Link Closed by Client and the RPState field set to Idle.
MESH/SR/RPR/LNK/BV-14-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Link Close] Additional Reference: [10] 4.4.5.5.3.3	PB-GATT	
MESH/SR/RPR/LNK/BV-15-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Link Closed by Device] Additional Reference: [10] 4.4.5.5.3.5	PB-ADV	Lower Tester 2 closes the provisioning link with the Reason field set to Fail, and the IUT sends a Remote Provisioning Link Report message with the Status field set to Link Closed by Device, the RPState field set to Idle, and the Reason field set to Fail.
MESH/SR/RPR/LNK/BV-16-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Link Closed by Device] Additional Reference: [10] 4.4.5.5.3.5	PB-GATT	
MESH/SR/RPR/LNK/BV-17-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Resending the Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	PB-ADV	Lower Tester 2 delivers a Provisioning Capabilities PDU over the provisioning link, and the IUT successfully sends a Remote Provisioning PDU Report to the Lower Tester 1 containing the provisioning PDU and the InboundPDUNumber field set to 1. Lower Tester 1 resends the Remote Provisioning PDU Send message from Step 1d, and the IUT sends a Remote Provisioning Outbound PDU Report message with the OutboundPDUNumber field set to 1.
MESH/SR/RPR/LNK/BV-18-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Resending the Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	PB-GATT	

TCID	Step 1a	Step 2
MESH/SR/RPR/LNK/BV-19-C [Link Active State with Unprovisioned Device over PB-ADV – Handle Network Key Removal] Additional Reference: [10] 4.4.5.4 Additional Initial Condition :A second network key has been configured on the IUT.	PB-ADV	Delete the network key on the IUT that was used to open the remote provisioning link. The IUT terminates the provisioning link. Then Lower Tester 1 sends a Remote Provisioning Link Get message to the IUT using the second network key, and the IUT responds with a Remote Provisioning Link Status message with the RPState field set to Idle.
MESH/SR/RPR/LNK/BV-22-C [Link Active State with Unprovisioned Device over PB-GATT – Handle Network Key Removal] Additional Reference: [10] 4.4.5.4 Additional Initial Condition: A second network key has been configured on the IUT.	PB-GATT	
MESH/SR/RPR/LNK/BI-02-C [Link Active State with Unprovisioned Device – Handle Invalid Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	Any	Lower Tester 2 delivers a Provisioning Capabilities PDU over the provisioning link, and the IUT successfully sends a Remote Provisioning PDU Report to Lower Tester 1 containing the provisioningPDU and the InboundPDUNumber field set to 1. Lower Tester 1 sends a Remote Provisioning PDU Send message containing a Provisioning Start PDU but with the OutboundPDUNumber field set to 3, and the IUT sends a Remote Provisioning Outbound PDU Report message with the OutboundPDUNumber field set to 1.

Table 4.152: Additional test procedure steps for Handle Requests and Events in the Link Active State with Unprovisioned Device

- Expected Outcome

  Pass verdict

  The IUT sends all messages with the appropriate state values and status codes.

MESH/SR/RPR/LNK/BV-20-C [Respond to Redundant Link Requests with Device Key Interface Procedures]

- Test Purpose

  Verify that the IUT responds correctly to redundant link requests when executing a Device Key Interface procedure.

- Reference

[10] 3.10.8, 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT. The IUT is configured with two network keys.
  - The Lower Tester has access to the device key of the IUT.
  - If the IUT supports more than one network key, then a second network key has been configured on the IUT.
- Test Procedure

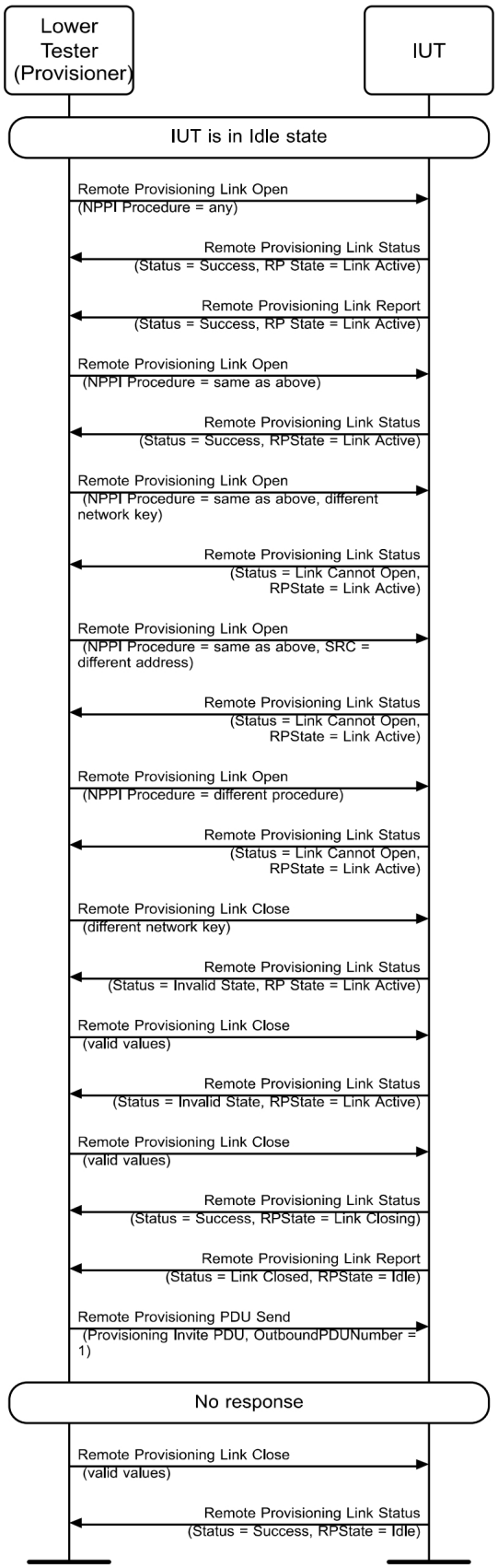


Figure 4.328: Test procedure for MESH/SR/RPR/LNK/BV-20-C MSC

1. The Lower Tester sends a Remote Provisioning Link Open message with the NPPI Procedure field set to any value supported by the IUT. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active, followed by a Remote Provisioning Link Report message with the Status field set to Success, the RPState field set to Link Active, and without the Reason field.
2. The Lower Tester resends the message from Step 1 and the IUT sends the same response.
3. If the IUT supports more than one network key, the Lower Tester resends the message from Step 2, but encrypted with a different network key, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.
4. The Lower Tester resends the message from Step 2, but with a different source unicast address, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.
5. The Lower Tester resends the message from Step 2, but with a different value in the NPPI Procedure field, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.\
6. If the IUT supports more than one network key, the Lower Tester sends a Remote Provisioning Link Close message encrypted with a different network key, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Invalid State and the RPState field set to Link Active.
7. The Lower Tester sends a Remote Provisioning Link Close message with a different source unicast address, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Invalid State and the RPState field set to Link Active.
8. The Lower Tester sends a Remote Provisioning Link Close message (with the correct network key and source unicast address), and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report messages with the Status field set to Link Closed by Client, the RPState field set to Idle, and without the Reason field.
9. The Lower Tester sends a Remote Provisioning PDU Send message containing a Provisioning Invite PDU and with the OutboundPDUNumber field set to 1, and the IUT does not respond. If the IUT sends a Remote Provisioning PDU Outbound Report message, the test ends with a Fail verdict.
10. The Lower Tester resends the Remote Provisioning Link Close message from Step 8 and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Idle.

- Expected Outcome

  Pass verdict

  The IUT responds to all messages with the appropriate state values and status codes.

4.22.2.5 Requests and Events in the Link Active State with Device Key Interface Procedures

- Test Purpose

  Verify that the IUT correctly handles link requests and various events when in the Link Active state when executing a Device Key Interface procedure.

- Reference

  [10] 3.10.8, 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT. The IUT is configured with two network keys.
  - The Lower Tester has access to the device key of the IUT.
- Test Procedure

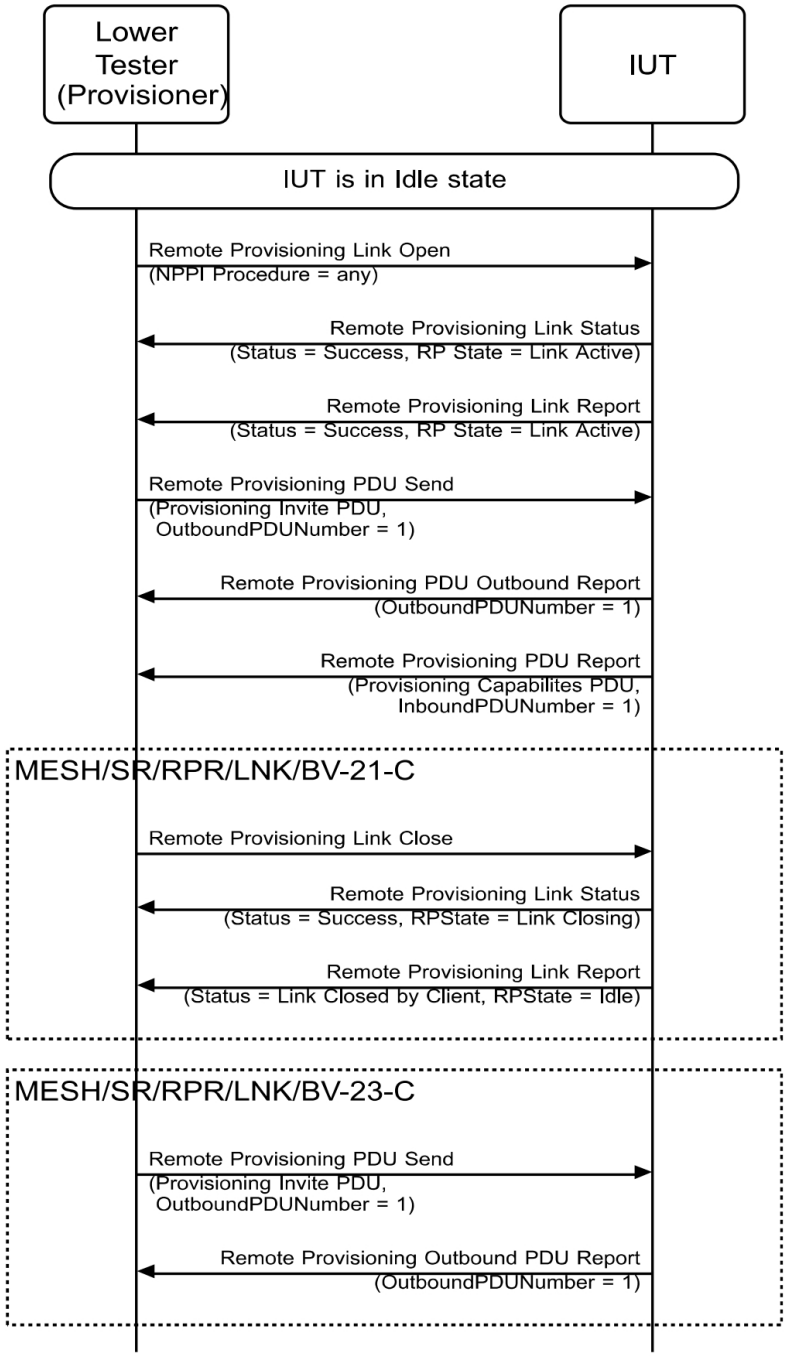


Figure 4.329: Test procedure - Handle Requests and Events in the Link Active State with Device Key Interface Procedures MSC - Page 1 of 2

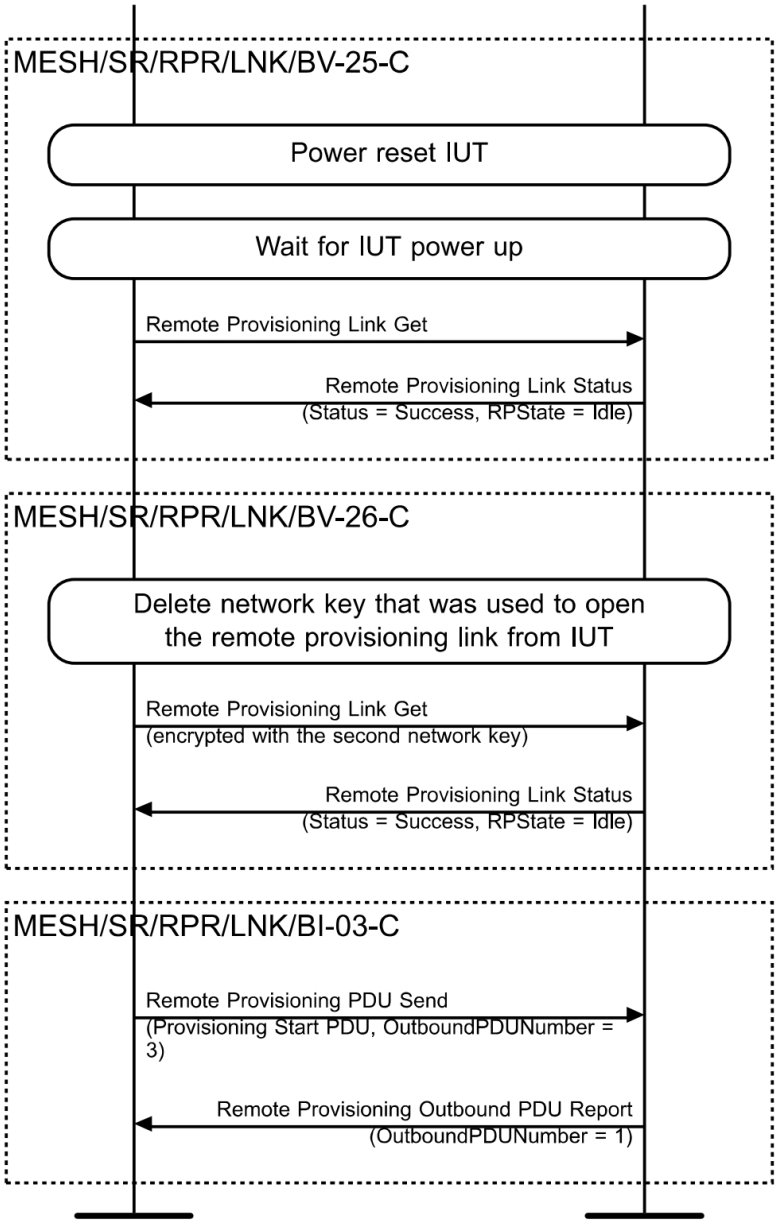


Figure 4.330: Test procedure - Handle Requests and Events in the Link Active State with Device Key Interface Procedures MSC - Page 2 of 2

1. Bring the IUT in the Link Active state by executing the following sequence:
    a. The Lower Tester sends a Remote Provisioning Link Open message with the NPPI Procedure field set to a value supported by the IUT. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active, followed by a Remote Provisioning Link Report message with the Status field set to Success, the RPState field set to Link Active, and without the Reason field.
    b. The Lower Tester sends a Remote Provisioning PDU Send message containing the Provisioning Invite PDU and with the OutboundPDUNumber field set to 1; the IUT responds with a Remote Provisioning PDU Outbound Report message with the OutboundPDUNumber field set to 1.

c. The IUT sends a Remote Provisioning PDU Report message containing a Provisioning Capabilities PDU and with the InboundPDUNumber field set to 1.

2. Execute the instructions listed in the Step 2 column in Table 4.153.

- Test Case Configuration

TCID	Step 2
MESH/SR/RPR/LNK/BV-21-C [Link Active State with Device Key Interface Procedures – Handle Link Close] Additional Reference: [10] 4.4.5.5.3.3	The Lower Tester sends a Remote Provisioning Link Close message, and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Closing, followed by a Remote Provisioning Link Report message with the Status field set to Link Closed by Client and the RPState field set to Idle.
MESH/SR/RPR/LNK/BV-23-C [Link Active State with Device Key Interface Procedures – Handle Resending the Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	Lower Tester 1 resends the Remote Provisioning PDU Send message from Step 1b and the IUT sends a Remote Provisioning Outbound PDU Report message with the OutboundPDUNumber field set to 1.
MESH/SR/RPR/LNK/BV-25-C [Link Active State with Device Key Interface Procedures – Handle Power Reset] Additional Reference: [10] 3.10.8.3	Trigger a power reset on the IUT. After the IUT has completed its power-up behavior, Lower Tester 1 sends a Remote Provisioning Link Get message and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Idle.
MESH/SR/RPR/LNK/BV-26-C [Link Active State with Device Key Interface Procedures – Handle Network Key Removal] Additional Reference: [10] 4.4.5.4 Additional Initial Condition: A second network key has been configured on the IUT.	Delete the network key on the IUT that was used to open the remote provisioning link. Then the Lower Tester sends a Remote Provisioning Link Get message to the IUT using the second network key, and the IUT responds with a Remote Provisioning Link Status message with the RPState field set to Idle.
MESH/SR/RPR/LNK/BI-03-C [Link Active State with Device Key Interface Procedures – Handle Invalid Outbound PDU] Additional Reference: [10] 4.4.5.5.4.1	Lower Tester 1 sends a Remote Provisioning PDU Send message containing a Provisioning Start PDU but with the OutboundPDUNumber field set to 3, and the IUT sends a Remote Provisioning Outbound PDU Report message with the OutboundPDUNumber field set to 1.

Table 4.153: Additional test procedure steps for Requests and Events in the Link Active State with Device Key Interface Procedures

- Expected Outcome

Pass verdict

The IUT sends all messages with the appropriate state values and status codes.

MESH/SR/RPR/LNK/BI-04-C [Ignore Invalid Link Requests]

- Test Purpose

  Verify that the IUT ignores invalid link requests.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT. The IUT is configured with two network keys.
  - The Lower Tester has access to the device key of the IUT.

- Test Procedure

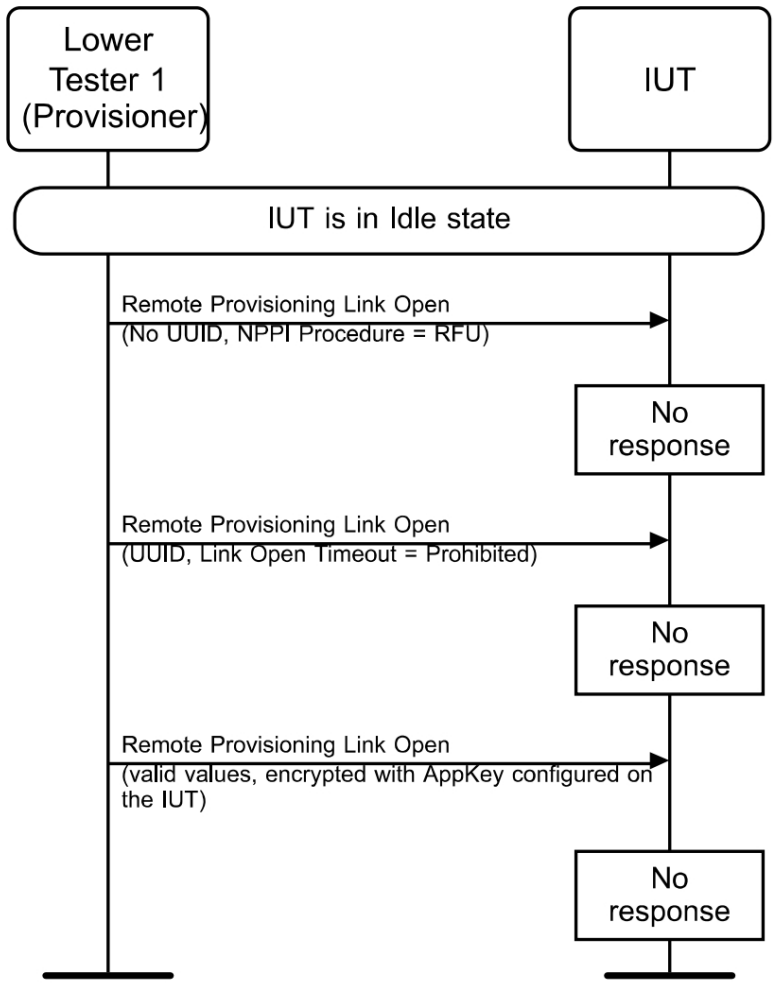


Figure 4.331: Test procedure for MESH/SR/RPR/LNK/BI-04-C MSC

1. The Lower Tester sends a Remote Provisioning Link Open message with the UUID field omitted and the NPPI Procedure field set to an RFU value, and the IUT does not respond.
2. The Lower Tester sends a Remote Provisioning Link Open message with the UUID field set to a random value and the Link Open Timeout field set to a Prohibited value, and the IUT does not respond.
3. The Lower Tester sends a Remote Provisioning Link Open message with a valid parameter but encrypted with an application key that has been configured on the IUT, and the IUT does not respond.

- Expected Outcome

  Pass verdict

  The IUT ignores the invalid messages.

MESH/SR/RPR/LNK/BI-05-C [Reject Parallel Provisioning and Node Provisioning Protocol Interface Procedure]

- Test Purpose

  Verify that the IUT does not allow a Node Provisioning Protocol Interface procedure while provisioning an unprovisioned device, or vice versa.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

- Test Procedure

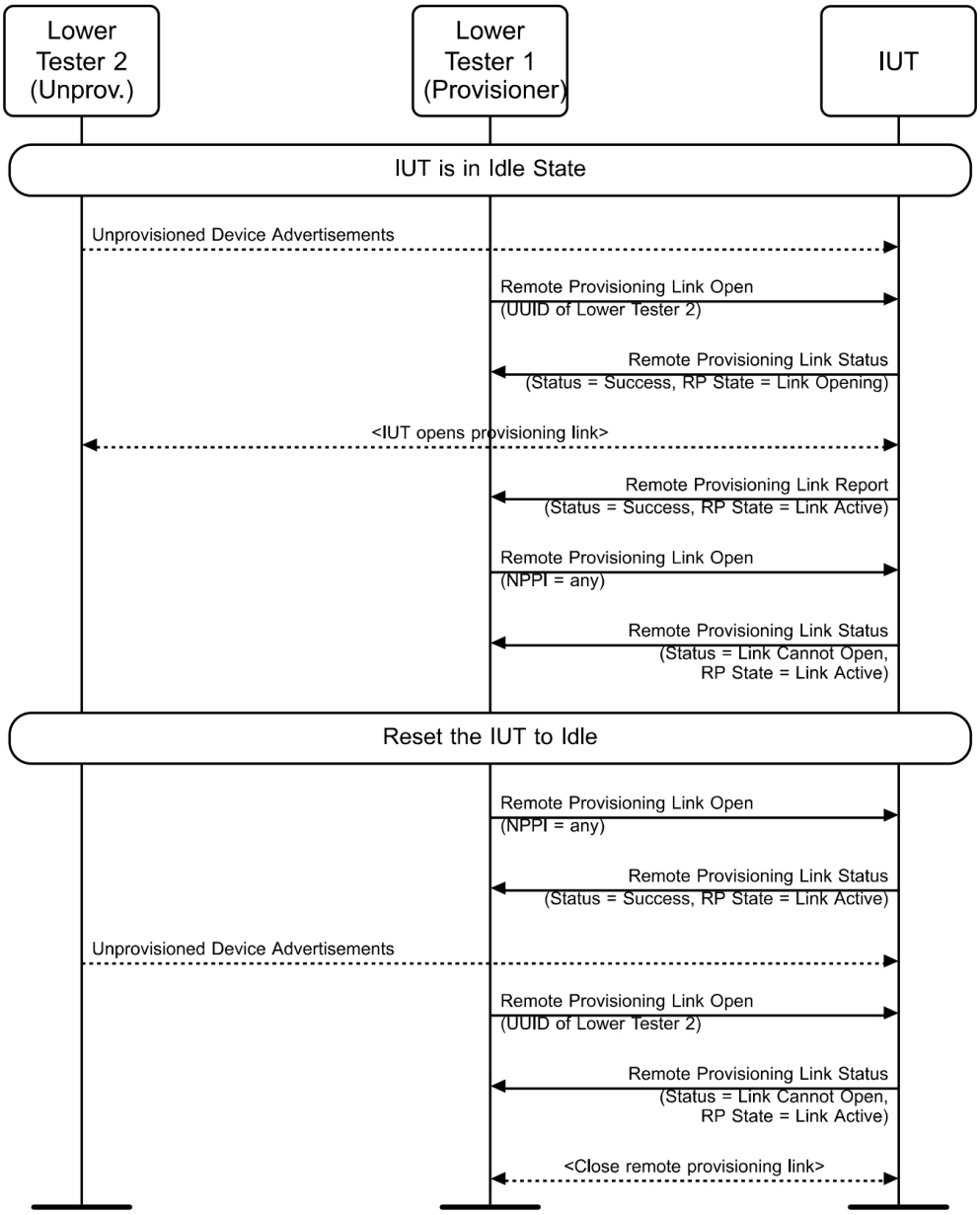


Figure 4.332: Test procedure for MESH/SR/RPR/LNK/BI-05-C MSC

1. Lower Tester 2 advertises as an unprovisioned device over a provisioning bearer supported by the IUT, and Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the UUID used by Lower Tester 2. The IUT opens a provisioning link and reports it to Lower Tester 1.
2. Lower Tester 1 sends a Remote Provisioning Link Open message with the NPPI Procedure field set to any value supported by the IUT. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.
3. Reset the IUT to the Idle state.
4. Lower Tester 1 sends a Remote Provisioning Link Open message with the NPPI Procedure field set to any value supported by the IUT. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active.

5. Lower Tester 2 advertises as an unprovisioned device over a provisioning bearer supported by the IUT, and Lower Tester 1 sends a Remote Provisioning Link Open message with the UUID field set to the UUID used by Lower Tester 2. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Link Active.

- Expected Outcome

  Pass verdict

  The IUT does not open the Node Provisioning Protocol Interface while provisioning another device and does not open a provisioning link with another device while a Node Provisioning Protocol Interface procedure is active.

MESH/CL/RPR/LNK/BV-01-C [Send Link Commands and Process Link Reports]

- Test Purpose

  Verify that the IUT can send link commands and can process link reports.

- Reference

  [10] 4.3.4, 4.4.6

- Initial Condition

  - The Lower Tester acts as Remote Provisioning Server and shares previously established network security credentials with the IUT.
  - The IUT has access to the device key of the Lower Tester.

- Test Procedure

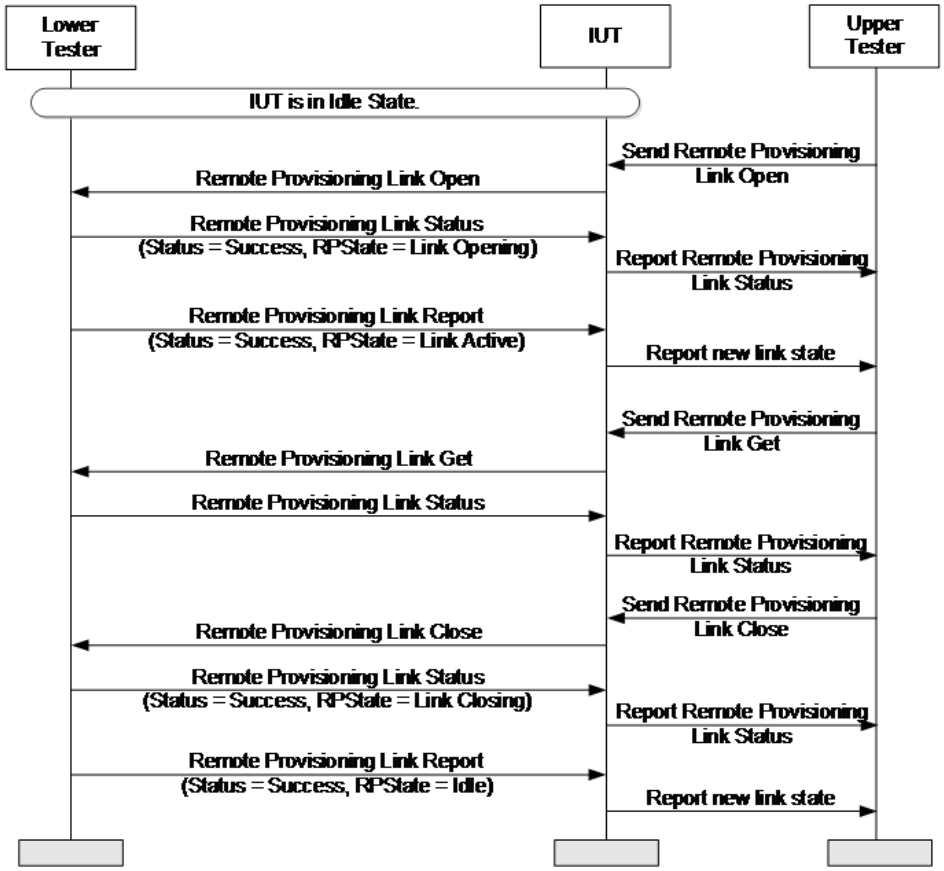


Figure 4.333: Test procedure for MESH/CL/RPR/LNK/BI-01-C MSC

1. The Upper Tester orders and the IUT sends a Remote Provisioning Link Open message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.
2. The Lower Tester sends a Remote Provisioning Link Report message to the IUT, indicating that the link has been opened, and the IUT reports the new link state.
3. The Upper Tester orders and the IUT sends a Remote Provisioning Link Get message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.
4. The Upper Tester orders and the IUT sends a Remote Provisioning Link Close message. The Lower Tester responds with a valid message and the IUT reports the result of the procedure to the Upper Tester.

- Expected Outcome

  Pass verdict

  The IUT sends the requested messages with correct format and parameter values and reports procedure completion and link changes to the Upper Tester.

4.22.3 Provisioning over PB-Remote

4.22.3.1 Provision an Unprovisioned Device

- Test Purpose

  Verify that the IUT can transfer provisioning PDUs between the Provisioner and the unprovisioned device to successfully complete a provisioning protocol.

- Reference

  [10] 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - Lower Tester 1 acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - Lower Tester 1 has access to the device key of the IUT.
  - Lower Tester 2 acts as an unprovisioned device with a randomly generated Device UUID.

- Test Procedure

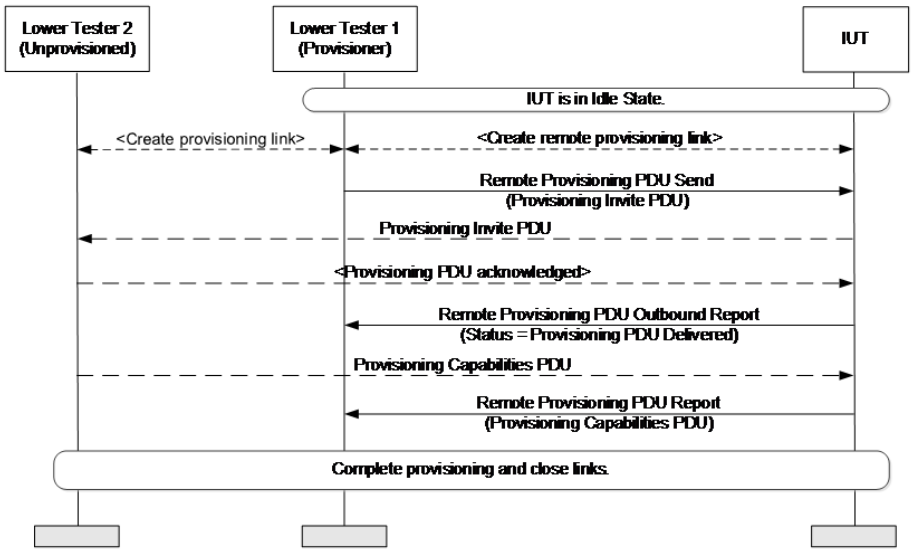


Figure 4.334: Test procedure - Provision an Unprovisioned Device MSC

1. Lower Tester 2 advertises as an unprovisioned device as indicated in the Step 1 column in Table 4.154, and Lower Tester 1 commands the IUT to create a provisioning link. The IUT reports that the link is open.
2. Lower Tester 1 sends a Remote Provisioning PDU Send message containing a Provisioning Invite PDU.
3. The IUT sends the Provisioning Invite PDU over the provisioning link, and then IUT sends a Remote Provisioning PDU Outbound Report message with the Status field set to Provisioning PDU Delivered.

4. Lower Tester 2 sends a Provisioning Capabilities PDU to the IUT over the provisioning link, and the IUT sends a Remote Provisioning PDU Report containing the PDU from Lower Tester 2.
5. Complete the provisioning procedure by repeating Steps 2 and 3 for every provisioning PDU originated by Lower Tester 1, in which case the IUT increments the value of the OutboundPDUNumber field for each new PDU, and by repeating Step 4 for every provisioning PDU originating from Lower Tester 2, in which case the IUT increments the value of the InboundPDUNumber field for each new PDU.
6. Close the provisioning link and the remote provisioning link.

- Test Case Configuration

TCID	Step 1
MESH/SR/RPR/PDU/BV-01-C [Provision an Unprovisioned Device over PB-ADV]	Lower Tester 2 begins sending Unprovisioned Device Beacons.
MESH/SR/RPR/PDU/BV-05-C [Provision an Unprovisioned Device over PB-GATT]	Lower Tester 2 begins sending connectable advertisements containing the Mesh Provisioning Service UUID.

Table 4.154: Additional test procedure steps for Provision an Unprovisioned Device

- Expected Outcome

Pass verdict

The IUT transfers all provisioning PDUs between the Lower Testers using Remote Provisioning PDU Status, Outbound Report, and Report messages. The IUT increments the value of the OutboundPDUNumber and InboundPDUNumber fields by 1 for each new PDU.
The Remote Provisioning PDU Report messages are sent as segmented access messages.

4.22.3.2 Node Provisioning Protocol Interface Procedures

- Test Purpose

Verify that the IUT can send and receive provisioning PDUs and use them for Node Provisioning Protocol Interface procedures.

- Reference

[10] 3.10.8, 4.2.1, 4.2.46, 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The Lower Tester has access to the device key of the IUT.
- Test Procedure

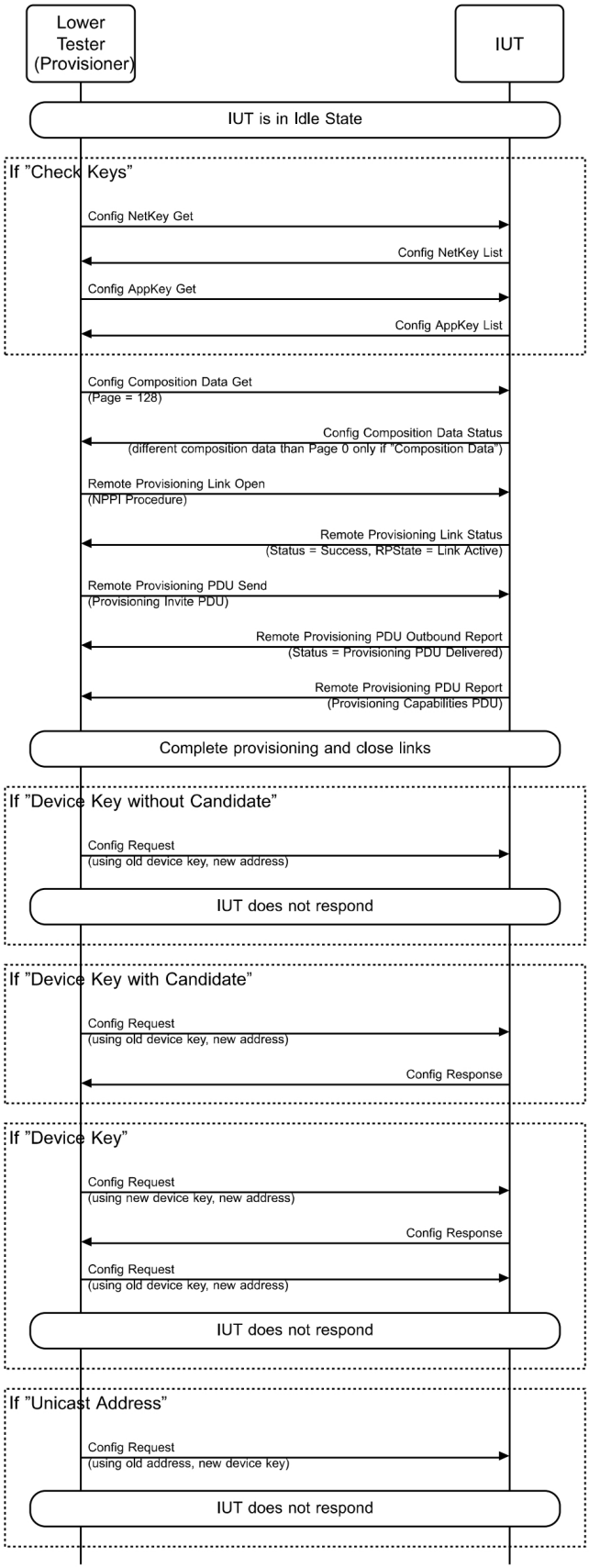


Figure 4.335: Test procedure - Node Provisioning Protocol Interface Procedures MSC - Page 1 of 2

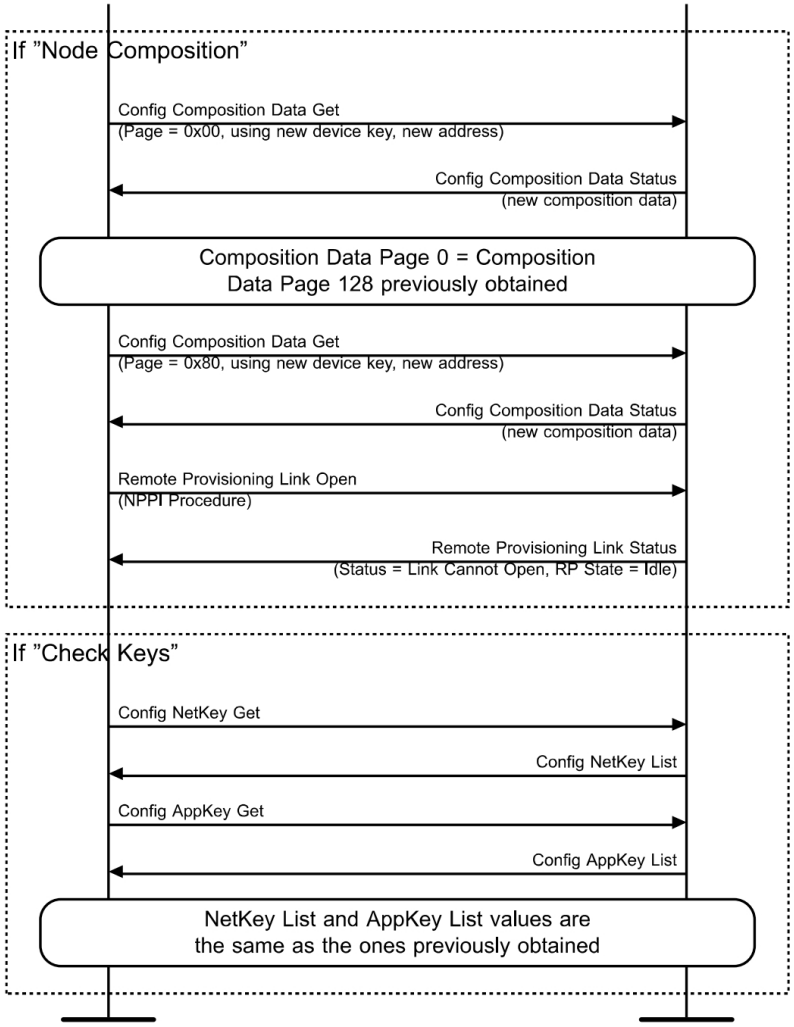


Figure 4.336: Test procedure - Node Provisioning Protocol Interface Procedures MSC - Page 2 of 2

1. If the Check Keys column in Table 4.155 indicates "Yes", then the Lower Tester reads the NetKey List and AppKey List states on the IUT and saves their values.
2. The Lower Tester reads the Composition Data Page 128 state on the IUT and saves its value. Additionally:
    a. If the Change Confirmation column in Table 4.155 indicates "Node Composition", then the Lower Tester verifies that Composition Page 128 is different than Composition Page 0.
    b. If the Large Composition Data Server Model is supported, as indicated by the ICS item MESH 11/22, then the Lower Tester reads the Models Metadata Page 128 state on the IUT and saves its value, and if the Change Confirmation column indicates "Node Composition", then the Lower Tester verifies that Models Metadata Page 128 is different than Models Metadata Page 0.
3. The Lower Tester sends a Remote Provisioning Link Open message with the UUID field omitted and the NPPI Procedure field set to the value listed in the Procedure column in Table 4.155. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active.
4. The Lower Tester sends a Remote Provisioning PDU Send message containing a Provisioning Invite

PDU.

5. The IUT sends a Remote Provisioning PDU Outbound Report message with the Status field set to Provisioning PDU Delivered.
6. The IUT sends a Remote Provisioning PDU Report containing a Provisioning Capabilities PDU.
7. Complete the provisioning procedure by repeating Steps 4 and 5 for every provisioning PDU originated by the Lower Tester, in which case the IUT increments the value of the OutboundPDUNumber field for each new PDU, and by repeating Step 6 for every provisioning PDU originated by the IUT, in which case the IUT increments the value of the InboundPDUNumber field for each new PDU. The Provisioning Data PDU contains the same network information available to the IUT prior to the test procedure, with the exceptions listed in the column Provisioning Data PDU.
8. Close the remote provisioning link.
9. If the Change Confirmation column in Table 4.155 indicates "Device Key without Candidate", then the Lower Tester sends any Configuration Model message using the old device key and the new address of the IUT, and the IUT does not respond.
10. If the Change Confirmation column in Table 4.155 indicates "Device Key with Candidate", then the Lower Tester sends any Configuration Model message using the old device key and the new address of the IUT, and the IUT responds with the appropriate message.
11. If the Change Confirmation column in Table 4.155 indicates "Device Key with Candidate" or "Device Key without Candidate", then the Lower Tester sends any Configuration Model message using the new device key and the new address of the IUT, and the IUT responds with the appropriate message.
12. If the Change Confirmation column in Table 4.155 indicates "Device Key with Candidate" or "Device Key without Candidate", then the Lower Tester sends any Configuration Model message using the old device key and the new address of the IUT, and the IUT does not respond.
13. If the Change Confirmation column in Table 4.155 indicates "Unicast Address", then the Lower Tester sends any Configuration Model message using the new device key and the old address of the IUT, and the IUT does not respond.
14. If the Change Confirmation column in Table 4.155 indicates "Node Composition", then execute the following sequence:
    a. The Lower Tester reads the Composition Data Page 0 state on the IUT using the new device key and the new address of the IUT, and the new value for the Composition Data Page 0 state is equal to the old value of the Composition Data Page 128 obtained in Step 2.
    b. The Lower Tester reads the Composition Data Page 128 state on the IUT and if it has not changed (i.e., it is the same value as the new Composition Data Page 0), then the Lower Tester resends the message in Step 3 and the IUT responds with a Remote Provisioning Link Status message with the Status field set to Link Cannot Open and the RPState field set to Idle.
    c. If the Large Composition Data Server Model is supported, then the Lower Tester reads the Models Metadata Page 0 state on the IUT using the new device key and the new address of the IUT, and the new value for the Models Metadata Page 0 state is equal to the old value of the Models Metadata Page 128 obtained in Step 2.
15. If the Check Keys column in Table 4.155 indicates "Yes", then the Lower Tester reads the NetKey List and AppKey List states on the IUT and verifies that their values are the same as the ones obtained in Step 1.

- Test Case Configuration

TCID	Check Keys (Steps 1 and 14)	Procedure (Step 3)	Provisioning Data PDU (Step 7)	Change Confirmation (Steps 9–14)
MESH/SR/RPR/PDU/BV-02-C [Device Key Refresh Procedure]	No	0x00	No difference	Device Key with Candidate
MESH/SR/RPR/PDU/BV-03-C [Node Address Refresh Procedure]	Yes	0x01	Different unicast address	Device Key without Candidate, Unicast Address
MESH/SR/RPR/PDU/BV-04-C [Node Composition Refresh Procedure]	No	0x02	No difference	Device Key with Candidate, Node Composition

Table 4.155: Additional test procedure steps for Node Provisioning Protocol Interface Procedures

- Expected Outcome

  Pass verdict

  The IUT exchanges all provisioning PDUs with the Lower Tester over the remote provisioning link.

  The IUT increments the value of the OutboundPDUNumber and InboundPDUNumber fields by 1 for each new PDU.

  The Remote Provisioning PDU Report messages are sent as segmented access messages.

  The IUT is able to use the new values of the parameters changed by the procedure.

  The IUT no longer uses the old values of the parameters changed by the procedure.

  The IUT preserves and updates its configuration states as required by the procedure.

4.22.3.3 Node Provisioning Protocol Interface Procedure Failures

- Test Purpose

  Verify that the IUT properly fails the Node Provisioning Protocol Interface procedures on receiving invalid data.

- Reference

  [10] 3.10.8, 4.3.4, 4.4.5

- Initial Condition
  - The IUT is in the Idle state.
  - The Lower Tester acts as Remote Provisioning Client and shares previously established network security credentials with the IUT.
  - The Lower Tester has saved the device key, the unicast address, and the Composition Data Page 0 of the IUT.

- Test Procedure

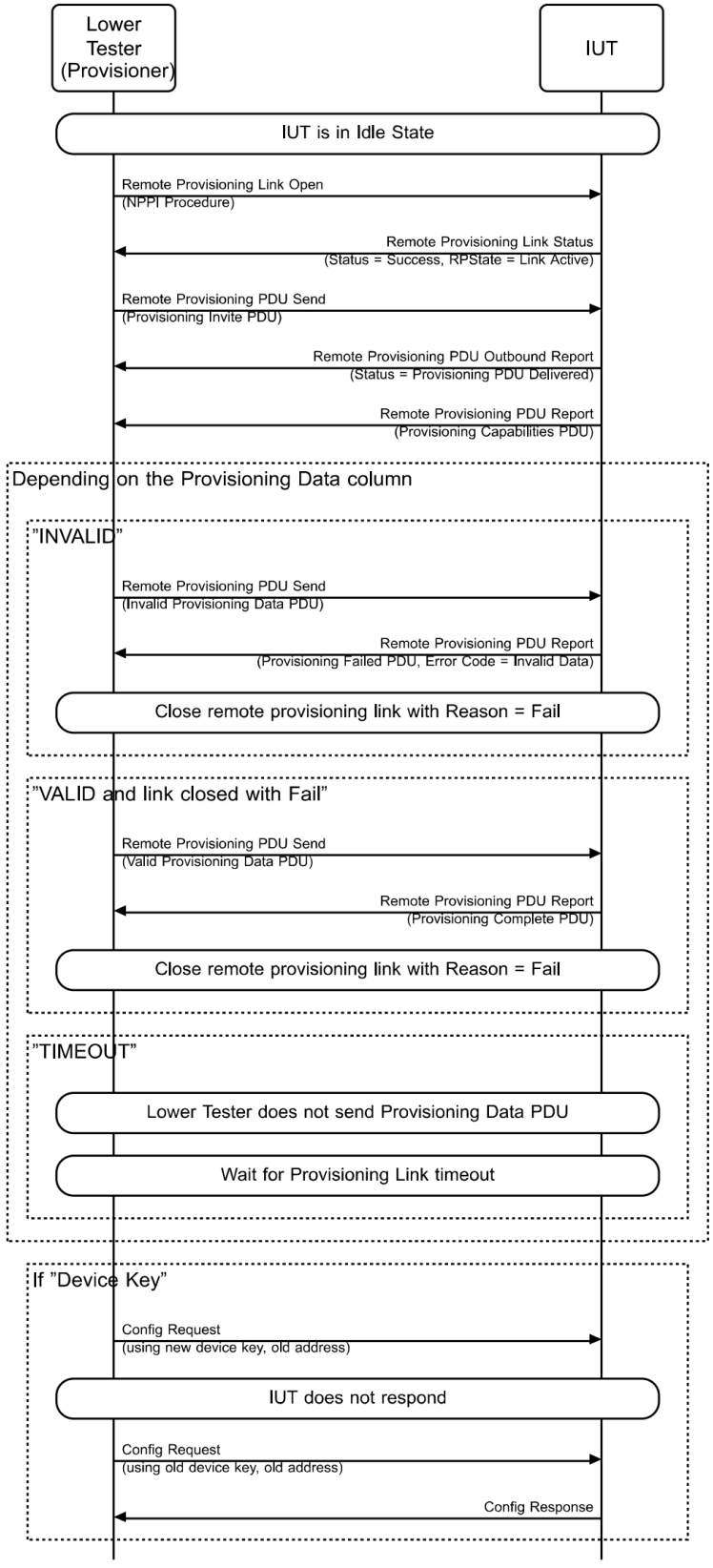


Figure 4.337: Test procedure for Node Provisioning Protocol Interface Procedure Failures MSC - Page 1 of 2

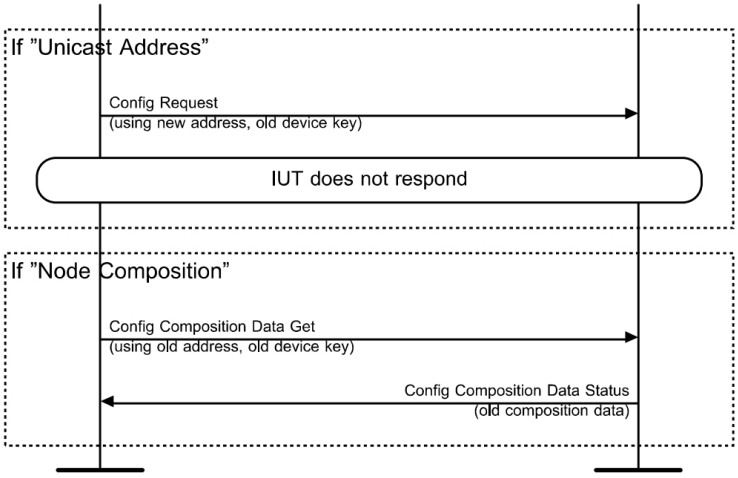


Figure 4.338: Test procedure for Node Provisioning Protocol Interface Procedure Failures MSC - Page 2 of 2

Execute Steps 1–10 for each round applicable to the test case as indicated in the Round column in Table 4.156:

1. The Lower Tester sends a Remote Provisioning Link Open message with the UUID field omitted and the NPPI Procedure field set to the value listed in the Procedure column in Table 4.156. The IUT responds with a Remote Provisioning Link Status message with the Status field set to Success and the RPState field set to Link Active.
2. The Lower Tester sends a Remote Provisioning PDU Send message containing a Provisioning Invite PDU.
3. The IUT sends a Remote Provisioning PDU Outbound Report message with the Status field set to Provisioning PDU Delivered.
4. The IUT sends a Remote Provisioning PDU Report containing a Provisioning Capabilities PDU.
5. Follow the provisioning procedure by repeating Steps 2 and 3 with the proper Provisioning PDUs contained in the remote provisioning messages.
6. When the Lower Tester needs to send the Provisioning Data PDU, follow the instructions indicated in the Provisioning Data column in Table 4.156, with one of the options as follows:
    a. If the Provisioning Data column indicates "INVALID", then the Provisioning Data PDU contains the same network information available to the IUT prior to the test procedure with the exception listed in Table 4.156; therefore, the PDU is invalid and the IUT sends a Provisioning Failed PDU in response, with the Error Code field set to Invalid Data. The Lower Tester closes the remote provisioning link with the Reason field set to Fail.
    b. If the Provisioning Data column indicates "VALID", then the Provisioning Data PDU contains valid fields (network information is consistent), and the IUT sends a Provisioning Complete PDU in response. The Lower Tester then closes the remote provisioning link with the Reason field set to Fail.
    c. If the Provisioning Data column indicates "TIMEOUT", then the Lower Tester does not send the Provisioning Data PDU and waits for the provisioning link to timeout.
7. If the Discard Confirmation column in Table 4.156 indicates "Device Key", then the Lower Tester sends any Configuration Model message using the new device key and the old address of the IUT, and the

IUT does not respond.

8. If the Discard Confirmation column in Table 4.156 indicates "Device Key", the Lower Tester sends any Configuration Model message using the old device key and the old address of the IUT, and the IUT responds with the appropriate message.
9. If the Discard Confirmation column in Table 4.156 indicates "Unicast Address", then the Lower Tester sends any Configuration Model message using the old device key and the new address of the IUT, and the IUT does not respond.
10. If the Discard Confirmation column in Table 4.156 indicates "Node Composition", then the Lower Tester sends a Composition Data Get message using the old device key and the old address of the IUT, and the IUT returns the old composition data page 0.

- Test Case Configuration

TCID	Round	Procedure (Step 1)	Provisioning Data (Step 6)	Discard Confirmation (Steps 7–10)
MESH/SR/RPR/PDU/BI-01-C [Device Key Refresh Failures]	1	0x00	INVALID: Different network key value, unknown to the IUT.	Device Key
	2		INVALID: Different NetKey Index, unknown to the IUT.	Device Key
	3		INVALID: Different IV Index.	Device Key
	4		INVALID: Different Unicast Address.	Device Key, Unicast Address
	5		VALID (No difference.)	Device Key
	6		TIMEOUT	Device Key

TCID	Round	Procedure (Step 1)	Provisioning Data (Step 6)	Discard Confirmation (Steps 7–10)
MESH/SR/RPR/PDU/BI-02-C [Node Address Refresh Failures]	1	0x01	INVALID: Different network key value, unknown to the IUT. Different unicast address.	Device Key, Unicast Address, Node Composition
	2		INVALID: Different NetKey Index, unknown to the IUT. Different unicast address.	Device Key, Unicast Address, Node Composition
	3		INVALID: Different IV Index. Different unicast address.	Device Key, Unicast Address, Node Composition
	4		INVALID: No difference.	Device Key, Node Composition
	5		VALID (Different unicast address.)	Device Key, Unicast Address, Node Composition
	6		TIMEOUT	Device Key, Node Composition
MESH/SR/RPR/PDU/BI-03-C [Node Composition Refresh Failures]	1	0x02	INVALID: Different network key value, unknown to the IUT.	Device Key, Node Composition
	2		INVALID: Different NetKey Index, unknown to the IUT.	Device Key, Node Composition
	3		INVALID: Different IV Index.	Device Key, Node Composition
	4		INVALID: Different Unicast Address.	Device Key, Unicast Address, Node Composition
	5		VALID (No difference.)	Device Key, Node Composition
	6		TIMEOUT	Device Key, Node Composition

Table 4.156: Additional test procedure steps for Node Provisioning Protocol Interface Procedure Failures

- Expected Outcome

  Pass verdict

  The IUT discards all parameter changes when failing the Device Key Interface procedure.

MESH/CL/RPR/PDU/BV-01-C [Execute a Complete Provisioning Procedure over PB-Remote]

- Test Purpose

  Verify that the IUT can run a complete provisioning protocol over PB-Remote.

- Reference

  [10] 4.3.4, 4.4.6

- Initial Condition
  - The Lower Tester acts as Remote Provisioning Server and shares previously established network security credentials with the IUT.
  - The IUT has access to the device key of the Lower Tester.
  - The Lower Tester can execute the Device Key Refresh procedure and can simulate a provisioning link with an unprovisioned device.
- Test Procedure

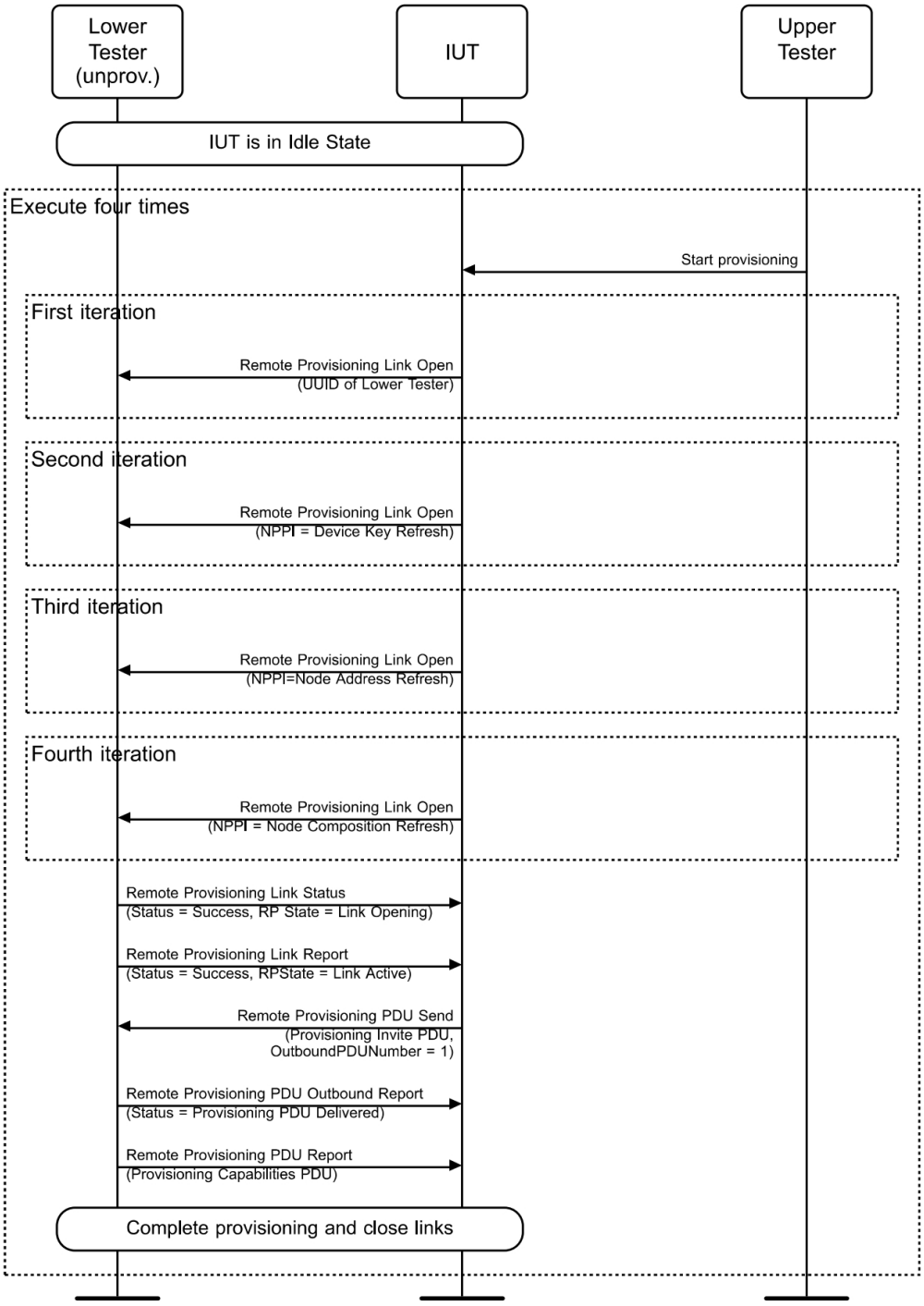


Figure 4.339: Test procedure for MESH/CL/RPR/PDU/BV-01-C MSC

1. The Upper Tester orders the IUT to begin provisioning an unprovisioned device with the UUID used by the Lower Tester that can be provisioned. The IUT creates a remote provisioning link using the Remote Provisioning Link Open message that contains the indicated Device UUID and replies with a Remote Provisioning Link Status message followed by a Remote Provisioning Link Report message indicating that the provisioning link has been open.
2. The IUT sends Remote Provisioning PDU Send message, containing a Provisioning Invite PDU and

with the OutboundPDUNumber field set to 1, and replies with a Remote Provisioning PDU Outbound Report message with the Status field set to Provisioning PDU Delivered.

3. The Lower Tester sends a Remote Provisioning PDU Report, containing a Provisioning Capabilities PDU.
4. The IUT continues the provisioning procedure until is it complete, by repeating Step 2 every time the IUT is sending a provisioning PDU and by repeating Step 3 every time the IUT is waiting for a provisioning PDU.
5. Repeat Steps 1–4 with the IUT running a Device Key Refresh procedure with the Lower Tester; the Remote Provisioning Link Open message contains the NPPI Procedure field with the value set to Device Key Refresh.
6. Repeat Steps 1–4 with the IUT running a Node Address Refresh procedure with the Lower Tester; the Remote Provisioning Link Open message contains the NPPI Procedure field with the value set to Node Address Refresh.
7. Repeat Steps 1–4 with the IUT running a Node Composition Refresh procedure with the Lower Tester; the Remote Provisioning Link Open message contains the NPPI Procedure field with the value set to Node Composition Refresh.

- Expected Outcome

  Pass verdict

  The IUT completes the provisioning protocol in all four iterations.

  When sending a Remote Provisioning PDU Send message, the IUT sets the OutboundPDUNumber field to 1 when sending the first provisioning PDU in a new procedure, and increments its value by 1 for each subsequent provisioning PDU sent within the same procedure.

4.23 On-demand Proxy Models

The objective of these tests is to verify the functionality of the On-demand Private Proxy Server Model and the On-demand Private Proxy Client Model.

4.23.1 On-demand Proxy State

MESH/SR/ODP/BV-01-C [On-Demand Private GATT Proxy State Get/Set Behavior]

- Test Purpose

  Verify that an On-demand Private Proxy Server IUT can process ON_DEMAND_PRIVATE_PROXY_GET and ON_DEMAND_PRIVATE_PROXY_SET messages from an On-demand Private Proxy Client, maintaining the binding between the On-Demand Private GATT Proxy state and the Private GATT Proxy state.

- Reference

  [10] 4.2.22, 4.3.4, 4.4.5, 4.4.11, 4.4.12

- Initial Condition

- The IUT supports the On-demand Private Proxy Server model on the Primary Element.
- The Lower Tester supports the On-demand Private Proxy Client Model.
- The IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT defined in the IXIT [6] parameters.
- The Lower Tester has read the value of Private GATT Proxy state on the IUT.

• Test Procedure

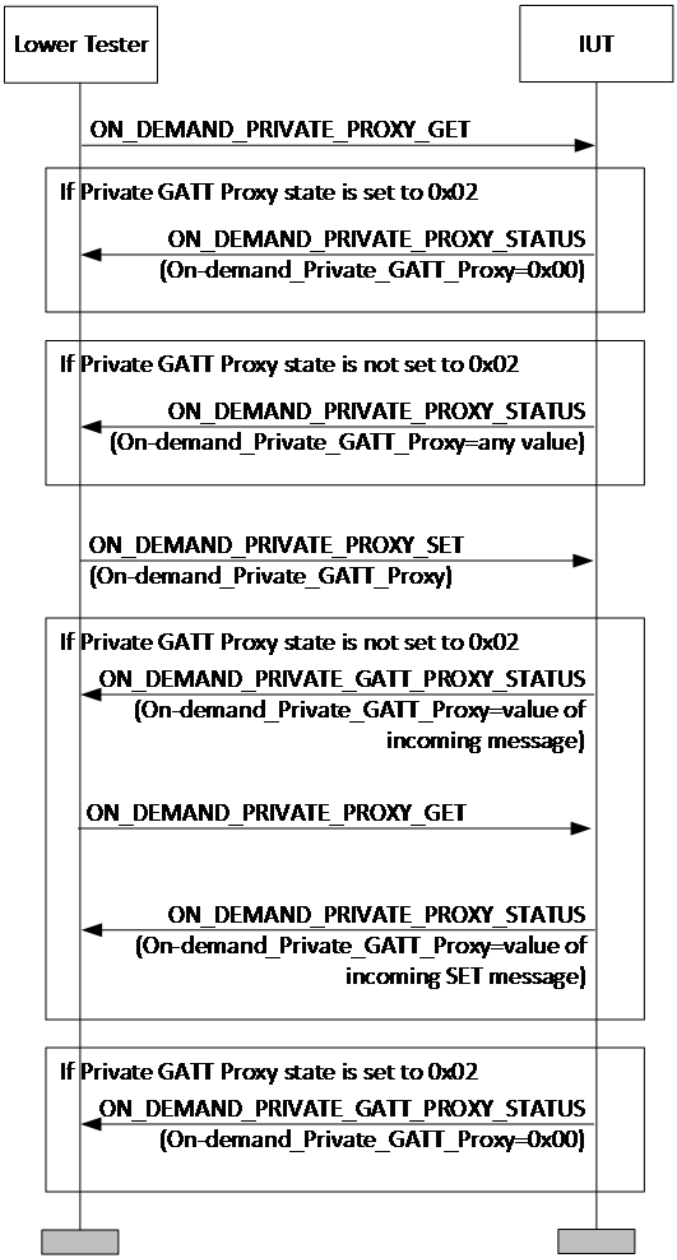


Figure 4.340: Test procedure for MESH/SR/ODP/BV-01-C MSC

1. The Lower Tester sends an ON_DEMAND_PRIVATE_PROXY_GET message to the IUT.
2. The IUT responds with an ON_DEMAND_PRIVATE_PROXY_STATUS message with the On-demand_Private_GATT_Proxy field set to 0x00 if the Private GATT Proxy state is set to 0x02, or any value otherwise.
3. The Lower Tester sends an ON_DEMAND_PRIVATE_PROXY_SET message to the IUT with On-demand_Private_GATT_Proxy field having a random valid value different than the one received in Step 2.
4. If the Private GATT Proxy state value is not set to 0x02 (the Private Proxy feature is not supported), the IUT responds with an ON_DEMAND_PRIVATE_PROXY_STATUS message with the On-demand_Private_GATT_Proxy field set to the value from Step 3. Repeat Steps 1 and 2 expecting the On-demand_Private_GATT_Proxy field set to the value from Step 3.
5. If the Private GATT Proxy state value is set to 0x02 (the Private Proxy feature is not supported), the IUT responds with an ON_DEMAND_PRIVATE_PROXY_STATUS message with the On-demand_Private_GATT_Proxy field value set to 0x00.

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages from the Lower Tester.

  All messages are encrypted with the device key of the IUT.

MESH/CL/ODP/BV-01-C [On-Demand Private GATT Proxy State Get/Set Procedure]

- Test Purpose

  Verify that an On-demand Private Proxy Client IUT can configure the On-Demand Private GATT Proxy state on an On-demand Private Proxy Server.

- Reference

  [10] 4.2.22, 4.3.4, 4.4.6, 4.4.11, 4.4.12

- Initial Condition
  - The IUT supports the On-demand Private Proxy Client Model.
  - The Lower Tester supports the On-demand Private Proxy Server model on the Primary Element.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

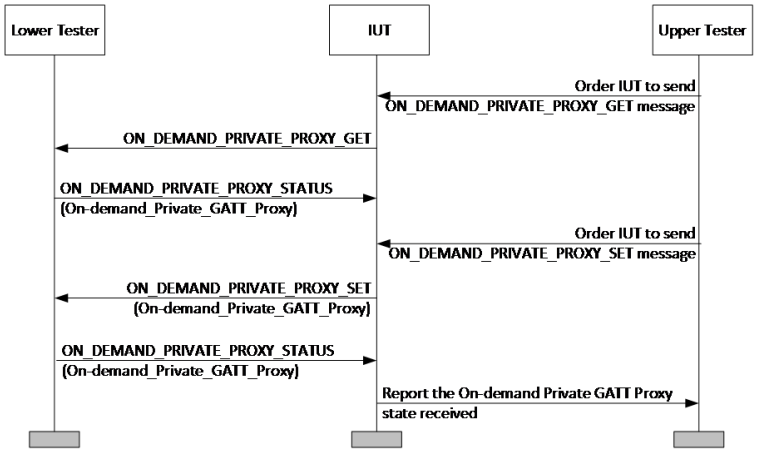


Figure 4.341: Test procedure for MESH/CL/ODP/BV-01-C MSC

1. The Upper Tester orders the IUT to send an ON_DEMAND_PRIVATE_PROXY_GET message to the Lower Tester.
2. The IUT sends an ON_DEMAND_PRIVATE_PROXY_GET message to the Lower Tester.
3. The Lower Tester responds with an ON_DEMAND_PRIVATE_PROXY_STATUS with the On-demand_Private_GATT_Proxy field set to a random value.
4. The Upper Tester orders the IUT to send an ON_DEMAND_PRIVATE_PROXY_SET message to the Lower Tester with a new random valid value for the state.
5. The IUT sends an ON_DEMAND_PRIVATE_PROXY_SET message with the On-demand_Private_GATT_Proxy field set to the value requested by the Upper Tester.
6. The Lower Tester responds with an ON_DEMAND_PRIVATE_PROXY_STATUS message with the On-demand_Private_GATT_Proxy field set to the value requested in Step 5.
7. The IUT reports to the Upper Tester the On-Demand Private GATT Proxy state received from the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends correctly formatted ON_DEMAND_PRIVATE_PROXY_GET and ON_DEMAND_PRIVATE_PROXY_SET messages.

  The IUT correctly reports the value of the On-Demand Private GATT Proxy state received from the Lower Tester.

  All messages are encrypted with the device key of the Lower Tester.

4.24 SAR Configuration Models

The objective of these tests is to verify the functionality of the SAR Configuration Server Model and the SAR Configuration Client Model.

4.24.1 SAR Transmitter State

MESH/SR/SAR/BV-01-C [SAR Transmitter State Get/Set Behavior]

- Test Purpose

  Verify that a SAR Configuration Server IUT can process SAR_TRANSMITTER_GET and SAR_TRANSMITTER_SET messages from a SAR Configuration Client.

- Reference

  [10] 4.2.23, 4.3.5, 4.4.7, 4.4.13, 4.4.14

- Initial Condition

  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

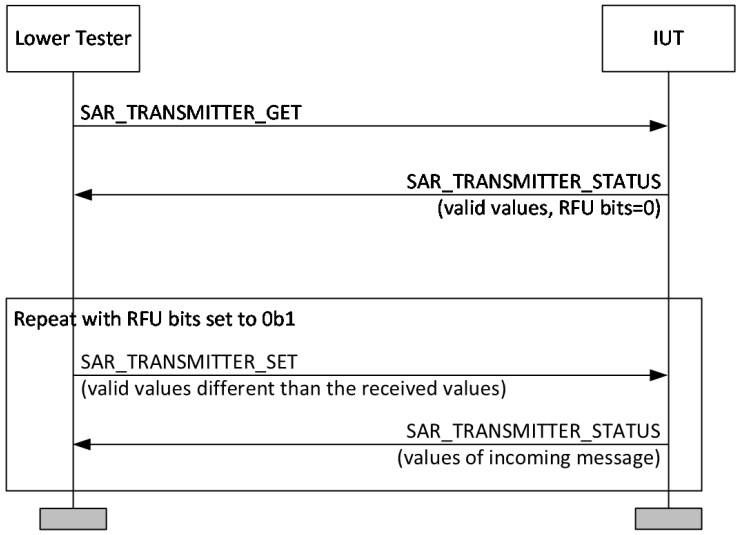


Figure 4.342: Test procedure for MESH/SR/SAR/BV-01-C MSC

1. The Lower Tester sends a SAR_TRANSMITTER_GET message to the IUT.
2. The IUT responds with a SAR_TRANSMITTER_STATUS message with valid values and all RFU bits set to 0.
3. The Lower Tester sends a SAR_TRANSMITTER_SET message with all fields set to random valid values different than the received values.
4. The IUT responds with a SAR_TRANSMITTER_STATUS message with parameters set as in Step 3.
5. Repeat Steps 3 and 4 using in Step 3 all RFU bits set to 0b1.

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages from the Lower Tester.

  In Step 2 and both iterations of Step 4, the IUT sends a SAR_TRANSMITTER_STATUS message to the Lower Tester with the indicated field values, and with the RFU bits set to 0.

  All messages are encrypted with the device key of the IUT.

MESH/CL/SAR/BV-01-C [SAR Transmitter State Get/Set Procedure]

- Test Purpose

  Verify that a SAR Configuration Client IUT can configure the SAR Transmitter state on a SAR Configuration Server.

- Reference

  [10] 4.2.23, 4.3.5, 4.4.8, 4.4.13, 4.4.14

- Initial Condition
  - The Lower Tester supports the SAR Configuration Server Model on the Primary Element.
  - The IUT supports the SAR Configuration Client Model.
  - The Lower Tester has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The IUT has access to the Device Key for the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

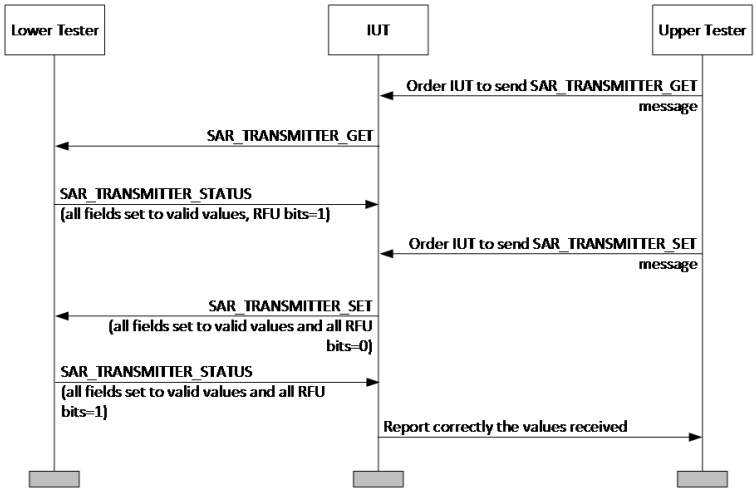


Figure 4.343: Test procedure for MESH/CL/SAR/BV-01-C MSC

1. The Upper Tester orders the IUT to send a SAR_TRANSMITTER_GET message to the Lower Tester.
2. The IUT sends a SAR_TRANSMITTER_GET message.
3. The Lower Tester responds with a SAR_TRANSMITTER_STATUS message with all fields set to valid values and all RFU bits set to 1.
4. The Upper Tester orders the IUT to send a SAR_TRANSMITTER_SET message to the Lower Tester.
5. The IUT sends a SAR_TRANSMITTER_SET message with all fields set to valid values and all RFU bits set to 0.
6. The Lower Tester responds with a SAR_TRANSMITTER_STATUS message with parameters set as in Step 5 except all RFU bits, which are set to 1.
7. The IUT reports to the Upper Tester the values received from the Lower Tester.

- Expected Outcome

  Pass verdict

  The IUT sends correctly formatted SAR_TRANSMITTER_GET and SAR_TRANSMITTER_SET messages.

  The IUT correctly reports to the Upper Tester the values received from the Lower Tester ignoring RFU bits set to 1.

  All messages are encrypted with the device key of the Lower Tester.

4.24.2 SAR Receiver State

MESH/SR/SAR/BV-02-C [SAR Receiver State Get/Set Behavior]

- Test Purpose

  Verify that a SAR Configuration Server IUT can process SAR_RECEIVER_GET and SAR_RECEIVER_SET messages from a SAR Configuration Client.

- Reference

  [10] 4.2.24, 4.3.5, 4.4.7, 4.4.13, 4.4.14

- Initial Condition
  - The IUT supports the SAR Configuration Server Model on the Primary Element.
  - The Lower Tester supports the SAR Configuration Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.
- Test Procedure

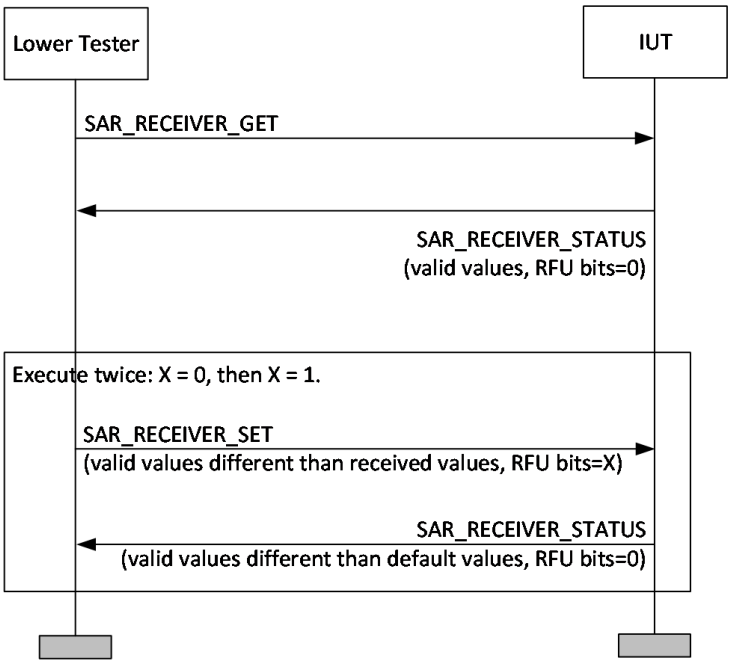


Figure 4.344: Test procedure for MESH/SR/SAR/BV-02-C MSC

1. The Lower Tester sends a SAR_RECEIVER_GET message to the IUT.
2. The IUT responds with a SAR_RECEIVER_STATUS message with valid values for all fields and all RFU bits set to 0.
3. The Lower Tester sends a SAR_RECEIVER_SET message with all fields set to valid values different than default values.
4. The IUT responds with a SAR_RECEIVER_STATUS message with parameters set as in Step 3.

5. Repeat Steps 3 and 4 using, in Step 3, all RFU bits set to 0b1 and expecting their value in Step 4 to be 0.

- Expected Outcome

  Pass verdict

  The IUT responds correctly to all messages from the Lower Tester.

  All messages are encrypted with the device key of the IUT.

MESH/CL/SAR/BV-02-C [SAR Receiver State Get/Set Procedure]

- Test Purpose

  Verify that a SAR Configuration Client IUT can configure the SAR Receiver state on a SAR Configuration Server.

- Reference

  [10] 4.2.24, 4.3.5, 4.4.7, 4.4.13, 4.4.14

- Initial Condition

  - The Lower Tester supports the SAR Configuration Server Model on the Primary Element.
  - The IUT supports the SAR Configuration Client Model.
  - The Lower Tester has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The IUT has access to the Device Key for the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

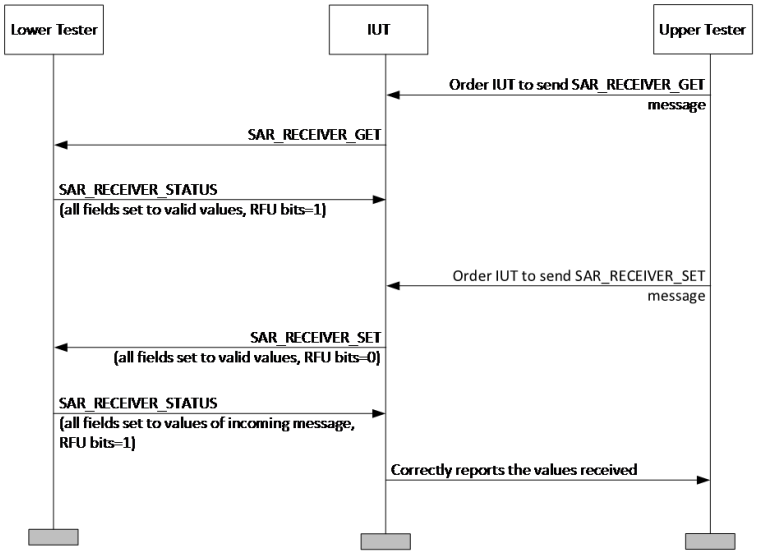


Figure 4.345: Test procedure for MESH/CL/SAR/BV-02-C MSC

1. The Upper Tester orders the IUT to send a SAR_RECEIVER_GET message to the Lower Tester.
2. The IUT sends a SAR_RECEIVER_GET message.
3. The Lower Tester responds with a SAR_RECEIVER_STATUS message with all fields set to valid values and all RFU bits set to 1.
4. The Upper Tester orders the IUT to send a SAR_RECEIVER_SET message to the Lower Tester.
5. The IUT sends a SAR_RECEIVER_SET message with all fields set to valid values and all RFU bits set to 0.
6. The Lower Tester responds with a SAR_RECEIVER_STATUS message with parameters set as in Step 5 excepting all RFU bits that are set to 1.
7. The IUT reports the values received to the Upper Tester.

- Expected Outcome

  Pass verdict

  The IUT sends correctly formatted SAR_RECEIVER_GET and SAR_RECEIVER_SET messages and reports the values received to the Upper Tester.

  All messages are encrypted with the device key of the Lower Tester.

4.25 Opcodes Aggregator Models

The objective of these tests is to verify the functionality of the Opcodes Aggregator Server Model and the Opcodes Aggregator Client Model.

MESH/SR/AGG/BV-01-C [Message Request List Processing Behavior – Success]

- Test Purpose

Verify that an Opcodes Aggregator Server IUT can successfully process OPCODES_AGGREGATOR_SEQUENCE messages from an Opcodes Aggregator Client, as well as the encapsulated Access PDUs from these messages.

- Reference

  [10] 4.3.6, 4.4.9

- Initial Condition
  - The IUT supports the Opcodes Aggregator Server Model on the Primary Element.
  - The Lower Tester supports the Opcodes Aggregator Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network and application security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters. A shared application key has been bound to the Opcodes Aggregator Server Model.
- Test Procedure

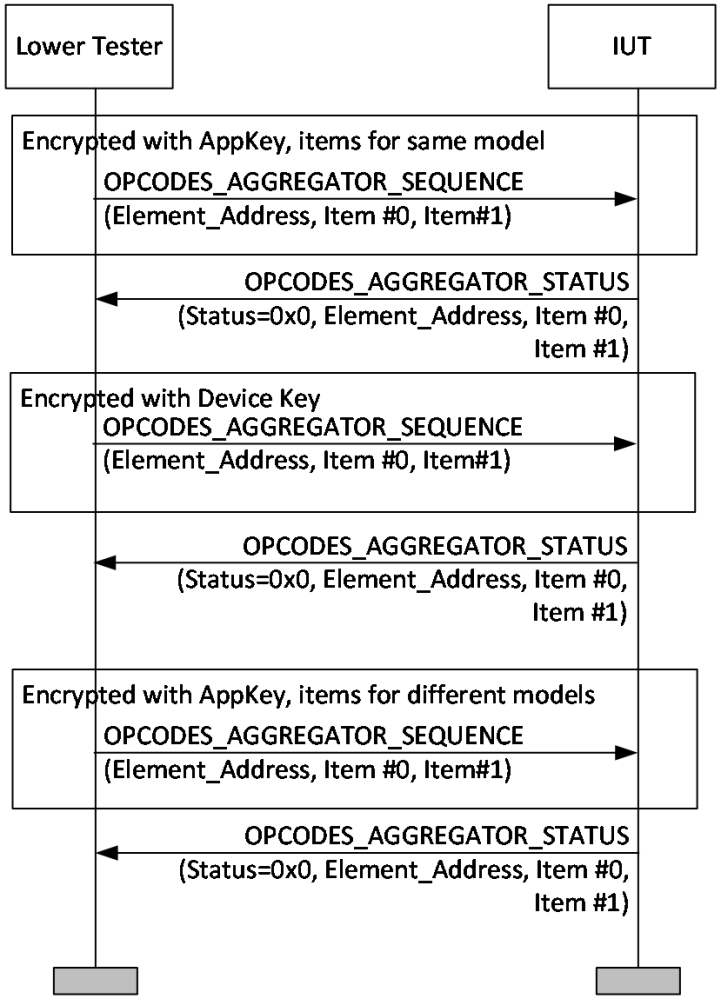


Figure 4.346: Test procedure for MESH/SR/AGG/BV-01-C MSC

1. The Lower Tester sends an OPCODES_AGGREGATOR_SEQUENCE message to the IUT with Element_Address field set to the unicast address of the element, Item #0 identifying an acknowledged message and Item #1 identifying an unacknowledged message, with both opcodes belonging to the same model. The message is encrypted with the AppKey bound to the applicable model.
2. The IUT responds with an OPCODES_AGGREGATOR_STATUS message with Status set to 0x0 (Success), Element_Address field set to the value from the incoming message, Item #0 set to the response to the acknowledged message, and Item #1 set to an empty item.
3. The Lower Tester sends an OPCODES_AGGREGATOR_SEQUENCE message to the IUT with Element_Address field set to the unicast address of the element, and two items identifying two acknowledged messages of the Configuration Server Model. The message is encrypted with the Device Key.
4. The IUT responds with an OPCODES_AGGREGATOR_STATUS message with Status set to 0x0 (Success), Element_Address field set to the value from the incoming message, Item #0 set to the response to the first acknowledged message, and Item #1 set to the response to the second acknowledged message.

5. If the IUT does not support at least two models on the same element to allow execution of Steps 6 and 7, then skip those steps and end the test with a Pass verdict.
6. The Lower Tester sends an OPCODES_AGGREGATOR_SEQUENCE message to the IUT with Element_Address field set to the unicast address of the element, Item #0 identifying an acknowledged message, and Item #1 identifying an unacknowledged message, with each opcode belonging to a different model. The message is encrypted with the AppKey bound to the two models and to the Opcode Aggregator Server model. OPCODES_AGGREGATOR_SEQUENCE and Element_Address are specified in the IXIT [6].
7. The IUT responds with an OPCODES_AGGREGATOR_STATUS message with Status set to 0x0 (Success), Element_Address field set to the value from the incoming message, Item #0 set to the response to the acknowledged message, and Item #1 set to an empty item.

- Expected Outcome

  Pass verdict

  The IUT responds correctly to the OPCODES_AGGREGATOR_SEQUENCE messages from the Lower Tester.

MESH/SR/AGG/BV-02-C [Message Request List Processing Behavior – Errors]

- Test Purpose

  Verify that an Opcodes Aggregator Server IUT can process OPCODES_AGGREGATOR_SEQUENCE messages from an Opcodes Aggregator Client when some of the encapsulated Access PDUs generate errors.

- Reference

  [10] 4.3.6, 4.4.9

- Initial Condition
  - The IUT supports the Opcodes Aggregator Server Model on the Primary Element.
  - The Lower Tester supports the Opcodes Aggregator Client Model.
  - The IUT has been provisioned and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

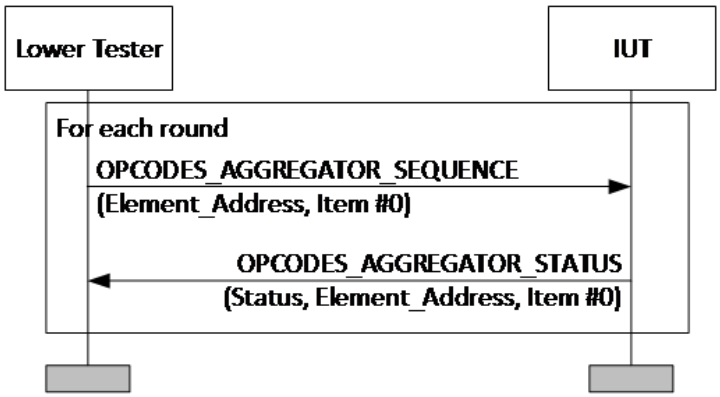


Figure 4.347: Test procedure for MESH/SR/AGG/BV-02-C MSC

Repeat Steps 1 and 2 for each round in Table 4.157.

1. The Lower Tester sends an OPCODES_AGGREGATOR_SEQUENCE message to the IUT with Element_Address field and one or two items set according to Table 4.157.
2. The IUT responds with an OPCODES_AGGREGATOR_STATUS message with Status set according to Table 4.157 and Element_Address and Item #0 fields set as in the incoming message.

Round	Error Description	Status Error
1	Element_Address field does not contain the IUT's unicast address	0x01 (Invalid Address)
2	Configuration message encrypted with an application key present on the IUT	0x02 (WrongAccessKey)
3	Access message encrypted with a device key, and the identified model's access-layer security is not using a device key	0x02 (WrongAccessKey)
4	Access message encrypted with an application key not bound to the identified model	0x02 (WrongAccessKey)
5	Item #1's Opcode not supported by the identified element (but Item #0 is valid)	0x03 (WrongOpCode)
6	Item #0's Opcode not supported by the identified element (but Item #1 is valid)	0x03 (WrongOpCode)
7	Message parameters not understood by the identified model	0x04 (MessageNotUnderstood)

Table 4.157: Message Request List Processing Behavior - Errors Rounds

- Expected Outcome

  Pass verdict

  The IUT responds with the correct status code for each error scenario.

MESH/SR/AGG/BI-01-C [Message Request List Processing Behavior – Invalid Parameters]

- Test Purpose

  Verify that an Opcodes Aggregator Server IUT ignores OPCODES_AGGREGATOR_SEQUENCE messages from an Opcodes Aggregator Client with invalid parameters.

- Reference

  [10] 4.3.6, 4.4.9, 4.4.15, 4.4.16

- Initial Condition

  - The IUT supports the Opcodes Aggregator Server Model on the Primary Element.
  - The Lower Tester supports the Opcodes Aggregator Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the Device Key for the IUT defined in the IXIT [6] parameters.

- Test Procedure

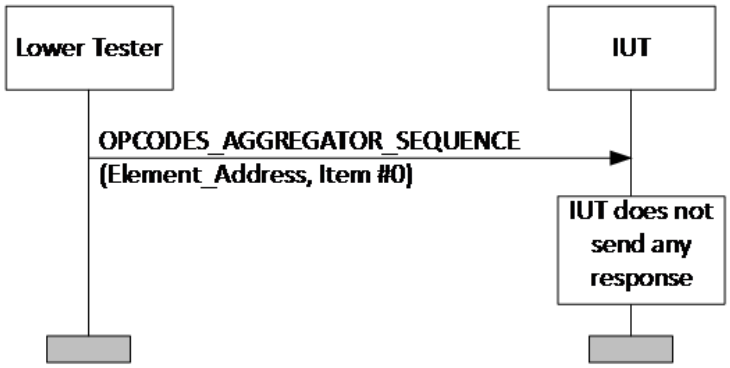


Figure 4.348: Test procedure for MESH/SR/AGG/BI-01-C MSC

1. The Lower Tester sends an OPCODES_AGGREGATOR_SEQUENCE message to the IUT with an invalid address type in the Element_Address field and Item #0 indicating an acknowledged message.
2. The IUT ignores the message and does not send any response.

- Expected Outcome

  Pass verdict

  The IUT does not respond to the invalid message.

MESH/CL/AGG/BV-01-C [Message Request List Processing Procedure]

- Test Purpose

Verify that an Opcodes Aggregator Client IUT can execute the Message Request List Processing Procedure on an Opcodes Aggregator Server.

- Reference

  [10] 4.3.6, 4.4.10, 4.4.15, 4.4.16

- Initial Condition

  - The Lower Tester supports the Opcodes Aggregator Server Model on the Primary Element.
  - The IUT supports the Opcodes Aggregator Client Model.
  - The Lower Tester has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The IUT has access to the Device Key for the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

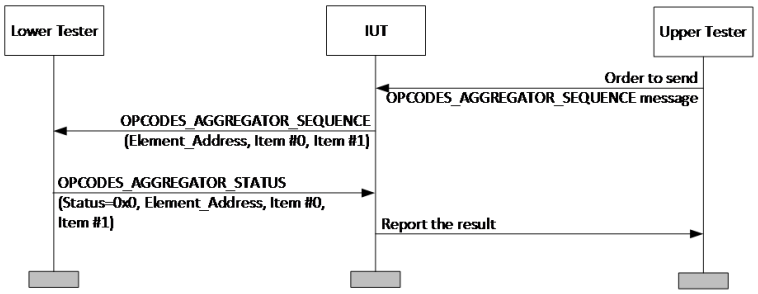


Figure 4.349: Test procedure for MESH/CL/AGG/BV-01-C MSC

1. The Upper Tester orders the IUT to send an OPCODES_AGGREGATOR_SEQUENCE message with an acknowledged message and an unacknowledged message.
2. The IUT sends an OPCODES_AGGREGATOR_SEQUENCE message to the Lower Tester with Element_Address field set to the unicast address of the element, Item #0 identifying an acknowledged message and Item #1 identifying an unacknowledged message.
3. The Lower Tester responds with an OPCODES_AGGREGATOR_STATUS message with Status set to 0x0 (Success), Element_Address field set to the value from the incoming message, Item #0 set to the response to the acknowledged message (from Item #0 of the message in Step 2), and Item #1 set to an empty item.
4. The IUT reports the result of the procedure to the Upper Tester.

- Expected Outcome

  Pass verdict

  The IUT sends an OPCODES_AGGREGATOR_SEQUENCE message with correct format and reports the result of each message.

4.26 Large Composition Data Models

The objective of these tests is to verify the functionality of the Large Composition Data Server Model and the Large Composition Data Client Model.

4.26.1 Large Composition Data Procedures

This section validates the Large Composition Data procedures for both the Large Configuration Server Model and the Large Configuration Client Model.

MESH/SR/LCD/LCMP/BV-01-C [Respond to LARGE_COMPOSITION_DATA_GET Messages]

- Test Purpose

  Verify that an IUT node supporting the Large Composition Data Server Model can respond to LARGE_COMPOSITION_DATA_GET messages and that the IUT supports the mandatory models.

- Reference

  [10] 4.3.9, 4.4.17, 4.4.18

- Initial Condition
  - The IUT supports the Large Composition Data Server Model on the primary Element.
  - The Lower Tester supports the Large Composition Data Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key of the IUT and the number of pages of the IUT's Composition Data defined in the IXIT [6] parameters.
- Test Procedure

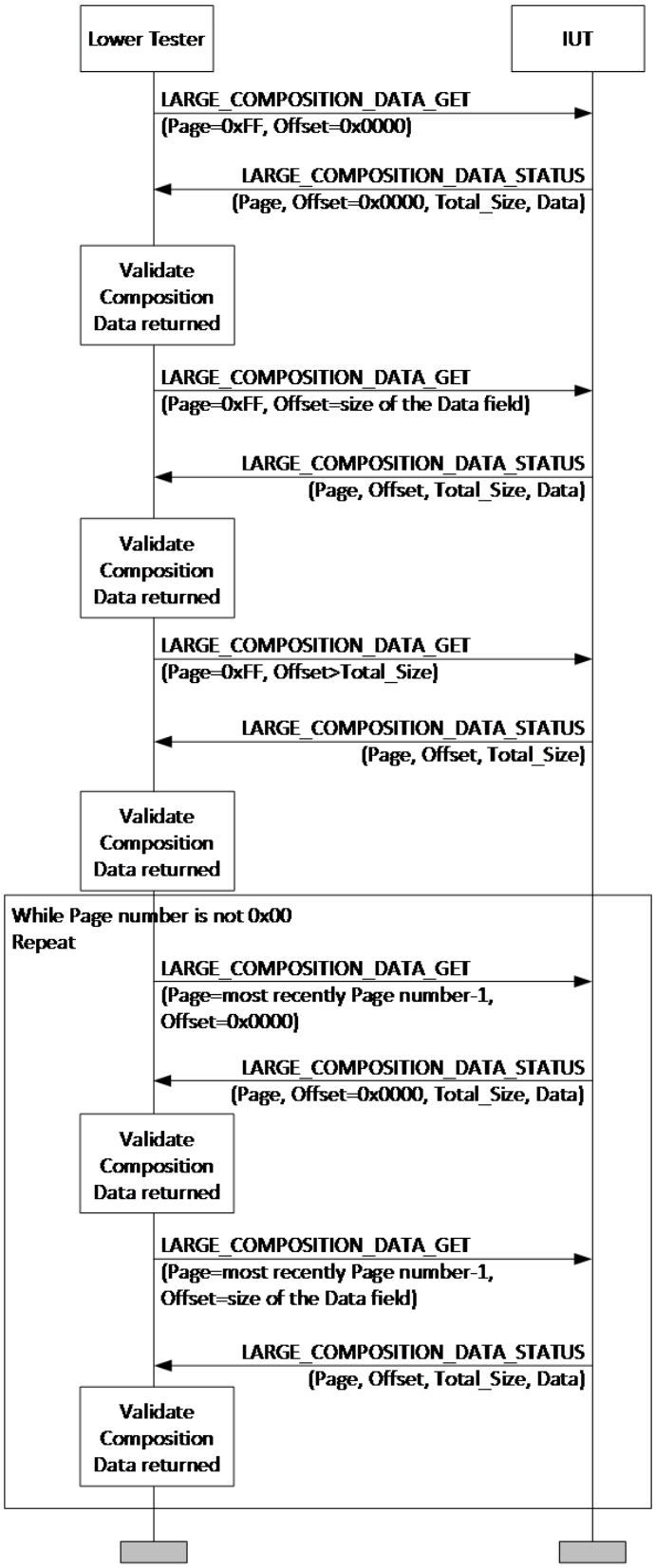


Figure 4.350: Test procedure for MESH/SR/LCD/LCMP/BV-01-C MSC

1. The Lower Tester sends a LARGE_COMPOSITION_DATA_GET message to the IUT with the Page field value set to 0xFF and the Offset field value set to 0x0000.
2. The IUT responds with a LARGE_COMPOSITION_DATA_STATUS message with the Page field set to the maximum Composition Data Page number defined in IXIT [6], the Offset field set to 0x0000, the Total_Size field set to the total size of that Composition Data Page, and the Data field set to the portion of the information contained by that Composition Data Page starting from 0x0000.
3. The Lower Tester sends a LARGE_COMPOSITION_DATA_GET message to the IUT with the Page field value set to 0xFF and the Offset field value set to the size of the Data field from Step 2.
4. The IUT responds with a LARGE_COMPOSITION_DATA_STATUS message with the Page field set to the maximum Composition Data Page number defined in IXIT [6], the Offset field set to the size of the Data field from Step 2, the Total_Size field set to the total size of that Composition Data Page, and the Data field set to the portion of the information contained by that Composition Data Page starting from the size of the Data field from Step 2.
5. The Lower Tester sends a LARGE_COMPOSITION_DATA_GET message to the IUT with the Page field value set to 0xFF and the Offset field set to a value greater than the Total_Size field value received in Step 2.
6. The IUT responds with a LARGE_COMPOSITION_DATA_STATUS message with the Page field set to the maximum Composition Data Page number defined in IXIT [6], the Offset field set as in the incoming message, the Total_Size field set to the total size of that Composition Data Page, and with no Data field.
7. While the page number received in Steps 2 and 4 is not 0x00, repeat Steps 1–4, setting the Page field in Steps 1 and 3 to the most recently received page number minus one.
8. The Lower Tester parses the Composition Data returned by the IUT in all iterations through Step 2 and checks that all fields are valid and consistent and that the mandatory root models are supported.

- Expected Outcome

  Pass verdict

  In all iterations through Step 2, the IUT sends complete and correctly formatted Composition Data. The Configuration Model and the Health Model are supported on the primary Element. All fields from the Composition Data information contain valid values.
  All configuration messages are encrypted using the device key of the IUT.

MESH/CL/LCD/LCMP/BV-01-C [Send LARGE_COMPOSITION_DATA_GET Messages]

- Test Purpose

  Verify that an IUT node supporting the Large Composition Data Client Model can send LARGE_COMPOSITION_DATA_GET messages to a Large Composition Data Server.

- Reference

  [10] 4.3.9, 4.4.17, 4.4.18

- Initial Condition

  - The IUT supports the Large Composition Data Client Model.

- The Lower Tester supports the Large Composition Data Server Model on the Primary Element.
- The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

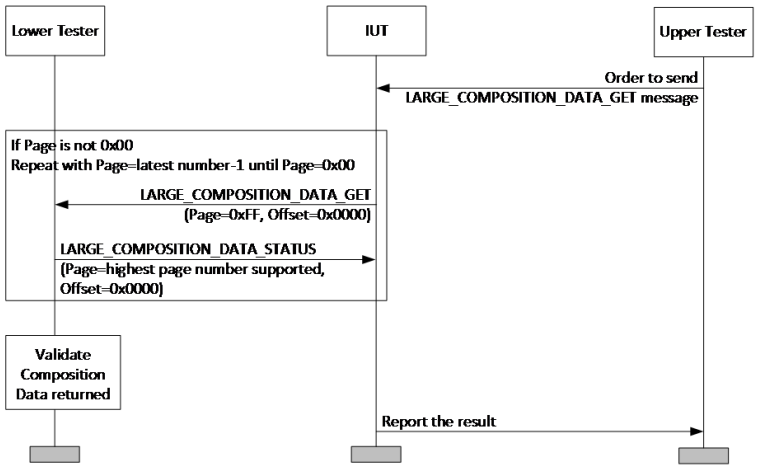


Figure 4.351: Test procedure for MESH/CL/LCD/LCMP/BV-01-C MSC

1. The Upper Tester orders the IUT to obtain the device composition data of the Lower Tester using the Large Composition Data Client Model.
2. The IUT sends a LARGE_COMPOSITION_DATA_GET message with the highest supported page number or 0xFF and the Offset field set to 0x0000.
3. The Lower Tester responds with a LARGE_COMPOSITION_DATA_STATUS message with the Page field value set to the highest page number supported by the Lower Tester and the Offset field set to 0x0000.
4. If the page number sent in Step 3 is greater than 0x00, then repeat Steps 2 and 3, expecting the IUT always to request a page number equal to the one previously received, decremented by 1, until Page 0 has been sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a properly formatted LARGE_COMPOSITION_DATA_GET message to the Lower Tester.

  If applicable, the IUT requests a page number equal to the previously received page number minus 1.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.26.2 Models Metadata Procedures

This section validates the Models Metadata procedures for both the Large Configuration Server Model and the Large Configuration Client Model.

MESH/SR/LCD/MMD/BV-01-C [Respond to MODELS_METADATA_GET Messages]

- Test Purpose

  Verify that an IUT node supporting the Large Composition Data Server Model can respond to MODELS_METADATA_GET messages.

- Reference

  [10] 4.2.46, 4.4.17, 4.4.18

- Initial Condition
  - The IUT supports the Large Composition Data Server Model on the Primary Element.
  - The Lower Tester supports the Large Composition Data Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to the device key for the IUT defined in the IXIT [6] parameters.
- Test Procedure

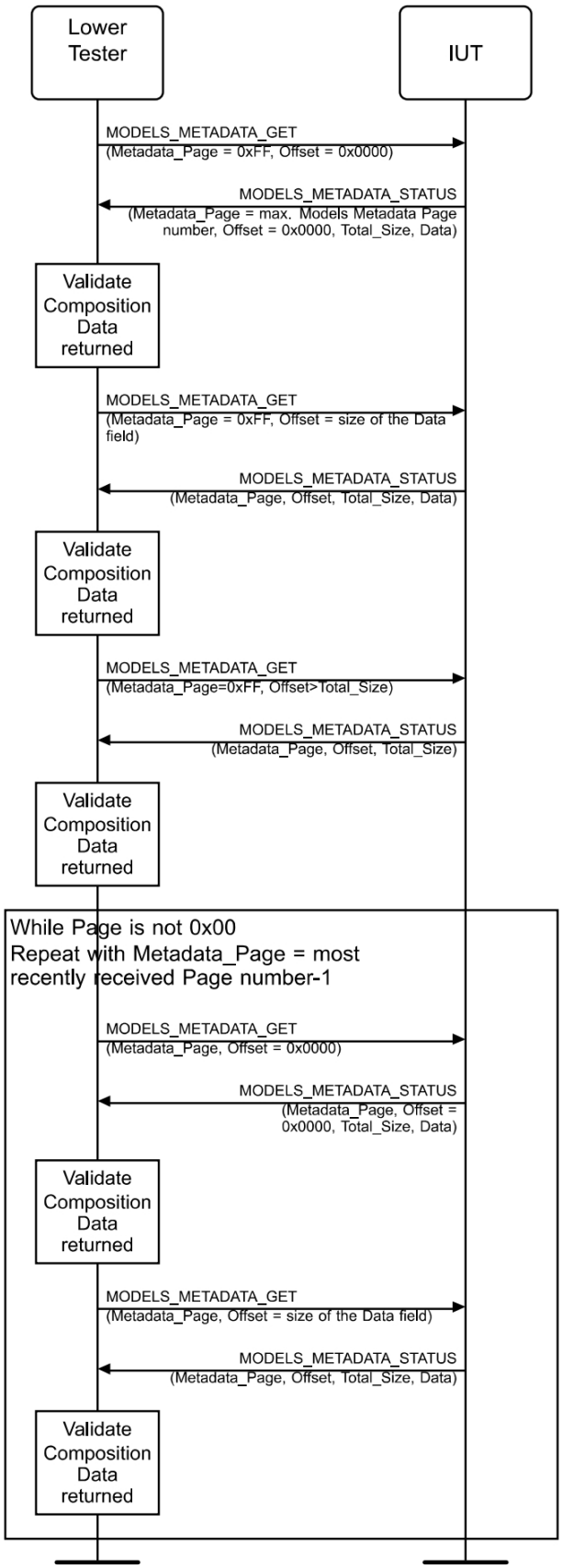


Figure 4.352: Test procedure for MESH/SR/LCD/MMD/BV-01-C MSC

1. The Lower Tester sends a MODELS_METADATA_GET message to the IUT with the Metadata_Page field value set to 0xFF and the Offset field value set to 0x0000.
2. The IUT responds with a MODELS_METADATA_STATUS message with the Metadata_Page field set to the maximum Models Metadata Page number defined in IXIT [6], the Offset field set to 0x0000, the Total_Size field set to the total size of that Models Metadata Page, and the Data field set to the a portion of the information contained by that Models Metadata Page starting from 0x0000.
3. The Lower Tester sends a MODELS_METADATA_GET message to the IUT with the Metadata_Page field value set to 0xFF and the Offset field value set to the size of the Data field from Step 2.
4. The IUT responds with a MODELS_METADATA_STATUS message with the Metadata_Page field set to the maximum Models Metadata Page number defined in IXIT [6], the Offset field set to the size of the Data field from Step 2, the Total_Size field set to the total size of that Models Metadata Page, and the Data field set to a portion of the information contained by that Models Metadata Page starting from the size of the Data field from Step 2.
5. The Lower Tester sends a MODELS_METADATA_GET message to the IUT with the Metadata_Page field value set to 0xFF and the Offset field set to a value greater than the Total_Size field value received in Step 4.
6. The IUT responds with a MODELS_METADATA_STATUS message with the Metadata_Page field set to the maximum Models Metadata Page number defined in IXIT [6], the Offset field set to the size of the Data field from Step 4, the Total_Size field set to the total size of that Models Metadata Page, and with no Data field.
7. While the page number received in Steps 2 and 4 is not 0x00, repeat Steps 1–4, setting the Metadata_Page field in Steps 1 and 3 to the most recently received page number minus 1.
8. The Lower Tester parses the Models Metadata returned by the IUT in all iterations through Step 2 and checks that all fields are valid and consistent.

- Expected Outcome

  Pass verdict

  In all iterations through Step 2, the IUT sends complete and correctly formatted Models Metadata. All fields from the Models Metadata information contain valid values.

  When Metadata_Page is 0x00, the order of the list of metadata is the same as in the corresponding Composition Data Page.

  The Items_Number_SIG_Models and the Items_Number_Vendor_Models field values are less or equal than the corresponding elements in Composition Data Page.

  All configuration messages are encrypted using the device key of the IUT.

MESH/CL/LCD/MMD/BV-01-C [Send MODELS_METADATA_GET Messages]

- Test Purpose

  Verify that an IUT node supporting the Large Composition Data Client Model can send MODELS_METADATA_GET messages to a Large Composition Data Server.

- Reference

[10] 4.4.17, 4.4.18

- Initial Condition
  - The IUT supports the Large Composition Data Client Model.
  - The Lower Tester supports the Large Composition Data Server Model on the Primary Element.
  - The IUT and the Lower Tester share previously established network security credentials. The IUT has access to the device key of the Lower Tester defined in the IXIT [6] parameters.
- Test Procedure

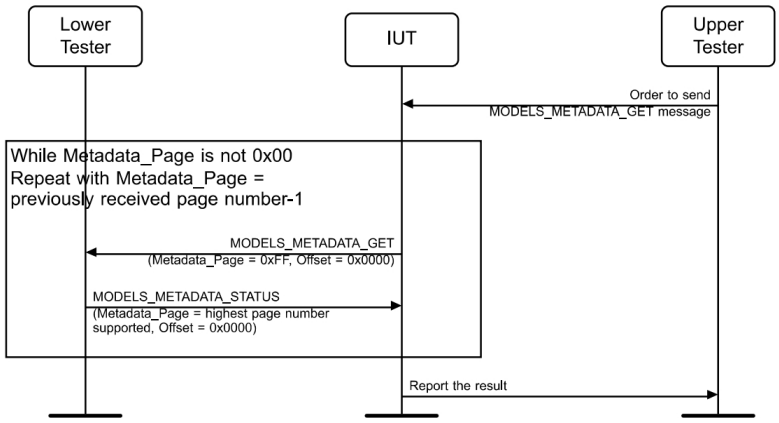


Figure 4.353: Test procedure for MESH/CL/LCD/MMD/BV-01-C MSC

1. The Upper Tester orders the IUT to obtain the device Models Metadata of the Lower Tester using the Large Composition Data Client Model.
2. The IUT sends a MODELS_METADATA_GET message with the highest supported page number or 0xFF and the Offset field set to 0x0000.
3. The Lower Tester responds with a MODELS_METADATA_STATUS message with the Metadata_Page field value set to set to the highest page number supported by the Lower Tester and the Offset field set to 0x0000.
4. If the page number sent in Step 3 is greater than 0x00, then repeat Steps 2 and 3, expecting the IUT to always request a page number equal to the one previously received, decremented by 1, until Page 0 has been sent by the Lower Tester.

- Expected Outcome

  Pass verdict

  In Step 2, the IUT sends a properly formatted MODELS_METADATA_GET message to the Lower Tester.

  If applicable, the IUT requests a page number equal to the previously received page number minus 1.

  All configuration messages are encrypted using the device key of the Lower Tester.

4.27 Solicitation PDU RPL Configuration Models

The tests verify the functionality of the Solicitation PDU RPL Configuration Server Model and the SAR

Configuration Client Model.

MESH/SR/SRPL/BV-01-C [RPL Item Clear Behavior]

- Test Purpose

  Verify that a Solicitation PDU RPL Configuration Server IUT can process SOLICITATION_PDU_RPL_ITEM_CLEAR and SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED messages from a Solicitation PDU RPL Configuration Client.

- Reference

  [10] 4.3.7, 4.4.13

- Initial Condition
  - The IUT supports the Solicitation PDU RPL Configuration Server Model on the Primary Element.
  - The Lower Tester supports the Solicitation PDU RPL Configuration Client Model.
  - The IUT has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The Lower Tester has access to an Application Key of the IUT defined in the IXIT [6] parameters.

- Test Procedure

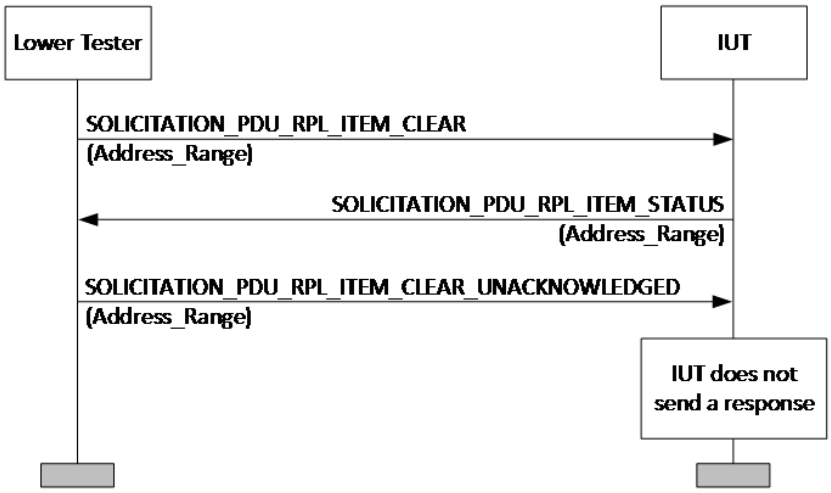


Figure 4.354: Test procedure for MESH/SR/SRPL/BV-01-C MSC

1. The Lower Tester sends a SOLICITATION_PDU_RPL_ITEM_CLEAR message to the IUT with a valid Address_Range field value.
2. The IUT responds with a SOLICITATION_PDU_RPL_ITEM_STATUS message with the Address_Range field set to the same value as in the incoming message.
3. The Lower Tester sends a SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED message to the IUT with a valid Address_Range field value.

4. The IUT processes the message and does not send a response.

- Expected Outcome

  Pass verdict

  The IUT correctly processes all messages from the Lower Tester.

  All messages are encrypted with the application key configured for the model.

MESH/CL/SRPL/BV-01-C [RPL Item Clear Procedure]

- Test Purpose

  Verify that a Solicitation PDU RPL Configuration Client IUT can send SOLICITATION_PDU_RPL_ITEM_CLEAR and SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED messages.

- Reference

  [10] 4.3.7, 4.4.14

- Initial Condition

  - The Lower Tester supports the Solicitation PDU RPL Configuration Server Model on the Primary Element.
  - The IUT supports the Solicitation PDU RPL Configuration Client Model.
  - The Lower Tester has been provisioned, and the IUT and the Lower Tester share previously established network security credentials. The IUT has access to the Application Key of the Lower Tester defined in the IXIT [6] parameters.

- Test Procedure

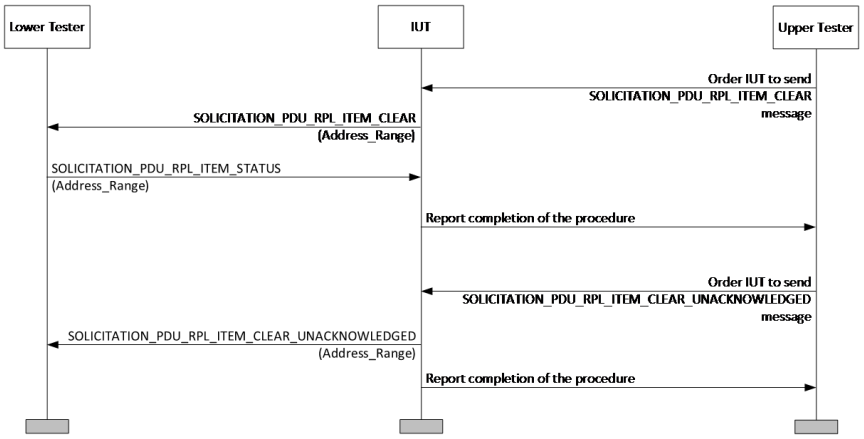


Figure 4.355: Test procedure for MESH/CL/SRPL/BV-01-C MSC

1. The Upper Tester orders the IUT to send a SOLICITATION_PDU_RPL_ITEM_CLEAR message to the Lower Tester.
2. The IUT sends a SOLICITATION_PDU_RPL_ITEM_CLEAR message with the Address_Range field set to a valid value.
3. The Lower Tester responds with a SOLICITATION_PDU_RPL_ITEM_STATUS message with the Address_Range field set to the same value as in the incoming message.
4. The IUT reports the completion of the procedure to the Upper Tester.
5. The Upper Tester orders the IUT to send a SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED message to the Lower Tester.
6. The IUT sends a SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED message with the Address_Range field set to a valid value.
7. The IUT reports the completion of the procedure to the Upper Tester.

- Expected Outcome

  Pass verdict

  The IUT sends correctly formatted SOLICITATION_PDU_RPL_ITEM_CLEAR and SOLICITATION_PDU_RPL_ITEM_CLEAR_UNACKNOWLEDGED messages.

  All messages are encrypted with the application key configured for the model.

4.28 Certificate-Based Provisioning

4.28.1 Get provisioned using X.509 Certificates retrieved out-of-band

- Test Purpose

  Verify that the IUT provides a valid device certificate over an out-of-band transport and can be provisioned using the public key contained by the certificate.

- Reference

  [10] 5.5

- Initial Condition
  - The Device UUID of the IUT and the certificate chain that was used to sign the device certificate are available in the IXIT.
- Test Procedure

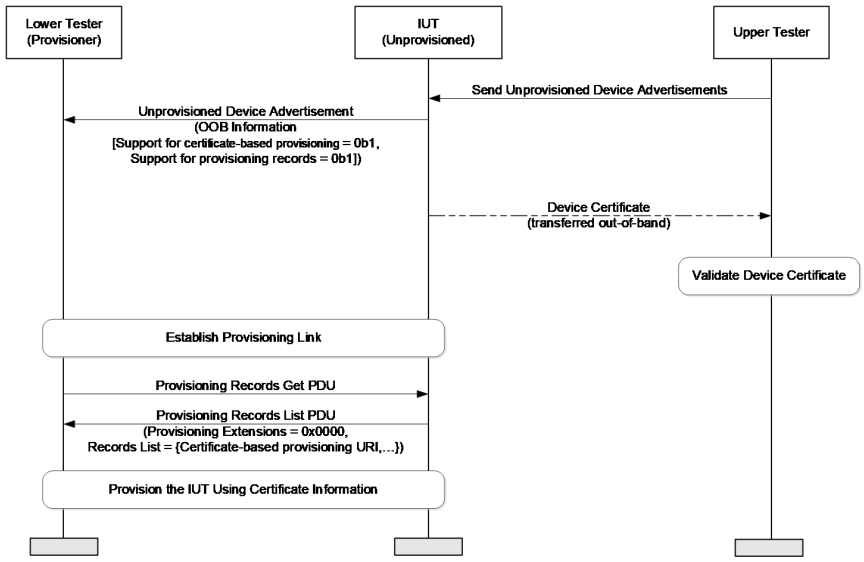


Figure 4.356: Test Procedure — Get provisioned using X.509 Certificates retrieved out-of-band MSC

1. The Upper Tester orders the IUT to send unprovisioned device advertisements as indicated in the Step 1 column of Table 4.158.
2. The Lower Tester verifies that the OOB Information field from the unprovisioned device advertisements has the Support for certificate-based provisioning and Support for provisioning records bits set.
3. The Upper Tester loads the device certificate from a local file.
4. The Upper Tester validates the format of the device certificate.
5. If intermediate certificates are available locally as indicated in the IXIT, then the Upper Tester loads all of them together with the root certificate.
6. The Upper Tester validates the signature of the device certificate using the certificate chain information provided in the IXIT.
7. The Lower Tester establishes a provisioning link with the IUT over the advertised provisioning bearer.
8. The Lower Tester sends a Provisioning Records Get PDU and the IUT responds with a Provisioning Records List PDU, with the Provisioning Extensions field set to 0x0000 and the Records List field containing the Certificate-based Provisioning URI record.
9. The Lower Tester provisions the IUT using the data from the OOB device certificate.
10. The Lower Tester sends a Config Composition Data Get message to the IUT with the Page field value set to 0x00.
11. The Lower Tester expects the IUT to respond with a Config Composition Data Status message with the Page field set to 0 and the Data field set to the information contained by the Composition Data Page.

- Test Case Configuration

TCID	Step 1
MESH/NODE/CBP/BV-01-C [Get Provisioned using a X.509 Certificate over PB-ADV]	Unprovisioned Device Beacons
MESH/NODE/CBP/BV-02-C [Get Provisioned using a X.509 Certificate over PB-GATT]	Connectable Mesh Provisioning Service Advertisements

Table 4.158: Unprovisioned Device Advertisement Selection

- Expected Outcome

  Pass verdict

  The IUT sends unprovisioned device advertisements with the correct OOB Information field value.
  The IUT provides a valid device certificate with a valid signature (verified by the certificate chain).
  The IUT sends the correct list of provisioning records over the provisioning bearer, containing the Certificate-based Provisioning URI record. The URI uses an HTTPS scheme.
  The IUT is successfully provisioned using the information from the OOB device certificate.
  If the certificate's subject field contains the CID or PID fields, then both are present and both are equal to the corresponding values in the Device Composition Data Page 0.
  If the certificate policies extension is present, it is marked as critical.
  The public key of the IUT is not the same as the one from the sample data.

4.28.2 Send X.509 Certificates over the Provisioning Bearer

- Test Purpose

  Verify that the IUT sends a valid OOB device certificate over the provisioning bearer and can be provisioned using the public key contained by the certificate.

- Reference

  [10] 5.4.1.11, 5.4.1.12

- Initial Condition

  - The Device UUID of the IUT and the certificate chain that was used to sign the device certificate are available in the IXIT.

- Test Procedure

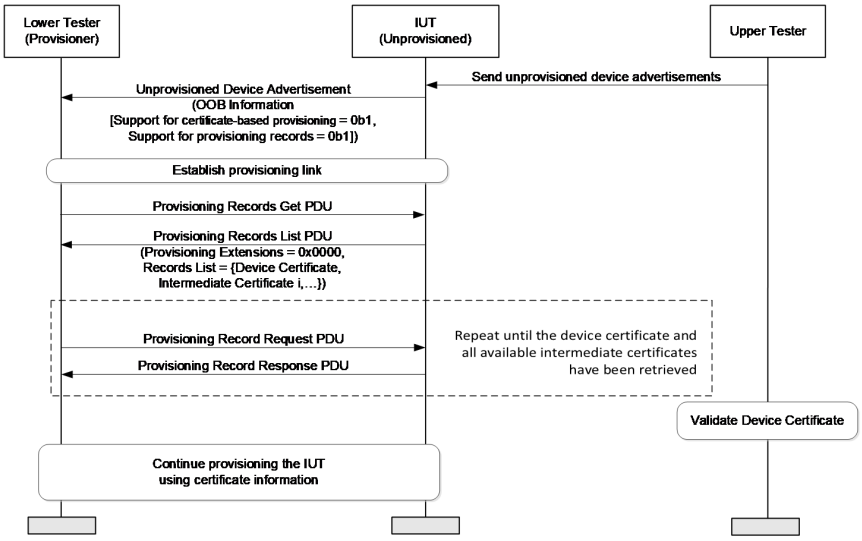


Figure 4.357: Test Procedure - Send X.509 Certificates over the Provisioning Bearer MSC

1. The Upper Tester orders the IUT to send unprovisioned device advertisements as indicated in the Step 1 column of Table 4.159.
2. The Lower Tester verifies that the OOB Information field from the IUT's unprovisioned device advertisements has the Support for certificate-based provisioning and the Support for provisioning records bits set.
3. The Lower Tester creates a provisioning link over the advertised provisioning bearer.
4. The Lower Tester sends a Provisioning Records Get PDU and the IUT responds with a Provisioning Records List PDU, with the Provisioning Extensions field set to 0x0000, the Records List field containing the Device Certificate record, and optionally a list of consecutive Intermediate Certificate records, starting with the Intermediate Certificate 1 record.
5. The Lower Tester sends a Provisioning Record Request PDU with the Record ID field set to "Device certificate".
6. The IUT responds with a Provisioning Record Response PDU with the Status field set to Success and the Record ID and Fragment Offset fields set to the values of the corresponding fields from the request.
7. Repeat Steps 4 and 5 until the complete device certificate has been retrieved. The Total Length field has a constant and correct value across all fragments, and the length of the Data field is not greater than the requested size (the value of the Fragment Maximum Size field).
8. Repeat Steps 4 and 5 as necessary with the Record ID set to increasing values of the certificate index in order to retrieve all the available intermediate certificates, if any. Verify that each intermediate certificate retrieved in this manner correctly signs the certificate retrieved in the previous iteration (or that it signs the device certificate, if this is the first intermediate certificate retrieved over the provisioning bearer).
9. If further intermediate certificates are available locally as indicated in the IXIT, then the Upper Tester loads all of them together with the root certificate and validates the complete certificate chain.
10. The Lower Tester continues the provisioning procedure (with a Provisioning Invite PDU) and successfully provisions the IUT using the data from the OOB device certificate.

11. The Lower Tester sends a Config Composition Data Get message to the IUT with the Page field value set to 0x00.
12. The Lower Tester expects the IUT to respond with a Config Composition Data Status message with the Page field set to 0 and the Data field set to the information contained by the Composition Data Page.

- Test Case Configuration

TCID	Step 1
MESH/NODE/CBP/BV-03-C [Send X.509 Certificates over PB-ADV]	Unprovisioned Device Beacons
MESH/NODE/CBP/BV-04-C [Send X.509 Certificates over PB-GATT]	Connectable Mesh Provisioning Service Advertisements

Table 4.159: Unprovisioned Device Advertisement Selection

- Expected Outcome

  Pass verdict

  The IUT sends unprovisioned device advertisements with the correct information and field values.
  The IUT sends valid device and intermediate certificates over the provisioning bearer and the entire certificate chain is correctly signed up to the root certificate.
  The list of intermediate certificates sent in the Provisioning Records List PDU matches the list of intermediate certificates retrieved over the provisioning bearer.
  The IUT is successfully provisioned using the information from the OOB device certificate.
  If the certificate's subject field contains the CID or PID fields, then both are present and both are equal to the corresponding values in the Device Composition Data Page 0.
  If the certificate policies extension is present, it is marked as critical.
  The public key of the IUT is not the same as the one from the sample data.

MESH/NODE/CBP/BI-01-C [Handle Invalid Provisioning Record Request PDUs]

- Test Purpose

  Verify that the IUT correctly handles invalid Provisioning Record Request PDUs.

- Reference

  [10] 5.4.1.11, 5.4.1.12

- Initial Condition

  - The Device UUID of the IUT is available in the IXIT.

- Test Procedure

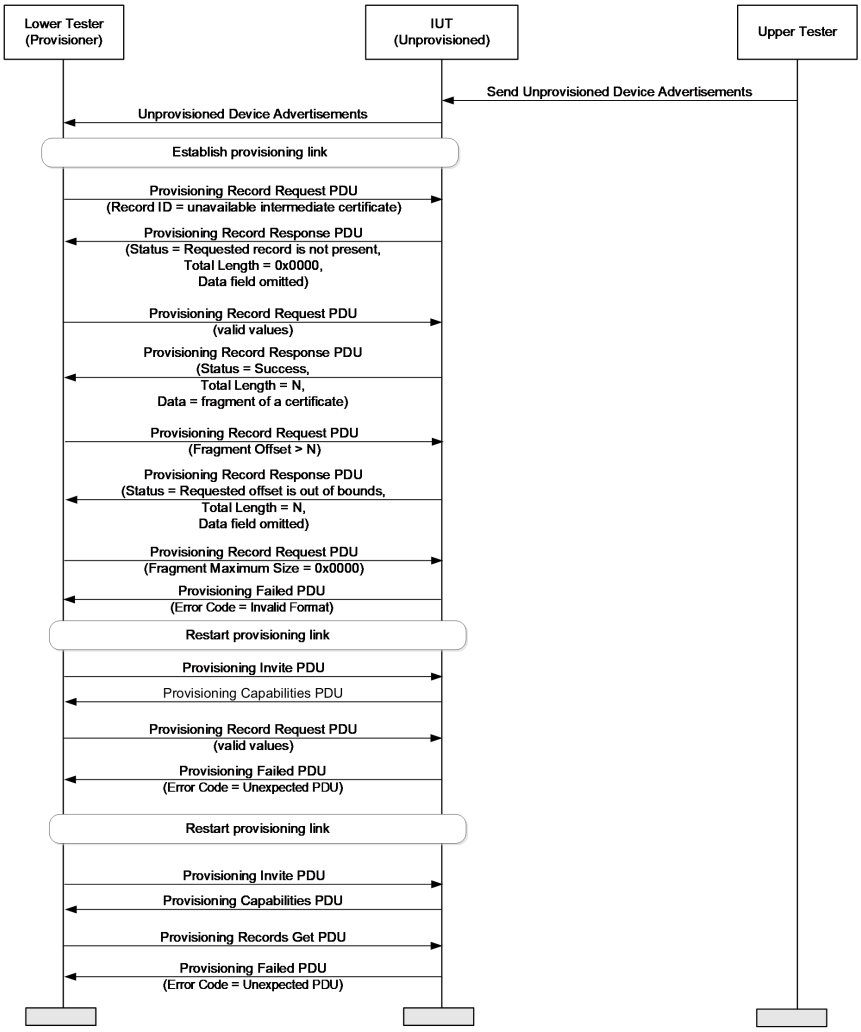


Figure 4.358: Test Procedure for MESH/NODE/CBP/BI-01-C MSC

1. The Upper Tester orders the IUT to send unprovisioned device advertisements for any supported provisioning bearer, and the Lower Tester creates a provisioning link.
2. If the number of intermediate certificates located on the IUT, as determined from the IXIT, is equal to the maximum allowed (15), then skip this step; otherwise, the Lower Tester sends a Provisioning Record Request PDU with the Record ID field set to a value that indicates a certificate index greater than the number of intermediate certificates on the IUT, and the IUT responds with a Provisioning Record Response PDU with the Status field set to "Requested Record Is Not Present", the Total Length field set to 0x0000, the Data field omitted, and the Record ID and Fragment Offset fields set to the values of the corresponding fields from the request.
3. The Lower Tester sends a Provisioning Record Request PDU with valid values and the IUT responds with a fragment of the requested certificate. Then the Lower Tester sends another Provisioning Record Request PDU for the same certificate, but with the Fragment Offset field set to a value greater than the total length of the certificate (as indicated by the IUT in the previous response). The IUT now responds with a Provisioning Record Response PDU with the Status field set to "Requested Offset Is Out Of

Bounds", the Total Length field set as in the previous response, and the Data field omitted.

4. The Lower Tester sends a Provisioning Record Request PDU with the Fragment Maximum Size field set to 0x0000 and the IUT responds with a Provisioning Failed PDU with the Error Code field set to Invalid Format.
5. The Lower Tester recreates a new provisioning link, then sends a Provisioning Invite PDU and the IUT responds with a Provisioning Capabilities PDU. Then the Lower Tester sends a Provisioning Record Request PDU with valid parameters, and the IUT responds with a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.
6. The Lower Tester recreates a new provisioning link, then sends a Provisioning Invite PDU and the IUT responds with a Provisioning Capabilities PDU. Then the Lower Tester sends a Provisioning Records Get PDU, and the IUT responds with a Provisioning Failed PDU with the Error Code field set to Unexpected PDU.

- Expected Outcome

  Pass verdict

  The IUT rejects the invalid requests and only sends an error response when appropriate.

4.28.3 Retrieve X.509 Certificates over HTTPS and Provision Device

- Test Purpose

  Verify that the IUT can obtain the URI string, can retrieve and validate an OOB device certificate over HTTPS, and can use the certificate information to provision the unprovisioned device.

- Reference

  [10] 5.5

- Initial Condition

  - The Device UUID of the Lower Tester and the certificate authority information are available on the IUT side.

- Test Procedure

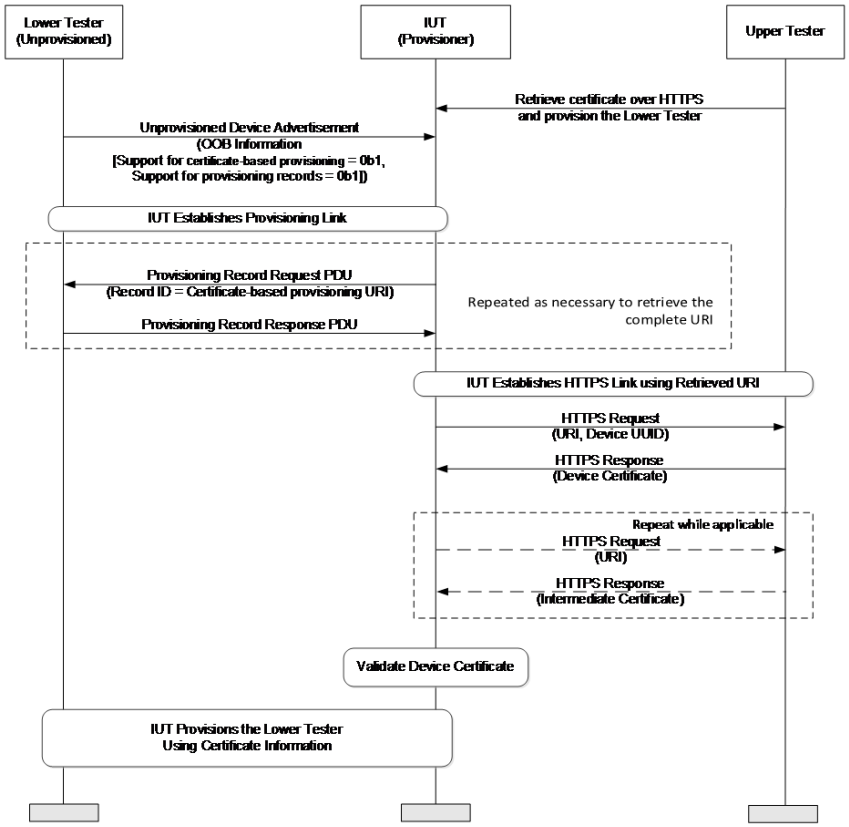


Figure 4.359: Test Procedure — Retrieve X.509 Certificates over HTTPS and Provision Device MSC

1. The Lower Tester advertises unprovisioned device advertisements as indicated in the Step 1 column of Table 4.160, with the OOB Information field containing the Support for certificate-based provisioning and Support for provisioning records bits set to 1.
2. The Upper Tester orders the IUT to retrieve the OOB device certificate over HTTPS and to provision the Lower Tester.
3. The IUT creates a provisioning link over the advertised provisioning bearer, then retrieves the URI contained in the Certificate-based provisioning URI record on the Lower Tester, by sending as many Provisioning Record Request PDU messages with valid fields as needed.
4. The IUT then creates an HTTPS link with the Upper Tester using the URI obtained from the previous step and sends an HTTPS Request containing the Device UUID of the Lower Tester and properly formatted fields.
5. The Upper Tester sends a device certificate in response to the IUT, with valid format and signature.
6. If intermediate device certificates are available over HTTPS, the IUT issues subsequent HTTPS Requests and the Upper Tester responds with the intermediate certificates.
7. The IUT uses the complete certificate chain information to validate and accept the device certificate.
8. The IUT completes the provisioning procedure successfully.

- Test Case Configuration

TCID	Step 1
MESH/PVNR/CBP/BV-01-C [Retrieve X.509 Certificates over HTTPS and Provision Device over PB-ADV]	Unprovisioned Device Beacons
MESH/PVNR/CBP/BV-02-C [Retrieve X.509 Certificates over HTTPS and Provision Device over PB-GATT]	Connectable Mesh Provisioning Service Advertisements

Table 4.160: Unprovisioned Device Advertisement Selection

- Expected Outcome

  Pass verdict

  The IUT retrieves the URI string from the Lower Tester's Certificate-based provisioning URI record.
  The IUT sends a properly formatted HTTPS Request to retrieve the device certificate and as many additional requests as needed to retrieve the intermediate certificates.
  The IUT validates and accepts the device certificate and successfully provisions the Lower Tester over the provisioning bearer.
  The public key of the IUT is not the same as the one from the sample data.

4.28.4 Retrieve X.509 Certificates over the Provisioning Bearer

- Test Purpose

  Verify that the IUT can retrieve a valid OOB device certificate over the provisioning bearer and use the public key contained by the certificate to provision the device.

- Reference

  [10] 5.4.1.11, 5.4.1.12

- Initial Condition

  - The Device UUID of the Lower Tester and the certificate authority information are available on the IUT side.

- Test Procedure

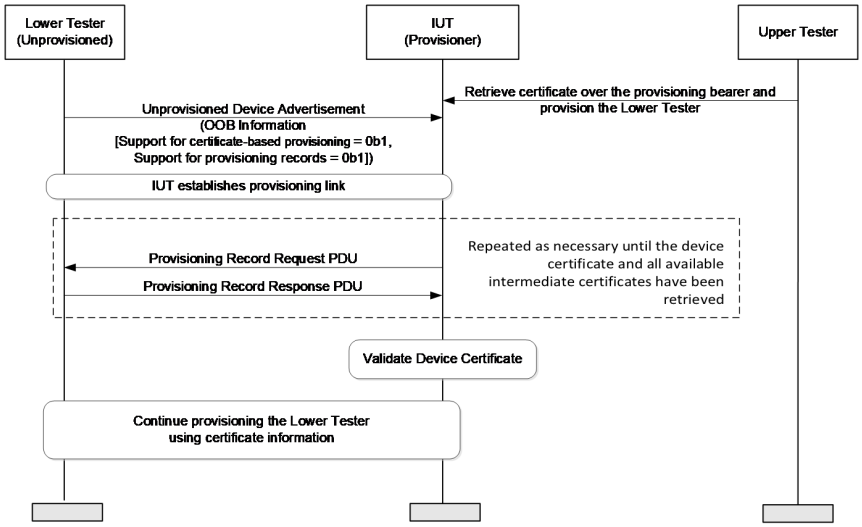


Figure 4.360: Test Procedure - Retrieve X.509 Certificates over the Provisioning Bearer MSC

1. The Lower Tester sends unprovisioned device advertisements as indicated in Table 4.161 with the OOB Information field containing the Support for certificate-based provisioning and the Support for provisioning records bits set to 1.
2. The Upper Tester orders the IUT to retrieve the OOB device certificate over the provisioning bearer under test, using the Device UUID of the Lower Tester.
3. The Upper Tester expects the IUT to create a provisioning link with the Lower Tester and to begin downloading the certificate chain by sending Provisioning Record Request PDUs.
4. The Lower Tester replies to each request with a Provisioning Record Response PDU with valid fields. The Lower Tester sends at least one intermediate certificate.
5. After the device certificate has been validated by the IUT through the entire certificate chain, the Lower Tester and the IUT complete the provisioning procedure.

- Test Case Configuration

TCID	Step 1
MESH/PVNR/CBP/BV-03-C [Retrieve X.509 Certificates over PB-ADV]	Unprovisioned Device Beacons
MESH/PVNR/CBP/BV-04-C [Retrieve X.509 Certificates over PB-GATT]	Connectable Mesh Provisioning Service Advertisements

Table 4.161: Unprovisioned Device Advertisement Selection

- Expected Outcome

  Pass verdict

  The IUT retrieves, reassembles, and validates the certificate chain.
  The IUT successfully provisions the Lower Tester with the public key contained in the device certificate.
  The public key of the IUT is not the same as the one from the sample data.

MESH/PVNR/CBP/BI-01-C [Reject Invalid X.509 Certificates]

- Test Purpose

  Verify that the IUT rejects invalid OOB device certificates.

- Reference

  [10] 5.5

- Initial Condition

  - The Device UUID of the Lower Tester and the certificate authority information are available on the IUT side.

- Test Procedure

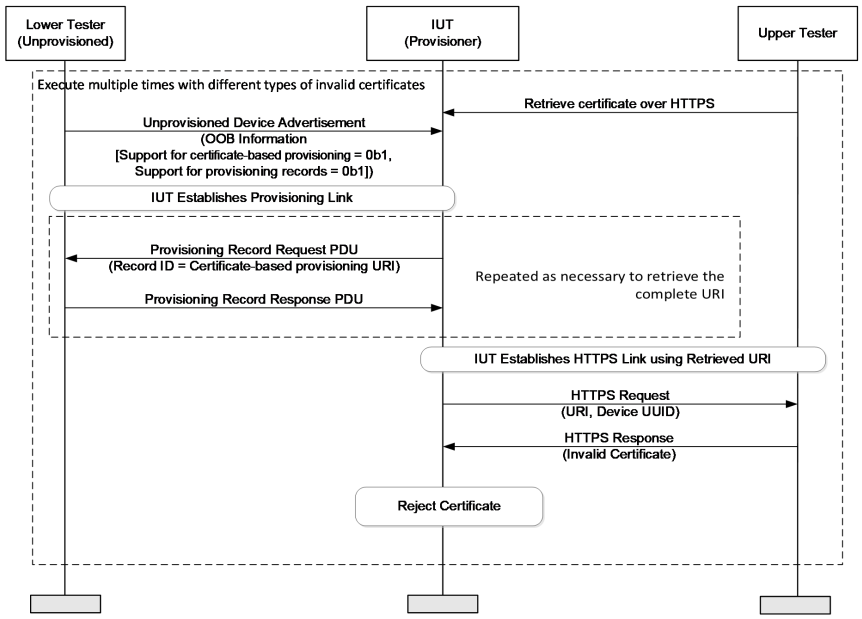


Figure 4.361: Test Procedure for MESH/PVNR/CBP/BI-01-C MSC

1. The Upper Tester orders the IUT to retrieve the OOB device certificate over HTTPS and expects the IUT to establish a provisioning link over the advertised bearer, to retrieve the URI from the Certificate-based provisioning URI record, then to establish an HTTPS link and send an HTTPS Request containing the Device UUID of the Lower Tester and properly formatted fields.
2. The Upper Tester sends a device certificate in response to the IUT, with a valid format and with invalid signature.
3. The Upper Tester expects the IUT to reject the device certificate sent in the previous step.
4. For each round in Table 4.162, repeat Steps 1–3 using, in Step 2, a device certificate with valid format except for the fields or values described by the Field and Value/Condition columns, and with valid signature.
5. Repeat Steps 1–3 using, in Step 2, a device certificate with valid format and with valid signature, but which has expired.

6. Repeat Steps 1–3 using, in Step 2, a device certificate that is a root certificate, with valid format and with valid signature.
7. Repeat Steps 1 and 2 using, in Step 2, a device certificate with valid format and with valid signature, signed by an intermediate certificate.
8. The IUT sends subsequent HTTPS Requests and the Upper Tester responds with an intermediate certificate which is a root certificate.
9. The Upper Tester expects the IUT to reject the root certificate sent in Step 8.
10. The Upper Tester orders the IUT to retrieve the OOB device certificate over HTTPS and expects the IUT to establish a provisioning link over the advertised bearer and to retrieve the URI from the Certificate-based provisioning URI record, but in this step the Lower Tester sends a URI with an HTTP schema (not HTTPS). If the IUT proceeds to establish an HTTP link, the test ends with a Fail verdict.

- Expected Outcome

  Pass verdict

  The IUT verifies and rejects the invalid device certificate or intermediate certificates.

  The IUT does not retrieve the device certificate over HTTP.

Round	Field	Value/Condition	Reference
1	signatureAlgorithm	different than "ecdsa-with-SHA256"	[10] 5.5.1.1.2
2	version	different than "v3"	[10] 5.5.1.1.4.1
3	validity	*notBefore* field set to 99991231235959Z	[10] 5.5.1.1.4.5
4	subjectPublicKeyInfo	*algorithm* field different than "id-ecPublicKey"	[10] 5.5.1.1.4.7
5	Key usage extension	field is missing	[10] 5.5.1.1.4.12
6	Key usage extension	*keyAgreement* bit cleared and at least one other bit set	[10] 5.5.1.1.4.12
7	Key usage extension	*keyAgreement* bit set and at least one other bit set	[10] 5.5.1.1.4.12
8	Policy mappings extension	random valid value	[10] 5.5.1.1.4.14
9	Subject alternative name extension	random valid value	[10] 5.5.1.1.4.15
10	Basic constraints extension	field is missing	[10] 5.5.1.1.4.18
11	Basic constraints extension	*cA* field set to TRUE	[10] 5.5.1.1.4.18
12	Basic constraints extension	*pathLenConstraint* field set to a valid value	[10] 5.5.1.1.4.18
13	Name constraints extension	random valid value	[10] 5.5.1.1.4.19

Round	Field	Value/Condition	Reference
14	Policy constraints extension	random valid value	[10] 5.5.1.1.4.20
15	Extended key usage extension	random valid value	[10] 5.5.1.1.4.21
16	Inhibit anyPolicy extension	random valid value	[10] 5.5.1.1.4.22
17	subject	CID is present and valid, PID is not present	[10] 5.5.1.1.4.6
18	subject	PID is present and valid, CID is not present	[10] 5.5.1.1.4.6
19	Certificate policies extension	Present but not marked as critical	

Table 4.162: MESH/PVNR/CBP/BI-01-C invalid field rounds

5 Test case mapping

The Test Case Mapping Table (TCMT) maps test cases to specific requirements in the ICS. The IUT is tested in all roles for which support is declared in the ICS document.

The columns for the TCMT are defined as follows:

Item: Contains a logical expression based on specific entries from the associated ICS document. Contains a logical expression (using the operators AND, OR, NOT as needed) based on specific entries from the applicable ICS document(s). The entries are in the form of y/x references, where y corresponds to the table number and x corresponds to the feature number as defined in the ICS document for the Mesh Protocol Specification [4].

If a test case is mandatory within the respective layer, then the y/x reference is omitted.

Feature: A brief, informal description of the feature being tested.

Test Case(s): The applicable test case identifiers, required for Bluetooth Qualification if the corresponding y/x references defined in the Item column are supported. Further details about the function of the TCMT are elaborated in [2].

For the purpose and structure of the ICS/IXIT, refer to [2].

Item	Feature	Test Case(s)
MESH 4/1	PB-ADV - Node	MESH/NODE/PBADV/BV-01-C MESH/NODE/PBADV/BV-02-C MESH/NODE/PBADV/BV-03-C MESH/NODE/PBADV/BV-04-C MESH/NODE/PBADV/BI-01-C MESH/NODE/PBADV/BI-02-C MESH/NODE/PBADV/BI-03-C MESH/NODE/PBADV/BI-04-C
MESH 18/2	PB-ADV - Provisioner	MESH/PVNR/PBADV/BV-01-C MESH/PVNR/PBADV/BV-02-C MESH/PVNR/PBADV/BV-03-C MESH/PVNR/PBADV/BI-01-C

Item	Feature	Test Case(s)
MESH 4/2	PB-GATT - Node	MESH/NODE/MPS/BV-01-C MESH/NODE/MPS/BV-02-C MESH/NODE/MPS/BV-03-C MESH/NODE/MPS/BV-04-C MESH/NODE/MPS/BV-05-C MESH/NODE/MPS/BV-06-C MESH/NODE/MPS/BV-07-C MESH/NODE/MPS/BV-08-C MESH/NODE/MPS/BV-09-C MESH/NODE/MPS/BV-10-C MESH/NODE/MPS/BI-01-C
MESH 18/5	PB-GATT - Provisioner	MESH/PVNR/MPS/BV-01-C MESH/PVNR/MPS/BV-02-C MESH/PVNR/MPS/BV-03-C MESH/PVNR/MPS/BV-04-C MESH/PVNR/MPS/BV-05-C MESH/PVNR/MPS/BV-06-C MESH/PVNR/MPS/BV-08-C MESH/PVNR/MPS/BV-09-C
MESH 4/8 AND MESH 4/9 AND MESH 4/13	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Output OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-01-C
MESH 4/8 AND MESH 4/10 AND MESH 4/13	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Input OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-02-C
MESH 4/8 AND MESH 4/11 AND MESH 4/13	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Static OOB, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-03-C

Item	Feature	Test Case(s)
MESH 4/7 AND MESH 4/9 AND MESH 4/13	Provisioning: Public Key OOB - Node, Provisioning: Authentication Output OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-04-C
MESH 4/7 AND MESH 4/10 AND MESH 4/13	Provisioning: Public Key OOB - Node, Provisioning: Authentication Input OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-05-C
MESH 4/7 AND MESH 4/11 AND MESH 4/13	Provisioning: Public Key OOB - Node, Provisioning: Authentication Static OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-06-C
MESH 4/8 AND MESH 4/12 AND MESH 4/13	Provisioning: Public Key No OOB - Node, Provisioning: Authentication No OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-07-C
MESH 4/7 AND MESH 4/12 AND MESH 4/13	Provisioning: Public Key OOB - Node, Provisioning: Authentication No OOB - Node, Provisioning algorithm: BTM_ECDH_P256_CMAC_AES128_AES_CCM	MESH/NODE/PROV/BV-08-C
MESH 4/8 AND MESH 4/9 AND MESH 4/14	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Output OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-13-C

Item	Feature	Test Case(s)
MESH 4/8 AND MESH 4/10 AND MESH 4/14	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Input OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-14-C
MESH 4/8 AND MESH 4/11 AND MESH 4/14	Provisioning: Public Key No OOB - Node, Provisioning: Authentication Static OOB, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-15-C
MESH 4/7 AND MESH 4/9 AND MESH 4/14	Provisioning: Public Key OOB - Node, Provisioning: Authentication Output OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-16-C
MESH 4/7 AND MESH 4/10 AND MESH 4/14	Provisioning: Public Key OOB - Node, Provisioning: Authentication Input OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-17-C
MESH 4/7 AND MESH 4/11 AND MESH 4/14	Provisioning: Public Key OOB - Node, Provisioning: Authentication Static OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-18-C
MESH 4/8 AND MESH 4/12 AND MESH 4/14	Provisioning: Public Key No OOB - Node, Provisioning: Authentication No OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-19-C

Item	Feature	Test Case(s)
MESH 4/7 AND MESH 4/12 AND MESH 4/14	Provisioning: Public Key OOB - Node, Provisioning: Authentication No OOB - Node, Provisioning algorithm: BTM_ECDH_P256_HMAC_SHA256_AES_CCM	MESH/NODE/PROV/BV-20-C
MESH 4/6	Provisioning Protocol (Provisioning Server)	MESH/NODE/PROV/BV-09-C MESH/NODE/PROV/BV-10-C MESH/NODE/PROV/BV-11-C MESH/NODE/PROV/BI-01-C MESH/NODE/PROV/BI-02-C MESH/NODE/PROV/BI-03-C MESH/NODE/PROV/BV-12-C MESH/NODE/PROV/BI-13-C MESH/NODE/PROV/BI-17-C
MESH 18/4	Provisioning Protocol (Provisioning Client)	MESH/PVNR/PROV/BI-14-C MESH/PVNR/PROV/BI-16-C MESH/PVNR/PROV/BI-18-C
MESH 4/6	Provisioning Protocol (Provisioning Server)	MESH/NODE/PROV/BI-15-C
MESH 18/4	Provisioning Protocol (Provisioning Client)	MESH/PVNR/PROV/BV-09-C MESH/PVNR/PROV/BV-10-C
MESH 18/4 AND MESH 18/12	Provisioning Protocol (Provisioning Client) with BTM_ECDH_P256_CMAC_AES128_AES_CCM algorithm	MESH/PVNR/PROV/BV-01-C MESH/PVNR/PROV/BV-02-C MESH/PVNR/PROV/BV-03-C MESH/PVNR/PROV/BV-04-C MESH/PVNR/PROV/BV-05-C MESH/PVNR/PROV/BV-06-C MESH/PVNR/PROV/BV-07-C MESH/PVNR/PROV/BV-08-C
MESH 18/4 AND MESH 18/13	Provisioning Protocol (Provisioning Client) with BTM_ECDH_P256_HMAC_SHA256_AES_CCM algorithm	MESH/PVNR/PROV/BV-11-C MESH/PVNR/PROV/BV-12-C MESH/PVNR/PROV/BV-13-C MESH/PVNR/PROV/BV-14-C MESH/PVNR/PROV/BV-15-C MESH/PVNR/PROV/BV-16-C MESH/PVNR/PROV/BV-17-C MESH/PVNR/PROV/BV-18-C
MESH 10/1	Secure Network Beacons	MESH/NODE/BCN/SNB/BV-01-C
MESH 10/5	Private Beacon	MESH/NODE/BCN/PRB/BV-01-C MESH/NODE/BCN/PRB/BV-02-C

Item	Feature	Test Case(s)
MESH 5/1	Transmitting and Receiving Secured Network Layer Messages	MESH/NODE/NET/BV-01-C MESH/NODE/NET/BV-02-C MESH/NODE/NET/BV-03-C MESH/NODE/NET/BV-04-C MESH/NODE/NET/BV-05-C MESH/NODE/NET/BV-06-C MESH/NODE/NET/BV-07-C MESH/NODE/NET/BV-08-C MESH/NODE/NET/BV-09-C MESH/NODE/NET/BV-10-C MESH/NODE/NET/BV-14-C MESH/NODE/NET/BI-01-C MESH/NODE/NET/BI-02-C MESH/NODE/NET/BI-03-C
MESH 5/2	Relay feature	MESH/NODE/NET/BV-13-C MESH/NODE/RLY/BV-01-C MESH/NODE/RLY/BI-01-C MESH/NODE/RLY/BI-02-C MESH/NODE/RLY/BV-02-C MESH/NODE/CFG/HBP/BV-03-C MESH/NODE/CFG/HBP/BI-01-C
MESH 6/1	Transmitting and Receiving a Lower Transport PDU	MESH/NODE/TNPT/BV-01-C MESH/NODE/TNPT/BV-02-C MESH/NODE/TNPT/BV-03-C MESH/NODE/TNPT/BV-04-C MESH/NODE/TNPT/BV-05-C MESH/NODE/TNPT/BV-06-C MESH/NODE/TNPT/BV-07-C MESH/NODE/TNPT/BV-08-C MESH/NODE/TNPT/BV-09-C MESH/NODE/TNPT/BV-10-C MESH/NODE/TNPT/BV-11-C MESH/NODE/TNPT/BV-22-C MESH/NODE/TNPT/BI-01-C MESH/NODE/TNPT/BI-02-C MESH/NODE/TNPT/BI-03-C
MESH 6/1 AND MESH 0/1	Transmitting and Receiving a Lower Transport PDU - Mesh 1.0 only	MESH/NODE/TNPT/BV-12-C
MESH 9/1	Message Replay Protection	MESH/NODE/TNPT/BV-13-C

Item	Feature	Test Case(s)
MESH 10/3	IV Update Procedure	MESH/NODE/IVU/BV-01-C MESH/NODE/IVU/BV-02-C MESH/NODE/IVU/BV-04-C MESH/NODE/IVU/BV-05-C MESH/NODE/IVU/BI-01-C MESH/NODE/IVU/BI-02-C MESH/NODE/IVU/BI-04-C MESH/NODE/IVU/BI-05-C MESH/NODE/IVU/BI-06-C
MESH 10/4	IV Index Recovery Procedure	MESH/NODE/IVU/BV-03-C
MESH 10/3 AND MESH 10/5	IV Update Procedure, Private Beacon	MESH/NODE/IVU/BV-06-C MESH/NODE/IVU/BV-07-C MESH/NODE/IVU/BV-09-C MESH/NODE/IVU/BV-10-C MESH/NODE/IVU/BI-07-C MESH/NODE/IVU/BI-08-C MESH/NODE/IVU/BI-09-C MESH/NODE/IVU/BI-10-C MESH/NODE/IVU/BI-11-C
MESH 10/4 AND MESH 10/5	IV Index Recovery Procedure, Private Beacon	MESH/NODE/IVU/BV-08-C
MESH 10/2	Key Refresh Procedure	MESH/NODE/KR/BV-01-C MESH/NODE/KR/BV-02-C MESH/NODE/KR/BV-03-C MESH/NODE/KR/BI-01-C MESH/NODE/KR/BI-02-C MESH/NODE/KR/BI-03-C MESH/NODE/KR/BI-04-C MESH/NODE/KR/BI-05-C
MESH 10/2 AND MESH 10/5	Key Refresh Procedure, Private Beacon	MESH/NODE/KR/BV-04-C MESH/NODE/KR/BV-05-C MESH/NODE/KR/BI-06-C

Item	Feature	Test Case(s)
MESH 13/2	Mesh Proxy Service	MESH/SR/MPXS/BV-01-C MESH/SR/MPXS/BV-02-C MESH/SR/MPXS/BV-03-C MESH/SR/MPXS/BV-04-C MESH/SR/MPXS/BV-05-C MESH/SR/MPXS/BV-06-C MESH/SR/MPXS/BV-07-C MESH/SR/MPXS/BV-09-C MESH/NODE/CFG/HBP/BV-07-C MESH/NODE/CFG/GPXY/BV-02-C MESH/NODE/NET/BV-11-C
MESH 11/1	Configuration Server Model	MESH/NODE/CFG/SNBP/BV-01-C MESH/NODE/CFG/SNBP/BI-01-C MESH/NODE/CFG/COMP/BV-01-C MESH/NODE/CFG/DTTL/BV-01-C MESH/NODE/CFG/DTTL/BI-01-C MESH/NODE/CFG/MP/BV-01-C MESH/NODE/CFG/MP/BV-02-C MESH/NODE/CFG/MP/BV-03-C MESH/NODE/CFG/MP/BI-01-C MESH/NODE/CFG/MP/BI-02-C MESH/NODE/CFG/MP/BI-03-C MESH/NODE/CFG/SL/BV-01-C MESH/NODE/CFG/SL/BV-02-C MESH/NODE/CFG/SL/BV-03-C MESH/NODE/CFG/SL/BI-01-C MESH/NODE/CFG/SL/BI-02-C MESH/NODE/CFG/SL/BI-03-C MESH/NODE/CFG/SL/BI-04-C MESH/NODE/CFG/SL/BI-05-C MESH/NODE/CFG/SL/BI-06-C MESH/NODE/CFG/SL/BI-07-C MESH/NODE/CFG/SL/BI-08-C MESH/NODE/CFG/SL/BI-09-C MESH/NODE/CFG/SL/BI-10-C

Item	Feature	Test Case(s)
MESH 11/1	Configuration Server Model	MESH/NODE/CFG/NKL/BV-01-C MESH/NODE/CFG/NKL/BV-02-C MESH/NODE/CFG/NKL/BV-04-C MESH/NODE/CFG/NKL/BI-01-C MESH/NODE/CFG/NKL/BI-02-C MESH/NODE/CFG/NKL/BI-03-C MESH/NODE/CFG/AKL/BV-01-C MESH/NODE/CFG/AKL/BV-02-C MESH/NODE/CFG/AKL/BV-03-C MESH/NODE/CFG/AKL/BI-01-C MESH/NODE/CFG/AKL/BI-02-C MESH/NODE/CFG/AKL/BI-03-C MESH/NODE/CFG/AKL/BI-04-C MESH/NODE/CFG/MAKL/BV-01-C MESH/NODE/CFG/MAKL/BV-02-C MESH/NODE/CFG/MAKL/BI-01-C MESH/NODE/CFG/MAKL/BI-02-C MESH/NODE/CFG/MAKL/BI-03-C MESH/NODE/CFG/MAKL/BI-04-C MESH/NODE/CFG/RST/BV-01-C MESH/NODE/CFG/GPXY/BV-01-C MESH/NODE/CFG/GPXY/BI-01-C MESH/NODE/CFG/NTX/BV-01-C MESH/NODE/CFG/CFGF/BV-01-C MESH/NODE/CFG/CFGF/BI-01-C MESH/NODE/CFG/CFGR/BV-01-C MESH/NODE/CFG/CFGR/BI-01-C MESH/NODE/CFG/HBP/BV-01-C MESH/NODE/CFG/HBP/BV-02-C MESH/NODE/CFG/HBP/BI-02-C MESH/NODE/CFG/HBP/BV-06-C MESH/NODE/CFG/HBS/BV-01-C MESH/NODE/CFG/HBS/BV-02-C MESH/NODE/CFG/HBS/BV-03-C MESH/NODE/CFG/HBS/BV-04-C MESH/NODE/CFG/HBS/BI-01-C
MESH 11/1 AND MESH 5/4	Configuration Server Model Multiple Subnet Support	MESH/NODE/CFG/NKL/BV-03-C

Item	Feature	Test Case(s)
MESH 11/3	Configuration Client Model	MESH/CFGCL/KR/BV-01-C MESH/CFGCL/KR/BV-02-C MESH/CFGCL/KR/BV-03-C MESH/CFGCL/KR/BV-04-C MESH/CFGCL/CFG/SNBP/BV-01-C MESH/CFGCL/CFG/COMP/BV-01-C MESH/CFGCL/CFG/DTTL/BV-01-C MESH/CFGCL/CFG/SL/BV-01-C MESH/CFGCL/CFG/SL/BV-02-C MESH/CFGCL/CFG/NKL/BV-01-C MESH/CFGCL/CFG/NKL/BV-02-C MESH/CFGCL/CFG/NKL/BV-03-C MESH/CFGCL/CFG/AKL/BV-01-C MESH/CFGCL/CFG/AKL/BV-02-C MESH/CFGCL/CFG/AKL/BV-03-C MESH/CFGCL/CFG/AKL/BV-04-C MESH/CFGCL/CFG/MAKL/BV-01-C MESH/CFGCL/CFG/MAKL/BV-02-C MESH/CFGCL/CFG/MAKL/BV-03-C MESH/CFGCL/CFG/MAKL/BV-04-C MESH/CFGCL/CFG/NID/BV-01-C MESH/CFGCL/CFG/RST/BV-01-C MESH/CFGCL/CFG/NTX/BV-01-C MESH/CFGCL/CFG/LPNPT/BV-01-C MESH/CFGCL/CFG/GPXY/BV-01-C MESH/CFGCL/CFG/MP/BV-01-C MESH/CFGCL/CFG/CFGF/BV-01-C MESH/CFGCL/CFG/CFGR/BV-01-C MESH/CFGCL/CFG/HBP/BV-01-C MESH/CFGCL/CFG/HBS/BV-01-C
MESH 11/3 AND MESH 10/5	Configuration Client Model, Private Beacon	MESH/CFGCL/KR/BV-05-C MESH/CFGCL/KR/BV-06-C
MESH 12/4	Advertising with Node Identity	MESH/NODE/CFG/NID/BV-01-C MESH/NODE/CFG/NID/BV-02-C MESH/NODE/CFG/NID/BV-03-C MESH/NODE/CFG/NID/BI-01-C MESH/NODE/CFG/NID/BI-03-C

Item	Feature	Test Case(s)
MESH 11/2	Health Server Model	MESH/SR/HM/CFS/BV-01-C MESH/SR/HM/CFS/BV-02-C MESH/SR/HM/RFS/BV-01-C MESH/SR/HM/RFS/BV-02-C MESH/SR/HM/RFS/BV-03-C MESH/SR/HM/RFS/BI-01-C MESH/SR/HM/RFS/BI-02-C MESH/SR/HM/HPS/BV-01-C MESH/SR/HM/ATS/BV-01-C
MESH 11/4	Health Client Model	MESH/CL/HM/RFS/BV-01-C MESH/CL/HM/HPS/BV-01-C MESH/CL/HM/ATS/BV-01-C
MESH 7/3	Friend Feature	MESH/NODE/FRND/FN/BV-01-C MESH/NODE/FRND/FN/BV-02-C MESH/NODE/FRND/FN/BV-03-C MESH/NODE/FRND/FN/BV-04-C MESH/NODE/FRND/FN/BV-05-C MESH/NODE/FRND/FN/BV-06-C MESH/NODE/FRND/FN/BV-07-C MESH/NODE/FRND/FN/BV-08-C MESH/NODE/FRND/FN/BV-09-C MESH/NODE/FRND/FN/BV-10-C MESH/NODE/FRND/FN/BV-11-C MESH/NODE/FRND/FN/BV-12-C MESH/NODE/FRND/FN/BV-13-C MESH/NODE/FRND/FN/BV-14-C MESH/NODE/FRND/FN/BV-15-C MESH/NODE/FRND/FN/BV-16-C MESH/NODE/FRND/FN/BV-17-C MESH/NODE/FRND/FN/BV-18-C MESH/NODE/FRND/FN/BV-19-C MESH/NODE/FRND/FN/BV-20-C MESH/NODE/FRND/FN/BV-21-C MESH/NODE/FRND/FN/BV-22-C MESH/NODE/FRND/FN/BV-23-C MESH/NODE/FRND/FN/BI-01-C MESH/NODE/FRND/FN/BI-02-C MESH/NODE/FRND/FN/BI-03-C MESH/NODE/CFG/HBP/BV-04-C MESH/NODE/CFG/HBS/BV-05-C MESH/NODE/CFG/LPNPT/BI-01-C MESH/NODE/NET/BV-12-C

Item	Feature	Test Case(s)
MESH 7/3 AND MESH 10/5	Friend Feature, Private Beacon	MESH/NODE/FRND/FN/BV-24-C MESH/NODE/FRND/FN/BV-25-C MESH/NODE/FRND/FN/BV-26-C
MESH 7/4	Low Power Feature	MESH/NODE/FRND/LPN/BV-01-C MESH/NODE/FRND/LPN/BV-02-C MESH/NODE/FRND/LPN/BV-03-C MESH/NODE/FRND/LPN/BV-04-C MESH/NODE/FRND/LPN/BV-05-C MESH/NODE/FRND/LPN/BV-06-C MESH/NODE/FRND/LPN/BV-07-C MESH/NODE/FRND/LPN/BV-08-C MESH/NODE/FRND/LPN/BI-01-C MESH/NODE/FRND/LPN/BI-02-C MESH/NODE/FRND/LPN/BI-03-C MESH/NODE/CFG/HBP/BV-05-C
MESH 12/1	Proxy Server	MESH/SR/PROX/BV-01-C MESH/SR/PROX/BV-02-C MESH/SR/PROX/BV-03-C MESH/SR/PROX/BV-04-C MESH/SR/PROX/BV-05-C MESH/SR/PROX/BV-06-C MESH/SR/PROX/BV-07-C MESH/SR/PROX/BV-08-C MESH/SR/PROX/BV-09-C MESH/SR/PROX/BV-10-C MESH/SR/PROX/BV-11-C MESH/SR/PROX/BV-12-C MESH/SR/PROX/BV-13-C MESH/SR/PROX/BV-14-C MESH/SR/PROX/BI-01-C MESH/SR/PROX/BI-02-C
MESH 12/1	Proxy Server	MESH/SR/PROX/BV-15-C
MESH 12/1 AND MESH 10/5	Proxy Server, Private Beacon	MESH/SR/PROX/BV-16-C MESH/SR/PROX/BV-17-C MESH/SR/PROX/BV-18-C MESH/SR/PROX/BV-20-C MESH/SR/PROX/BV-21-C MESH/SR/PROX/BV-22-C
MESH 12/1 AND MESH 10/5 AND MESH 5/4	Proxy Server, Private Beacon, Multiple Subnet Support	MESH/SR/PROX/BV-23-C MESH/SR/PROX/BV-24-C
MESH 12/1 AND MESH 10/5 AND MESH 12/8	Proxy Server, Private Beacon, Advertising with Private Node Identity	MESH/SR/PROX/BV-19-C

Item	Feature	Test Case(s)
MESH 12/5	Proxy Client	MESH/CL/MPXS/BV-01-C MESH/CL/MPXS/BV-02-C MESH/CL/MPXS/BV-03-C MESH/CL/MPXS/BV-04-C MESH/CL/MPXS/BV-05-C MESH/CL/MPXS/BV-06-C MESH/CL/PROX/BV-01-C MESH/CL/PROX/BV-02-C MESH/CL/PROX/BV-03-C MESH/CL/PROX/BV-04-C MESH/CL/PROX/BV-05-C MESH/CL/PROX/BV-06-C MESH/CL/PROX/BV-07-C MESH/CL/PROX/BV-08-C MESH/CL/PROX/BV-09-C MESH/CL/PROX/BV-10-C MESH/CL/PROX/BI-01-C MESH/CL/PROX/BI-02-C
MESH 12/5 AND MESH 10/5	Proxy Client, Private Beacon	MESH/CL/PROX/BV-11-C MESH/CL/PROX/BV-12-C MESH/CL/PROX/BV-13-C
MESH 12/8	Advertising with Private Node Identity	MESH/SR/PRB/PNID/BV-01-C MESH/SR/PRB/PNID/BI-01-C MESH/SR/PRB/PNID/BI-02-C
MESH 12/8 AND MESH 5/4	Advertising with Private Node Identity, Multiple Subnet Support	MESH/SR/PRB/PNID/BV-02-C MESH/SR/PRB/PNID/BV-03-C
MESH 12/7 AND MESH 5/4	Advertising with Private Network Identity, Multiple Subnet Support	MESH/SR/PRB/PGPXY/BV-02-C
MESH 11/5	Private Beacon Server Model	MESH/SR/PRB/PBS/BV-01-C MESH/SR/PRB/PBS/BI-01-C MESH/SR/PRB/PGPXY/BV-01-C MESH/SR/PRB/PGPXY/BI-01-C
MESH 11/6	Private Beacon Client Model	MESH/CL/PRB/PBS/BV-01-C MESH/CL/PRB/PGPXY/BV-01-C MESH/CL/PRB/PNID/BV-01-C
MESH 13/2 AND MESH 12/8	Mesh Proxy Service, Advertising with Private Node Identity	MESH/SR/MPXS/BV-11-C
MESH 13/2 AND MESH 12/7	Mesh Proxy Service, Advertising with Private Network Identity	MESH/SR/MPXS/BV-10-C
MESH 12/5 AND MESH 10/5	Proxy Client, Private Beacon	MESH/CL/MPXS/BV-07-C MESH/CL/MPXS/BV-08-C

Item	Feature	Test Case(s)
MESH 12/9	Directed Proxy Server	MESH/NODE/DF/INIT/BV-03-C MESH/NODE/DF/INIT/BV-04-C MESH/NODE/DF/DISC/BV-05-C MESH/NODE/DF/DISC/BV-06-C MESH/NODE/DF/DISC/BV-08-C MESH/NODE/DF/DISC/BV-11-C MESH/NODE/DF/EST/BV-05-C MESH/NODE/DF/UPDATE/BV-02-C MESH/NODE/DF/UPDATE/BV-03-C MESH/NODE/DF/UPDATE/BV-05-C MESH/NODE/DF/UPDATE/BV-06-C MESH/NODE/DF/UPDATE/BV-07-C MESH/NODE/DF/UPDATE/BV-08-C MESH/NODE/DF/UPDATE/BV-09-C MESH/NODE/DF/UPDATE/BV-10-C MESH/NODE/DF/UPDATE/BV-11-C MESH/NODE/DF/SOL/BV-03-C MESH/SR/DFM/DFS/BV-03-C
MESH 12/9	Directed Proxy Server	MESH/SR/DPROX/BV-01-C MESH/SR/DPROX/BV-02-C MESH/SR/DPROX/BV-03-C MESH/SR/DPROX/BV-05-C MESH/SR/DPROX/BI-01-C
MESH 12/9 AND MESH 5/4	Directed Proxy Server Multiple Subnet Support	MESH/SR/DPROX/BV-04-C MESH/SR/DPROX/BV-06-C
NOT MESH 0/1 AND MESH 12/1 AND NOT MESH 12/9	Mesh 1.1 or later Proxy Server without Directed Proxy Server	MESH/SR/DPROX/BV-07-C
NOT MESH 0/1 AND MESH 12/5 AND NOT MESH 12/10	Mesh 1.1 or later Proxy Client without Directed Proxy Client	MESH/CL/DPROX/BV-02-C
MESH 12/10	Directed Proxy Client	MESH/CL/DPROX/BV-01-C
MESH 11/9 AND MESH 7/3	Directed Forwarding Configuration Server Model Friend Feature	MESH/NODE/DF/INIT/BV-02-C MESH/NODE/DF/DISC/BV-04-C MESH/NODE/DF/DISC/BV-07-C MESH/NODE/DF/DISC/BV-10-C MESH/NODE/DF/EST/BV-04-C MESH/NODE/DF/UPDATE/BV-01-C MESH/NODE/DF/UPDATE/BV-04-C MESH/NODE/DF/SOL/BV-02-C MESH/SR/DFM/DFS/BV-04-C

Item	Feature	Test Case(s)
MESH 11/9	Directed Forwarding Configuration Server Model	MESH/NODE/DF/INIT/BV-01-C MESH/NODE/DF/INIT/BV-05-C MESH/NODE/DF/INIT/BV-06-C MESH/NODE/DF/INIT/BV-07-C MESH/NODE/DF/INIT/BV-08-C MESH/NODE/DF/INIT/BV-09-C MESH/NODE/DF/INIT/BV-10-C MESH/NODE/DF/DISC/BV-01-C MESH/NODE/DF/DISC/BV-02-C MESH/NODE/DF/DISC/BV-03-C MESH/NODE/DF/DISC/BV-09-C MESH/NODE/DF/DISC/BV-14-C MESH/NODE/DF/DISC/BV-15-C MESH/NODE/DF/DISC/BV-16-C MESH/NODE/DF/DISC/BV-17-C MESH/NODE/DF/DISC/BI-01-C MESH/NODE/DF/EST/BV-01-C MESH/NODE/DF/EST/BV-02-C MESH/NODE/DF/EST/BV-03-C MESH/NODE/DF/EST/BV-06-C MESH/NODE/DF/EST/BV-07-C MESH/NODE/DF/EST/BV-08-C MESH/NODE/DF/EST/BI-01-C

Item	Feature	Test Case(s)
MESH 11/9	Directed Forwarding Configuration Server Model	MESH/NODE/DF/CONF/BV-01-C MESH/NODE/DF/CONF/BV-02-C MESH/NODE/DF/CONF/BV-03-C MESH/NODE/DF/CONF/BV-04-C MESH/NODE/DF/CONF/BI-01-C MESH/NODE/DF/UPDATE/BV-12-C MESH/NODE/DF/UPDATE/BV-13-C MESH/NODE/DF/UPDATE/BI-01-C MESH/NODE/DF/ECHO/BV-01-C MESH/NODE/DF/ECHO/BV-02-C MESH/NODE/DF/ECHO/BV-03-C MESH/NODE/DF/ECHO/BV-04-C MESH/NODE/DF/ECHO/BV-05-C MESH/NODE/DF/ECHO/BV-06-C MESH/NODE/DF/ECHO/BI-01-C MESH/NODE/DF/ECHO/BI-02-C MESH/NODE/DF/ECHO/BI-03-C MESH/NODE/DF/ECHO/BI-04-C MESH/NODE/DF/SOL/BV-01-C MESH/NODE/DF/SOL/BV-04-C MESH/NODE/DF/SOL/BI-01-C MESH/NODE/DF/FIX/BV-01-C MESH/NODE/DF/FIX/BV-02-C MESH/NODE/DF/FIX/BV-03-C MESH/NODE/DF/FIX/BV-04-C MESH/NODE/DF/FIX/BV-05-C MESH/NODE/DF/FIX/BV-06-C MESH/NODE/DF/FIX/BV-07-C MESH/SR/DFM/DFS/BV-01-C MESH/SR/DFM/DFS/BV-02-C MESH/SR/DFM/DFS/BV-06-C MESH/SR/DFM/DFS/BI-01-C MESH/SR/DFM/PMS/BV-01-C MESH/SR/DFM/PMS/BI-01-C MESH/SR/DFM/DTCS/BV-01-C MESH/SR/DFM/DTCS/BI-01-C MESH/SR/DFM/FTS/BV-01-C MESH/SR/DFM/FTS/BV-02-C MESH/SR/DFM/FTS/BV-03-C MESH/SR/DFM/FTS/BV-04-C MESH/SR/DFM/FTS/BV-05-C MESH/SR/DFM/FTS/BV-06-C MESH/SR/DFM/FTS/BV-07-C

Item	Feature	Test Case(s)
MESH 11/9	Directed Forwarding Configuration Server Model	MESH/SR/DFM/FTS/BV-08-C MESH/SR/DFM/FTS/BV-09-C MESH/SR/DFM/FTS/BV-10-C MESH/SR/DFM/FTS/BV-11-C MESH/SR/DFM/FTS/BV-12-C MESH/SR/DFM/FTS/BI-01-C MESH/SR/DFM/FTS/BI-02-C MESH/SR/DFM/FTS/BI-03-C MESH/SR/DFM/FTS/BI-04-C MESH/SR/DFM/WLS/BV-01-C MESH/SR/DFM/WLS/BI-01-C MESH/SR/DFM/TWPS/BV-01-C MESH/SR/DFM/TWPS/BI-01-C MESH/SR/DFM/PEIPS/BV-01-C MESH/SR/DFM/PEIPS/BV-02-C MESH/SR/DFM/PEIPS/BV-03-C MESH/SR/DFM/PEIPS/BI-01-C MESH/SR/DFM/PEIPS/BI-02-C MESH/SR/DFM/DNTS/BV-01-C MESH/SR/DFM/DRTS/BV-01-C MESH/SR/DFM/RTS/BV-01-C MESH/SR/DFM/RTS/BI-01-C MESH/SR/DFM/DPS/BV-01-C MESH/SR/DFM/PPS/BV-01-C MESH/SR/DFM/PPS/BI-01-C MESH/SR/DFM/DTS/BV-01-C MESH/SR/DFM/DTS/BI-01-C MESH/SR/DFM/DCNTS/BV-01-C MESH/SR/DFM/DCRTS/BV-01-C

Item	Feature	Test Case(s)
MESH 11/10	Directed Forwarding Configuration Client Model	MESH/CL/DFM/DFS/BV-01-C MESH/CL/DFM/DFS/BI-01-C MESH/CL/DFM/PMS/BV-01-C MESH/CL/DFM/PMS/BI-01-C MESH/CL/DFM/PMS/BI-02-C MESH/CL/DFM/DTCS/BV-01-C MESH/CL/DFM/DTCS/BI-01-C MESH/CL/DFM/DNTS/BV-01-C MESH/CL/DFM/DCNTS/BV-01-C MESH/CL/DFM/FTS/BV-01-C MESH/CL/DFM/FTS/BV-02-C MESH/CL/DFM/FTS/BV-03-C MESH/CL/DFM/WLS/BV-01-C MESH/CL/DFM/WLS/BI-01-C MESH/CL/DFM/TWPS/BV-01-C MESH/CL/DFM/TWPS/BI-01-C MESH/CL/DFM/PEIPS/BV-01-C MESH/CL/DFM/PEIPS/BI-01-C MESH/CL/DFM/DRTS/BV-01-C MESH/CL/DFM/RTS/BV-01-C MESH/CL/DFM/RTS/BI-01-C MESH/CL/DFM/DPS/BV-01-C MESH/CL/DFM/PPS/BV-01-C MESH/CL/DFM/PPS/BI-01-C MESH/CL/DFM/DTS/BV-01-C MESH/CL/DFM/DCRTS/BV-01-C
MESH 11/9 AND MESH 7/4	Directed Forwarding Configuration Server Model Low Power Feature	MESH/SR/DFM/DFS/BV-05-C

Item	Feature	Test Case(s)
MESH 11/7	Bridge Server Model	MESH/NODE/SBR/NET/BV-01-C MESH/NODE/SBR/NET/BV-02-C MESH/SR/BCM/SBS/BV-01-C MESH/SR/BCM/SBS/BI-01-C MESH/SR/BCM/BDS/BV-01-C MESH/SR/BCM/BDS/BV-02-C MESH/SR/BCM/BDS/BV-03-C MESH/SR/BCM/BDS/BV-04-C MESH/SR/BCM/BDS/BV-05-C MESH/SR/BCM/BDS/BV-06-C MESH/SR/BCM/BDS/BI-01-C MESH/SR/BCM/BDS/BI-02-C MESH/SR/BCM/BDS/BI-03-C MESH/SR/BCM/BTS/BV-01-C
MESH 11/8	Bridge Client Model	MESH/CL/BCM/SBS/BV-01-C MESH/CL/BCM/BDS/BV-01-C MESH/CL/BCM/BTS/BV-01-C
MESH 11/7 AND MESH 11/9	Directed Forwarding Configuration Server Model Bridge Server Model	MESH/NODE/DF/INIT/BV-11-C MESH/NODE/DF/DISC/BV-12-C MESH/NODE/DF/DISC/BV-13-C MESH/NODE/DF/EST/BV-09-C MESH/NODE/DF/UPDATE/BV-14-C MESH/NODE/DF/UPDATE/BV-15-C MESH/NODE/DF/SOL/BV-05-C
MESH 11/11	Remote Provisioning Server Model	MESH/SR/RPR/SCN/BV-01-C MESH/SR/RPR/SCN/BV-02-C MESH/SR/RPR/SCN/BV-03-C MESH/SR/RPR/SCN/BV-06-C MESH/SR/RPR/SCN/BV-07-C MESH/SR/RPR/SCN/BI-01-C MESH/SR/RPR/SCN/BI-02-C MESH/SR/RPR/LNK/BI-02-C MESH/SR/RPR/LNK/BI-04-C MESH/SR/RPR/LNK/BI-05-C

Item	Feature	Test Case(s)
MESH 11/11 AND MESH 18/2	Remote Provisioning over PB-ADV	MESH/SR/RPR/SCN/BV-04-C MESH/SR/RPR/SCN/BV-08-C MESH/SR/RPR/LNK/BV-01-C MESH/SR/RPR/LNK/BV-03-C MESH/SR/RPR/LNK/BV-05-C MESH/SR/RPR/LNK/BV-06-C MESH/SR/RPR/LNK/BV-07-C MESH/SR/RPR/LNK/BV-09-C MESH/SR/RPR/LNK/BV-10-C MESH/SR/RPR/LNK/BV-11-C MESH/SR/RPR/LNK/BV-13-C MESH/SR/RPR/LNK/BV-15-C MESH/SR/RPR/LNK/BV-17-C MESH/SR/RPR/LNK/BI-01-C MESH/SR/RPR/PDU/BV-01-C
MESH 11/11 AND MESH 18/5	Remote Provisioning over PB-GATT	MESH/SR/RPR/SCN/BV-05-C MESH/SR/RPR/LNK/BV-02-C MESH/SR/RPR/LNK/BV-04-C MESH/SR/RPR/LNK/BV-14-C MESH/SR/RPR/LNK/BV-16-C MESH/SR/RPR/LNK/BV-18-C MESH/SR/RPR/PDU/BV-05-C
MESH 11/11 AND MESH 5/4 AND MESH 18/2	Remote Provisioning Server Model with Multiple Network Keys and PB-ADV	MESH/SR/RPR/LNK/BV-19-C
MESH 11/11 AND MESH 5/4 AND MESH 18/5	Remote Provisioning Server Model with Multiple Network Keys and PB-GATT	MESH/SR/RPR/LNK/BV-22-C
MESH 4/15	Node Provisioning Protocol Interface	MESH/SR/RPR/LNK/BV-20-C MESH/SR/RPR/LNK/BV-21-C MESH/SR/RPR/LNK/BV-23-C MESH/SR/RPR/LNK/BV-25-C MESH/SR/RPR/LNK/BI-03-C MESH/SR/RPR/PDU/BV-02-C MESH/SR/RPR/PDU/BI-01-C
MESH 4/15 AND MESH 5/4	Node Provisioning Protocol Interface with Multiple Network Keys	MESH/SR/RPR/LNK/BV-26-C
MESH 4/16	Node Address Refresh Procedure	MESH/SR/RPR/PDU/BV-03-C MESH/SR/RPR/PDU/BI-02-C
MESH 4/17	Node Composition Refresh Procedure	MESH/SR/RPR/PDU/BV-04-C MESH/SR/RPR/PDU/BI-03-C

Item	Feature	Test Case(s)
MESH 11/12	Remote Provisioning Client Model	MESH/CL/RPR/SCN/BV-01-C MESH/CL/RPR/LNK/BV-01-C MESH/CL/RPR/PDU/BV-01-C
MESH 11/14	On-demand Proxy Server Model	MESH/SR/MPXS/BV-12-C MESH/SR/MPXS/BV-13-C MESH/SR/ODP/BV-01-C
MESH 11/15	On-demand Proxy Client Model	MESH/CL/ODP/BV-01-C
MESH 11/16	Solicitation PDU RPL Configuration Server Model	MESH/SR/SRPL/BV-01-C
MESH 11/17	Solicitation PDU RPL Configuration Client Model	MESH/CL/SRPL/BV-01-C
MESH 11/18	SAR Configuration Server Model	MESH/NODE/TNPT/BV-14-C MESH/NODE/TNPT/BV-15-C MESH/NODE/TNPT/BV-16-C MESH/NODE/TNPT/BV-17-C MESH/NODE/TNPT/BV-18-C MESH/NODE/TNPT/BV-19-C MESH/NODE/TNPT/BV-20-C MESH/NODE/TNPT/BV-21-C MESH/SR/SAR/BV-01-C MESH/SR/SAR/BV-02-C
MESH 11/19	SAR Configuration Client Model	MESH/CL/SAR/BV-01-C MESH/CL/SAR/BV-02-C
MESH 11/20	Opcodes Aggregator Server Model	MESH/SR/AGG/BV-01-C MESH/SR/AGG/BV-02-C MESH/SR/AGG/BI-01-C
MESH 11/21	Opcodes Aggregator Client Model	MESH/CL/AGG/BV-01-C
MESH 11/22	Large Composition Data Server Model	MESH/SR/LCD/LCMP/BV-01-C MESH/SR/LCD/MMD/BV-01-C
MESH 11/23	Large Composition Data Client Model	MESH/CL/LCD/LCMP/BV-01-C MESH/CL/LCD/MMD/BV-01-C
MESH 12/11	Sending Proxy Solicitation	MESH/CL/MPXS/BV-09-C
MESH 4/20 AND MESH 4/1	Unprovisioned Device using X.509 Certificates and PB-ADV	MESH/NODE/CBP/BV-01-C
MESH 4/20 AND MESH 4/2	Unprovisioned Device using X.509 Certificates and PB-GATT	MESH/NODE/CBP/BV-02-C
MESH 4/21 AND MESH 4/1	Send X.509 Certificates over PB-ADV	MESH/NODE/CBP/BV-03-C
MESH 4/21 AND MESH 4/2	Send X.509 Certificates over PB-GATT	MESH/NODE/CBP/BV-04-C

Item	Feature	Test Case(s)
MESH 4/19	In-band Certificate Transfer	MESH/NODE/CBP/BI-01-C
MESH 18/14 AND MESH 18/2	Provisioner using X.509 Certificates and PB-ADV	MESH/PVNR/CBP/BV-01-C
MESH 18/14 AND MESH 18/5	Provisioner using X.509 Certificates and PB-GATT	MESH/PVNR/CBP/BV-02-C
MESH 18/15 AND MESH 18/2	Retrieve X.509 Certificates over PB-ADV	MESH/PVNR/CBP/BV-03-C
MESH 18/15 AND MESH 18/5	Retrieve X.509 Certificates over PB-GATT	MESH/PVNR/CBP/BV-04-C
MESH 18/14	Provisioner using X.509 Certificates	MESH/PVNR/CBP/BI-01-C
MESH 2/3	Unprovisioned device role	MESH/NODE/BV-01-C
MESH 2/1	Node role	MESH/NODE/BV-02-C MESH/NODE/BV-03-C

Table 5.1: Test case mapping

6 Annex: Supplementary Mesh Test Cases

This annex defines mesh test procedures that are tested in the Interoperable Product Testing (IOPT) Test Suite.

6.1 Mesh Node role supplementary tests

MESH/NODE/BV-01-C [Provisioning and reprovisioning]

- Test Purpose

  Verify that a device can be provisioned, de-provisioned, and re-provisioned.

- Reference

  [3] 3.11.7, 5

- Initial Condition

  - If the IUT is unprovisioned, it is sending unprovisioned device advertisements over the supported bearer(s).

- Test Procedure

  1. If the IUT is provisioned, the Lower Tester executes the Node Removal procedure to remove all mesh network information from the IUT.
  2. The Lower Tester provisions the IUT over any supported bearer.
  3. If multiple provisioning bearers are supported, execute Steps 1 and 2 again for each additional provisioning bearer.

- Expected Outcome

  Pass verdict

  The IUT allows node removal and re-provisioning.

MESH/NODE/BV-02-C [Mesh device composition data]

- Test Purpose

  Verify that the composition data returned by the IUT is valid and accurately represents the device's capabilities.

- Reference

  [3] 2.3.6, 3.8.4
  [11] 1.4.4

- Initial Condition

  - The IUT has been provisioned by the Lower Tester.

- Test Procedure

1. The Lower Tester reads and verifies the format of the IUT's composition data, as described in [12] MMDL/SR-CL/COMP/BV-01-C.
2. For each supported server model instance, the Lower Tester binds an application key to the instance or uses the device key (depending on what is allowed in the model definition), and then executes a single exchange of messages on that model, such as reading a state value.
3. For each supported client model instance that supports publishing, the Lower Tester binds an application key to the instance and configures the IUT to publish for that model.

- Expected Outcome

  Pass verdict

  At least Composition Data Pages 0 and 1 are present and have a valid structure according to their definition.
  All mesh features indicated as supported in the IUT's Mesh Protocol ICS [4] are marked as supported in Composition Data Page 0.
  Each mesh model indicated as supported in the IUT's Mesh Model ICS [13] has at least one instance present on any of the elements of the IUT.
  The arrangement of model instances on all elements conforms to the requirements defined in [3], [11] (e.g., Section 1.4.4), and each model's definition.
  The IUT responds to all model-specific requests.

MESH/NODE/BV-03-C [Network message cache and replay protection]

- Test Purpose

  Verify proper network cache operation and functional replay protection list.

- Reference

  [3] 3.4.6.5, 3.9.8

- Initial Condition

  - The IUT has been provisioned by the Lower Tester.

- Test Procedure

  1. If the IUT supports the Relay feature, then execute test case MESH/NODE/RLY/BV-02-C [Network Message Cache]. The size of the network cache is specified in the Mesh Protocol IXIT [6].
  2. Execute test case MESH/NODE/TNPT/BV-13-C [Ignore Replayed Message].

- Expected Outcome

  Pass verdict

  If the IUT supports the Relay feature, the IUT implements proper network message cache.
  The IUT implements proper replay protection.

7 Revision history and acknowledgments

Revision History

Publication Number	Revision Number	Date	Comments
0	1.0.0	2017-07-12	Approved by BTI. Prepared for TCRL 2017-1 publication.
	1.0.1r00	2017-08-25	TSE 9589: Deleted test case MESH/NODE/CFG/NID/BI-02-C from the TS and TCMT. TSE 9626: Revised MESH/NODE/PROV/BI-01-C test purpose.
	1.0.1r01	2017-08-28	TSE 9716: Removed test case MESH/NODE/IVU/BI-03-C from text and the TCMT. TSE 9741: Clarified test case MESH/NODE/IVU/BI-05-C and updated pass verdict step numbers. TSE 9742: Corrected the test procedure calculation in MESH/NODE/CFG/HBP/BV-02-C.
	1.0.1r02	2017-09-05	TSE 9746: Updated pass verdict step 4 for MESH/NODE/KR/BI-01-C.
	1.0.1r03	2017-09-14	TSE 9795: Revised text and figure for test case MESH/NODE/CFG/HBS/BV-05-C. TSE 9750: Revised text and figure for test case MESH/NODE/CFG/HBS/BV-01-C. TSE 9714: Removed test case MESH/NODE/BCN/SNB/BV-02-C. TSE 9713: Revised text for test case MESH/NODE/IVU/BI-06-C.
	1.0.1r04	2017-09-15	TSE 9774: Revised text and figure for test case MESH/NODE/FRND/LPN/BI-02-C.
	1.0.1r05	2017-09-29	TSE 9760: Revised text and figure for test case MESH/NODE/FRND/FN/BV-20-C.
	1.0.1r06	2017-10-02	TSE 9824: Added missing friend enables to test cases: MESH/NODE/FRND/FN/BV-01-C - 21-C, and MESH/NODE/FRND/FN/BI-01-C - 03-C.
1	1.0.1	2017-11-22	Approved by BTI. Prepared for TCRL 2017-2 publication.
	1.0.2r00-08	2018-02-20 - 2018-06-28	TSE 10040 (rating 1): Corrected MSC titles: MESH/NODE/IVU/BV-04-C, MESH/NODE/KR/BV-03-C03-C, MESH/NODE/CFG/NKL/BI-02-C - 03-C, MESH/CFGCL/CFG/NKL/BV-03-C, MESH/NODE/CFG/NID/BV-03-C, MESH/NODE/CFG/HBP/BV-07-C.

Publication Number	Revision Number	Date	Comments
	1.0.2r00-08	2018-02-20 - 2018-06-28	Replaced MSCs: MESH/NODE/FRND/FN/BV-04-C, 06-C, 09-C - 11-C, 14-C, 21-C, MESH/NODE/FRND/FN/BI-01-C, MESH/NODE/FRND/LPN/BV-03-C - 06-C. TSE 10229 (rating 1): Updated Test Procedure step 6 and Pass Verdict in Section 4.3.5 (MESH/PVNR/PBADV/BV-01-C [PB-ADV Link Establishment and Link Close - Provisioner]). TSE 10145 (rating 3): Revised initial condition in MESH/NODE/FRND/LPN/BV-05-C to add test procedure execution text and reference MESH/NODE/FRND/LPN/BV-04-C instead of MESH/NODE/FRND/LPN/BV-01-C. TSE 10009 (rating 3): Added fail verdict to all Friend node BV test cases (MESH/NODE/FRND/FN/BV-01-C to 21-C) verifying that a Friend node uses a single Network PDU to send a message to its LPN. TSE 9905 (rating 3): Revised initial condition and test procedure for MESH/SR/PROX/BV-12-C and 14-C, and replaced MESH/SR/PROX/BV-14-C MSC. TSE 9864 (rating 4): Removed test case MESH/PVNR/MPS/BV-07-C from text and TCMT. TSE 10279 (rating 1): Corrected test procedure step 1 for MESH/NODE/FRND/FN/BV-13-C. TSE 10461 (rating 3): Added clarification to MESH/NODE/TNPT/BV-05-C initial condition. TSE 10330 (rating 3): Revised test procedure step 13, MSC, and expected outcome pass verdict for MESH/NODE/CFG/HBS/BV-02-C. TSE 10407 (rating 3): Corrected the expectation in the MESH/NODE/CFG/MP/BV-01-C test procedure steps 14 and 16. Replaced the test procedure MSC and updated the Pass Verdict. TSE 10286 (rating 3): Removed OOB from the MESH/PVNR/PROV/PVN/BV-10-C test procedure steps 3 and 4.
	1.0.2r00-08	2018-02-20 - 2018-06-28	E10395: Added 3 new test cases to TS and TCMT: MESH/NODE/PROV/UPD/BV-12-C, MESH/NODE/PROV/UPD/BV-13-C, and MESH/PVNR/PROV/UPD/BI-14-C. Corrected implementation of TSE 10330 by changing “a group address” to “the same address” in step 13.
2	1.0.2	2018-06-27	Approved by BTI. Prepared for TCRL 2018-1 publication.

Publication Number	Revision Number	Date	Comments
	1.0.3r00–r03	2018-10-02 – 2018-11-05	TSE 10696 (rating 3): Added new test case MESH/NODE/PROV/BI-03-C, and added it to the TCMT. TSE 10698 (rating 3): Updated test procedure steps and pass verdict for test case MESH/NODE/PROV/UPD/BV-10-C. Updated description, test purpose, test procedure steps, and pass verdict for test case MESH/NODE/PROV/BI-02-C. TSE 11166 (rating 3): Updated initial condition for test cases MESH/SR/MPXS/BV-09-C and MESH/NODE/CFG/NID/BV-02-C. TSE 11167 (rating 3): Updated test purpose, test procedure steps, pass verdict for test case MESH/NODE/PBADV/BV-02-C. Editorial correction: Stray “TBD” removed from pass verdict MESH/NODE/BCN/SNB/BV-01-C. TSE 11230 (rating 1): Updated TCIDs for the majority of the test cases in section 4.5 Provisioning Procedure to match the correct format, by removing particles /UPD/ and /PVN/. Moved the two invalid key BI tests at the end of 4.5.
	1.0.1.0	2018-11-09	Updated version number from 1.0.3 to 1.0.1.0 to align with adoption of specification version 1.0.1.
3	1.0.1.0	2018-11-21	Approved by BTI. Prepared for TCRL 2018-2 publication.
	1.0.1.1 r00-r05	2019-04-04 - 2019-06-19	TSE 11180 (rating 2): Updated step 13 in test case MESH/NODE/CFG/HBS/BV-02-C so that the exception covers only step 2. TSE 11250 (rating 3): Updated to new test case figure for MESH/NODE/CFG/HBS/BV-02-C. TSE 11123 (rating 3): Updated test cases MESH/NODE/TNPT/BV-07-C and -08-C to enhance transport tests. TSE 11445 (rating 4): Deleted test case MESH/SR/MPXS/BV-08-C and updated TCMT accordingly. TSE 11760 (rating 2): Updated initial condition and/or test steps and/or pass verdict in test cases MESH/NODE/KR/BV-01-C - -03-C; MESH/NODE/KR/BI-05-C; MESH/NODE/CFG/NKL/BV-01-C, -02-C, and -04-C; MESH/NODE/CFG/NKL/BI-01-C - -03-C; MESH/NODE/CFG/AKL/BV-01-C - -03-C; and MESH/NODE/CFG/AKL/BI-02-C and -04-C. Updated TCMT accordingly.
4	1.0.1.1	2019-07-25	Approved by BTI. Prepared for TCRL 2019-1 publication.

Publication Number	Revision Number	Date	Comments
	1.0.1.2 r00–r03	2019-09-16 – 2019-10-17	TSE 11776 (rating 4): Deleted test case MESH/NODE/NET/BI-04-C; updated test case MESH/NODE/RLY/BV-01-C with additional reference and updated test step #7, adding a table to indicate rounds of testing, and updated the MSC. Updated TCMT accordingly. TSE 12241 (rating 3): Updated test step #6 of test case MESH/NODE/FRND/FN/BV-05-C to align better with Mesh specification. TSE 12295 (rating 3): Updated test case MESH/NODE/TNPT/BV-05-C by adding a new reference and updating test procedure step 2 and the pass verdict to align better with the Mesh spec recommendations. TSE 12391 (rating 3): Updated test case MESH/SR/PROX/BV-13-C to align test steps 8 and 9 better with the correct IUT expectation. TSE 12074 (rating 1): Updated test case MESH/NODE/FRND/FN/BV-20-C to correctly reflect Lower Tester 1 and 2 roles. TSE 12423 (rating 1): Updated test case MESH/NODE/CFG/HBP/BV-05-C to correctly reflect Lower Tester 1 and 2 roles. TSE 12037 (rating 3): Updated test cases MESH/NODE/MPS/BV-02-C, MESH/PVNR/MPS/BV-01-C, and MESH/PVNR/MPS/BV-02-C and the "State: Node PB-GATT Link Established" section to address an issue with required interleave Unprovisioned Device beacon for PB-GATT. TSE 12298 (rating 1): Updated test procedure, MSC, and pass verdict for test case MESH/CFGCL/CFG/HBS/BV-01-C to resolve issue with IUT/Lower Tester roles caused by a cut-and-paste error from a reverse test case. TSE 12718 (rating 1): Updated Initial Conditions and step 1 of the Test Procedure of test case MESH/NODE/CFG/NKL/BI-03-C to fix and invalid expected value for NetKey Add. TSE 12393 (rating 4): Added new test cases MESH/NODE/PROV/BI-15-C and MESH/PVNR/PROV/BI-16-C to cover testing for receiving an RFU Provisioning PDU Type, and updated the TCMT accordingly.
5	1.0.1.2	2020-01-07	Approved by BTI on 2019-11-03. Prepared for TCRL 2019-2 publication.

Publication Number	Revision Number	Date	Comments
	p6r00-r12	2020-01-15 - 2020-11-18	TSE 11756 (rating 4): Added new test case MESH/SR/PROX/BV-15-C. Updated TCMT accordingly. TSE 12323 (rating 4): For MESH/SR/HM/RFS/BV-01-C, added an initial condition, a test procedure step, and a pass verdict and updated the MSC. TSE 12770 (rating 4): Added new section "Ignore RFU Proxy configuration message" featuring new test cases MESH/SR/PROX/BI-02-C and MESH/CL/PROX/BI-02-C. Updated the TCMT accordingly. TSE 12773 (rating 4): Updated test case name, test purpose, test steps, and MSC for test case MESH/NODE/FRND/FN/BV-18-C; added an initial condition for test cases MESH/NODE/FRND/FN/BV-05-C and MESH/NODE/FRND/LPN/BV-01-C and -04-C; and added new test cases MESH/NODE/FRND/FN/BV-22-C and -23-C with requisite TCMT entries. TSE 12774 (rating 4): Updated test case name, test purpose, test steps, and expected outcome for test case MESH/NODE/PBADV/BV-02-C. TSE 13068 (rating 2): Updated the MSC for test case MESH/NODE/FRND/LPN/BV-05-C to align with test procedure. TSE 13137 (rating 3): Added an initial condition for test case MESH/NODE/IVU/BV-04-C so that the value of the IUT's segment transmission timer is known to the Lower Tester. TSE 13513 (rating 1): Replaced the MSC for test case MESH/NODE/CFG/NKL/BI-02 to fix the NetKeyIndex field. TSE 13518 (rating 1): Replaced the MSC for test case MESH/NODE/CFG/AKL/BV-03-C to fix the AppKey Status field. TSE 13537 (rating 3): Updated test procedure and added a rounds-based test case config table to test case MESH/NODE/PROV/BI-03-C to address an issue with allowing a Provisioner not to use an OOB public key when it is available. TSE 13590 (rating 3): Updated the pass verdict for test case MESH/NODE/CFG/COMP/BV-01-C so that the tester checks the RFU model IDs for correctness. TSE 14760 (rating 2): Updated TC MESH/NODE/PROV/BI-15-C to fix the Expected Code Field value for Provisioning a Failed PDU.

Publication Number	Revision Number	Date	Comments
	p6r00–r12	2020-01-15 – 2020-11-18	TSE 15221 (rating 2): For test case MESH/NODE/CFG/HBS/BV-02-C, updated step numbering in step 13 and in two of the pass verdicts, and replaced MSC with new one. TSE 16063 (rating 1): Per Errata 15155 and 15991, globally changed “master” to “managed flooding”, “black list” or “blacklist” to “reject list”, and “white list” to “accept list”. Editorials related to template updates and consistency checker findings. Added Publication Number column to the Revision History table and assigned last published version 1.0.1.2 as p5.
6	p6	2020-12-22	Approved by BTI on 2020-12-02. Prepared for TCRL 2020-1 publication.
	p7r00	2021-03-19	TSE 16434 (rating 4): To address Erratum 16350, added new test cases MESH/NODE/PROV/BI-17-C and MESH/PVNR/PROV/BI-18-C. Updated TCMT accordingly. Modified Test Procedure of MESH/NODE/PROV/BI-13-C and MESH/PVNR/PROV/BI-14-C. Concatenated 2020-1 rev history entries.
7	p7	2021-05-24	Approved by BTI on 2021-04-21. Prepared for publication.
	p8r00	2021-05-24	TSE 12296 (rating 4): To address E11394, added new TC MESH/NODE/CFG/MP/BV-03-C. Updated TCMT accordingly. TSE 16075 (rating 3): Updated test steps, MSC, and pass verdict for TC MESH/NODE/CFG/MP/BI-03-C. TSE 16076 (rating 2): Updated test procedure and pass verdict of TC MESH/SR/MPXS/BV-09-C. TSE 16309 (ratin 1): To correct the key type, updated MSCs for TCs MESH/NODE/KR/BV-01-C and MESH/CFGCL/KR/BV-01-C. TSE 16786 (rating 2): Updated test steps and pass verdicts for TCs MESH/SR/PROX/BV-06-C, -08-C, and -09-C.
8	p8	2021-07-13	Approved by BTI on 2021-06-03. Prepared for TCRL 2021-1 publication.
	p8ed2r00	2021-08-03	TSE 17317 (rating 1): Removed duplicated TCID from the TCMT.
	p8 edition 2	2021-09-30	Approved by BTI on 2021-09-27. Prepared for edition 2 publication.

Publication Number	Revision Number	Date	Comments
	p9r00	2021-10-12	TSE 17003 (rating 2): Updated test procedure and Pass verdict for MESH/NODE/CFG/HBP/BV-01-C to specify the CountLog value. TSE 17076 (rating 3): Added power reset steps to the test procedure for MESH/NODE/TNPT/BV-13-C. TSE 17208 (rating 2): Updated the initial conditions for MESH/NODE/FRND/FN/BV-04-C. Editorials, including performing template-related formatting fixes, updating the introduction text before the TCMT to align with the latest template, and updating the copyright page to align with v2 of the DNMD.
9	p9	2022-01-25	Approved by BTI on 2021-12-13. Prepared for TCRL 2021-2 publication.
	p10r00–r02	2022-03-15 – 2022-05-09	TSE 18479 (rating 2): Fixed the Reason Field in a test step and a pass verdict in MESH/PVNR/PBADV/BV-01-C.
10	p10	2022-06-28	Approved by BTI on 2022-05-22. Prepared for TCRL 2022-1 publication.

Publication Number	Revision Number	Date	Comments
	p11r00–r15	2022-07-26 – 2022-12-22	Incorporated changes from MESH_CBP_Test_CR_r10-E19129.docx: Added Certificate-Based Provisioning section with 10 new TCs: MESH/NODE/CBP/BV-01-C - -04-C, MESH/NODE/CBP/BI-01-C, MESH/PVNR/CBP/BV-01-C - -04-C, and MESH/PVNR/CBP/BI-01-C. Updated the TCMT accordingly. Incorporated changes from Mesh.DF.CR.TSr19-ba_PTS_comment-ba: Added two new subsections to the Test Strategy. Added Directed Forwarding to the Test groups section. Added abbreviations to the Test case identification conventions table. Updated the Network and Application Security Creditials setup preamble section. Added new setup preamble sections: Common Initial Test Conditions for Directed Forwarding Tests; State: Path Established, as Origin, Unicast Address Destination; State: Path Established as Directed Relay, Unicast Address Destination; State: Path Established as Destination, Unicast Address Destination; State: Proxy connection with Proxy client, IUT as Directed Proxy Server; and State: Proxy connection with Directed Proxy client, IUT as Directed Proxy Server. Updated the expected outcome for MESH/NODE/CFG/COMP/BV-01-C. Added a Directed Forwarding test section with 86 new test cases: MESH/NODE/DF/INIT/BV-01-C – -10-C, MESH/NODE/DF/DISC/BV-01-C – -11-C, -14-C – -17-C, MESH/NODE/DF/DISC/BI-01-C, MESH/NODE/DF/EST/BV-01-C – -08-C, MESH/NODE/DF/EST/BI-01-C, MESH/NODE/DF/CONF/BV-01-C – -04-C, MESH/NODE/DF/CONF/BI-01-C, MESH/NODE/DF/UPDATE/BV-01-C – -13-C, MESH/NODE/DF/UPDATE/BI-01-C, MESH/NODE/DF/ECHO/BV-01-C – -06-C, MESH/NODE/DF/ECHO/BI-01-C – -04-C, MESH/NODE/DF/SOL/BV-01-C – -04-C, MESH/NODE/DF/SOL/BI-01-C, MESH/NODE/DF/FIX/BV-01-C – -07-C, MESH/SR/DPROX/BV-01-C – -07-C, MESH/SR/DPROX/BI-01-C, MESH/CL/DPROX/BV-01-C and -02-C.

Publication Number	Revision Number	Date	Comments
			Added a new Director Forwarding Configuration Model test section, including 73 new test cases: MESH/SR/DFM/DFS/BV-01-C - -06-C, MESH/SR/DFM/DFS/BI-01-C, MESH/CL/DFM/DFS/BV-01-C, MESH/CL/DFM/DFS/BI-01-C, MESH/SR/DFM/PMS/BV-01-C, MESH/SR/DFM/PMS/BI-01-C, MESH/CL/DFM/PMS/BV-01-C, MESH/CL/DFM/PMS/BI-01-C and -02-C, MESH/SR/DFM/DTCS/BV-01-C, MESH/SR/DFM/DTCS/BI-01-C, MESH/CL/DFM/DTCS/BV-01-C, MESH/CL/DFM/DTCS/BI-01-C, MESH/SR/DFM/FTS/BV-01-C - -12-C, MESH/SR/DFM/FTS/BI-01-C - -04-C, MESH/CL/DFM/FTS/BV-01-C - -03-C, MESH/SR/DFM/WLS/BV-01-C, MESH/SR/DFM/WLS/BI-01-C, MESH/CL/DFM/WLS/BV-01-C, MESH/CL/DFM/WLS/BI-01-C, MESH/SR/DFM/TWPS/BV-01-C, MESH/SR/DFM/TWPS/BI-01-C, MESH/CL/DFM/TWPS/BV-01-C, MESH/CL/DFM/TWPS/BI-01-C, MESH/SR/DFM/PEIPS/BV-01-C - -03-C, MESH/SR/DFM/PEIPS/BI-01-C and -02-C, MESH/CL/DFM/PEIPS/BV-01-C, MESH/CL/DFM/PEIPS/BI-01-C, MESH/SR/DFM/DNTS/BV-01-C, MESH/CL/DFM/DNTS/BV-01-C, MESH/SR/DFM/DRTS/BV-01-C, MESH/CL/DFM/DRTS/BV-01-C, MESH/SR/DFM/RTS/BV-01-C, MESH/SR/DFM/RTS/BI-01-C, MESH/CL/DFM/RTS/BV-01-C, MESH/CL/DFM/RTS/BI-01-C, MESH/SR/DFM/DPS/BV-01-C, MESH/CL/DFM/DPS/BV-01-C, MESH/SR/DFM/PPS/BV-01-C, MESH/SR/DFM/PPS/BI-01-C, MESH/CL/DFM/PPS/BV-01-C, MESH/CL/DFM/PPS/BI-01-C, MESH/SR/DFM/DTS/BV-01-C, MESH/SR/DFM/DTS/BI-01-C, MESH/CL/DFM/DTS/BV-01-C, MESH/SR/DFM/DCNTS/BV-01-C, MESH/CL/DFM/DCNTS/BV-01-C, MESH/SR/DFM/DCRTS/BV-01-C, and MESH/CL/DFM/DCRTS/BV-01-C. Updated the TCMT accordingly.

Publication Number	Revision Number	Date	Comments
			Incorporated changes from MshPRF.ENH.TS.CRr05-ba8-MWG-ba2.docx: Added Send Segmented Messages Using the SAR Transmitter State Values - Unicast test section with 3 new TCs: MESH/NODE/TNPT/BV-14-C - -16-C. Added Receive Segmented Messages Using the SAR Receiver State Values - Unicast test section with 6 new TCs: MESH/NODE/TNPT/BV-17-C - -21-C and MESH/NODE/TNPT/BI-03-C. Added 3 new TCs to the Mesh Proxy Service section: MESH/SR/MPXS/BV-12-C and -13-C and MESH/CL/MPXS/BV-09-C. Updated the MSC for MESH/NODE/CFG/COMP/BV-01-C. Updated the MSC, test procedure, and expected outcome for MESH/CFGCL/CFG/COMP/BV-01-C. Added a reference and updated the test procedure for MESH/SR/RPR/PDU/BV-02-C - -04-C. Added an On-demand Proxy Models test section with 2 new TCs: MESH/SR/ODP/BV-01-C and MESH/CL/ODP/BV-01-C. Added an SAR Configuration Models test section with 4 new TCs: MESH/SR/SAR/BV-01-C and -02-C, MESH/CL/SAR/BV-01-C and -02-C. Added an Opcodes Aggregator Models test section with 4 new TCs: MESH/SR/AGG/BV-01-C and -02-C, MESH/SR/AGG/BI-01-C, and MESH/CL/AGG/BV-01-C. Added a Large Composition Data Models test section with 4 new TCs: MESH/SR/LCD/LCMP/BV-01-C, MESH/CL/LCD/LCMP/BV-01-C, MESH/SR/LCD/MMD/BV-01-C, and MESH/CL/LCD/MMD/BV-01-C. Added a Solicitation PDU RPL Configuration Models test section with 2 new TCs: MESH/SR/SRPL/BV-01-C and MESH/CL/SRPL/BV-01-C. Updated the TCMT accordingly.

Publication Number	Revision Number	Date	Comments
			Incorporated changes from MESH_PRB_TEST_CR_r06: Updated the Beacons, IV Update, and Key Refresh sections. Added abbreviations to the Test case identification conventions table. Added a new Private Beacon section, including 13 new TCs: MESH/SR/PRB/PBS/BV-01-C and -01-C, MESH/CL/PRB/PBS/BV-01-C, MESH/SR/PRB/PGPXY/BV-01-C and -02-C, MESH/SR/PRB/PGPXY/BI-01-C, MESH/CL/PRB/PGPXY/BV-01-C, MESH/SR/PRB/PNID/BV-01-C - -03-C, MESH/SR/PRB/PNID/BI-01-C and -02-C, and MESH/CL/PRB/PNID/BV-01-C. Updated the Secure Network Beacons section: renamed it “Mesh Beacons”, added numbering to the MESH/NODE/BCN/SNB/BV-01-C section, added new TC MESH/NODE/BCN/PRB/BV-01-C, and updated the test purpose, reference, test procedure, and expected outcome for MESH/NODE/BCN/SNB/BV-01-C and MESH/NODE/BCN/PRB/BV-01-C. Added new TC MESH/NODE/BCN/PRB/BV-02-C. Added numbering to the MESH/NODE/IVU/BV-01-C section, added new TC MESH/NODE/IVU/BV-06-C, and updated the test purpose, reference, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BV-01-C and -06-C. Added numbering to the MESH/NODE/IVU/BV-02-C section, added new TC MESH/NODE/IVU/BV-07-C, and updated the initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BV-02-C and -07-C. Added numbering to the MESH/NODE/IVU/BV-03-C section, added new TC MESH/NODE/IVU/BV-08-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BV-03-C and -08-C. Added numbering to the MESH/NODE/IVU/BV-04-C section, added new TC MESH/NODE/IVU/BV-09-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BV-04-C and -09-C.

Publication Number	Revision Number	Date	Comments
			Added numbering to the MESH/NODE/IVU/BV-05-C section, added new TC MESH/NODE/IVU/BV-10-C, and updated the test purpose, initial condition, test procedure, and MSC for MESH/NODE/IVU/BV-05-C and -10-C. Added numbering to the MESH/NODE/IVU/BI-01-C section, added new TC MESH/NODE/IVU/BI-07-C, and updated the test purpose, initial condition, test procedure, and MSC for MESH/NODE/IVU/BI-01-C and -07-C. Added numbering to the MESH/NODE/IVU/BI-02-C section, added new TC MESH/NODE/IVU/BI-08-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BI-02-C and -08-C. Added numbering to the MESH/NODE/IVU/BI-04-C section, added new TC MESH/NODE/IVU/BI-09-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BI-04-C and -09-C. Added numbering to the MESH/NODE/IVU/BI-05-C section, added new TC MESH/NODE/IVU/BI-10-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/IVU/BI-05-C and -10-C. Added numbering to the MESH/NODE/IVU/BI-06-C section, added new TC MESH/NODE/IVU/BI-11-C, and updated the test purpose, initial condition, test procedure, and expected outcome for MESH/NODE/IVU/BI-06-C and -11-C. Added numbering to the MESH/NODE/KR/BV-02-C section, added new TC MESH/NODE/KR/BV-04-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/KR/BV-02-C and -04-C. Added numbering to the MESH/NODE/KR/BV-03-C section, added new TC MESH/NODE/KR/BV-05-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/KR/BV-03-C and -05-C. Added numbering to the MESH/CFGCL/KR/BV-02-C section, added new TC MESH/CFGCL/KR/BV-05-C, and updated the initial condition, test procedure, MSC, and expected outcome for MESH/CFGCL/KR/BV-02-C and -05-C.

Publication Number	Revision Number	Date	Comments
			Added numbering to the MESH/CFGCL/KR/BV-04-C section, added new TC MESH/CFGCL/KR/BV-06-C, and updated the test purpose, initial condition, test procedure, and expected outcome for MESH/CFGCL/KR/BV-04-C and -06-C. Added numbering to the MESH/NODE/KR/BI-05-C section, added new TC MESH/NODE/KR/BI-06-C, and updated the test purpose, initial condition, and test procedure for MESH/NODE/KR/BI-05-C and -06-C. Added numbering to the MESH/NODE/FRND/FN/BV-04-C section, added new TC MESH/NODE/FRND/FN/BV-24-C, and updated the initial condition, test procedure, MSC, and expected outcome for MESH/NODE/FRND/FN/BV-04-C and -24-C. Added numbering to the MESH/NODE/FRND/FN/BV-14-C section, added new TC MESH/NODE/FRND/FN/BV-25-C, and updated the test purpose, initial condition, test procedure, MSC, and expected outcome for MESH/NODE/FRND/FN/BV-14-C and -25-C. Added numbering to the MESH/NODE/FRND/FN/BV-20-C section, added new TC MESH/NODE/FRND/FN/BV-26-C, and updated the initial condition, test procedure, and MSC for MESH/NODE/FRND/FN/BV-20-C and -26-C. Added numbering to the MESH/SR/PROX/BV-02-C section, added new TC MESH/SR/PROX/BV-16-C, and updated the test purpose, reference, initial condition, test procedure, and MSC caption for MESH/SR/PROX/BV-02-C and -16-C. Added numbering to the MESH/SR/PROX/BV-04-C section, added new TC MESH/SR/PROX/BV-17-C, and updated the test purpose, reference, initial condition, and test procedure for MESH/SR/PROX/BV-04-C and -17-C. Added numbering to the MESH/SR/PROX/BV-11-C section, added new TCs MESH/SR/PROX/BV-18-C and -19-C, and updated the test purpose, reference, initial condition, test procedure, and expected outcome for MESH/SR/PROX/BV-11-C, -18-C, and -19-C.

Publication Number	Revision Number	Date	Comments
			Added numbering to the MESH/SR/PROX/BV-12-C section, added new TC MESH/SR/PROX/BV-20-C, and updated the test purpose, reference, initial condition, test procedure, and expected outcome for MESH/SR/PROX/BV-12-C and -20-C. Added numbering to the MESH/SR/PROX/BV-13-C section, added new TC MESH/SR/PROX/BV-21-C, and updated the test purpose, reference, initial condition, test procedure, MSC, and expected outcome for MESH/SR/PROX/BV-13-C and -21-C. Added numbering to the MESH/SR/PROX/BV-14-C section, added new TC MESH/SR/PROX/BV-22-C, and updated the initial condition, test procedure, MSC, and expected outcome for MESH/SR/PROX/BV-14-C and -22-C. Added numbering to the MESH/CL/PROX/BV-08-C section, added new TC MESH/CL/PROX/BV-11-C, and updated the test purpose, initial condition, test procedure, and expected outcome for MESH/CL/PROX/BV-08-C and -11-C. Added numbering to the MESH/CL/PROX/BV-09-C section, added new TC MESH/CL/PROX/BV-12-C, and updated the test purpose, initial condition, and test procedure for MESH/CL/PROX/BV-09-C and -12-C. Added numbering to the MESH/CL/PROX/BV-10-C section, added new TC MESH/CL/PROX/BV-13-C, and updated the test purpose, initial condition, and test procedure for MESH/CL/PROX/BV-10-C and -13-C. Added new TCs MESH/SR/PROX/BV-23-C and -24-C, MESH/SR/MPXS/BV-10-C and -11-C, and MESH/CL/MPXS/BV-07-C and -08-C. Updated the TCMT accordingly.

Publication Number	Revision Number	Date	Comments
			Incorporated changes from MESH_RPR_TEST_CR_r15-E18559-E18520-E18895: Per Errata 18520, 18559, and 18895, added a Remote Provisioning test section with 50 new TCs: MESH/SR/RPR/SCN/BV-01-C - -08-C; MESH/SR/RPR/SCN/BI-01-C and -02-C; MESH/CL/RPR/SCN/BV-01-C; MESH/SR/RPR/LNK/BV-01-C - -07-C, -09-C - -23-C, -25-C, -26-C; MESH/SR/RPR/LNK/BI-01-C - -05-C; MESH/CL/RPR/LNK/BV-01-C; MESH/SR/RPR/PDU/BV-01-C - -05-C; MESH/SR/RPR/PDU/BI-01-C - -03-C; and MESH/CL/RPR/PDU/BV-01-C. Updated the TCMT accordingly. Updated the Acknowledgments list. Incorporated changes from SBR.TS.CRr02-ba4: Added the Mesh Profile Specification v1.1 to the References list. Added Subnet Bridge to the Test groups section. Added abbreviations to the Test case identification conventions table. Added a new setup preamble for State: Adding a bridging table entry, IUT as Subnet Bridge. Updated the expected outcome for MESH/NODE/CFG/COMP/BV-01-C. Added new TC MESH/NODE/DF/INIT/BV-11-C per E18278 and E18504. Added new TCs MESH/NODE/DF/DISC/BV-12-C and MESH/NODE/DF/SOL/BV-05-C. Added new TCs MESH/NODE/DF/DISC/BV-13-C and MESH/NODE/DF/UPDATE/BV-14-C and -15-C per E18504. Added a Subnet Bridge test section with 2 new TCs: MESH/NODE/SBR/NET/BV-01-C and -02-C. Added a Bridge Configuration Models test section with 15 new TCs: MESH/SR/BCM/SBS/BV-01-C, MESH/SR/BCM/SBS/BI-01-C, MESH/CL/BCM/SBS/BV-01-C, MESH/SR/BCM/BDS/BV-01-C - -05-C, and -07-C, MESH/SR/BCM/BDS/BI-01-C - -03-C, MESH/CL/BCM/BDS/BV-01-C, MESH/SR/BCM/BTS/BV-01-C, and MESH/CL/BCM/BTS/BV-01-C. Updated the TCMT accordingly.

Publication Number	Revision Number	Date	Comments
			Incorporated changes from MshPRF_EPA_Test_CR_r03-ba3: For MESH/NODE/PROV/BV-01-C - -08-C, updated the test procedure, expected outcome, and TC descriptions; added an Algorithm column to the TCID table; and added new TCs MESH/NODE/PROV/BV-13-C - -20-C. For MESH/PVNR/PROV/BV-01-C - -08-C, updated the test procedure and TC descriptions, added an Algorithm column to the TCID table, and added new TCs MESH/PVNR/PROV/BV-11-C - -18-C. Updated the test case configuration for MESH/NODE/PROV/BI-03-C. Updated the TCMT and the Acknowledgments list. TSE 18821 (rating 2): Updated the initial condition, test procedure, and MSC and added notes for MESH/NODE/TNPT/BV-13-C. TSE 18823 (rating 2): Updated the expected outcome for MESH/NODE/PROV/BV-01-C - 08-C and -13-C - -20-C, MESH/PVNR/PROV/BV-01-C - -08-C and -11-C - -18-C, MESH/CFGCL/KR/BV-01-C - 06-C, MESH/CFGCL/CFG/NKL/BV-01-C and -02-C, and MESH/CFGCL/CFG/AKL/BV-01-C and -02-C. TSE 18991 (rating 2): Updated the test procedure for MESH/NODE/IVU/BI-06-C and -11-C. TSE 19039 (rating 2): Updated the test procedure for MESH/NODE/IVU/BV-01-C - -10-C; MESH/NODE/IVU/BI-01-C, -02-C, -04-C - -11-C; MESH/NODE/KR/BV-02-C - -05-C; MESH/NODE/KR/BI-05-C and -06-C; and MESH/NODE/FRND/FN/BV-14-C and -25-C and the expected outcome for MESH/NODE/IVU/BI-02-C, -05-C, -08 -C, and -10-C. TSE 20627 (rating 1): Updated the test procedure for MESH/NODE/IVU/BI-05-C and -10-C to define “n” as the current IV index. Performed template-related formatting fixes. Added Section 2.3 to explain provisional gray v1.1 text present in this TS.
11	p11	2023-02-07	Approved by BTI on 2022-12-11. Prepared for TCRL 2022-2 publication.

Publication Number	Revision Number	Date	Comments
	p11ed2r00	2023-02-15	TSE 22511 (rating 1): In the Pass verdict of MESH/SR/MPXS/BV-09-C, reversed the order of the Hash and random values in the expected PDU.
	p11 edition 2	2023-03-15	Approved by BTI on 2023-03-13. Prepared for edition 2 publication.
	p12r00-r05	2023-03-15 - 2023-05-31	TSE 22252 (rating 2): Updated the additional initial conditions for MESH/NODE/FRND/BV-25-C (provisional text) in Table 4.30. TSE 22944 (rating 2): Updated the test procedure for MESH/NODE/IVU/BI-06-C and -11-C. Incorporated Mesh Working Group's edits to provisional v1.1 content. Globally changed "master" to "managed flooding" in new text.
12	p12	2023-06-29	Approved by BTI on 06-01-2023. Prepared for TCRL 2023-1 publication.
	p13r00-r06	2023-06-30 - 2023-08-13	Updated the test procedure, MSC, and expected outcome in MESH/SR/PRB/PNID/BV-02-C and MESH/NODE/CFG/NID/BV-02-C per test issue ES-22859. Updated the additional initial condition for MESH/SR/PROX/BV-11-C, -18-C, and -19-C per test issue ES-22701. Updated the expected outcome for MESH/NODE/CBP/BV-01-C - -04-C and MESH/PVNR/CBP/BI-01-C per test issue ES-22954. Updated the expected outcome for MESH/PVNR/CBP/BI-01-C per test issue ES-23390. Removed provisional formatting in preparation for release with v1.1 spec adoption.
13	p13	2023-09-19	Approved by BTI on 2023-08-27. Mesh Protocol v1.1 adopted by the BoD on 2023-09-12. Prepared for publication.
	p13ed2r00	2023-10-01	TSEs 23953 (rating 1): Updated a cross-reference to a section in the Mesh specification in MESH/NODE/DF/DISC/BV-04-C and -07-C. TSE 23954 (rating 1): Updated the test procedure for MESH/CL/BCM/BDS/BV-01-C. TSE 23955 (rating 1): Updated step 3 of the test procedure in MESH/CL/LCD/LCMP/BV-01-C and MESH/CL/LCD/MMD/BV-01-C. TSE 24065 (rating 1): Added a missing Pass verdict for MESH/NODE/IVU/BV-06-C.

Publication Number	Revision Number	Date	Comments
	p13 edition 2	2023-10-17	Approved by BTI on 2023-10-09. Prepared for edition 2 publication.
	p14r00-r03	2024-02-14 - 2024-03-06	TSE 24034 (rating 2): Updated the MSC and deleted step 3 of the test procedure for MESH/NODE/DF/DISC/BV-09-C - -11-C. TSE 24062 (rating 2): Updated the test procedure, MSC, and expected outcome for MESH/NODE/TNPT/BV-11-C. TSE 24118 (rating 3): Updated the MSC and test procedure for MESH/NODE/DF/SOL/BV-02-C, -03-C, and -05-C. TSE 24124 (rating 2): Updated the TCMT for MESH/CL/PROX/BV-11-C - -13-C. TSE 24178 (rating 2): Updated the section heading for IV Index Recovery tests and the test case configuration table for MESH/NODE/IVU/BV-03-C and -08-C. Updated the test case configuration table for MESH/NODE/IVU/BI-01-C and -07-C. TSEs 24190 (rating 2): In the TCMT, deleted MESH 0/1 from the conditions for MESH/NODE/PROV/BV-01-C - -06-C. TSE 24237 (rating 2): In the TCMT, updated the item conditions and feature description for MESH/SR/DPROX/BV-07-C and MESH/CL/DPROX/BV-02-C. TSE 24280 (rating 2): Updated the MSC and test procedure for MESH/NODE/DF/ECHO/BI-04-C. TSE 24300 (rating 2): In the TCMT, updated the item column for MESH/CL/PRB/PBS/BV-01-C, MESH/CL/PRB/PGPXY/BV-01-C, and MESH/CL/PRB/PNID/BV-01-C. TSE 24301 (rating 1): Updated the initial condition, test procedure, and expected outcome for MESH/NODE/CFG/COMP/BV-01-C. Performed editorial work to align with the current TS template, including updates to the captions for multi-page MSCs.
14	p14	2024-07-01	Approved by BTI on 2024-04-18. Prepared for TCRL 2024-1 publication.
	p14ed2r00	2024-06-27	Baselined from 2024-1 pre-publication document. TSE 23053 (rating 1): Added test cases MESH/NODE/BV-01-C - -03-C in the annex Supplementary Mesh Test Cases for use in IOPT TS testing. Updated the references list.
	p14 edition 2	2024-07-09	Approved by BTI on 2024-07-09. Prepared for edition 2 publication.

Publication Number	Revision Number	Date	Comments
	p15r00-r03	2024-10-21 - 2024-11-22	TSE 24667 (rating 4): Added new test case MESH/NODE/TNPT/BV-22-C. Updated the TCMT accordingly. TSE 25544 (rating 1): Deleted MESH/SR/RPR/LNK/BV-12-C and updated the TCMT accordingly. TSE 25619 (rating 2): Updated the TCMT for MESH/NODE/BV-01-C - -03-C. TSE 25831 (rating 1): Updated the MSC for MESH/SR/BCM/BDS/BV-01-C.
15	p15	2025-02-18	Approved by BTI on 2025-02-05. Prepared for TCRL 2025-1 publication.
	p16r00-r01	2025-02-20 - 2025-04-02	TSE 26747 (rating 3): Per E26350, updated the initial condition, test procedure, MSC, and expected outcome for test cases MESH/NODE/CFG/MAKL/ BV-01-C and -02-C. TSE 27185 (rating 2): Updated the test procedure and pass verdict for MESH/NODE/BV-03-C.
16	p16	2025-07-08	Approved by BTI on 2025-05-28. Prepared for TCRL pkg100 publication.
	p17r00	2025-07-23	TSE 27974 (rating 1): Updated the "Authentication Method" to include size for MESH/PVNR/PROV/BV-01-C, -02-C, -04 -C, -05-C, -11-C, -12-C, -14-C, and -15-C and for MESH/NODE/PROV/BV-01-C, -02-C, -04-C, -05-C, -13-C, -14-C, -16-C, and -17-C. Revised the test procedures for the two related sections accordingly. TSE 27786 (rating 1): Updated the MSC for MESH/NODE/CFG/HBP/BV-02-C.
17	p17	2025-11-04	Approved by BTI on 2025-09-03. Prepared for TCRL pkg101 publication.
	p18r00	2026-05-06	TSE 25351 (rating 3): Updated the pass verdict in MESH/NODE/CFG/COMP/BV-01-C.
18	p18	2026-06-23	Approved by BTI on 2026-06-02. Prepared for TCRL pkg104 publication.

Publication Number	Revision Number	Date	Comments
	p19r00	2026-07-10	TSE 28358 (rating 1): Removed IXIT MESH items no longer used by PTS. Updated references to IXIT items in the setup preambles and the following TCs: MESH/NODE/PROV/BV-01-C – 08-C, -13-C – -20-C, MESH/NODE/CFG/CFGR/BV-01-C, MESH/NODE/CFG/MP/BI-01-C – -03-C, MESH/NODE/CFG/SL/BI-01-C – -06-C, MESH/CFGCL/CFG/SL/BV-01-C and -02-C, MESH/SR/HM/RFS/BV-03-C, MESH/SR/DFM/FTS/BV-05-C, and MESH/SR/AGG/BV-01-C.
19	p19	2026-11-03	Approved by BTI on 2026-10-06. Prepared for TCRL pkg105 publication.

Acknowledgments

Name	Company
Bogdan Alexandru	Bluetooth SIG, Inc.
Alexandru Andreescu	Bluetooth SIG, Inc.
Christopher Badder	Bluetooth SIG, Inc.
Jeff Drake	Bluetooth SIG, Inc.
Andrei Frincu	Bluetooth SIG, Inc.
Jim Harper	Bluetooth SIG, Inc.
Seong-Ho Kim	Bluetooth SIG, Inc.
Tibi Marinescu	Bluetooth SIG, Inc.
Jawid Mirani	Bluetooth SIG, Inc.
Nathaniel Roby	Bluetooth SIG, Inc.
Florin Toma	Bluetooth SIG, Inc.
Josh Toole	Bluetooth SIG, Inc.
Dexter Wong	Bluetooth SIG, Inc.
Per Elmdahl	Ericsson
Omkar Kulkarni	Nordic Semiconductor ASA
Piotr Winiarczyk	Silvair, Inc.
Danilo Blasi	STMicroelectronics