Coordinated Set Identification Service

***Bluetooth*® Service Specification**

- **Version:** v1.1
- **Version Date:** 2025-11-03
- **Prepared by:** Generic Audio Working Group

Abstract

This service specifies how devices can be identified and treated as part of a Coordinated Set.

Version History

Version Number	Date	Comments
v1.0	2021-03-23	Adopted by the Bluetooth SIG Board of Directors.
v1.0.1	2022-06-21	Adopted by the Bluetooth SIG Board of Directors.
v1.1	2025-11-03	Adopted by the Bluetooth SIG Board of Directors.

Acknowledgments

Name	Company
Riccardo Cavallari	Sivantos GmbH
Stefan Mijovic	Sivantos GmbH
Georg Dickmann	Sonova AG
Scott Walsh	Plantronics Inc.
Leif-Alexandre Aschehoug	Nordic Semiconductor ASA
Stephan Gehring	Sonova AG
Oren Haggai	Intel Corporation
Khaled Elsayed	Synopsys, Inc.
Rasmus Abildgren	Bose Corporation
Andrew Credland	Samsung Electronics Co., Ltd.
Jeff Solum	Starkey Hearing Technologies
Nick Hunn	GN Hearing A/S
Masahiko Seki	Sony Corporation
Bjarne Klemmensen	Demant A/S

Contents

1 Introduction

The Coordinated Set Identification Service (CSIS) can be used by devices to be discovered as part of a Coordinated Set.

A Coordinated Set is defined as a group of devices that are configured to support a specific scenario. Examples of Coordinated Sets include a pair of hearing aids, a pair of earbuds, or a speaker set that receives multi-channel audio and that reacts to control commands in a coordinated way (e.g., volume up and volume down). Other examples of Coordinated Sets include a group of sensor nodes (e.g., electrocardiogram (EKG) leads, tire pressure sensors, etc.) that trigger a specific measurement when instructed by a client device.

CSIS is agnostic to the actual features and functions of the devices. The purpose of CSIS is to specify how a device can be discovered as part of a Coordinated Set and how to grant a client exclusive access to the Coordinated Set to avoid race conditions when multiple clients want to access the Coordinated Set at the same time.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

1.1 Conformance

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

1.2 Service dependencies

This service does not depend on any other services.

1.3 Bluetooth Core Specification release compatibility

This service is compatible with the Bluetooth Core Specification Version 4.2 or later [2].

1.4 GATT sub-procedure requirements

Requirements in this section represent a minimum set of requirements for a server. Other Generic Attribute Profile (GATT) sub-procedures may be used if supported by both client and server.

Table 1.1 summarizes additional GATT sub-procedure requirements beyond those required by all GATT servers over Unenhanced Attribute Protocol (ATT) bearers.

GATT Sub-Procedure	Requirements
Read Long Characteristic Values	C.2
Write Characteristic Value	C.1
Notifications	C.1
Read Characteristic Descriptors	C.1

GATT Sub-Procedure	Requirements
Write Characteristic Descriptors	C.1

Table 1.1: GATT sub-procedure requirements, Unenhanced ATT bearers

C.1: Mandatory if Set Member Lock characteristic is supported, otherwise Optional.

C.2: Mandatory if the server supports characteristic values larger than the minimum ATT_MTU, otherwise Optional.

1.5 Transport dependencies

This service uses GATT, and therefore has no additional transport dependencies.

Notifications with GATT are considered unreliable when used with an Unenhanced Attribute Protocol (ATT) bearer. See Volume 3, Part F, Section 3.3.2 in the Bluetooth Core Specification [3].

A higher-layer profile can specify the use of an Enhanced ATT bearer. See Volume 3, Part F, Section 3.3.2 in the Bluetooth Core Specification [3].

1.6 Application error codes

This service defines the Attribute Protocol Application error codes.

Name	Error Code	Description
Lock Denied	0x80	The lock cannot be granted because the server is already locked.
Lock Release Not Allowed	0x81	The lock cannot be released because another client has locked the Coordinated Set.
Invalid Lock Value	0x82	The client attempts to write an RFU value to the Set Member Lock characteristic value.
OOB SIRK Only	0x83	The server only supports exposing the Set Identity Resolving Key (SIRK) via an out-of-band (OOB) procedure.
Lock Already Granted	0x84	The client that made the request is the current owner of the lock.
Value Changed During Read Long	0x85	A characteristic value has changed while a Read Long Characteristic Values sub-procedure is in progress.

Table 1.2: Attribute Protocol Application error codes defined by this service

1.7 Byte transmission order

All characteristics used with this service shall be transmitted with the least significant octet (LSO) first (i.e., little endian). The LSO is identified in the characteristic definitions in [1].

1.8 Change History

This section summarizes changes at a moderate level of detail and should not be considered representative of every change made.

1.8.1 New and updated features

Feature Name	Description	Location
Coordinated Set Name	Added the Coordinated Set Name feature.	Many

Table 1.3: New and/or updated features

1.8.2 Changes from v1.0.1 to v1.1

Section	Errata
1.1: Conformance	23788
1.5: Transport dependencies	18850
5.3.1: Characteristic behavior	18457
8: References	19302

Table 1.4: Errata incorporated in v1.1

1.9 Language

1.9.1 Language conventions

In the development of a specification, the Bluetooth SIG has established the following conventions for use of the terms "*shall*", "*mandatory*", "*shall not*", "*should*", "*should not*", "*may*", "*optional*", "*must*", and "*can*". In this Bluetooth specification, the terms in Table 1.5 have the specific meanings given in that table, irrespective of other meanings that exist.

Term	Definition
shall *or* mandatory	—used to express what is required by the specification and is to be implemented exactly as written without deviation
shall not	—used to express what is forbidden by the specification
should *or* may *or* optional	—not mandatory. Used to express either: 1. what is recommended by the specification without forbidding anything ("should") 2. what is permissible within the limits of the specification (“may” or “optional”)
should not	—used to indicate that something is discouraged but not forbidden by the specification
must	—used to indicate either: 1. an indisputable statement of fact that is always true regardless of the circumstances 2. an implication or natural consequence if a separately-stated requirement is followed

Term	Definition
can	—used to express a statement of possibility or capability

Table 1.5: Language conventions terms and definitions

Where more than one item is permitted but not required, the choices to include or support those items are independent from one another unless the specification explicitly states otherwise. Each item that is implemented shall be implemented exactly as written without deviation.

1.9.2 Reserved for Future Use

Where a field in a packet, Protocol Data Unit (PDU), or other data structure is described as "Reserved for Future Use" (irrespective of whether in uppercase or lowercase), the device creating the structure shall set its value to zero unless otherwise specified. Any device receiving or interpreting the structure shall ignore that field; in particular, it shall not reject the structure because of the value of the field.

Where a field, parameter, or other variable object can take a range of values, and some values are described as "Reserved for Future Use," a device sending the object shall not set the object to those values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous; however, this does not apply in a context where the object is described as being ignored or it is specified to ignore unrecognized values.

When a field value is a bit field, unassigned bits can be marked as Reserved for Future Use and shall be set to 0. Implementations that receive a message that contains a Reserved for Future Use bit that is set to 1 shall process the message as if that bit was set to 0, except where specified otherwise.

The acronym RFU is equivalent to Reserved for Future Use.

1.9.3 Prohibited

When a field value is an enumeration, unassigned values can be marked as “Prohibited.” These values shall never be used by an implementation, and any message received that includes a Prohibited value shall be ignored and shall not be processed and shall not be responded to.

Where a field, parameter, or other variable object can take a range of values, and some values are described as “Prohibited,” devices shall not set the object to any of those Prohibited values. A device receiving an object with such a value should reject it, and any data structure containing it, as being erroneous.

“Prohibited” is never abbreviated.

1.10 Terminology

Table 1.6 defines terms that are needed to understand features used in this service.

Term	Definition
Coordinated Set	A group of devices that are configured to support a specific scenario.
EATT	An ATT bearer feature introduced in Volume 3, Part F, Section 3.2.11 in the Bluetooth Core Specification [3].

Term	Definition
Enhanced ATT bearer	An ATT bearer using the Enhanced Credit Based Flow Control Logical Link Control and Adaptation Protocol (L2CAP) channel mode introduced in Volume 3, Part A, Section 10.2 in [3].
Unenhanced ATT bearer	An ATT bearer not using the Enhanced Credit Based Flow Control L2CAP channel mode introduced in Volume 3, Part A, Section 10.2 in [3].

Table 1.6: Terminology

2 Service

2.1 Declaration

The CSIS shall be instantiated as a «Primary Service». The service universally unique identifier (UUID) shall be set to «Coordinated Set Identification Service» as defined in [1].

2.2 Behavior

As required in Section 5.1, to be discovered as part of a Coordinated Set, devices must share a Set Identity Resolving Key (SIRK), which is used to generate and resolve a random Resolvable Set Identifier (RSI) that can be advertised.

This specification does not include instructions to provision a device with a SIRK or to renew the SIRK across devices that are part of the same Coordinated Set. The SIRK can be set during the manufacturing of the device, or another method could be used.

An RSI is a random set identifier that reduces the ability to track a device over a period by changing on a frequent basis. Devices that are members of the same Coordinated Set use the same SIRK to generate RSIs.

To discover a Coordinated Set and its members, a device discovers at least one member of the Coordinated Set connects to it, and obtains the SIRK. The SIRK can be exposed in encrypted form or in plain text form. The discovering device uses the SIRK to resolve RSIs from other advertising devices and to determine that the advertising devices belong to the Coordinated Set identified by that SIRK when the RSI is successfully resolved.

The SIRK of the Coordinated Set may also be obtained by using an OOB procedure.

This service also specifies the Set Member Lock and Set Member Rank characteristics that are used to minimize race conditions when multiple clients access the Coordinated Set at the same time. The values of the Set Identity Resolving Key, Coordinated Set Size, and Set Member Rank characteristics are read-only; these values can be set during the manufacturing of the device, or another method that is not described in this specification could be used.

3 Advertising data types

This section defines the RSI AD Type and the operations to generate and resolve an RSI. This section also defines the Service Data associated with this service.

3.1 RSI AD Type

3.1.1 Description

An RSI identifies the advertising device as belonging to a Coordinated Set. An RSI shall be generated by using the Resolvable Set Identifier generation operation (Section 4.8) and shall be resolved by using the Resolvable Set Identifier resolution operation (Section 4.9).

3.1.2 Format

Data Type	Description
«Resolvable Set Identifier»	Size: 6 octets. The value of the RSI is generated according to the Resolvable Set Identifier generation operation (Section 4.8).

Table 3.1: RSI AD Type

The value of the Data Type is listed in [1].

3.2 Service Data

The Service Data exposes both the human-readable name of a Coordinated Set instance that the advertising device is a member of and the UUIDs of services that include this instance of CSIS.

The Service Data shall list the UUIDs of all services that include this instance of CSIS.

The Coordinated Set Name string in the Service Data shall be the same as the value of the Coordinated Set Name characteristic (see Section 5.5) in the CSIS instance that is included by the services referenced by the UUIDs contained in the Service Data.

3.2.1 Format

Table 3.2 defines the format of the Service Data.

Parameter		Size (Octets)	Description
Length		1	Length of Type and Value fields
Type: «Service Data – 16-bit UUID»		1	Defined in Bluetooth Assigned Numbers [1]
Value		Varies	
	Coordinated Set Identification Service UUID	2	Defined in Bluetooth Assigned Numbers [1]

Parameter		Size (Octets)	Description	
	Coordinated Set Associations	1	Bitfield	
			Bits 0-1	Number of 16-bit Service UUIDs in this Service Data structure
			Bits 2-3	Number of 32-bit Service UUIDs in this Service Data structure
			Bits 4-5	Number of 128-bit Service UUIDs in this Service Data structure
			Bits 6-7	RFU
	List of 16-bit UUIDs	Varies	UUIDs of services that include CSIS with this Coordinated Set Name	
	List of 32-bit UUIDs	Varies		
	List of 128-bit UUIDs	Varies		
	Name Length	1	Length in octets of the Coordinated Set Name	
	Coordinated Set Name	0-128	UTF-8 encoded string Shall exist only if the Name Length field ≠ 0	

Table 3.2: Service Data

This section describes several examples of the Service Data. In the first example shown in Table 3.3, a Common Audio Service with UUID 0x1853 includes an instance of CSIS with the Coordinated Set Name “Amy’s Earbuds”. The Set Member exposes the association between a Coordinated Set and an including service by setting the 16-bit UUID value 0x1853 in the same Service Data structure as the Coordinated Set Name “Amy’s Earbuds”.

Parameter		Size (Octets)	Value
Length		1	0x14
Type: «Service Data – 16-bit UUID»		1	0x16
Value		Varies	
	Coordinated Set Identification Service UUID	2	0x1846
	Coordinated Set Associations	1	00000001b (0x01) (one 16-bit Service UUID, zero 32-bit, or 128-bit UUIDs)
	List of 16-bit UUIDs	2	0x1853

Parameter		Size (Octets)	Value
	Name Length	1	0x0D
	Coordinated Set Name	13	“Amy’s Earbuds”

Table 3.3: Service Data example for "Amy’s Earbuds"

In the second example shown in Table 3.4, a fictitious service with the 128-bit UUID 0x42DC30535BBD4761A0C6C5237E572E83 includes another CSIS instance with the Coordinated Set Name “Björn’s Set”. A device that is a member of both Coordinated Sets therefore adds both instances of the Service Data, a first instance with the 16-bit UUID value 0x1853 shown in Table 3.3, and a second instance with the 128-bit UUID value 0x42DC30535BBD4761A0C6C5237E572E83 and the Coordinated Set Name “Björn’s Set” shown in Table 3.4 to the Advertising Data or Scan Response Data.

Parameter		Size (Octets)	Value
Length		1	0x23
Type: «Service Data – 16-bit UUID»		1	0x16
Value		Varies	
	Coordinated Set Identification Service UUID	2	0x1846
	Coordinated Set Associations	1	00010000b (0x10) (zero 16 bit or 32-bit UUIDs, one 128-bit UUID)
	List of 128-bit UUIDs	16	0x42DC30535BBD4761A0C6C5237E572E83
	Name Length	1	0x0C
	Coordinated Set Name	12	“Björn’s Set”

Table 3.4: Service Data example for “Björn’s Set”

4 Security toolbox

This section describes the security functions and operations used in this specification.

4.1 Encryption function e

The encryption function *e*, as defined in Volume 3, Part H, Section 2.2.1 in [2], shall be used. This is summarized as:

ciphertext = *e*(key, plaintext)

4.2 CMAC function

RFC4493 [4], also known as AES-CMAC, shall be used. AES-CMAC defines the Cipher-based Message Authentication Code (CMAC) that uses AES-128 as the block cipher function. The inputs to AES-CMAC are:

k is the 128-bit key

m is the variable length data to be authenticated

To generate the 128-bit message authentication code (MAC), the AES-CMAC function shall be used as follows:

$MAC = AES\text{-}CMAC_k (m)$

Inside the *s1* and *k1* functions (see Section 4.3 and Section 4.4), when a multi-octet integer parameter is used as input to AES-CMAC, the most significant octet (MSO) of the integer shall be the first octet of the stream and the LSO of the integer shall be the last octet of the stream. The output of AES-CMAC inside these functions is a multi-octet integer, where the first octet is the MSO, and the last octet is the LSO of this integer.

A device can implement AES functions in the host or can use the HCI_LE_Encrypt command (see Volume 2, Part E, Section 7.8.22 in [2]) to use the AES function in the controller.

4.3 *s1* SALT generation function

The inputs to function *s1* are:

M is a non-zero length octet array or ASCII encoded string

If M is an ASCII encoded string, M shall be converted into an integer number by replacing each string character with its ASCII code preserving the order. For example, if M is the string “CSIS”, M is converted into the integer number: 0x4353 4953.

ZERO is the 128-bit value:

0x0000 0000 0000 0000 0000 0000 0000 0000

The output of the salt generation function *s1* shall be calculated as follows:

$s1(M) = AES\text{-}CMAC_{ZERO} (M)$

Where $AES\text{-}CMAC_{ZERO}$ is the CMAC function defined in Section 4.2.

4.4 *k1* derivation function

The key derivation function *k1* is used to derive a key. The derived key is used to encrypt and decrypt the value of the Set Identity Resolving Key characteristic (see Section 5.1).

The definition of this key generation function uses the MAC function AES-CMAC$_T$ with a 128-bit key T.

The inputs to function *k1* are:

N is 0 or more octets

SALT is 128 bits

P is 0 or more octets

The key (T) shall be computed as follows:

T = AES-CMAC$_{SALT}$(N)

Where AES-CMAC$_{SALT}$ is the CMAC function defined in Section 4.2.

The output of the key generation function *k1* shall be calculated as follows:

k1(N, SALT, P) = AES-CMAC$_T$ (P)

Where AES-CMAC$_T$ is the CMAC function defined in Section 4.2.

4.5 SIRK encryption function *sef*

The SIRK encryption function *sef* shall be used by the server to encrypt the SIRK with a key K. The value of K depends on the transport on which the Set Identity Resolving Key characteristic is read or notified.

If the Set Identity Resolving Key characteristic is read or notified on the Basic Rate/Enhanced Data Rate (BR/EDR) transport, K shall be equal to the Link Key shared by the server and the client.

K = Link Key

If the Set Identity Resolving Key characteristic is read or notified on the Bluetooth Low Energy (LE) transport, K shall be equal to the LTK shared by the server and client. That is,

K = LTK

The inputs to the function *sef* are:

K is the key defined above in this section

SIRK is the value of the SIRK to be encrypted

The output of the SIRK encryption function *sef* is as follows:

sef(K, SIRK) = *k1*(K, s1(“SIRKenc”), “csis”) ^ SIRK

Where ^ is the bitwise exclusive or operation.

4.6 SIRK decryption function *sdf*

The SIRK decryption function *sdf* shall be used by a client to decrypt the SIRK with a key K.

The inputs to the *sdf* function are:

K is the key derived as specified in Section 4.5

EncSIRK is the value of the SIRK in encrypted form

The output of the SIRK decryption function *sdf* is as follows

sdf(K, EncSIRK) = *k1*(K, *s1*("SIRKenc"), "csis") ^ EncSIRK

Where ^ is the bitwise exclusive or operation.

4.7 Resolvable Set Identifier hash function *sih*

The RSI hash function *sih* is used to generate a hash value that is used in RSIs.

The following variables are the inputs to the RSI hash function *sih*:

k is 128 bits

r is 24 bits

padding is 104 bits, all set to 0

r is concatenated with padding to generate r', which is used as the 128-bit input parameter plaintextData to security function e:

r' = padding || r

The LSO of r becomes the LSO of r', and the MSO of padding becomes the MSO of r'.

For example, if the 24-bit value r is 0x3A98B5, then r' is 0x000000000000000000000000003A98B5.

The output of the Resolvable Set Identifier function *sih* is:

sih(k, r) = e(k, r') mod 2^{24}

The output of the security function e is truncated to 24 bits by taking the least significant 24 bits of the output of e as the result of *sih*.

4.8 Resolvable Set Identifier generation operation

A device generates an RSI (resolvableSetIdentifier) by using a 24-bit random number (prand) and the SIRK used to hash prand.

The random number prand shall meet the following requirements:

- The two most significant bits (MSBs) of prand shall be equal to 0 and 1 as shown in Figure 4.1.
- At least one bit of the random part of prand shall be 0.
- At least one bit of the random part of prand shall be 1.

The format of the RSI is shown in Figure 4.1.

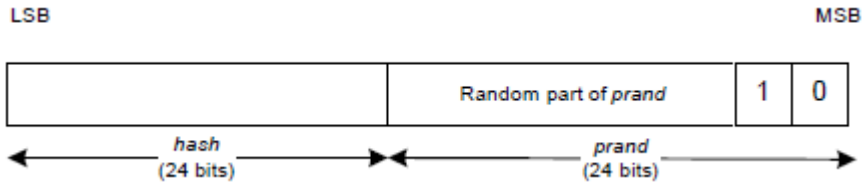

Figure 4.1: Format of RSI

The hash is generated by using the RSI hash function *sih*, defined in Section 4.7, with the input parameter k set to the device's SIRK, and the input parameter r set to prand:

hash = *sih*(SIRK, prand)

The prand and hash are concatenated to generate the RSI resolvableSetIdentifier in the following manner:

resolvableSetIdentifier = hash || prand

Where || means concatenation.

4.9 Resolvable Set Identifier resolution operation

An RSI can be resolved if the corresponding SIRK is available by using the Resolvable Set Identifier resolution operation.

The RSI is divided into a 24-bit random part (prand) and a 24-bit hash part (hash). The LSO of the RSI becomes the LSO of hash, and the MSO of RSI becomes the MSO of prand. A localHash value is then generated by using the random set identifier hash function *sih*, defined in Section 4.7, with the input parameter k set to the SIRK of the known device, and the input parameter r set to the prand value that is extracted from the RSI:

localHash = *sih*(SIRK, prand)

The localHash value is then compared with the hash value extracted from RSI. If the localHash value matches the extracted hash value, then the RSI has been resolved.

5 Service characteristics

This section defines the characteristic and descriptor requirements.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

Characteristic Name	Requirement	Mandatory Properties	Optional Properties	Security Permissions
Set Identity Resolving Key	M	Read	Notify	Encryption required
Coordinated Set Size	O	Read	Notify	Encryption required
Set Member Lock	O	Read, Write, Notify	None	Encryption required
Set Member Rank	C.1	Read	None	Encryption required
Coordinated Set Name	O	Read	Notify	Encryption required

Table 5.1: Coordinated Set Identification Service characteristics

C.1: Mandatory if Set Member Lock is supported, otherwise Optional.

Properties that are not listed as Mandatory or Optional in Table 5.1 are Excluded.

As shown in Table 5.1, the Set Identity Resolving Key, Coordinated Set Size, and Coordinated Set Name characteristics are read-only (mandatory properties) and optionally notifiable (optional properties). If notifiable, their values may be updated via an OOB procedure. If not notifiable, their values shall not change.

5.1 Set Identity Resolving Key

The Set Identity Resolving Key characteristic exposes the SIRK that is associated with the Coordinated Set. All servers that are part of the same Coordinated Set shall use the same SIRK.

The SIRK shall be a 128-bit long random number. The method used to generate the SIRK shall meet the criteria for random number generation as defined in Volume 2, Part H, Section 2 in [2]. A Coordinated Set is uniquely identified by the SIRK.

When the term SIRK is used in this document, it refers to the 128-bit random number described in the immediately preceding paragraph. When referring to the characteristic, the term Set Identity Resolving Key characteristic is used.

The value of the Set Identity Resolving Key characteristic shall be formatted according to Table 5.2.

Field Name	Size (Octets)	Format
Type	1	uint8
Value	16	uint8

Table 5.2: Set Identity Resolving Key characteristic value format

5.1.1 Characteristic behavior

The Set Identity Resolving Key characteristic returns its associated value when read by a client.

A server shall expose the SIRK in encrypted form or in plain text or by using an OOB procedure. A higher-layer specification defines when a server may expose the SIRK in encrypted form or a plain text SIRK or by using an OOB procedure. When using an OOB procedure, the SIRK may be sent in plain text or in encrypted form unless otherwise specified by a higher-layer specification.

If the server exposes a SIRK in encrypted form, the server shall set the Type field value to 0x00 (Encrypted SIRK). The Value field shall be set to the SIRK in encrypted form, calculated with the SIRK encryption function as specified in Section 4.5.

If the server exposes a plain text SIRK in the Value field, the server shall set the Type field value to 0x01 (Plain text SIRK).

If a server exposes a SIRK in encrypted form, the server shall not expose the same SIRK in plain text form.

A server that only exposes a SIRK via an OOB procedure shall reject the request to read the SIRK by replying to the request PDU with an Error Response with the error code OOB SIRK Only.

The characteristic may be configured for notifications by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor.

If a server is provisioned with a new SIRK when connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the new characteristic value to the client.

If a server is provisioned with a new SIRK when not connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the characteristic value when reconnecting to a bonded client.

5.2 Coordinated Set Size

The Coordinated Set Size characteristic exposes the number of devices comprising the Coordinated Set. The value of the Coordinated Set Size characteristic shall be formatted according to Table 5.3.

Allowed values for the Coordinated Set Size characteristic are integers in the range 0x01 to 0xFF. The value 0x00 is Prohibited.

Field Name	Size (Octets)	Format
Coordinated Set Size	1	uint8

Table 5.3: Coordinated Set Size characteristic value format

5.2.1 Characteristic behavior

The Coordinated Set Size characteristic returns its associated value when read by a client.

The characteristic may be configured for notifications by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor.

If the characteristic value changes when connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the new characteristic value to the client.

If the characteristic value changes when not connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the current characteristic value when reconnecting to a bonded client.

5.3 Set Member Lock

The Set Member Lock characteristic allows clients to have exclusive access to the server to minimize race conditions that might occur when multiple clients are accessing multiple servers of the same Coordinated Set at the same time.

A higher-layer specification defines when the Set Member Lock characteristic may be used and which resources are subject to exclusive access.

The value of the Set Member Lock characteristic shall be formatted according to Table 5.4.

Allowed values for the Set Member Lock characteristic are Unlocked (corresponding to the numeric value 0x01) and Locked (corresponding to the numeric value 0x02); all other values are RFU.

Field Name	Size (Octets)	Format
Set Member Lock	1	uint8

Table 5.4: Set Member Lock characteristic value format

5.3.1 Characteristic behavior

The Set Member Lock characteristic may be written by the client.

If a client attempts to write an RFU value to the Set Member Lock characteristic, the server shall reply with an Error Response with the Error Code Invalid Lock Value (see Table 1.2).

The Set Member Lock characteristic returns its associated value when read by a client. A server shall expose the same value of the Set Member Lock characteristic to all clients.

The characteristic can be configured for notifications by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor.

If the characteristic value changes when connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the new characteristic value to the client, unless the characteristic value changed as a consequence of that client writing the new value.

If the characteristic value changes at least once when not connected, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the current characteristic value when reconnecting to a bonded client.

If the server and client are bonded, then the value of the Set Member Lock characteristic shall be persistent across connections within the T_{CSIS}(lock_timeout) (see Section 5.3.1.1).

When the value of the Set Member Lock characteristic is set to Unlocked, the instance of the CSIS is said to be in an “unlocked state;” when the value of the Set Member Lock characteristic is set to Locked, the instance of the CSIS is said to be in a “locked state.”

5.3.1.1 Lock Request

If a client requests to write Locked to the Set Member Lock characteristic (i.e., the client requests the lock), and the value of the Set Member Lock characteristic is equal to Unlocked, then the server shall write the requested value, shall reply with the Write Response (i.e., the server grants the lock to the client), and shall set a timer to T_{CSIS}(lock_timeout).

If a client requests to write Locked to the Set Member Lock characteristic, the value of the Set Member Lock characteristic is equal to Locked, and the lock is granted to the client, then the server shall reply with an Error Response with the error code Lock Already Granted value (see Table 1.2) and shall not reset the timer to T_{CSIS}(lock_timeout).

If a client requests to write Locked to the Set Member Lock characteristic, the value of the Set Member Lock characteristic is equal to Locked, and the lock is not granted to the client, then the server shall reply with an Error Response with the error code Lock Denied value (see Table 1.2) and shall not reset the timer to T_{CSIS}(lock_timeout).

If the timer T_{CSIS}(lock_timeout) expires, then the server shall set the value of the Set Member Lock characteristic to Unlocked. The value of the timer T_{CSIS}(lock_timeout) should be 60 seconds or may be defined by a higher-layer specification. A T_{CSIS}(lock_timeout) value that is too high or too low can lead to undesired behaviors. For example, if a client locks a Coordinated Set to change the volume on the servers, and if the T_{CSIS}(lock_timeout) is set to 1 second, the T_{CSIS}(lock_timeout) could expire before the client has changed the volume on all servers in the Coordinated Set. On the other hand, if the T_{CSIS}(lock_timeout) is set to 360 seconds, clients cannot change the volume on the servers in the Coordinated Set for 6 minutes in case a previous client did not unlock the Coordinated Set.

When the server and the client are not bonded and they disconnect when the value of the Set Member Lock characteristic is set to Locked, the server shall set the value of the Set Member Lock characteristic to Unlocked immediately after the disconnection.

5.3.1.2 Lock Release

If a client requests to write Unlocked to the Set Member Lock characteristic (i.e., the client requests to release the lock), the value of the Set Member Lock characteristic is equal to Locked, and the lock is granted to the client, then the server shall write the requested value and shall reply with the Write Response. The server shall also stop the timer to T_{CSIS}(lock_timeout).

If a client requests to write Unlocked to the Set Member Lock characteristic, the value of the Set Member Lock characteristic is equal to Locked, and the Set Member Lock is not granted to the client, then the server shall not write the value, and the server shall reply with an Error Response with the error code *Lock Release Not Allowed value* (see Table 1.2).

If a client requests to write Unlocked to the Set Member Lock characteristic, and the value of the Set Member Lock characteristic is already set to Unlocked, then the server shall reply with the Write Response.

5.4 Set Member Rank

The Set Member Rank characteristic exposes a numeric value that shall be unique within a Coordinated Set and that allows a client to perform procedures with servers that are part of a Coordinated Set in a defined order (e.g., in order of increasing rank).

Allowed values for the Set Member Rank characteristics are positive integer numbers starting from 0x01.

The value of the Set Member Rank characteristic exposed by servers that are part of the same Coordinated Set shall be contiguous integers starting from 0x01 to the size of the Coordinated Set.

The value of the Set Member Rank characteristic shall be formatted according to Table 5.5.

Field Name	Size (Octets)	Format
Set Member Rank	1	uint8

Table 5.5: Set Member Rank characteristic value format

5.4.1 Characteristic behavior

The Set Member Rank characteristic returns its associated value when read by a client.

5.5 Coordinated Set Name

The Coordinated Set Name characteristic shall be the name of the Coordinated Set. The characteristic value is a UTF-8 string represented by a minimum of 0 octets and a maximum of 128 octets.

5.5.1 Characteristic behavior

The Coordinated Set Name characteristic exposes the name of the Coordinated Set. If notifications are supported, the Coordinated Set Name characteristic can be configured for notifications by using the GATT Write Characteristic Descriptors sub-procedure on the Client Characteristic Configuration descriptor.

If the characteristic value changes when in a connection, and the value of the Client Characteristic Configuration descriptor is configured for notifications, then the server shall notify the new characteristic value to the client. If the characteristic value is greater than (ATT_MTU-3), then the first (ATT_MTU-3) octets shall be included in the notification.

If the characteristic value has changed since the last zero-offset read, the server shall return an ATT Error Response with the application error code Value Changed During Read Long (0x85) as defined in Table 1.2 to any ATT_READ_BLOB_REQ with a non-zero Value Offset requested by the same client.

If the characteristic value changes when not in a connection, and the value of the Client Characteristic Configuration descriptor is configured for notifications, the server shall notify the new characteristic value when reconnecting to a bonded client.

6 SDP interoperability

For services that are exposed over BR/EDR, include a Service Discovery Protocol (SDP) Record table.

Requirements in this section are defined as “Mandatory” (M), “Optional” (O), “Excluded” (X), and “Conditional” (C.*n*). Conditional statements (C.*n*) are listed directly below the table in which they appear.

Item	Definition	Type	Value	Status
Service Class ID List	–	–	–	M
Service Class #0	–	UUID	«Coordinated Set Identification Service»	M
Protocol Descriptor List	–	Data Element Sequence	–	M
Protocol #0	–	UUID	«L2CAP»	M
Parameter #0 for Protocol #0	Protocol/Service Multiplexer (PSM)	uint16	PSM = ATT	M
Protocol #1	–	UUID	«ATT»	M
Additional Protocol Descriptor List	–	Data Element Sequence	–	C.1
Protocol Descriptor List	–	Data Element Sequence	–	C.1
Protocol #0	–	UUID	«L2CAP»	C.1
Parameter #0 for Protocol #0	PSM	uint16	PSM = EATT	C.1
Protocol #1	–	UUID	«ATT»	C.1
BrowseGroupList	–	–	PublicBrowseRoot Other browse UUIDs may also be included in the list.	M

Table 6.1: SDP Record

C.1: Mandatory if Enhanced Attribute Protocol (EATT), introduced in Volume 3, Part F, Section 3.2.11 in [3], is supported, otherwise Excluded.

7 Acronyms and abbreviations

Acronym/Abbreviation	Meaning
ATT	Attribute Protocol
BR/EDR	Basic Rate/Enhanced Data Rate
CMAC	Cipher-based Message Authentication Code
CSIS	Coordinated Set Identification Service
EATT	Enhanced ATT
EKG	electrocardiogram
GATT	Generic Attribute Profile
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy
LSO	least significant octet
MAC	message authentication code
MSB	most significant bit
MSO	most significant octet
OOB	out-of-band
PDU	Protocol Data Unit
PSM	Protocol/Service Multiplexer
RFU	Reserved for Future Use
RSI	Resolvable Set Identifier
SDP	Service Discovery Protocol
SIRK	Set Identity Resolving Key
UUID	universally unique identifier

Table 7.1: Acronyms and abbreviations

8 References

[1] Bluetooth SIG Assigned Numbers, https://www.bluetooth.com/specifications/assigned-numbers

[2] Bluetooth Core Specification (amended) Version 4.2 or later

[3] Bluetooth Core Specification (amended) Version 5.2 or later

[4] Song, et al., RFC4493, “The AES-CMAC Algorithm”, June 2006, https://tools.ietf.org/html/rfc4493

Appendix A. Sample data

In each sample data set in this section, the octets are ordered from most significant on the left to least significant on the right. ‘M’ represents the message byte array for which the AES-128 is calculated.

A.1 *sih* Resolvable Set Identifier hash function

```
SIRK            457d7d09 21a1fd22 cecd8c86 dd72cccd
prand           00000000 00000000 00000000 0069f563
M               00000000 00000000 00000000 0069f563
AES_128         3fb05349 94e48cd8 022b5590 9b1948da
sih             1948da
```

A.2 *sef* SIRK Encryption Function

```
SIRK                            457d7d09 21a1fd22 cecd8c86 dd72cccd
LTK                             676e1b9b d448696f 061ec622 3ce5ced9
K = LTK                         676e1b9b d448696f 061ec622 3ce5ced9
s1("SIRKenc")                   6901983f 18149e82 3c7d133a 7d774572
k1(K, s1("SIRKenc"), "csis")    5277453c c094d982 b0e8ee53 2f2d1f8b
sef(K, SIRK)                    170a3835 e13524a0 7e2562d5 f25fd346
```