Authorization Control Profile

***Bluetooth*® Profile Specification**

- **Revision:** v1.0
- **Revision Date:** 2022-09-13
- **Group Prepared By**: Medical Devices Working Group

Abstract:

This profile enables an Authorization Control (AC) Client to access specific, protected Generic Attribute Profile (GATT) resources of an AC Server.

Revision History

Revision Number	Date	Comments
v1.0	2022-09-13	Adopted by the Bluetooth SIG Board of Directors.

Acknowledgments

Name	Company
Christoph Fischer	F. Hoffmann-La Roche AG
Felix Bootz	F. Hoffmann-La Roche AG
Wolfgang Heck	F. Hoffmann-La Roche AG
Daniel Pletea	Koninklijke Philips N.V.
Leif-Alexandre Aschehoug	Nordic Semiconductor ASA
Laurence Richardson	Qualcomm Technologies, Inc.

Contents

1 Introduction

1.1 Language

1.1.1 Language conventions

The Bluetooth SIG has established the following conventions for use of the words ***shall***, ***must***, ***will***, ***should***, ***may***, ***can***, and ***note*** in the development of specifications:

shall	is required to – used to define requirements.
must	is used to express: a natural consequence of a previously stated mandatory requirement. OR an indisputable statement of fact (one that is always true regardless of the circumstances).
will	it is true that – only used in statements of fact.
should	is recommended that – used to indicate that among several possibilities one is recommended as particularly suitable, but not required.
may	is permitted to – used to allow options.
can	is able to – used to relate statements in a causal manner.
note	Text that calls attention to a particular point, requirement, or implication or reminds the reader of a previously mentioned point. It is useful for clarifying text to which the reader ought to pay special attention. It shall not include requirements. A note begins with “Note:” and is set off in a separate paragraph. When interpreting the text, the relevant requirement shall take precedence over the clarification.

If there is a discrepancy between the information in a figure and the information in other text of the specification, the text prevails. Figures are visual aids including diagrams, message sequence charts (MSCs), tables, examples, sample data, and images. When specification content shows one of many alternatives to satisfy specification requirements, the alternative shown is not intended to limit implementation options. Other acceptable alternatives to satisfy specification requirements may also be possible.

1.1.2 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.

1.1.3 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.

1.2 Conformance

If conformance to this specification is claimed, all capabilities indicated as mandatory for this specification shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated.

1.3 Scope

The Authorization Control Profile (ACP) is used to access protected GATT resources of an Authorization Control (AC) Server that exposes the Authorization Control Service (ACS) [1].

1.4 Profile dependencies

This profile requires GATT.

1.5 Bluetooth specification release compatibility

This specification is compatible with any Bluetooth Core Specification, v4.2 or later [2] that includes GATT.

2 Configuration

2.1 Roles

This profile defines two roles: the AC Server and the AC Client. The AC Server is the device with protected GATT resources, and the AC Client is the device that accesses the protected GATT resources.

The AC Server shall be a GATT Server.

The AC Client shall be a GATT Client.

2.2 Profile/service role relationship

Figure 2.1 shows the relationship between service and profile roles defined by this profile:

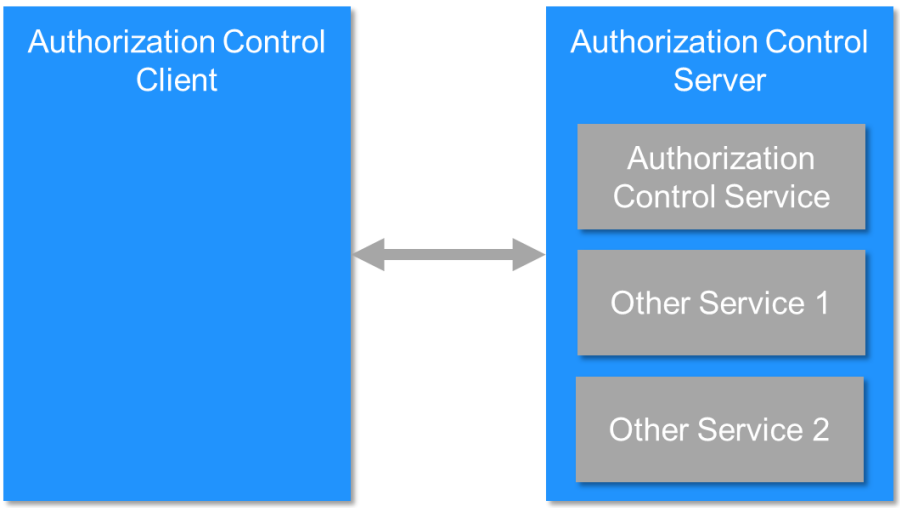


Figure 2.1: Relationship between services and profile roles; the profile roles defined by this profile are represented by the blue boxes, and the services are represented by the gray boxes

This profile is a generic profile that is intended to be referenced by another (higher-level) profile specification with some requirements being overwritten to support the use case (e.g., the required security controls). Figure 2.1 shows the dependency of the higher-level profile specification with two other services named Other Service 1 and Other Service 2.

2.3 Concurrency limitations/restrictions

There are no concurrency limitations or restrictions for the AC Client or AC Server roles defined by this profile.

2.4 Topology limitations/restrictions

There are no topology limitations or restrictions defined by this profile.

2.5 Transport dependencies

There are no transport dependencies defined by this profile.

3 Authorization Control overview

Authorization control supports application layer end-to-end information security by providing the capability of secure Plug & Play interoperability in [3].

This is achieved using the following approach:

- ACS [1] provides a scalable information security toolbox, which includes security controls for key exchange, message authentication codes, and authenticated encryption, among others.
- ACP applies the ACS security toolbox to the personal health device (PHD) domain and use cases, which reduces the ACS toolbox to a set of security controls and makes interoperability possible. ACP is developed taking into account the requirements from the National Institute of Standards and Technology (NIST) [5] and the recommendations of the European Network and Information Security Agency (ENISA) [6]. Other similar higher-level profiles can apply the ACS security toolbox to the same or other domains and use cases.

ACS and ACP are the Bluetooth transport implementations of the transport-independent international consensus standard ISO/IEEE 11073-40102:2022 Health informatics — Device interoperability — Part 40102: Foundational — Cybersecurity — Capabilities for mitigation [4].

4 Authorization Control Server role requirements

The AC Server shall instantiate only one instance of ACS [1] as a primary service.

Service	Authorization Control Server
Authorization Control Service	M

Table 4.1: AC Server service requirements

M: Mandatory

4.1 Incremental Authorization Control Service requirements

4.1.1 Additional requirements

4.1.1.1 Key exchange

The Elliptic Curve Diffie-Hellman (ECDH) key exchange (also called the ECDH key agreement scheme) shall support the NIST-approved [5] and ENISA-recommended [6] elliptic ‘Curve P-256’ and key derivation function (KDF) ‘HKDF SHA-256 128-bit’ based on HMAC-SHA-256 as described in [1].

4.1.1.2 Authentication Encryption with Associated Data

If the AC Server supports confidentiality, integrity, and authentication by setting the value of 1 to the corresponding bits in the Protection_Methods field of the ACS Feature characteristic, the AC Server shall support Galois/Counter Mode (GCM) authenticated encryption. In the case where the AC Server supports authentication and integrity by setting the corresponding bits to a value of 1 and setting the corresponding bit for confidentiality to a value of 0, the AC Server shall use the Galois/Counter Mode Message Authentication Code (GMAC) variant of GCM for authenticated messaging.

The initialization vector (IV) used for GCM and GMAC shall be a fixed field concatenated with an invocation field. The size of the IV fixed field shall be 4 octets. The size of the IV invocation field shall be 8 octets. The IV invocation field shall be implemented as a ‘Sequence Number Different Fixed Parts’.

5 Authorization Control Client role requirements

The AC Client shall support the ACS [1].

Service	AC Client
Authorization Control Service	M

Table 5.1: AC Client service requirements

M: Mandatory

Table 5.2 shows the requirements for support of the discovery of services and characteristics.

Profile Requirement	Section	Support in Collector
Service Discovery	5.2	M
Authorization Control Service	5.2	M
Characteristic Discovery	5.3	M
ACS Status	5.3.1	M
ACS Data In	5.3.2	M
ACS Data Out Notify	5.3.3	M
ACS Data Out Indicate	5.3.4	M
ACS Control Point	5.3.5	M

Table 5.2: AC Client profile requirements

M: Mandatory

5.1 GATT sub-procedure requirements

Requirements in this section represent a minimum set of requirements for an AC Client. Other GATT sub-procedures may be used if supported by both the AC Client and the AC Server.

Table 5.3 summarizes additional GATT sub-procedures requirements beyond those required by all GATT Clients.

GATT Sub-procedure	AC Client Requirements
Discover All Primary Services	C.1
Discover Primary Services by Service UUID	C.1
Discover All Characteristics of a Service	C.2
Discover Characteristics by UUID	C.2
Discover All Characteristic Descriptors	M

GATT Sub-procedure	AC Client Requirements
Notifications	M
Indications	M
Read Characteristic Value	M
Write Characteristic Value	M
Write Long Characteristic Values	O
Read Characteristic Descriptors	M
Write Characteristic Descriptors	M

Table 5.3: Additional GATT sub-procedure requirements

M: Mandatory
O: Optional
C.1: Mandatory to support at least one of the Service Discovery sub-procedures.
C.2: Mandatory to support at least one of the Characteristic Discovery sub-procedures.

5.2 Service discovery

The AC Client shall perform the GATT Service Discovery to discover the services according to Table 5.1.

When using the LE transport, the AC Client shall perform Primary Service Discovery using either the GATT Discover All Primary Services sub-procedure or the GATT Discover Primary Services by Service universally unique identifier (UUID) sub-procedure.

When using the Basic Rate/Enhanced Data Rate (BR/EDR) transport, the AC Client shall perform service discovery by retrieving the Service Discovery protocol (SDP) record of the Authorization Control Service as defined in ACS [1].

5.3 Characteristic discovery

As required by GATT, the AC Client must be tolerant of additional optional characteristics in the service records of services used with this profile.

The AC Client shall perform either the GATT Discover All Characteristic of a Service sub-procedure or the GATT Discover Characteristics by UUID sub-procedure to discover the characteristics of the service.

The AC Client shall perform the GATT Discover All Characteristic Descriptors sub-procedure to discover the characteristic descriptors described in Sections 5.3.1, 5.3.2, 5.3.3, 5.3.4, and 5.3.5.

5.3.1 ACS Status characteristic

The AC Client shall discover the ACS Status characteristic.

The AC Client shall discover the AC Client Characteristic Configuration descriptor of the ACS Status characteristic.

5.3.2 ACS Data In characteristic

The AC Client shall discover the ACS Data In characteristic.

5.3.3 ACS Data Out Notify characteristic

The AC Client shall discover the ACS Data Out Notify characteristic.

The AC Client shall discover the AC Client Characteristic Configuration descriptor of the ACS Data Out Notify characteristic.

5.3.4 ACS Data Out Indicate characteristic

The AC Client shall discover the ACS Data Out Indicate characteristic.

The AC Client shall discover the AC Client Characteristic Configuration descriptor of the ACS Data Out Indicate characteristic.

5.3.5 ACS Control Point characteristic

The AC Client shall discover the ACS Control Point (CP) characteristic.

The AC Client shall discover the AC Client Characteristic Configuration descriptor of the ACS CP characteristic.

5.4 Security controls

This section defines the default security controls for the AC Client.

5.4.1 Key exchange

The AC Client shall support the ECDH key exchange (also called the ECDH key agreement scheme).

5.4.1.1 ECDH key exchange

The ECDH key exchange shall support the NIST-approved [5] and ENISA-recommended [6] elliptic ‘Curve P-256’ and key derivation function ‘HKDF SHA-256 128-bit’ based on HMAC-SHA-256 as described in ACS [1].

The AC Client shall confirm the exchange of the ECDH public key using either No Confirmation OOB Method Used, Confirmation Output OOB Number Action Used, Confirmation Input OOB Number Action Used, or Confirmation Static OOB Number Used as supported by the AC Server. When the AC Client requests the start of the key exchange, the AC Client shall include the selected method to be used.

If the AC Client selects an output action, the AC Client shall prompt the user to input the Output OOB Number provided by the AC Server.

If the AC Client selects an input action, the AC Client shall generate a random Input OOB Number to be observed by the user and entered into the AC Server as described in the selected input action.

ACS [1] defines the algorithms and parameters to calculate the AC Client confirmation code used to confirm the exchanged key.

5.4.2 Authenticated Encryption with Authenticated Data

The AC Client shall support both the GCM and the GMAC variant. The fixed value of the IV is provided by the AC Server.

5.5 Authorization Control Service characteristics

5.5.1 General requirements for characteristics

The ACS characteristics that are exposed by the AC Server are listed in Table 5.2. Refer to ACS [1] for definitions of the format, properties, and behaviors of each characteristic.

If required, the AC Server shall conduct payload segmentation, which is described in detail in ACS [1]. When a payload is segmented, the AC Client shall wait until the complete payload is received and reassembled before processing its contents or wait until a timeout occurs.

The AC Client shall use the characteristics of the AC Server as described in Section 5.5.

5.5.2 ACS Status characteristic behavior

The AC Client may read the ACS Status characteristic to receive the status of the AC Server. The ACS Status characteristic describes the current restriction map ID, whether the AC Server information security controls are enabled, and whether security is established.

The AC Client may configure the AC Client Characteristic Configuration descriptor of the ACS Status characteristic for indications to receive updates to the ACS Status characteristic value.

5.5.3 ACS Data characteristics behavior

The AC Server may support up to three data characteristics: the ACS Data In characteristic used for requests, and the ACS Data Out Notify and the ACS Data Out Indicate characteristics used for responses.

The AC Client may write to the ACS Data In characteristic using the GATT Write Long Characteristic Values sub-procedures, if supported.

The ACS Data In characteristic shall be used by the AC Client to send secure write or read requests to ACS-protected GATT resources. Each secure request sent via the ACS Data In characteristic shall have one or more secure responses sent via one of the ACS Data Out characteristics.

The AC Server shall use the ACS Data Out Notify characteristic to send secure responses from protected GATT resources through notifications. The ACS Data Out Notify characteristic is used for a secure response when the protected resource issues a notification or a read response.

The AC Server shall use the ACS Data Out Indicate characteristic to send secure response indications from protected GATT resources. The ACS Data Out Indicate characteristic is used for a secure response when the protected resource issues an indication.

Before writing to the ACS Data In characteristic, the AC Client shall configure the ACS Data Out Notify or the ACS Data Out Indicate characteristics when supported and appropriate.

The structure of all three ACS Data characteristics are the same and described in detail in ACS [1].

To exchange data securely with a protected GATT resource, the AC Client shall write to the ACS Data In characteristic and receive indications or notifications from the appropriate ACS Data Out Indicate or ACS Data Out Notify characteristics. The Payload fields of the ACS Data characteristics contain the secure payload going to or coming from the protected GATT resource. This payload is secured by the information security configuration identified by the Information_Security_Configuration_ID field. If the total

size of the ATT header, Segmentation header, and Payload field exceeds the ATT_MTU size, the ACS payload is segmented. Therefore, the ACS Data characteristics include a Segmentation_Header field to support segmenting and reconstructing the payload. See Appendix A for protected resource data flow examples using the ACS Data characteristics. For additional details, see ACS [1].

If the exchange of secure data fails, the ACS Data In characteristic provides an application error response, and the ACS Data Out characteristic provides protected resource error responses. See Appendix B for error response examples using the ACS Data characteristic.

5.5.3.1 Response timeout

In the context of the ACS Data Out characteristics, a response to the ACS Data In characteristic request may be segmented. If the AC Client receives a response on an ACS Data Out characteristic that identifies additional segments are remaining, that response is considered to have timed out if the additional segment is not received within the Attribute Protocol (ATT) transaction timeout period, defined as 30 seconds, from the last segment received. If a timeout occurs, the AC Client shall consider the response to have failed.

5.5.4 ACS Control Point characteristic behavior

Before performing any ACS CP procedures, the AC Client shall configure the AC Client Characteristic Configuration descriptor of the ACS CP characteristic for indications.

The AC Client may perform a write to the ACS CP characteristic to request a specific procedure to be executed by the AC Server. This includes an operand with parameters that are valid within the context of that opcode. The opcodes, operands, and associated Server behavior are defined in ACS [1].

Procedures of the ACS CP characteristic may be protected by the AC Server, as described in the restriction map (see Section 5.5.4.3). If a procedure of the ACS CP characteristic is protected, then the AC Client requests the protected procedure on the ACS Data In characteristic (see Section 5.5.3) with applied security controls. For additional details, see ACS [1].

When writing to the ACS CP characteristic, the AC Client shall use the GATT Write Characteristic Value sub-procedure.

Only one ACS CP procedure may be executed at a time, with the exception of the Abort procedure (see Section 5.5.4.13). If the AC Client requests an ACS CP procedure while another procedure is being executed, the AC Server responds with a response error code set to Procedure Already in Progress.

If an error occurs, the AC Server responds by sending the appropriate ATT Error, ATT Application Error, or Response Code opcode with a Response Code Value other than Success, as defined in ACS [1].

5.5.4.1 Opcode support requirements

Table 5.4 shows the requirements for the AC Client to support ACS CP opcodes in the context of this profile.

Procedure (Opcode)	Section	Requirements
Get All Active Descriptors	5.5.4.2	M
Get Restriction Map Descriptor	5.5.4.3	M
Get Restriction Map ID List	5.5.4.4	M

Procedure (Opcode)	Section	Requirements
Activate Restriction Map	5.5.4.5	C.1
Get Resource Handle To UUID Map	5.5.4.6	M
Get Service And Characteristic UUIDs For Characteristic Resource Handle	5.5.4.7	M
Get Information Security Configuration Descriptor	5.5.4.8	M
Get Key Descriptor	5.5.4.9	M
Get Current Key List	5.5.4.10	C.2
Start Key Exchange	5.5.4.11	C.2
Invalidate All Established Security	5.5.4.12	O
Invalidate Key	5.5.4.13	C.2
Abort	5.5.4.14	M
Set Security Controls Switch	5.5.4.15	C.3
Get Key URI	5.5.4.16	C.4
Get ACS Feature	5.5.4.17	M
Key Exchange ECDH	5.5.4.18.1.1	C.2
Key Exchange ECDH Confirmation Code	5.5.4.18.1.2	C.2
Key Exchange ECDH Confirmation Random Number	5.5.4.18.1.3	C.2
Key Exchange KDF	5.5.4.18.2.1	C.2
Set AC Client Nonce Fixed	5.5.4.19	C.2
Get ATT_MTU	5.5.4.20	O
Initiate Pairing	5.5.4.21	O

Table 5.4: ACS CP procedure requirements

M: Mandatory
O: Optional
C.1: Mandatory if the AC Client supports multiple restriction maps, otherwise Excluded.
C.2: Mandatory if the AC Client supports key exchange, otherwise Excluded.
C.3: Mandatory if the AC Client supports setting the state of the security controls switch, otherwise Excluded.
C.4: Mandatory if the AC Client supports out-of-band (OOB) key exchange via URI, otherwise Excluded.

5.5.4.2 Get All Active Descriptors procedure

To get all of the active descriptors (i.e., current restriction map descriptor, information security configuration descriptor, and key descriptor), the AC Client shall write the Get All Active Descriptors opcode with no operand to the ACS CP characteristic.

Writing the Get All Active Descriptors opcode to the ACS CP characteristic triggers the execution of the following AC Server procedures:

1. Get Restriction Map Descriptor procedure (see Section 5.5.4.3)
2. Get Information Security Configuration Descriptor procedure (see Section 5.5.4.7)
3. If supported by the AC Server, Get Key Descriptor procedure (see Section 5.5.4.9)

See Appendix D for an example of the data flow for the Get All Active Descriptors procedure.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with an operand containing a Request Opcode set to the Get All Active Descriptors and a Response Code Value set to Success, which states the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.3 Get Restriction Map Descriptor procedure

To get an unprotected restriction map descriptor, the AC Client shall write the Get Restriction Map Descriptor opcode with an operand containing the desired restriction map ID and resource handle filter criterion to the ACS CP characteristic.

To get a protected restriction map descriptor, the AC Client shall write the Get Restriction Map Descriptor opcode with an operand containing the desired restriction map ID and resource handle filter criterion with applied information security controls to the ACS Data In characteristic.

The AC Client may request to filter the restriction map descriptor by Resource Handle. If a filtered restriction map descriptor is requested, then the AC Server responds with the descriptor records that match the filter criterion. If no records match the filter criterion, then the AC Client receives a Response Code opcode and an operand of the Request Opcode set to Get Restriction Map Descriptor opcode and a Response Code Value set to No Records Found.

If the payload of the Restriction Map Descriptor Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Restriction Map Descriptor Response opcode with an operand containing the last restriction map descriptor record stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23. For an unprotected restriction map, the response comes from the ACS CP characteristic. For a protected restriction map, the response comes from the ACS Data Out Indicate characteristic with applied information security controls.

5.5.4.4 Get Restriction Map ID List procedure

To get a list of the available restriction map IDs and their required Information Security Configuration IDs, the AC Client shall write the Get Restriction Map ID List opcode with no operand to the ACS CP characteristic.

The restriction map ID list is an array of entries. Each entry includes a restriction map ID and the required corresponding Information Security Configuration ID to interact with that restriction map (e.g., activating the restriction map or reading out the restriction map descriptor).

If the payload of the Restriction Map ID List Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the Restriction Map ID List Response opcode with an operand containing restriction map IDs and the corresponding Information Security Configuration IDs

stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.5 Activate Restriction Map procedure

To activate an unprotected restriction map, the AC Client shall write the Activate Restriction Map opcode with an operand containing the restriction map ID to be activated to the ACS CP characteristic.

To activate a protected restriction map, the AC Client shall write the Activate Restriction Map opcode with an operand containing the restriction map ID to be activated with applied information security controls to the ACS Data In characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with an operand containing the Requested Opcode set to Activate Restriction Map opcode and a Response Code Value set to Success, which states the successful completion of the procedure requested, or an error value as described in Section 5.5.4.23. For an unprotected restriction map, the response comes from the ACS CP characteristic. For a protected restriction map, the response comes from the ACS Data Out Indicate characteristic with applied information security controls.

5.5.4.6 Get Resource Handle To UUID Map procedure

To get the map between the AC Server defined Resource Handle and Attribute UUID, the AC Client shall write the Get Resource Handle To UUID Map opcode with no operand to the ACS CP characteristic.

If the payload of the Resource Handle To UUID Map Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Resource Handle To UUID Map Response opcode with an operand containing the last map record stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.7 Get Service And Characteristic UUIDs For Characteristic Resource Handle procedure

To get the service and characteristic UUIDs mapped to a specific AC Server defined Resource Handle, the AC Client shall write the Get Service And Characteristic UUIDs For Characteristic Resource Handle opcode with an operand containing the Resource Handle of a characteristic to the ACS CP characteristic.

If the payload of the Service And Characteristic UUIDs For Characteristic Resource Handle Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Service And Characteristic UUIDs For Characteristic Resource Handle Response opcode with an operand containing the last map record stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.8 Get Information Security Configuration Descriptor procedure

To get the information security configuration descriptor, the AC Client shall write the Get Information Security Configuration Descriptor opcode with an operand containing the information security configuration filter criterion to the ACS CP characteristic.

The AC Client may request to filter the information security configuration descriptor by the Information Security Configuration ID. If a filtered information security configuration descriptor is requested, then the AC Server responds with the descriptor records that match the filter criterion. If no records match the filter

criterion, then the AC Client receives a Response Code opcode and an operand of Request Opcode set to Get Information Security Configuration Descriptor opcode and a Response Code Value set to No Records Found.

If the payload of the Information Security Configuration Descriptor Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Information Security Configuration Descriptor Response opcode with an operand containing the last information security configuration descriptor record stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.9 Get Key Descriptor procedure

To get the key descriptor, the AC Client shall write the Get Key Descriptor opcode with an operand containing the key filter criterion to the ACS CP characteristic.

The AC Client may request to filter the key descriptor by key ID. If the AC Client requests to filter the key descriptor, then the AC Server responds with the descriptor records that match the filter criterion. If no records match the filter criterion, then the AC Client receives a Response Code opcode and an operand of Request Opcode set to Get Key Descriptor opcode and a Response Code Value set to No Records Found.

If the payload of the Key Descriptor Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Key Descriptor Response opcode with an operand containing the last key descriptor record stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.10 Get Current Key List procedure

To get the current list of IDs of valid keys between the AC Client and the AC Server, the AC Client shall write the Get Current Key List opcode with no operand to the ACS CP characteristic.

If the payload of the Current Key List Response operand exceeds the ATT_MTU size, the AC Client shall behave as described in Section 5.5.1.

The AC Client shall wait (see Section 5.5.4.22) for the last Current Key List Response opcode with an operand containing the last key ID stating the successful completion of the procedure requested or an error value as described in Section 5.5.4.23.

5.5.4.11 Start Key Exchange procedure

To indicate the start of a key exchange and confirmation, the AC Client shall write the Start Key Exchange opcode with an operand containing the key ID that will be exchanged, the selected confirmation method, and the selected confirmation action to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Request Opcode set to Start Key Exchange and the Response Code Value set to Success, which states the successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

In the case of an error during the key exchange and confirmation, the AC Server responds with the Key Exchange Response opcode and an operand containing the ID of the key and the response code Key

Exchange Failed. If the AC Client wants to retry the key exchange procedure after an error, then the AC Client shall do so from the beginning (i.e., write the Start Key Exchange opcode with an operand to the ACS CP characteristic) because no recovery procedure is defined.

For further details on exchanging keys, the corresponding ACS CP procedure, example data flows, and test vectors, see ACS [1].

Upon successful key exchange and confirmation, the AC Client shall wait (see Section 5.5.4.22) for the Key Exchange Response opcode with an operand containing the ID of the key and the response code Key Exchanged Successful.

5.5.4.12 Invalidate All Established Security procedure

To invalidate all of the established security for all AC Clients (e.g., expired exchanged keys), the AC Client shall write the Invalidate All Established Security opcode with no operand to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Response Code Value set to Success, which states the successful reception of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.13 Invalidate Key procedure

To invalidate a key for the requesting AC Client, the AC Client shall write the Invalidate Key opcode followed by an operand consisting of the Key_ID field to the ACS CP characteristic.

The AC Client may request that an exchanged key, a derived key, or all keys for the requesting AC Client be invalidated by setting the value of the Key_ID field to the specific key or to 0xFFFF. If no records match the key ID included in the request, then the AC Client will receive a Response Code opcode and the operand of the Request Opcode set to Invalidate Key opcode and the Response Code Value set to No Records Found.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Response Code Value set to Success, which states the successful reception of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.14 Abort procedure

To abort a procedure that the AC Client initiated, the AC Client shall write the Abort opcode with no operand to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Response Code Value set to Success, which states the successful aborting of the procedure, or an error value as described in Section 5.5.4.23. Although the AC Server stops the data transfer after the AC Server sends the Response Code Value of Success, the AC Server might still have some records queued for transfer. The queued records might be sent before the transfer is fully terminated. The AC Client might choose to process or ignore these additional responses.

The Request Opcode in the operand of the Response Code opcode is used to determine if the AC Server response is received in response to an Abort procedure or to the procedure that the Abort procedure is trying to abort. If the Abort procedure is completed successfully, then the AC Client receives the Response Code notification with the Request Opcode in the operand set to Abort but will not receive a response for the aborted procedure.

The AC Client may also receive a Response Code opcode with the Request Opcode in the operand set to Abort and the Response Code Value representing an error condition that occurred in processing the request. Though in practice not all Response Code Values may be returned for an Abort procedure, an AC Client shall handle receiving all defined Response Code Values in response to this procedure.

5.5.4.15 Set Security Controls Switch procedure

To enable or disable ACS information security controls, the AC Client shall write the Set Security Controls Switch opcode followed by an operand consisting of the Switch_State field to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Response Code Value set to Success, which states the successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.16 Get Key URI procedure

To receive a Uniform Resource Identifier (URI) to request the AC Server key OOB, the AC Client shall write the Get Key URI opcode with an operand containing the key ID to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Key URI Response opcode with an operand containing the key URI, stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.17 Get ACS Feature procedure

To get the features of the AC Server, the AC Client shall write the Get ACS Feature opcode with no operand to the ACS CP characteristic.

The AC Client shall request the Get ACS Feature procedure to determine the supported features of the AC Server to interpret the supported functionalities, the protection methods provided by the AC Server, the possible out-of-band (OOB) key exchange methods, and the input and output (I/O) capabilities of the AC Server. The provided information allows the AC Client to operate efficiently.

- If the Set Security Controls Switch Supported bit is set to a value of 1, the AC Client may turn off or turn on the security controls switch of the ACS using the ACS CP characteristic procedure. Otherwise, the AC Client cannot alter the state of the security controls switch.
- If the Key Exchange Supported bit is set to a value of 1, the AC Client can exchange keys with the AC Server when required by the information security controls to access protected resources. Otherwise, the AC Client shall not exchange keys with AC Server.
- If the Multiple Restriction Maps Supported bit is set to a value of 1, the AC Client should configure the ACS Status characteristic for indication to detect when the restriction map has changed. Otherwise, the AC Client might permanently cache the restriction map received from the AC Server.
- If the Resource Handle to UUID Map Supported bit is set to a value of 1, the AC Client can request a mapping between the AC Server defined Resource Handles and Attribute UUIDs, if required.
- If the Descriptors Supported bit is set to a value of 1, the AC Client can request the descriptors supported by the AC Server.
- If the Initiation of Pairing Supported bit is set to a value of 1, the AC Client can request the AC Server to initiate pairing.
- If the OOB Key Exchange Supported bit is set to a value of 1, the AC Client can get the AC Server key using an OOB method.

- If the Key URI Supported bit is set to a value of 1, the AC Client can get the AC Server key information using the AC Server provided URI.
- If the Invalidate Established Security Supported bit is set to a value of 1, the AC Client can request the AC Server to invalidate established security.
- If the ATT_MTU Supported bit is set to a value of 1, the AC Client can get the AC Server to provide the negotiated ATT_MTU size.
- If the ECDH Key Exchange Supported bit is set to a value of 1, the AC Client can exchange its public key and calculate a symmetric key using ECDH.
- If the Protected Resource uses Write Request bit is set to a value of 1, the AC Client can send a write request on the ACS Data In characteristic to write to a protected resource.
- If the Protected Resource uses Read Request bit is set to a value of 1, the AC Client can send a read request on the ACS Data In characteristic and receive a read response from the protected resource on the ACS Data Out Notify characteristic.
- If the Protected Resource uses Notification bit is set to a value of 1, the AC Client can receive a notification from a protected resource on the ACS Data Out Notify characteristic.
- If the Protected Resource uses Indication bit is set to a value of 1, the AC Client can receive an indication from a protected resource on the ACS Data Out Indicate characteristic.

The protection methods defined by the AC Server describe the type of information security controls that are supported by the AC Server.

The I/O capabilities of the AC Server could be used as part of a key exchange confirmation to help protect against man-in-the-middle attacks.

The AC Client shall wait (see Section 5.5.4.22) for the ACS Feature Response opcode with an operand containing the ACS Feature, stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.18 Key Exchange procedures

5.5.4.18.1 OOB and ECDH key exchange

To exchange a key using the OOB or ECDH key exchange, the AC Client shall request the AC Server to complete the following procedures:

1. Start Key Exchange procedure (see Section 5.5.4.11)
2. Key Exchange ECDH procedure (see Section 5.5.4.18.1.1)
3. Key Exchange KDF procedure (see Section 5.5.4.18.2.1)
4. Key Exchange ECDH Confirmation Code procedure (see Section 5.5.4.18.1.2)
5. Key Exchange ECDH Confirmation Random Number procedure (see Section 5.5.4.18.1.3)

For additional details on exchanging keys, the corresponding ACS CP procedure, example data flows, and test vectors, see ACS [1].

See Appendix E for an initial communication example and Appendix C for key exchange examples.

The AC Client shall wait (see Section 5.5.4.22) for the Key Exchange Response opcode with an operand containing the ID of the key and the response code Key Exchanged Successful or an error value as

described in Section 5.5.4.23. If the AC Client wants to retry the key exchange procedure after an error, then the AC Client shall do so from the beginning, because no recovery procedure is defined.

5.5.4.18.1.1 Key Exchange ECDH procedure

To exchange the public keys to be used with ECDH, the AC Client shall write the Key Exchange ECDH opcode with an operand containing the key ID and the AC Client public key to the ACS CP characteristic.

The AC Client application validates the AC Server public key as part of the ECDH key agreement scheme. If the public key provided by the AC Server fails key validation as described for P-256, P-384, and P-521 in [7] and for Curve25519 in [8] (e.g., the key is not on the curve), then the AC Client may retry the key exchange procedures from step 1 of the OOB and ECDH key exchange (e.g., to try another key) [1] or may stop the key exchange.

The AC Client shall wait (see Section 5.5.4.22) either for the Key Exchange ECDH Response opcode with an operand containing the key ID and AC Server public key (i.e., the AC Server public key is exchanged in-band) or for the Response Code opcode with an operand containing the Response Code Value set to Success (i.e., the AC Server public key is exchanged OOB), stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.18.1.2 Key Exchange ECDH Confirmation Code procedure

To exchange the confirmation codes used to confirm the exchange ECDH key, the AC Client shall write the Key Exchange ECDH Confirmation Code opcode and an operand containing the key ID and AC Client confirmation code to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Key Exchange ECDH Confirmation Code Response opcode with an operand containing the key ID and AC Server confirmation code stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.18.1.3 Key Exchange ECDH Confirmation Random Number procedure

To exchange the confirmation random numbers used to confirm the exchange ECDH key, the AC Client shall write the Key Exchange ECDH Confirmation Random Number opcode and an operand containing the key ID and AC Client confirmation random number to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Key Exchange ECDH Confirmation Random Number Response opcode with an operand containing the key ID and AC Server confirmation random number stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

The AC Client should validate the confirmation of the calculated symmetric key. If validation fails, then the AC Client may retry the key exchange procedures from step 1 of the OOB and ECDH key exchange (e.g., to try another key) [1] or may stop the key exchange.

5.5.4.18.2 KDF key exchange

To exchange a key using the KDF key exchange, the AC Client shall request the AC Server to complete the following procedures:

1. Start Key Exchange procedure (see Section 5.5.4.11)
2. Key Exchange KDF procedure (see Section 5.5.4.18.2.1)

For additional details on exchanging keys, the corresponding ACS CP procedure, example data flows, and test vectors, see ACS [1].

The AC Client shall wait (see Section 5.5.4.22) for the Key Exchange Response opcode with an operand containing the ID of the key and the response code Key Exchanged Successful or an error value as described in Section 5.5.4.23. If the AC Client wants to retry the key exchange procedure after an error, then the AC Client shall do so from the beginning, because no recovery procedure is defined.

5.5.4.18.2.1 Key Exchange KDF procedure

The Key Exchange KDF procedure may be called as part of a KDF key exchange or an ECDH key exchange.

To start the key derivation, the AC Client shall write the Key Exchange KDF opcode with an operand containing the key ID to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Key Exchange KDF Response opcode with an operand containing the key ID, KDF salt, and KDF information, stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

The AC Client shall derive the key with the parameters provided in the Key Exchange KDF Response. If the key derivation fails, then the AC Client may retry the key exchange procedures from step 1 of the OOB and ECDH key exchange (e.g., to try another key) [1] or may stop the key exchange.

5.5.4.19 Set AC Client Nonce Fixed procedure

To set the AC Client nonce fixed, the AC Client shall write the Set AC Client Nonce Fixed opcode and an operand containing the key ID and AC Client nonce fixed value to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with an operand containing the Requested Opcode set to Set AC Client Nonce Fixed opcode and a Response Code Value set to Success stating the successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.20 Get ATT_MTU procedure

To get the negotiated ATT_MTU value from the AC Server, the AC Client shall write the Get ATT_MTU opcode with no operand to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the ATT_MTU Response opcode with an operand containing the (ATT_MTU – 3) value stating successful completion of the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.21 Initiate Pairing procedure

To initiate the pairing procedure described in the Bluetooth Core Specification, v4.2 or later [2], the AC Client shall write the Initiate Pairing opcode with no operand to the ACS CP characteristic.

The AC Client shall wait (see Section 5.5.4.22) for the Response Code opcode with the Response Code Value set to Success stating receiving the procedure requested, or an error value as described in Section 5.5.4.23.

5.5.4.22 Procedure timeout

In the context of the CP characteristic, a procedure begins when the AC Server sends the write response to the AC Client write request. The procedure is considered to be completed when the CP characteristic responds with an indication of either an opcode set to Response Code or an opcode set to the corresponding response of the procedure followed by the required operand.

A procedure is considered to have timed out if a notification is not received within the ATT transaction timeout period, defined as 30 seconds, from the start of the procedure or from the last notification that was received as a result of the procedure.

If the link is lost while a CP procedure is in progress, the procedure shall be considered to have timed out.

5.5.4.23 Specific errors

If the AC Client writes an opcode to the CP characteristic and the AC Client Characteristic Configuration descriptor of the CP is not configured for indications, the AC Server will send an ATT Error Response with the Error Code set to AC Client Characteristic Configuration Descriptor Improperly Configured. To execute the procedure associated with the opcode, the AC Client shall configure the AC Client Characteristic Configuration descriptor of the CP characteristic for indications and may write the opcode to the CP characteristic again.

If the AC Client writes an opcode to the CP characteristic that is not supported by the AC Server (i.e., an RFU value) or does not refer to a feature (i.e., not an optional opcode), then the AC Server will send a Response Code opcode response with the Response Code Value set to Opcode Not supported.

If the AC Client writes an operand to the CP characteristic that is invalid (e.g., invalid structure, field with RFU value), then the AC Server will send a Response Code opcode response with the Response Code Value set to Invalid Operand.

If the AC Client writes to the CP characteristic and the procedure cannot be completed for any reason, then the AC Server will send a Response Code opcode response with the Response Code Value set to Procedure Not Completed.

If the AC Client writes any opcode, with the exception of the Abort opcode, to the CP characteristic procedure, and another CP procedure is running, the AC Server will send an ATT Error Response with the error code set to Procedure Already in Progress, and the AC Client should wait until the procedure completes before executing a new procedure.

If the AC Client writes to the CP characteristic and the procedure is protected, then the AC Server will send an ATT Error Response with the Error Code set to Insufficient Authorization.

If the AC Client writes an operand to the CP characteristic that contains a field with a value that is outside the range of values supported by the AC Sever, then the AC Server will send a Response Code opcode response with a Response Code Value set to Parameter Out Of Range.

If the AC Client writes to the CP characteristic and the procedure cannot be executed because the procedure is not applicable in the current AC Server context or the procedure refers to a feature that is not supported by the AC Server (i.e., the corresponding feature bit is not set in the ACS Feature characteristic Features field or an optional opcode is not supported), the AC Server will send a Response Code opcode response with the Response Code Value set to Procedure Not Applicable.

If the AC Client writes the Abort opcode to the CP characteristic and the running procedure cannot be aborted for any reason not stated in this section, then the AC Server will send a Response Code opcode response with a Response Code Value set to Abort Unsuccessful.

If the AC Client writes an operand to the CP characteristic that requests records that do not exist on the AC Server, then the AC Server will send a Response Code opcode response with a Response Code Value set to No Records Found.

If the AC Client writes a confirmation random number that does not correspond to the previously written confirmation code, then the AC Server will send a Response Code opcode response with a Response Code Value set to Invalid Key Exchange Confirmation Code.

If the AC Client writes an invalid public key, then the AC Server will send a Response Code opcode with a Response Code Value set to Invalid Public Key.

The AC Client behavior in the case of a received error, as described in this section, is left to the implementation.

6 Connection establishment procedures

The ACP is a generic profile that is intended to be referenced by another higher-level profile specification. The connection establishment procedures specified by the higher-level profile specification shall be followed.

7 Security considerations

This section describes the security considerations for an AC Server and an AC Client.

7.1 AC Server security considerations for Low Energy

This section describes the security requirements for the AC Server using an LE transport.

The ACS characteristics of the AC Server should be set to the same security mode and level as the other characteristics referred to by the higher-level profile specification that references this profile.

7.2 AC Client security considerations for Low Energy

This section describes the security requirements for the AC Client using an LE transport.

The AC Client may bond with the AC Server.

The AC Client shall accept any request by the AC Server for LE Security Mode 1 and Security Level 1 or higher.

7.3 Security consideration for BR/EDR

As required by the Generic Access Profile (GAP), Security Mode 4 (security level enforced security) shall be used for connections by the AC Server and the AC Client.

8 Acronyms and abbreviations

Any abbreviation or acronym used in the document, but not defined in the common specification sections (e.g., Volume 1 Part B in [2]), is defined in Table 8.1.

Acronym/Abbreviation	Meaning
AC	Authorization Control
ACP	Authorization Control Profile
ACS	Authorization Control Service
ATT	Attribute Protocol
BR/EDR	Basic Rate / Enhanced Data Rate
CP	Control Point
ECDH	Elliptic Curve Diffie-Hellman
ENISA	European Network and Information Security Agency
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GCM	Galois/Counter Mode
GMAC	Galois/Counter Mode Message Authentication Code
IV	Initialization Vector
KDF	key derivation function
LE	Low Energy
NIST	National Institute of Standards and Technology
OOB	out-of-band
PHD	personal health device
SDP	Service Discovery Protocol
URI	Uniform Resource Identifier
UUID	universally unique identifier

Table 8.1: Acronyms and abbreviations

9 References

[1] Authorization Control Service (ACS), Version 1.0

[2] Bluetooth Core Specification, Version 4.2 or later

[3] White Paper - IEEE PHD Cybersecurity Standards Roadmap, April 2019, https://ieeexplore.ieee.org/servlet/opac?punumber=8703256

[4] ISO/IEEE 11073-40102:2022 Health informatics — Device interoperability — Part 40102: Foundational — Cybersecurity — Capabilities for mitigation, https://www.iso.org/standard/83503.html

[5] NIST Framework for Improving Critical Infrastructure Cybersecurity, April 2018, https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf

[6] ENISA Algorithms, key size and parameters report – 2014, https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014/at_download/fullReport

[7] NIST SP 800-56A Rev. 3 – Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar3.pdf

[8] NIST SP 800-186 (Draft) – Recommendations for Discrete Logarithm-Based Cryptography: Elliptic Curve Domain Parameters, https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-186-draft.pdf

Appendix A Exchanging data securely

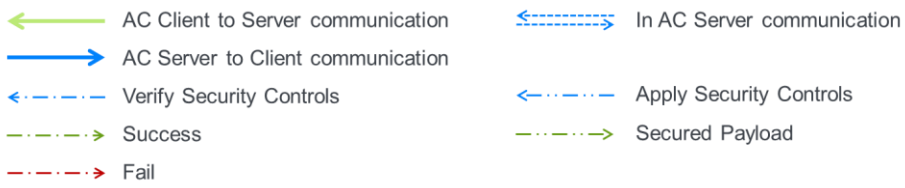
The following data flow examples show how data can be exchanged securely between the AC Client and an ACS protected resource. For these examples, Service X (SX) has a Characteristic Y (CY) and a CP that are protected by the AC Server. Characteristic Y supports Read and Write properties, and the CP supports Write and Indicate properties. Because the CP supports Indications, the ACS Data Out Indicate characteristic is used for responses from the CP. In each of the following examples, the Bluetooth Low Energy (LE) interface and application logic have been separated in the AC Server, and the application security module is responsible for validating the information security controls that are used to protect SX resources.

Interactions between the AC Client and ACS interface in the Bluetooth LE stack have the following nomenclature:

<ACS Data characteristic> <GATT sub-procedure> - <Service X characteristic> <GATT sub-procedure>

Where the Service X characteristic and following GATT sub-procedure are the embedded ATT packet within the secure payload of the ACS Data characteristic.

The legend below applies to the following examples:



A.1 Securely Read Characteristic Value

In this example, the AC Client reads the value of the protected resource CY, and Figure A.1 shows the data flow. In this case, the ACS Data In characteristic is used for the read request, and the ACS Data Out Notify characteristic is used for the read response.

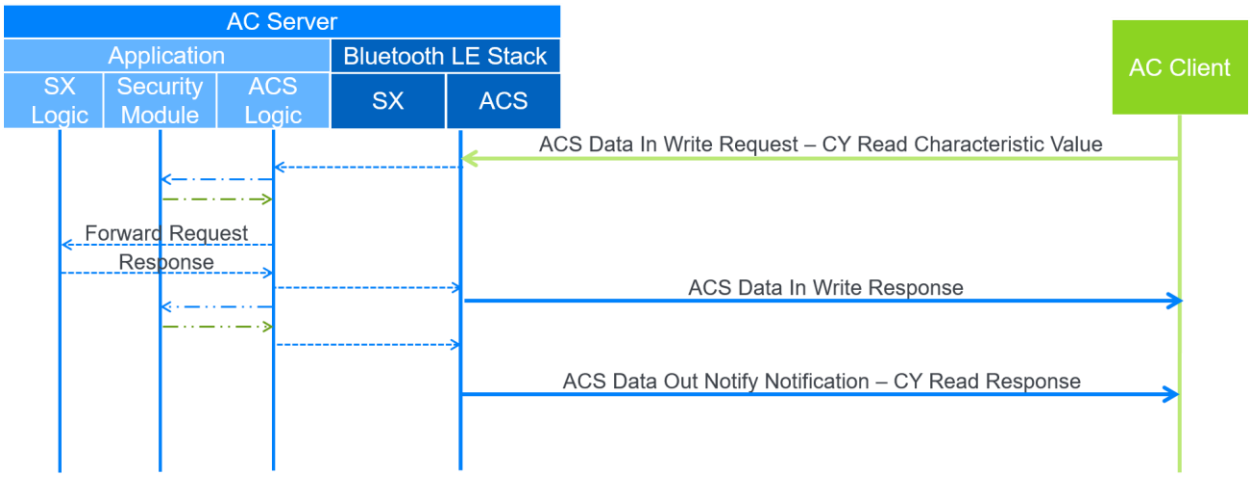


Figure A.1: Data flow to securely read a characteristic value; the Bluetooth LE interface and application logic are separated in the AC Server and show potential application layer communications

A.2 Securely Write Characteristic Value

In this example, the AC Client writes the value of the protected resource CY, and Figure A.2 shows the data flow. In this case, the ACS Data In characteristic is used for the write request and the ACS Data Out characteristic is not used.

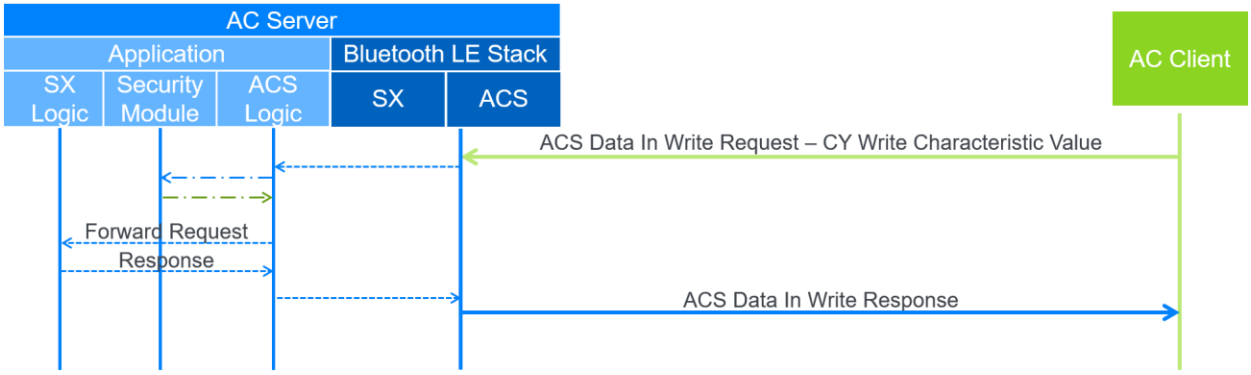


Figure A.2: Data flow to securely write a characteristic value; the Bluetooth LE interface and application logic are separated in the AC Server and show potential application layer communications

A.3 Securely Execute CP procedure

In this example, the AC Client writes to the SX CP to execute a protected procedure, and Figure A.3 shows the data flow. In this case, the ACS Data In characteristic is used to request the execution of the protected procedure and the ACS Data Out Indicate characteristic is used for the SX CP response, because the SX CP supports Indications. The ACS Logic module tracks when the Handle Value Confirmation is reported back to the SX Logic module.

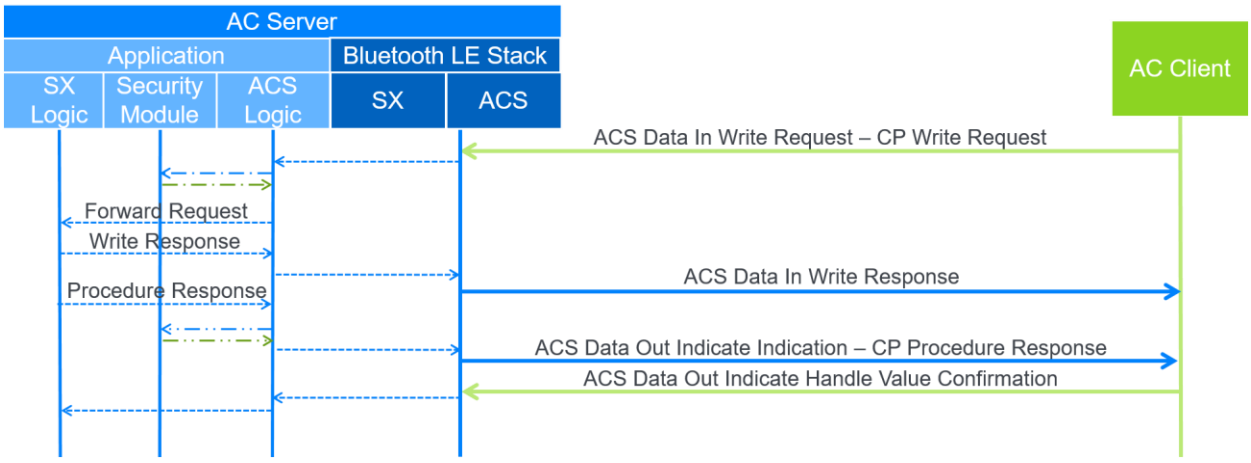


Figure A.3: Data flow to securely execute a protected CP procedure; the Bluetooth LE interface and application logic are separated in the AC Server and show potential application layer communications

Appendix B Application error response

The following data flow examples show how application error responses are sent to the AC Client from either the AC Server or a protected resource. For these examples, Service X (SX) has a Characteristic Y (CY) and a CP that are protected by the AC Server. Characteristic Y supports Read and Write properties, and the CP supports Write and Indicate properties. Because the CP supports Indications, the ACS Data Out Indicate characteristic is used for responses from the CP. In each of the following examples, the Bluetooth LE interface and application logic are separated in the AC Server, and the application security module is responsible for validating the information security controls used to protect the SX resources.

Interactions between the AC Client and ACS interface in the Bluetooth LE stack have either of the following nomenclatures:

<ACS Data characteristic> <GATT sub-procedure> - <Service X characteristic> <GATT sub-procedure>
or
<ACS Data characteristic> <GATT sub-procedure> - <ATT Error Response>

Where the Service X characteristic and following GATT sub-procedure or the ATT Error Response are the embedded ATT packet within the secure payload of the ACS Data characteristic.

The legend below applies to the following examples:

B.1 ACS ATT error response

In this example, the AC Client writes to the CY, but the AC Client includes the incorrect security configuration within the secure payload. Therefore, the application security module fails validation of the security controls and the AC Server will respond with an ATT Error Response. Figure B.1 shows the data flow. In this case, the ACS Data In characteristic is used for the write request and the ACS Data Out characteristic is not used.

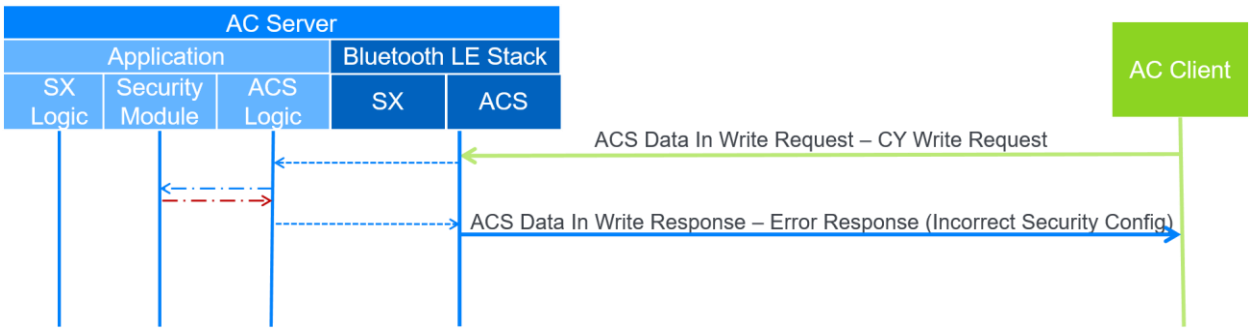


Figure B.1: Data flow of the ACS ATT Error Response

B.2 Protected Resource Error Response

In this example, the AC Client writes to the CP to execute a protected procedure, but the context of the Service controlling the CP cannot execute the procedure, and therefore provides an error response. Figure B.2 shows the data flow. In this case, the ACS Data In characteristic is used for the write request to execute the protected procedure, and the ACS Data Out Indicate characteristic is used for the CP response because the CP supports Indications. The ACS Logic module tracks when the Handle Value Confirmation should be reported back to the SX Logic module.

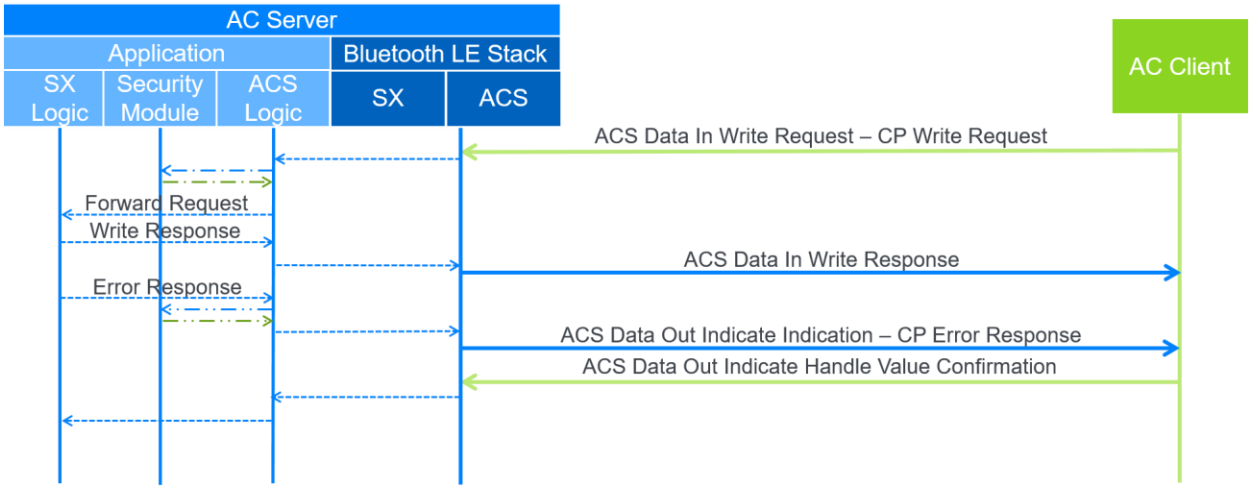


Figure B.2: Data flow of the protected resource error response

B.3 Protected Resource ATT Error Response

In this example, the AC Client writes to the CP to execute a protected procedure, but the CP is not configured for Indications as required by the Service X specification. Figure B.3 shows the data flow. Therefore, Service X provides an ATT Error Response. In this case, the ACS Data In characteristic is used for the write request to execute the protected procedure and the ACS Data Out characteristic is not used.

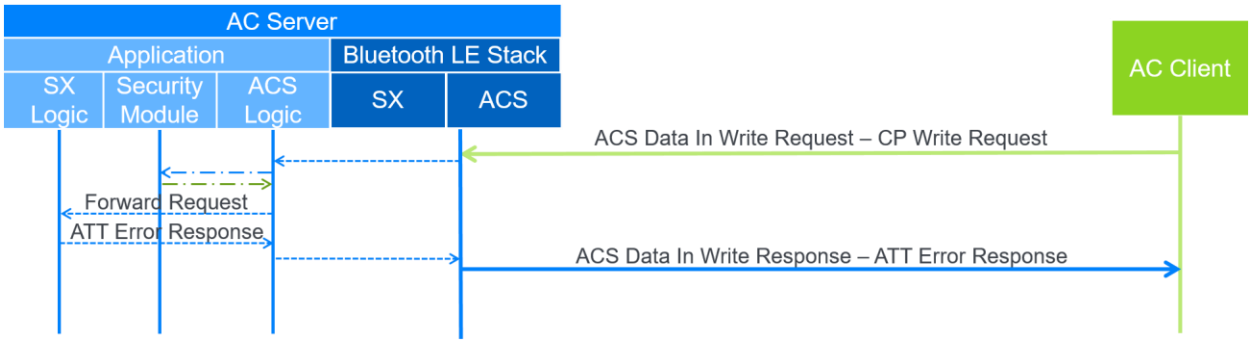
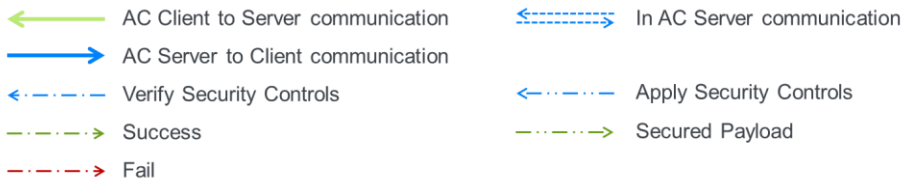


Figure B.3: Data flow of the protected resource ATT Error Response

Appendix C Exchanging keys

The following data flow examples show how keys are exchanged between the AC Client and the AC Server. For these examples, the AC Server supports specific CP procedures to handle the key exchange, which are described in detail in ACS [1]. In each of the following examples, the Bluetooth LE interface and application logic are separated in the AC Server, and the application security module is responsible for validating the information security controls used to protect the SX resources.

The legend below applies to the following examples:



C.1 ECDH key exchange

In this example, the AC Client writes to the ACS CP stating that the AC Client will start the ECDH key exchange. Figure C.1 shows the data flow. Next, the AC Client and the AC Server exchange key data and confirm that the key was exchanged successfully. Because the key exchange is successful, the AC Server responds with an ACS CP indication stating successful key exchange.

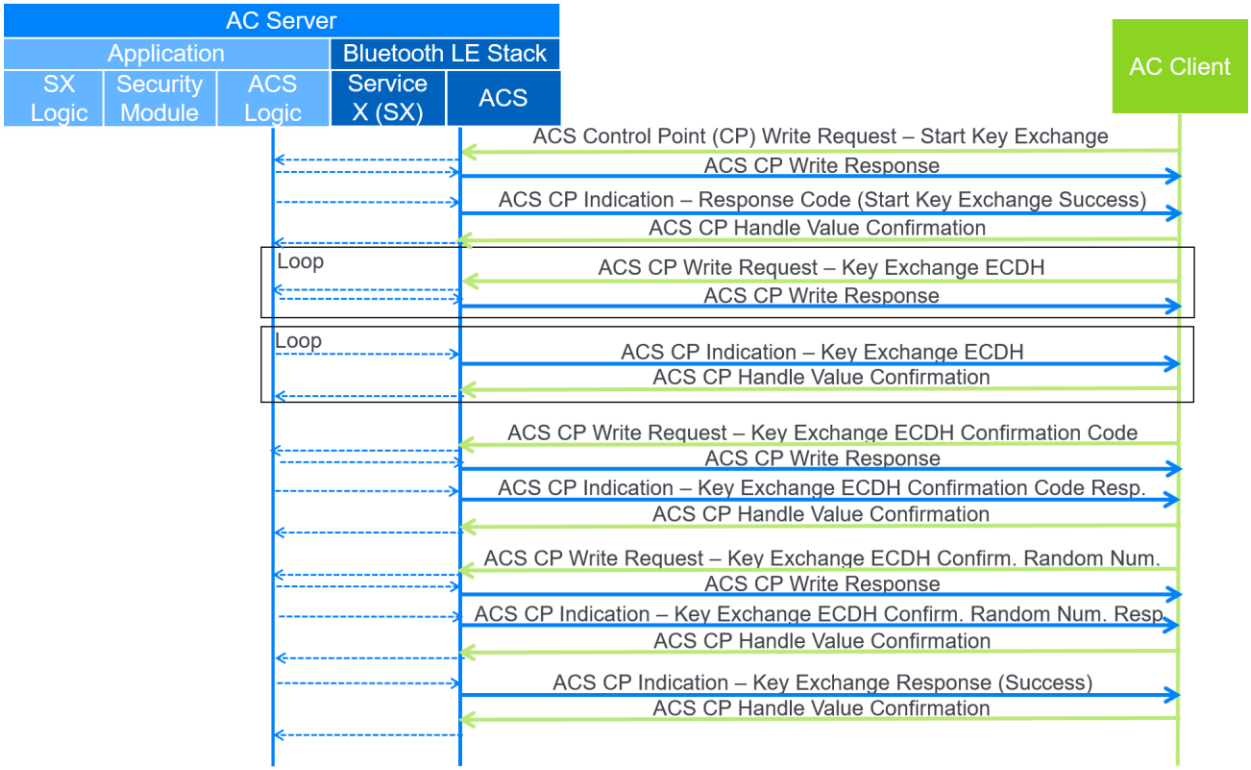


Figure C.1: Data flow of the ECDH key exchange between the AC Client and the AC Server

C.2 KDF key exchange

In this example, the AC Client writes to the ACS CP stating that the AC Client will start the KDF key exchange. Figure C.2 shows the data flow.

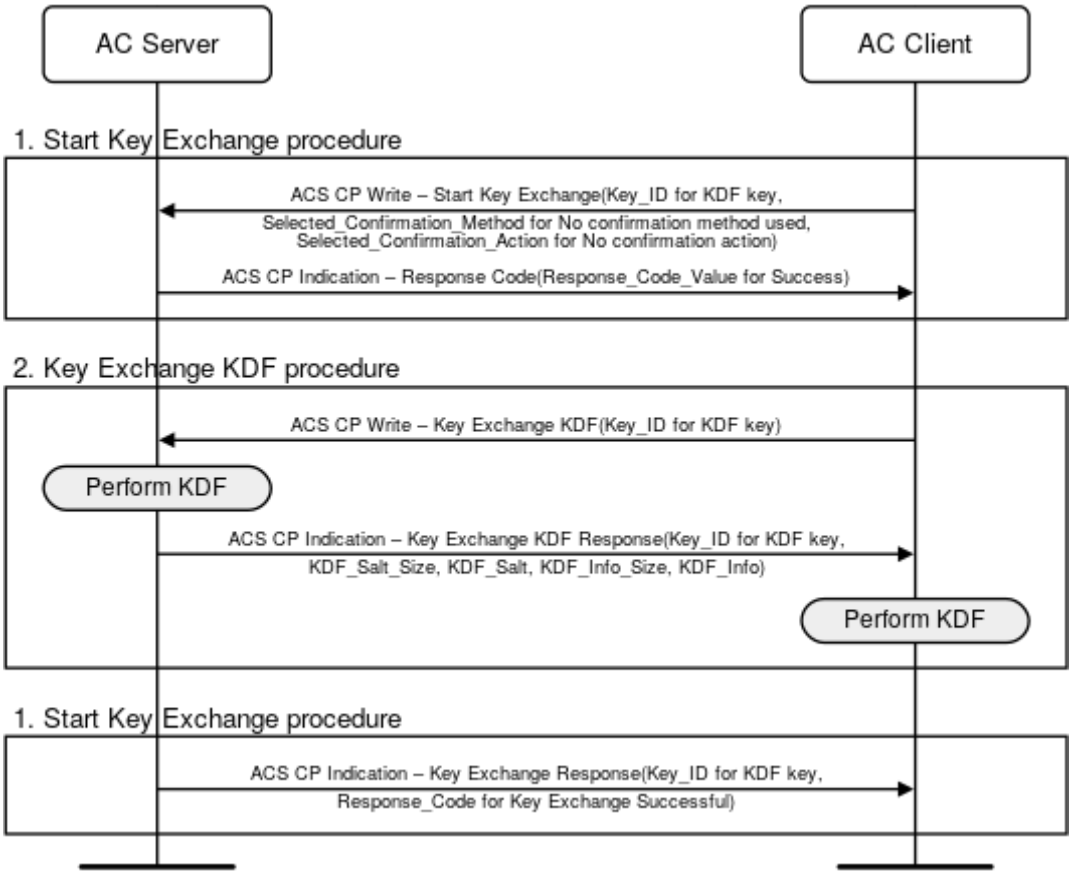


Figure C.2: Data flow of the KDF key exchange between the AC Client and the AC Server

Appendix D Get all active descriptors

In this example, the AC Client requesting the Get All Active Descriptors procedure of the ACS CP and the AC Server supports all descriptors, and none of the descriptors are protected by an information security configuration. Figure D.1 shows the data flow. Next, the AC Server responds with each descriptor. After the AC Server provides all of the descriptors, the AC Server confirms that the Get All Active Descriptors procedure has completed successfully.

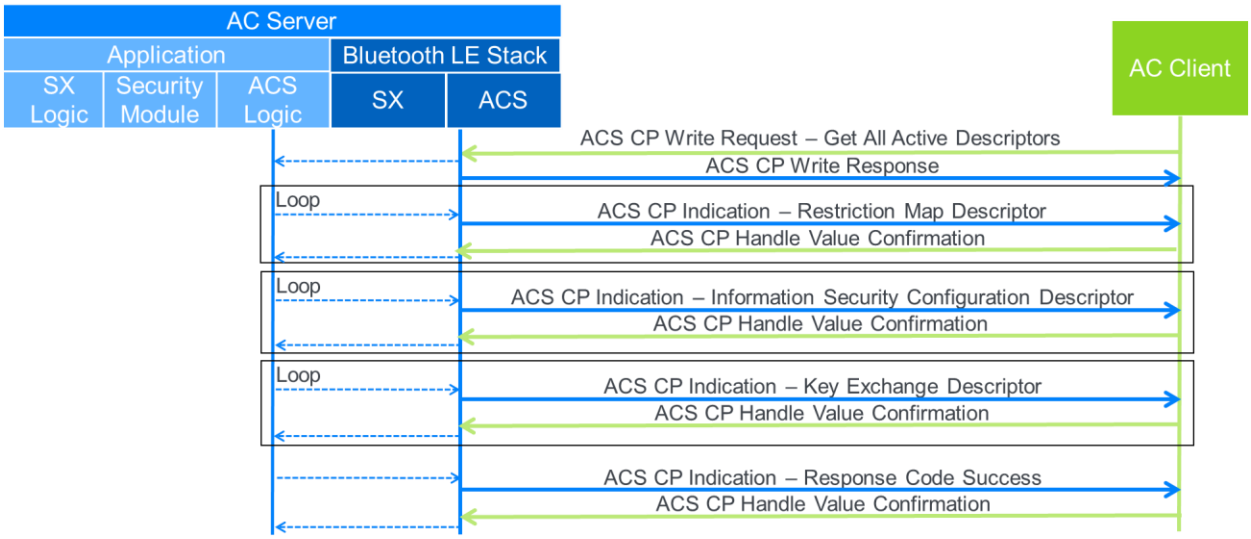


Figure D.1: Data flow of the get all active descriptors between the AC Client and the AC Server

Appendix E Initial communication

This example describes the initial communication between the AC Server and the AC Client. Figure E.1 shows the data flow. First, the AC Client reads the ACS Feature used by the ACS CP procedure and then reads the ACS Status characteristic. The ACS Feature describes the supported features of the AC Server. In this example, the AC Server supports the descriptors and key exchange. The ACS Status characteristic describes the current state of the AC Server. Next, the AC Client requests the restriction map ID list to learn whether the current restriction map is protected. In this example, the AC Server does not protect the current restriction map. The AC Client then requests all descriptors to learn which resources are protected, how they are protected, and how to exchange keys to access the protected resources. Afterwards, the AC Client requests the mapping between the AC Server defined Resource Handles and Attribute UUIDs. Finally, the AC Client exchanges keys with the AC Server. See Appendix C for examples of the key exchanges. After this initial communication is completed, the AC Client and the AC Server conduct secure data exchange.

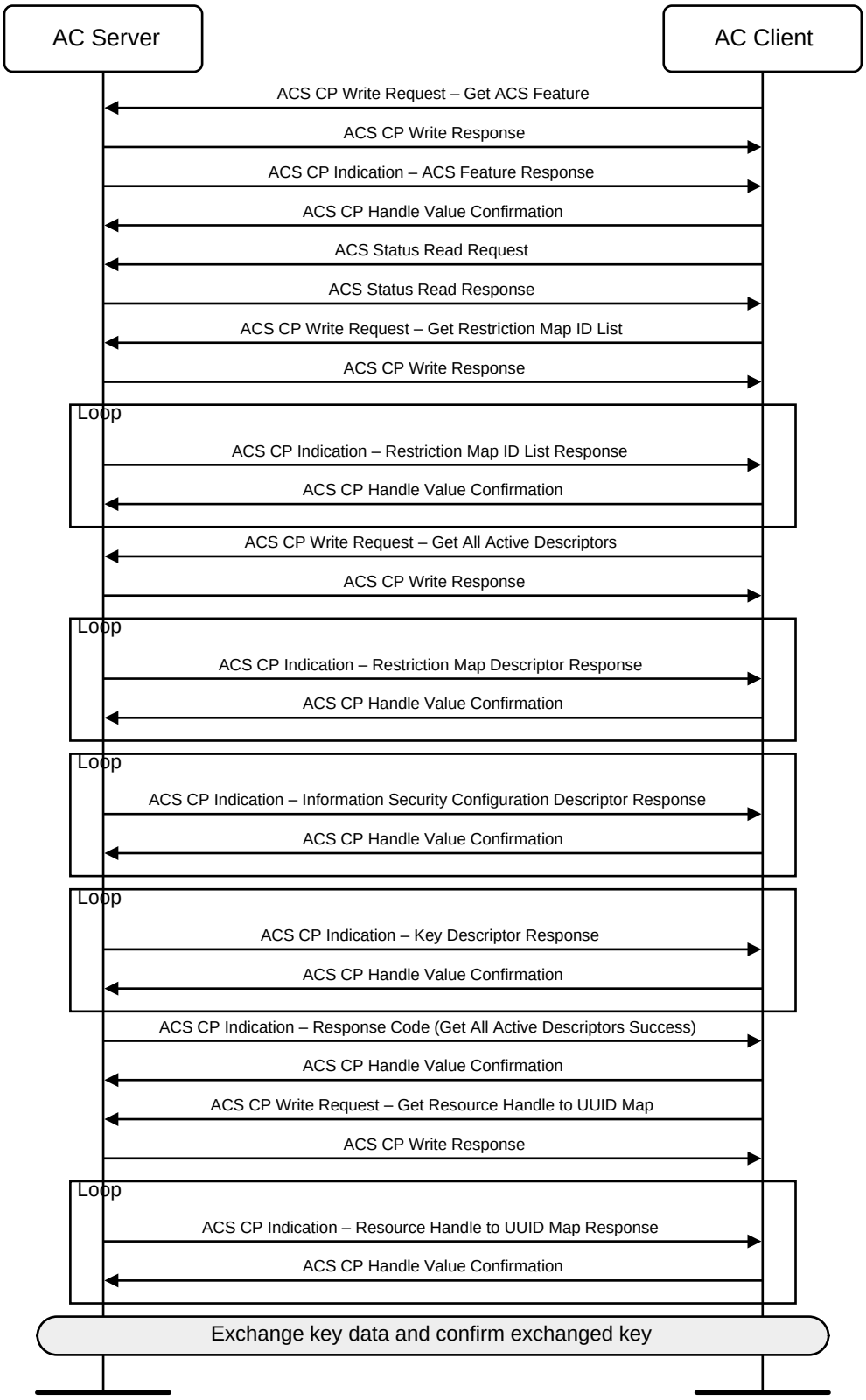


Figure E.1: Message sequence chart showing an example of the initial communication between the AC Client and the AC Server