Part F. Attribute Protocol (ATT)

This Part defines the Attribute Protocol; a protocol for discovering, reading, and \ device

1. Introduction

1.1. Scope

The Attribute Protocol allows a device referred to as the server to expose a set of attributes and their associated values to a peer device referred to as the client. These attributes exposed by the server can be discovered, read, and written by a client, and can be indicated and notified by the server.

1.2. Conformance

If conformance to this protocol is claimed, all capabilities indicated as mandatory for this protocol shall be supported in the specified manner (process-mandatory). This also applies to all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth qualification program.

1.3. Conventions

In this Part PDU names appear in italics. Error codes defined in Table 3.4[attribute-protocol--att-.html#UUID-eefd3e8d-9b16-3af8-1fb4-fa90f52262e8_informaltable-idm13358908515990] (see Section 3.4.1.1[attribute-protocol--att-.html#UUID-eefd3e8d-9b16-3af8-1fb4-fa90f52262e8]) appear in italics followed by the numeric code (e.g. *Attribute Not Found* (0x0A)). Other names such as parameters appear in Roman text.

2. Protocol overview

The Attribute Protocol defines two roles; a server role and a client role. It allows a server to expose a set of attributes to a client that are accessible using the Attribute Protocol.

An attribute is a discrete value that has the following three properties associated with it:

a. attribute type, defined by a UUID
b. attribute handle
c. a set of permissions that are defined by each higher layer specification that utilizes the attribute; these permissions cannot be accessed using the Attribute Protocol.

The attribute type specifies what the attribute represents. Bluetooth SIG defined attribute types are defined in Assigned Numbers[https://www.bluetooth.com/specifications/assigned-numbers] and used by an associated higher layer specification. Non-Bluetooth SIG attribute types may also be defined.

The attribute handle uniquely identifies an attribute on a server, allowing a client to reference the attribute in read or write requests; see Section 3.4.4[attribute-protocol--att-.html#UUID-e4edb1c5-1647-53d8-11be-cc50c78b6c76], Section 3.4.5[attribute-protocol--att-.html#UUID-6fdd15ca-b31d-b46c-4aeb-441a63ffa99e], and Section 3.4.6[attribute-protocol--att-.html#UUID-c31d9020-e006-b6f7-f651-40473c84f4f0]. It allows a client to uniquely identify the attribute being notified or indicated, see Section 3.4.7[attribute-protocol--att-.html#UUID-a2187d3c-304f-3cd2-93bc-423dbe0926e4]. Clients are able to discover the handles of the server's attributes; see Section 3.4.3[attribute-protocol--att-.html#UUID-558ebab6-e157-5c70-20f2-07b36a554076]. Permissions may be applied to an attribute to prevent applications from obtaining or altering an attribute's value. An attribute may be defined by a higher layer specification to be readable or writable or both, and may have additional security requirements. For more information, see Section 3.2.5[attribute-protocol--att-.html#UUID-010cde35-9579-fb7e-8882-0f4b5eec05b8].

A client may send Attribute Protocol requests to a server, and the server shall respond to all requests that it receives. A device can implement both client and server roles, and both roles can function concurrently in the same device and between the same devices. There shall be only one instance of a server on each Bluetooth device; this implies that the attribute handles shall be identical for all supported bearers. For a given client, the server shall have one set of attributes, which shall have the same value and properties irrespective of which bearer is used. The server can support multiple clients. The attribute values can be the same or different for each client as defined by GATT or a higher layer specification.

Note: Multiple services may be exposed on a single server by allocating separate ranges of handles for each service. The discovery of these handle ranges is defined by a higher layer specification.

The Attribute Protocol has notification and indication capabilities that provide an efficient way of sending attribute values to a client without the need for them to be read; see Section 3.3[attribute-protocol--att-.html#UUID-61ed2bef-83b1-9cdd-0191-9faa22366162].

All Attribute Protocol requests are sent over an ATT bearer. There can be multiple ATT bearers established between two devices. Each ATT bearer uses a separate L2CAP channel and can have a different configuration.

In LE, there is a single ATT bearer that uses a fixed channel that is available as soon as the ACL connection is established. Additional ATT bearers can be established using L2CAP (see Section 3.2.11[attribute-protocol--att-.html#UUID-4547fb09-c31e-8847-f3bf-b8baee920049]).

In BR/EDR, one or more ATT bearers can be established using L2CAP (see Section 3.2.11[attribute-protocol--att-.html#UUID-4547fb09-c31e-8847-f3bf-b8baee920049]).

3. Protocol requirements

3.1. Introduction

Each attribute has an attribute type that identifies, by means of a UUID (Universally Unique IDentifier), what the attribute represents so that a client can understand the attributes exposed by a server. Each attribute has an attribute handle that is used for accessing the attribute on a server, as well as an attribute value.

An attribute value is accessed using its attribute handle. The attribute handles are discovered by a client using Attribute Protocol PDUs (Protocol Data Unit). Attributes that have the same attribute type may exist more than once in a server. Attributes also have a set of permissions that controls whether they can be read or written, or whether the attribute value shall be sent over an encrypted link. Security aspects of the Attribute Protocol are defined in Section 4[attribute-protocol--att-.html#UUID-df9ddd0e-2f66-2e15-1f55-4e6c58bec4b0].

3.2. Basic concepts

3.2.1. Attribute type

A universally unique identifier (UUID) is used to identify every attribute type. A UUID is considered unique over all space and time. A UUID can be independently created by anybody and distributed or published as required. There is no central registry for UUIDs, as they are based off a unique identifier that is not duplicated. The Attribute Protocol allows devices to identify attribute types using UUIDs regardless of the local handle used to identify them in a read or write request.

Universal unique identifiers are defined in SDP [Vol 3] Part B, Section 2.5.1[service-discovery-protocol--sdp--specification.html#UUID-ef710684-4c7e-6793-4350-4a190ea9a7a4].

All 32-bit Attribute UUIDs shall be converted to 128-bit UUIDs when the Attribute UUID is contained in an ATT PDU.

3.2.2. Attribute handle

An attribute handle is a 16-bit value that is assigned by each server to its own attributes to allow a client to reference those attributes. An attribute handle shall not be reused while an ATT bearer exists between a client and its server.

Attribute handles on any given server shall have unique, non-zero values. Attributes are ordered by attribute handle.

An attribute handle of value 0x0000 is reserved for future use. An attribute handle of value 0xFFFF is known as the maximum attribute handle.

Note: Attributes can be added or removed while an ATT bearer is established; however, an attribute that has been removed cannot be replaced by another attribute with the same handle while any ATT bearer is established.

3.2.3. Attribute handle grouping

Grouping is defined by a specific attribute placed at the beginning of a range of other attributes that are grouped with that attribute, as defined by a higher layer specification. Clients can request the first and last handles associated with a group of attributes.

3.2.4. Attribute value

An attribute value is an octet array that may be either fixed or variable length. For example, it can be a one octet value, or a four octet integer, or a variable length string. An attribute may contain a value that is too large to transmit in a single PDU and can be sent using multiple PDUs. The values that are transmitted are opaque to the Attribute Protocol. The encoding of these octet arrays is defined by the attribute type.

When transmitting attribute values in a request, a response, a notification or an indication, the attribute value length is not sent in any field of the majority of PDUs. The length of a variable length field in those PDUs is implicitly given by the length of the packet that carries this PDU. This implies that:

a. only one attribute value can be placed in a single request, response, notification or indication unless the attribute values have lengths known by both the server and client, as defined by the attribute type.
b. This attribute value will always be the only variable length field of a request, response, notification or indication.
c. The bearer protocol (e.g. L2CAP) preserves datagram boundaries.

Note: Some responses include multiple attribute values, for example when client requests multiple attribute reads. For the client to determine the attribute value boundaries, the attribute values must have a fixed size defined by the attribute type.

There are some PDUs where the length of attribute values is included as a field within the PDU and therefore the above implications do not apply to these PDUs.

3.2.5. Attribute permissions

An attribute has a set of permission values associated with it. The permissions associated with an attribute specifies that it may be read and/or written. The permissions associated with the attribute specifies the security level required for read and/or write access, as well as notification and/or indication. The permissions of a given attribute are defined by a higher layer specification, and are not discoverable using the Attribute Protocol.

If access to a secure attribute requires an authenticated link, and the client is not already authenticated with the server with sufficient security, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). When a client receives this error code it may try to authenticate the link, and if the authentication is successful, it can then access the secure attribute.

If access to a secure attribute requires an encrypted link, and the link is not encrypted, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). When a client receives this error code it may try to encrypt the link and if the encryption is successful, it can then access the secure attribute.

If access to a secure attribute requires an encrypted link, and the link is encrypted but with an encryption key size that is too short for the level of security required, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). When a client receives this error code it may try to encrypt the link with a longer key size, and if the encryption is successful, it can then access the secure attribute.

Attribute permissions are a combination of access permissions, encryption permissions, authentication permissions and authorization permissions.

The following access permissions are possible:

- Readable
- Writeable
- Readable and writable

The following encryption permissions are possible:

- Encryption required
- No encryption required

The following authentication permissions are possible:

- Authentication Required
- No Authentication Required

The following authorization permissions are possible:

- Authorization Required
- No Authorization Required

Encryption, authentication, and authorization permissions can have different possibilities; for example, a specific attribute could require a particular kind of authentication or a certain minimum encryption key length. An attribute can have several combinations of permissions that apply; for example, a specific attribute could allow any of the following:

- Read if encrypted (authentication not required)
- Write if authenticated and encrypted
- Read or write if authenticated and authorized (irrespective of encryption)

Access permissions are used by a server to determine if a client can read and/or write an attribute value.

Authentication permissions are used by a server to determine if an authenticated physical link is required when a client attempts to access an attribute. Authentication permissions are also used by a server to determine if an authenticated physical link is required before sending a notification or indication to a client.

Authorization permissions determine if a client needs to be authorized before accessing an attribute value.

Different bearers for the same client may be on links with different security properties. Therefore, the server must not assume that when a client has been authenticated on the link carrying one bearer, it has been authenticated on all bearers. The different security properties might have other implications that an implementation needs to take into account.

3.2.6. Control-point attributes

Attributes that cannot be read, but can only be written, notified or indicated are called control-point attributes. These control-point attributes can be used by higher layers to enable device specific procedures, for example the writing of a command or the indication when a given procedure on a device has completed.

3.2.7. Protocol methods

The Attribute Protocol uses methods defined in Section 3.4[attribute-protocol--att-.html#UUID-9a2dc3de-5db8-cf7c-b394-3f13af7a11fa] to find, read, write, notify, and indicate attributes. A method is categorized as either a command, a request, a response, a notification, an indication, or a confirmation; see Section 3.3[attribute-protocol--att-.html#UUID-61ed2bef-83b1-9cdd-0191-9faa22366162]. Some Attribute Protocol PDUs can also include an Authentication Signature, to allow authentication of the originator of this PDU without requiring encryption. The method and signed bit are known as the opcode.

3.2.8. Exchanging MTU size

ATT_MTU is defined as the maximum size of any packet sent between a client and a server. A higher layer specification defines the default ATT_MTU value.

When using an L2CAP channel with a fixed CID, the client and server may optionally exchange the maximum size of a packet that can be received using the ATT_EXCHANGE_MTU_REQ and ATT_EXCHANGE_MTU_RSP PDUs. Both devices then use the minimum of these exchanged values for all further communication (see Section 3.4.2[attribute-protocol--att-.html#UUID-a1f22a10-ac5b-9887-badf-54c142565593]). A device that is acting as a server and client at the same time shall use the same value for Client Rx MTU and Server Rx MTU.

When using an L2CAP channel with a dynamically allocated CID, the ATT_MTU shall be set to the L2CAP MTU size.

The ATT_MTU value is a per ATT bearer value. A device with multiple ATT bearers may have a different ATT_MTU value for each ATT bearer.

3.2.9. Long attribute values

The longest attribute that can be sent in a single packet is (ATT_MTU-1) octets in size. At a minimum, the Attribute Opcode is included in an Attribute PDU.

An attribute value may be defined to be larger than (ATT_MTU-1) octets in size. These attributes are called long attributes.

To read the entire value of an attributes larger than (ATT_MTU-1) octets, the ATT_READ_BLOB_REQ PDU is used. It is possible to read the first (ATT_MTU-1) octets of a long attribute value using the ATT_READ_REQ PDU.

To write the entire value of an attribute larger than (ATT_MTU-3) octets, the ATT_PREPARE_WRITE_REQ and ATT_EXECUTE_WRITE_REQ PDUs are used. It is possible to write the first (ATT_MTU-3) octets of a long attribute value using the ATT_WRITE_CMD PDU.

It is not possible to determine if an attribute value is longer than (ATT_MTU-3) octets using this protocol. A higher layer specification will state that a given attribute can have a maximum length larger than (ATT_MTU-3) octets.

The maximum length of an attribute value shall be 512 octets.

Note: The protection of an attribute value changing when reading the value using multiple Attribute Protocol PDUs is the responsibility of the higher layer.

3.2.10. Atomic operations

The server shall treat each request or command as an atomic operation that cannot be affected by another ATT bearer sending a request or command at the same time. If an ATT bearer is terminated for any reason (user action or loss of the radio link), the value of any modified attribute is the responsibility of the higher layer specification.

Long attributes cannot be read or written in a single atomic operation.

3.2.11. ATT bearers

An ATT bearer is a channel used to send Attribute Protocol PDUs. Each ATT bearer uses an L2CAP channel which shall be either a dynamically allocated channel or the LE Attribute Protocol fixed channel (see [Vol 3] Part A, Section 2.1[logical-link-control-and-adaptation-protocol-specification.html#UUID-7a94afa8-64f4-93c3-a4cd-07b745296276]). A device may have any number of dynamically allocated channels and at most one fixed channel as ATT bearers to a peer device.

An ATT bearer connects an ATT Client on one device to an ATT Server on the peer device and may also connect an ATT Server on the first device to an ATT Client on the peer device. Whether a received Attribute PDU is intended for the ATT Client or for the ATT Server is determined by the PDU type (see Section 3.3[attribute-protocol--att-.html#UUID-61ed2bef-83b1-9cdd-0191-9faa22366162]).

The L2CAP channel mode determines the behavior of Attribute Protocol on that ATT bearer. If the L2CAP channel mode is using Enhanced Credit Based Flow Control Mode, the ATT bearer is known as an Enhanced ATT bearer. Any ATT bearer that is not an Enhanced ATT bearer, using any other L2CAP channel mode, is known as an Unenhanced ATT bearer.

Except where explicitly stated, the behavior of an Enhanced ATT bearer that uses Enhanced Credit Based Flow Control Mode shall be identical to the behavior of an ATT bearer that does not use Enhanced Credit Based Flow Control Mode.

An ATT bearer is terminated when either the L2CAP channel (if dynamically allocated) or the underlying physical link is disconnected.

An LE fixed channel can only be terminated by disconnecting the physical link.

A higher-layer specification may require an Enhanced ATT bearer.

3.3. Attribute PDU

Attribute PDUs have one of six types, which are indicated by the suffix to the PDU name as shown in Table 3.1[attribute-protocol--att-.html#UUID-61ed2bef-83b1-9cdd-0191-9faa22366162_informaltable-idm13358908308052]:

Type	Purpose	Suffix
Commands	PDUs sent to a server by a client that do not invoke a response.	CMD
Requests	PDUs sent to a server by a client that invoke a response.	REQ
Responses	PDUs sent to a client by a server in response to a request.	RSP
Notifications	Unsolicited PDUs sent to a client by a server that do not invoke a confirmation.	NTF
Indications	Unsolicited PDUs sent to a client by a server that invoke a confirmation.	IND
Confirmations	PDUs sent to a server by a client to confirm receipt of an indication.	CFM

Table 3.1: Attribute PDUs

A server shall be able to receive and properly respond to the following request PDUs:

- ATT_FIND_INFORMATION_REQ
- ATT_READ_REQ

Support for all other PDU types in a server can be specified in a higher layer specification, see Section 3.4.8[attribute-protocol--att-.html#UUID-2cdddfc5-98bd-4ed9-76e7-bcce33836208].

If a client sends a request, then the client shall support all possible response PDUs for that request.

If a server receives a request that it does not support, then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Request Not Supported* (0x06), with the Attribute Handle In Error set to 0x0000.

If a server receives a command that it does not support, indicated by the Command Flag of the PDU set to one, then the server shall ignore the command.

If the server receives an invalid request – for example, the PDU is the wrong length – then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid PDU* (0x04), with the Attribute Handle In Error set to 0x0000.

If a server does not have sufficient resources to process a request, then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Resources* (0x11), with the Attribute Handle In Error set to 0x0000.

If a server cannot process a request because an error was encountered during the processing of this request, then the server shall respond with the ATT_ERROR_RSP PDU with the Error Code parameter set to *Unlikely Error* (0x0E), with the Attribute Handle In Error set to 0x0000.

3.3.1. Attribute PDU format

Attribute PDUs have the following format:

Name	**Size (octets)**	**Description**
Attribute Opcode	1	The attribute PDU operation code bit 7: Authentication Signature Flag bit 6: Command Flag bits 5-0: Method

Attribute Parameters	0 to (ATT_MTU - X)	The attribute PDU parameters
		X = 1 if Authentication Signature Flag of the Attribute Opcode is 0
		X = 13 if Authentication Signature Flag of the Attribute Opcode is 1

Authentication Signature	0 or 12	Optional authentication signature for the Attribute Opcode and Attribute Parameters

Table 3.2: Format of attribute PDU

Multi-octet fields within the Attribute Protocol shall be sent least significant octet first (little-endian) with the exception of the Attribute Value field. The endian-ness of the Attribute Value field is defined by a higher layer specification.

The Attribute Opcode is composed of three fields, the Authentication Signature Flag, the Command Flag, and the Method. The Method is a 6-bit value that determines the format and meaning of the Attribute Parameters.

If the Authentication Signature Flag of the Attribute Opcode is set to one, the Authentication Signature value shall be appended to the end of the attribute PDU, and X is 13. If the Authentication Signature Flag of the Attribute Opcode is set to zero, the Authentication Signature value shall not be appended, and X is 1.

The Authentication Signature field is calculated as defined in Security Manager (see [Vol 3] Part H, Section 2.4.5[security-manager-specification.html#UUID-193f95ea-7252-1b51-853b-a1999393dddf]). This value provides an Authentication Signature for the variable length message (m) consisting of the following values in this order: Attribute Opcode, Attribute Parameters.

An Attribute PDU that includes an Authentication Signature should not be sent on an encrypted link.

Note: An encrypted link already includes authentication data on every packet and therefore adding more authentication data is not required.

If the Command Flag of the Attribute Opcode is set to one, the PDU shall be considered to be a command.

Only the ATT_WRITE_CMD PDU may include an Authentication Signature (and therefore becomes an ATT_SIGNED_WRITE_CMD PDU).

3.3.2. Sequential protocol

Many Attribute Protocol PDUs use a sequential request-response protocol.

Once a client sends a request to a server, that client shall send no other request to the same server on the same ATT bearer until a response PDU has been received.

Indications sent from a server also use a sequential indication-confirmation protocol. No other indications shall be sent to the same client from this server on the same ATT bearer until a confirmation PDU has been received. The client, however, is free to send commands and requests prior to sending a confirmation.

For notifications, which do not have a response PDU, there is no flow control and a notification can be sent at any time.

For commands, which do not have a response PDU, there is no flow control and a command can be sent at any time.

Note: A server can be flooded with commands, and a higher layer specification can define how to prevent this from occurring.

Commands that are received but cannot be processed, due to buffer overflows or a change-unaware client (see [Vol 3] Part G, Section 2.5.2.1[generic-attribute-profile--gatt-.html#UUID-787b0d4d-3112-9b07-6bef-89dda173a489]), shall be discarded. Therefore, those PDUs must be considered to be unreliable.

On an Unenhanced ATT bearer, notifications that are received but cannot be processed due to buffer overflows shall be discarded. Therefore, those PDUs must be considered to be unreliable.

On an Enhanced ATT bearer, notifications shall always be processed when received.

Note: Flow control for each client and a server is independent.

Note: It is possible for a server to receive a request, send one or more notifications, and then the response to the original request. The flow control of requests is not affected by the transmission of the notifications.

Note: It is possible for a server to receive a request and then a command before responding to the original request. The flow control of requests is not affected by the transmission of commands.

Note: It is possible for a notification from a server to be sent after an indication has been sent but the confirmation has not been received. The flow control of indications is not affected by the transmission of notifications.

Note: It is possible for a client to receive an indication from a server and then send a request or command to that server before sending the confirmation of the original indication.

3.3.3. Transaction

An Attribute Protocol request and response or indication-confirmation pair is considered a single transaction. A transaction shall always be performed on one ATT bearer, and shall not be split over multiple ATT bearers.

On the client, a transaction shall start when the request is sent by the client. A transaction shall complete when the response is received by the client.

On a server, a transaction shall start when a request is received by the server. A transaction shall complete when the response is sent by the server.

On a server, a transaction shall start when an indication is sent by the server. A transaction shall complete when the confirmation is received by the server.

On a client, a transaction shall start when an indication is received by the client. A transaction shall complete when the confirmation is sent by the client.

A transaction not completed within 30 seconds shall time out. Such a transaction shall be considered to have failed and the local higher layers shall be informed of this failure. No more Attribute Protocol requests, commands, indications or notifications shall be sent to the target device on this ATT bearer.

To send another Attribute Protocol PDU, a new ATT bearer must be established between these devices. The existing ATT bearer may need to be terminated before the new ATT bearer is established.

If the ATT bearer is terminated during a transaction, then the transaction shall be considered to be closed, and any values that were being modified on the server will be in an undetermined state.

Note: Each ATT_PREPARE_WRITE_REQ and each ATT_READ_BLOB_REQ PDU starts a separate request and therefore a separate transaction.

3.4. Attribute Protocol PDUs

3.4.1. Error handling

3.4.1.1. ATT_ERROR_RSP

The ATT_ERROR_RSP PDU is used to state that a given request cannot be performed, and to provide the reason.

Note: Commands (i.e. the ATT_WRITE_CMD and ATT_SIGNED_WRITE_CMD PDUs) do not generate this response.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x01 = ATT_ERROR_RSP PDU
Request Opcode In Error	1	The request that generated this ATT_ERROR_RSP PDU
Attribute Handle In Error	2	The attribute handle that generated this ATT_ERROR_RSP PDU
Error Code	1	The reason why the request has generated an ATT_ERROR_RSP PDU

Table 3.3: Format of the ATT_ERROR_RSP PDU

The Request Opcode In Error parameter shall be set to the Attribute Opcode of the request that generated this error.

The Attribute Handle In Error parameter shall be set to the attribute handle in the original request that generated this error. If there was no attribute handle in the original request or if the request is not supported, then the value 0x0000 shall be used for this field.

The Error Code parameter shall be set to one of the following values:

Name	**Error Code**	**Description**
Invalid Handle	0x01	The attribute handle given was not valid on this server.

Name	Error Code	Description
Read Not Permitted	0x02	The attribute cannot be read.
Write Not Permitted	0x03	The attribute cannot be written.
Invalid PDU	0x04	The attribute PDU was invalid.
Insufficient Authentication	0x05	The attribute requires authentication before it can be read or written.
Request Not Supported	0x06	ATT Server does not support the request received from the client.
Invalid Offset	0x07	Offset specified was past the end of the attribute.
Insufficient Authorization	0x08	The attribute requires authorization before it can be read or written.
Prepare Queue Full	0x09	Too many prepare writes have been queued.
Attribute Not Found	0x0A	No attribute found within the given attribute handle range.
Attribute Not Long	0x0B	The attribute cannot be read using the ATT_READ_BLOB_REQ PDU.
Encryption Key Size Too Short[1] [#ftn.idm46388861628080]	0x0C	The Encryption Key Size used for encrypting this link is too short.
Invalid Attribute Value Length	0x0D	The attribute value length is invalid for the operation.
Unlikely Error	0x0E	The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested.

Name	Error Code	Description
Insufficient Encryption	0x0F	The attribute requires encryption before it can be read or written.
Unsupported Group Type	0x10	The attribute type is not a supported grouping attribute as defined by a higher layer specification.
Insufficient Resources	0x11	Insufficient Resources to complete the request.
Database Out Of Sync	0x12	The server requests the client to rediscover the database.
Value Not Allowed	0x13	The attribute parameter value was not allowed.
Application Error	0x80 – 0x9F	Application error code defined by a higher layer specification.
Common Profile and Service Error Codes	0xE0 – 0xFF	Common profile and service error codes defined in [1[attribute-protocol--att-.html#UUID-9ebd9c5e-1ea8-291f-4639-b82e3e7059c6_bibliomixed-idm13391355185596]]
Reserved for future use	All other values	Reserved for future use.
[1] [#idm46388861628080]This was previously "Insufficient Encryption Key Size".		

Table 3.4: Error codes

The Error Code values listed in Section 3.4.2[attribute-protocol--att-.html#UUID-a1f22a10-ac5b-9887-badf-54c142565593] to Section 3.4.8[attribute-protocol--att-.html#UUID-2cdddfc5-98bd-4ed9-76e7-bcce33836208] are not necessarily the only ones permitted in response to those PDUs. See Section 3.4.9[attribute-protocol--att-.html#UUID-191d4479-95b1-b04e-88b1-229f52d4fb60] for the definitive list of which Error Code values are permitted.

If more than one error code applies, then it is vendor-specific which error code is transmitted in the ATT_ERROR_RSP PDU.

If an error code is received in the ATT_ERROR_RSP PDU that is not understood by the client, for example an error code that was reserved for future use that is now being used in a future version of the specification, then the ATT_ERROR_RSP PDU shall still be considered to state that the given request cannot be performed for an unknown reason.

Note: Sending an ATT_ERROR_RSP PDU should not cause the ATT Server to disconnect from the client. The client may upgrade the security and retry the request, so the server should give the client sufficient time to perform such an upgrade.

3.4.2. MTU exchange

3.4.2.1. ATT_EXCHANGE_MTU_REQ

The ATT_EXCHANGE_MTU_REQ PDU is used by the client to inform the server of the client's maximum receive MTU size and request the server to respond with its maximum receive MTU size.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x02 = ATT_EXCHANGE_MTU_REQ
Client Rx MTU	2	Client receive MTU size

Table 3.5: Format of ATT_EXCHANGE_MTU_REQ PDU

The Client Rx MTU shall be greater than or equal to the default ATT_MTU.

This request shall only be sent once during a connection by the client. The Client Rx MTU parameter shall be set to the maximum size of the Attribute Protocol PDU that the client can receive.

3.4.2.2. ATT_EXCHANGE_MTU_RSP

The ATT_EXCHANGE_MTU_RSP PDU is sent in reply to a received ATT_EXCHANGE_MTU_REQ PDU.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x03 = ATT_EXCHANGE_MTU_RSP
Server Rx MTU	2	ATT Server receive MTU size

Table 3.6: Format of ATT_EXCHANGE_MTU_RSP PDU

The Server Rx MTU shall be greater than or equal to the default ATT_MTU.

The Server Rx MTU parameter shall be set to the maximum size of the Attribute Protocol PDU that the server can receive.

The server and client shall set ATT_MTU to the minimum of the Client Rx MTU and the Server Rx MTU. The size is the same to ensure that a client can correctly detect the final packet of a long attribute read.

This ATT_MTU value shall be applied in the server after this response has been sent and before any other Attribute Protocol PDU is sent.

This ATT_MTU value shall be applied in the client after this response has been received and before any other Attribute Protocol PDU is sent.

If either Client Rx MTU or Service Rx MTU are incorrectly less than the default ATT_MTU, then the ATT_MTU shall not be changed and the ATT_MTU shall be the default ATT_MTU.

If a device is both a client and a server, the following rules shall apply:

1. A device's ATT_EXCHANGE_MTU_REQ PDU shall contain the same MTU as the device's ATT_EXCHANGE_MTU_RSP PDU (i.e. the MTU shall be symmetric).
2. If MTU is exchanged in one direction, that is sufficient for both directions.

3. It is permitted, (but not necessary - see 2.) to exchange MTU in both directions, but the MTUs shall be the same in each direction (see 1.)
4. If an Attribute Protocol Request is received after the MTU Exchange Request is sent and before the MTU Exchange Response is received, the associated Attribute Protocol Response shall use the default MTU. Figure 3.1[attribute-protocol--att-.html#UUID-f4afb5e3-c844-7547-4a20-2a6b5cfa3581_figure-idm457010448273443359399432146] shows an example that is covered by this rule. In this case device A and device B both use the default MTU for the Attribute Protocol Response.
5. Once the MTU Exchange Request has been sent, the initiating device shall not send an Attribute Protocol Indication or Notification until after the MTU Exchange Response has been received.

   Note: This stops the risk of a cross-over condition where the MTU size is unknown for the Indication or Notification.

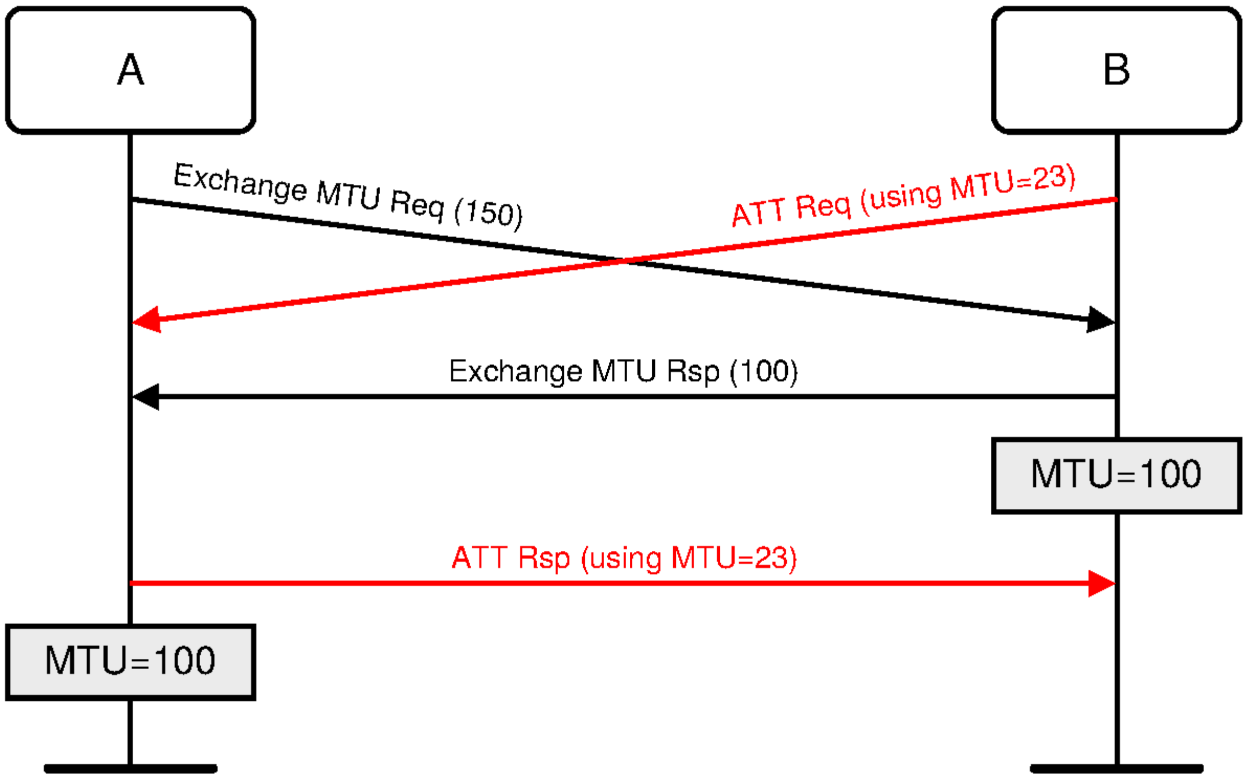

Figure 3.1: MTU Request and Response exchange

3.4.3. Find information

3.4.3.1. ATT_FIND_INFORMATION_REQ

The ATT_FIND_INFORMATION_REQ PDU is used to obtain the mapping of attribute handles with their associated types. This allows a client to discover the list of attributes and their types on a server.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x04 = ATT_FIND_INFORMATION_REQ
Starting Handle	2	First requested handle number
Ending Handle	2	Last requested handle number

Table 3.7: Format of ATT_FIND_INFORMATION_REQ PDU

Only attributes with attribute handles between the Starting Handle parameter and the Ending Handle parameter will be returned. To read all attributes, the Starting Handle parameter shall be set to 0x0001, and the Ending Handle parameter shall be set to 0xFFFF. The Starting Handle parameter shall be less than or equal to the Ending Handle parameter.

If one or more attributes will be returned, an ATT_FIND_INFORMATION_RSP PDU shall be sent.

If a server receives an ATT_FIND_INFORMATION_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01); the Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attributes will be returned, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A); the Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

The server shall not respond to the ATT_FIND_INFORMATION_REQ PDU with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05), *Insufficient Authorization* (0x08), *Encryption Key Size Too*

Short (0x0C), *Database Out of Sync* (0x12), *Application Error* (0x80 to 0x9F), or *Common Profile and Service Error Codes* (0xE0 to 0xFF).

3.4.3.2. ATT_FIND_INFORMATION_RSP

The ATT_FIND_INFORMATION_RSP PDU is sent in reply to a received ATT_FIND_INFORMATION_REQ PDU and contains information about this server.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x05 = ATT_FIND_INFORMATION_RSP
Format	1	The format of the information data.
Information Data	4 to (ATT_MTU-2)	The information data whose format is determined by the Format field

Table 3.8: Format of ATT_FIND_INFORMATION_RSP PDU

The *Find Information Response* shall have complete handle-UUID pairs. Such pairs shall not be split across response packets; this also implies that a handle-UUID pair shall fit into a single response packet. The handle-UUID pairs shall be returned in ascending order of attribute handles.

The Format parameter can contain one of two possible values.

Name	Format	Description
Handle(s) and 16-bit Bluetooth UUID(s)	0x01	A list of 1 or more handles with their 16-bit Bluetooth UUIDs
Handle(s) and 128-bit UUID(s)	0x02	A list of 1 or more handles with their 128-bit UUIDs

Table 3.9: Format field values

The information data field is comprised of a list of data defined in Table 3.10[attribute-protocol--att-.html#UUID-5b5e6160-4ba1-1138-1ced-1a00153eeb9c_informaltable-idm13358909072592] and Table 3.11[attribute-protocol--att-.html#UUID-5b5e6160-4ba1-1138-1ced-1a00153eeb9c_informaltable-idm13358909078082] depending on the value chosen for the format.

Handle	**16-bit Bluetooth UUID**
2 octets	2 octets

Table 3.10: Format 0x01 – handle and 16-bit Bluetooth UUIDs

Handle	**128-bit UUID**
2 octets	16 octets

Table 3.11: Format 0x02 – handle and 128-bit UUIDs

If sequential attributes have differing UUID sizes, the ATT_FIND_INFORMATION_RSP PDU shall end with the first attribute of the pair even though this may mean that it is not filled with the maximum possible amount of (handle, UUID) pairs. This is because it is not possible to include attributes with differing UUID sizes into a single response packet. In this situation, the client must use another ATT_FIND_INFORMATION_REQ PDU with its starting handle updated to fetch the second attribute of the pair and any further ones in its original request.

3.4.3.3. ATT_FIND_BY_TYPE_VALUE_REQ

The ATT_FIND_BY_TYPE_VALUE_REQ PDU is used to obtain the handles of attributes that have a 16-bit UUID attribute type and attribute value.This allows the range of handles associated with a given attribute to be discovered when the attribute type determines the grouping of a set of attributes.

Note: Generic Attribute Profile defines grouping of attributes by attribute type.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x06 = ATT_FIND_BY_TYPE_VALUE_REQ PDU
Starting Handle	2	First requested handle number
Ending Handle	2	Last requested handle number
Attribute Type	2	2 octet UUID to find
Attribute Value	0 to (ATT_MTU-7)	Attribute value to find

Table 3.12: Format of ATT_FIND_BY_TYPE_VALUE_REQ PDU

Only attributes with attribute handles between the Starting Handle parameter and the Ending Handle parameter that match the requested attribute type and the attribute value that have sufficient permissions to allow reading will be returned. To read all attributes, the Starting Handle parameter shall be set to 0x0001, and the Ending Handle parameter shall be set to 0xFFFF.

If one or more handles will be returned, an ATT_FIND_BY_TYPE_VALUE_RSP PDU shall be sent.

Note: Attribute values will be compared in terms of length and binary representation.

Note: It is not possible to use this request on an attribute that has a value longer than (ATT_MTU-7).

If a server receives an ATT_FIND_BY_TYPE_VALUE_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attributes will be returned, an ATT_ERROR_RSP PDU shall be sent by the server with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The server shall not respond to the ATT_FIND_BY_TYPE_VALUE_REQ PDU with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05), *Insufficient Authorization* (0x08), *Encryption Key Size Too Short* (0x0C), *Insufficient Encryption* (0x0F), *Database Out of Sync* (0x12), *Application Error* (0x80 to 0x9F), or *Common Profile and Service Error Codes* (0xE0 to 0xFF).

3.4.3.4. ATT_FIND_BY_TYPE_VALUE_RSP

The ATT_FIND_BY_TYPE_VALUE_RSP PDU is sent in reply to a received ATT_FIND_BY_TYPE_VALUE_REQ PDU and contains information about this server.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x07 = ATT_FIND_BY_TYPE_VALUE_RSP PDU
Handles Information List	4 to (ATT_MTU-1)	A list of 1 or more Handle Informations

Table 3.13: Format of ATT_FIND_BY_TYPE_VALUE_RSP PDU

The Handles Information List field is a list of one or more Handle Informations. The Handles Information field is an attribute handle range as defined in Table 3.14[attribute-protocol--att-.html#UUID-ee515de2-e411-3c7c-bcf1-17f159b1900d_informaltable-idm13358909132944].

Found Attribute Handle	**Group End Handle**
2 octets	2 octets

Table 3.14: Format of the Handles Information

The ATT_FIND_BY_TYPE_VALUE_RSP PDU shall contain one or more complete Handles Information. Such Handles Information shall not be split across response packets. The Handles Information List is ordered sequentially based on the found attribute handles.

For each handle that matches the attribute type and attribute value in the ATT_FIND_BY_TYPE_VALUE_REQ PDU a Handles Information shall be returned. The Found Attribute Handle shall be set to the handle of the attribute that has the exact attribute type and attribute value from the ATT_FIND_BY_TYPE_VALUE_REQ PDU. If the attribute type in the ATT_FIND_BY_TYPE_VALUE_REQ PDU is a grouping attribute as defined by a higher layer specification, the Group End Handle shall be defined by that higher layer specification. If the attribute type in the ATT_FIND_BY_TYPE_VALUE_REQ PDU is not a grouping attribute as defined by a higher layer specification, the Group End Handle shall be equal to the Found Attribute Handle.

Note: The Group End Handle may be greater than the Ending Handle in the ATT_FIND_BY_TYPE_VALUE_REQ PDU.

If a server receives an ATT_FIND_BY_TYPE_VALUE_REQ PDU, the server shall respond with the ATT_FIND_BY_TYPE_VALUE_RSP PDU containing as many handles for attributes that match the requested attribute type and attribute value that exist in the server that will fit into the maximum PDU size of (ATT_MTU-1).

3.4.4. Reading attributes

3.4.4.1. ATT_READ_BY_TYPE_REQ

The ATT_READ_BY_TYPE_REQ PDU is used to obtain the values of attributes where the attribute type is known but the handle is not known.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x08 = ATT_READ_BY_TYPE_REQ PDU
Starting Handle	2	First requested handle number
Ending Handle	2	Last requested handle number
Attribute Type	2 or 16	2 or 16 octet UUID

Table 3.15: Format of ATT_READ_BY_TYPE_REQ PDU

Only the attributes with attribute handles between the Starting Handle and the Ending Handle with the attribute type that is the same as the Attribute Type given will be returned. To search through all attributes, the starting handle shall be set to 0x0001 and the ending handle shall be set to 0xFFFF.

Note: All attribute types are effectively compared as 128-bit UUIDs, even if a 16-bit UUID is provided in this request or defined for an attribute. See [Vol 3] Part B, Section 2.5.1[service-discovery-protocol--sdp--specification.html#UUID-ef710684-4c7e-6793-4350-4a190ea9a7a4].

The starting handle shall be less than or equal to the ending handle. If a server receives an ATT_READ_BY_TYPE_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If no attribute with the given type exists within the handle range, then no attribute handle and value will be returned, and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The attributes returned shall be the attributes with the lowest handles within the handle range. These are known as the requested attributes.

If the attributes with the requested type within the handle range have attribute values that have the same length, then these attributes can all be read in a single request. However, if those attributes have different lengths, then multiple ATT_READ_BY_TYPE_REQ PDUs must be issued.

The ATT Server shall include as many attributes as possible in the response in order to minimize the number of PDUs required to read attributes of the same type.

When multiple attributes match, then the rules below shall be applied to each in turn.

- Only attributes that can be read shall be returned in an ATT_READ_BY_TYPE_RSP PDU.

- If an attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then this attribute cannot be included in an ATT_READ_BY_TYPE_RSP PDU and the attributes before this attribute shall be returned.
- If the first attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then no other attributes in the requested attributes can be considered.

The server shall respond with an ATT_READ_BY_TYPE_RSP PDU if the requested attributes have sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the requested attribute’s value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

Note: If there are multiple attributes with the requested type within the handle range, and the client would like to get the next attribute with the requested type, it would have to issue another ATT_READ_BY_TYPE_REQ PDU with its starting handle updated. The client can be sure there are no more such attributes remaining once it gets an ATT_ERROR_RSP PDU with the Error Code parameter set to *Attribute Not Found* (0x0A).

3.4.4.2. ATT_READ_BY_TYPE_RSP

The ATT_READ_BY_TYPE_RSP PDU is sent in reply to a received ATT_READ_BY_TYPE_REQ PDU and contains the handles and values of the attributes that have been read.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x09 = ATT_READ_BY_TYPE_RSP PDU
Length	1	The size of each attribute handle-value pair
Attribute Data List	2 to (ATT_MTU-2)	A list of Attribute Data

Table 3.16: Format of ATT_READ_BY_TYPE_RSP PDU

The ATT_READ_BY_TYPE_RSP PDU shall contain complete handle-value pairs. Such pairs shall not be split across response packets. The handle-value pairs shall be returned sequentially based on the attribute handle.

The Length parameter shall be set to the size of one attribute handle-value pair.

The maximum length of an attribute handle-value pair is 255 octets, bounded by the Length parameter that is one octet. Therefore, the maximum length of an attribute value returned in this response is (Length – 2) = 253 octets.

The attribute handle-value pairs shall be set to the value of the attributes identified by the attribute type within the handle range within the request. If the attribute value is longer than (ATT_MTU - 4) or 253 octets, whichever is smaller, then the first (ATT_MTU - 4) or 253 octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) can be used to read the remaining octets of a long attribute value.

The Attribute Data field is comprised of a list of attribute handle and value pairs as defined in Table 3.17[attribute-protocol--att-.html#UUID-5819983a-b26e-80c8-4592-6429394aacb0_informaltable-idm13358909202896].

Attribute Handle	**Attribute Value**
2 octets	(Length – 2) octets

Table 3.17: Format of the Attribute Data

3.4.4.3. ATT_READ_REQ

The ATT_READ_REQ PDU is used to request the server to read the value of an attribute and return its value in an ATT_READ_RSP PDU.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x0A = ATT_READ_REQ PDU
Attribute Handle	2	The handle of the attribute to be read

Table 3.18: Format of ATT_READ_REQ PDU

The attribute handle parameter shall be set to a valid handle.

The server shall respond with an ATT_READ_RSP PDU if the handle is valid and the attribute has sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

3.4.4.4. ATT_READ_RSP

The ATT_READ_RSP PDU is sent in reply to a received *Read Request* and contains the value of the attribute that has been read.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x0B = ATT_READ_RSP PDU
Attribute Value	0 to (ATT_MTU-1)	The value of the attribute with the handle given

Table 3.19: Format of ATT_READ_RSP PDU

The attribute value shall be set to the value of the attribute identified by the attribute handle in the request. If the attribute value is longer than

(ATT_MTU-1) then the first (ATT_MTU-1) octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) can be used to read the remaining octets of a long attribute value.

3.4.4.5. ATT_READ_BLOB_REQ

The ATT_READ_BLOB_REQ PDU is used to request the server to read part of the value of an attribute at a given offset and return a specific part of the value in an ATT_READ_BLOB_RSP PDU.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x0C = ATT_READ_BLOB_REQ PDU
Attribute Handle	2	The handle of the attribute to be read
Value Offset	2	The offset of the first octet to be read

Table 3.20: Format of ATT_READ_BLOB_REQ PDU

The attribute handle parameter shall be set to a valid handle.

The value offset parameter is based from zero; the first value octet has an offset of zero, the second octet has a value offset of one, etc.

The server shall respond with an ATT_READ_BLOB_RSP PDU if the handle is valid and the attribute and value offset is not greater than the length of the attribute value and has sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

If the value offset of the *Read Blob Request* is greater than the length of the attribute value, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Offset* (0x07).

If the attribute value has a fixed length that is less than or equal to (ATT_MTU - 1) octets in length, then an ATT_ERROR_RSP PDU may be sent with the Error Code parameter set to *Attribute Not Long* (0x0B).

If the value offset of the ATT_READ_BLOB_REQ PDU is equal to the length of the attribute value, then the length of the part attribute value in the response shall be zero.

Note: If the attribute is longer than (ATT_MTU-1) octets, the ATT_READ_BLOB_REQ PDU is the only way to read the additional octets of a long attribute. The first (ATT_MTU-1) octets may be read using an ATT_READ_RSP, an ATT_HANDLE_VALUE_NTF or an ATT_HANDLE_VALUE_IND PDU.

Note: Some, but not all, long attributes have their length specified by a higher layer specification. If the long attribute has a variable length, the only way to get to the end of it is to read it part by part until the value in the ATT_READ_BLOB_RSP PDU has a length shorter than (ATT_MTU-1) or an ATT_ERROR_RSP PDU is sent with the Error Code parameter set to *Invalid Offset* (0x07).

Note: The value of a Long Attribute may change between the server receiving one ATT_READ_BLOB_REQ PDU and the next ATT_READ_BLOB_REQ PDU. A higher layer specification should be aware of this and define appropriate behavior.

3.4.4.6. ATT_READ_BLOB_RSP

The ATT_READ_BLOB_RSP PDU is sent in reply to a received ATT_READ_BLOB_REQ PDU and contains part of the value of the attribute that has been read.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x0D = ATT_READ_BLOB_RSP

Parameter	Size (octets)	Description
Part Attribute Value	0 to (ATT_MTU-1)	Part of the value of the attribute with the handle given

Table 3.21: Format of ATT_READ_BLOB_RSP PDU

The part attribute value shall be set to part of the value of the attribute identified by the attribute handle and the value offset in the request. If the value offset is equal to the length of the attribute value, then the length of the part attribute value shall be zero. If the attribute value is longer than (Value Offset + ATT_MTU-1) then (ATT_MTU-1) octets from Value Offset shall be included in this response.

3.4.4.7. ATT_READ_MULTIPLE_REQ

The ATT_READ_MULTIPLE_REQ PDU is used to request the server to read two or more values of a set of attributes and return their values in an ATT_READ_MULTIPLE_RSP PDU. Only values that have a known fixed size can be read, with the exception of the last value that can have a variable length. The knowledge of whether attributes have a known fixed size is defined in a higher layer specification.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x0E = ATT_READ_MULTIPLE_REQ PDU
Set Of Handles	4 to (ATT_MTU-1)	A set of two or more attribute handles.

Table 3.22: Format of ATT_READ_MULTIPLE_REQ PDU

The attribute handles in the Set Of Handles parameter shall be valid handles.

The server shall respond with an ATT_READ_MULTIPLE_RSP PDU if all the handles are valid and all attributes have sufficient permissions to allow reading.

Note: The attribute values for the attributes in the Set Of Handles parameters do not have to all be the same size.

Note: The attribute handles in the Set Of Handles parameter do not have to be in attribute handle order; they are in the order that the values are required in the response.

If the client has insufficient authorization to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to read any of the attributes then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If any of the handles are invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If any of the attribute values cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02).

If an ATT_ERROR_RSP PDU is sent, the Attribute Handle In Error parameter shall be set to the handle of the first attribute causing the error.

3.4.4.8. ATT_READ_MULTIPLE_RSP

The ATT_READ_MULTIPLE_RSP PDU is sent in reply to a received ATT_READ_MULTIPLE_REQ PDU and contains the values of the attributes that have been read.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x0F = ATT_READ_MULTIPLE_RSP PDU

Parameter	Size (octets)	Description
Set Of Values	0 to (ATT_MTU-1)	A set of two or more values

Table 3.23: Format of ATT_READ_MULTIPLE_RSP PDU

The Set Of Values parameter shall be a concatenation of attribute values for each of the attribute handles in the request in the order that they were requested. If the Set Of Values parameter is longer than (ATT_MTU-1) then only the first (ATT_MTU-1) octets shall be included in this response.

Note: A client should not use this request for attributes when the Set Of Values parameter could be (ATT_MTU-1) as it will not be possible to determine if the last attribute value is complete, or if it overflowed.

3.4.4.9. ATT_READ_BY_GROUP_TYPE_REQ

The ATT_READ_BY_GROUP_TYPE_REQ PDU is used to obtain the values of attributes where the attribute type is known, the type of a grouping attribute as defined by a higher layer specification, but the handle is not known.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x10 = ATT_READ_BY_GROUP_TYPE_REQ PDU
Starting Handle	2	First requested handle number
Ending Handle	2	Last requested handle number
Attribute Group Type	2 or 16	2 or 16 octet UUID

Table 3.24: Format of ATT_READ_BY_GROUP_TYPE_REQ PDU

Only the attributes with attribute handles between the Starting Handle and the Ending Handle with the attribute type that is the same as the Attribute Group Type given will be returned. To search through all attributes, the starting handle shall be set to 0x0001 and the ending handle shall be set to 0xFFFF.

Note: All attribute types are effectively compared as 128-bit UUIDs, even if a 16-bit UUID is provided in this request or defined for an attribute. See [Vol 3] Part B, Section 2.5.1[service-discovery-protocol--sdp--specification.html#UUID-ef710684-4c7e-6793-4350-4a190ea9a7a4].

The starting handle shall be less than or equal to the ending handle. If a server receives an ATT_READ_BY_GROUP_TYPE_REQ PDU with the Starting Handle parameter greater than the Ending Handle parameter or the Starting Handle parameter is 0x0000, an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01). The Attribute Handle In Error parameter shall be set to the Starting Handle parameter.

If the Attribute Group Type is not a supported grouping attribute as defined by a higher layer specification then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Unsupported Group Type* (0x10). The Attribute Handle In Error parameter shall be set to the Starting Handle.

If no attribute with the given type exists within the handle range, then no attribute handle and value will be returned, and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Attribute Not Found* (0x0A). The Attribute Handle In Error parameter shall be set to the starting handle.

The attributes returned shall be the attributes with the lowest handles within the handle range. These are known as the requested attributes.

If the attributes with the requested type within the handle range have attribute values that have the same length, then these attributes can all be read in a single request. However, if those attributes have different lengths, then multiple ATT_READ_BY_GROUP_TYPE_REQ PDUs must be issued.

The ATT Server shall include as many attributes as possible in the response in order to minimize the number of PDUs required to read attributes of the same type.

When multiple attributes match, then the rules below shall be applied to each in turn.

- Only attributes that can be read shall be returned in an ATT_READ_BY_GROUP_TYPE_RSP PDU.
- If an attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then this attribute cannot be included in an ATT_READ_BY_GROUP_TYPE_RSP PDU and the attributes before this attribute shall be returned.
- If the first attribute in the set of requested attributes would cause an ATT_ERROR_RSP PDU then no other attributes in the requested attributes can be considered.

The server shall respond with an ATT_READ_BY_GROUP_TYPE_RSP PDU if the requested attributes have sufficient permissions to allow reading.

If the client has insufficient authorization to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has insufficient security to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has an encryption key size that is too short to read the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the client has not enabled encryption, and encryption is required to read the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

If the requested attribute’s value cannot be read due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Read Not Permitted* (0x02). The Attribute Handle In Error parameter shall be set to the handle of the attribute causing the error.

Note: If there are multiple attributes with the requested type within the handle range, and the client would like to get the next attribute with the requested type, it would have to issue another ATT_READ_BY_GROUP_TYPE_REQ PDU with its starting handle updated. The client can be sure there are no more such attributes remaining once it gets an ATT_ERROR_RSP PDU with the Error Code parameter set to *Attribute Not Found* (0x0A).

The server shall not respond to the ATT_READ_BY_GROUP_TYPE_REQ PDU with an ATT_ERROR_RSP PDU with the error code *Database Out of Sync* (0x12).

3.4.4.10. ATT_READ_BY_GROUP_TYPE_RSP

The ATT_READ_BY_GROUP_TYPE_RSP PDU is sent in reply to a received ATT_READ_BY_GROUP_TYPE_REQ PDU and contains the handles and values of the attributes that have been read.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x11 = ATT_READ_BY_GROUP_TYPE_RSP PDU
Length	1	The size of each Attribute Data
Attribute Data List	4 to (ATT_MTU-2)	A list of Attribute Data

Table 3.25: Format of ATT_READ_BY_GROUP_TYPE_RSP PDU

The ATT_READ_BY_GROUP_TYPE_RSP PDU shall contain complete Attribute Data. An Attribute Data shall not be split across response packets. The Attribute Data List is ordered sequentially based on the attribute handles.

The Length parameter shall be set to the size of the one Attribute Data.

The maximum length of an Attribute Data is 255 octets, bounded by the Length parameter that is one octet. Therefore, the maximum length of an attribute value returned in this response is (Length – 4) = 251 octets.

The Attribute Data List shall be set to the value of the attributes identified by the attribute type within the handle range within the request. If the attribute value is longer than (ATT_MTU - 6) or 251 octets, whichever is smaller, then the first (ATT_MTU - 6) or 251 octets shall be included in this response.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) can be used to read the remaining octets of a long attribute value.

The Attribute Data List is comprised of a list of Attribute Data as defined in Table 3.26[attribute-protocol--att-.html#UUID-6a4d7c47-0d76-73e0-b418-fab96fd9a14f_table-idm13358909789874].

Attribute Handle	**End Group Handle**	**Attribute Value**
2 octets	2 octets	(Length - 4) octets

Table 3.26: Format of the Attribute Data

3.4.4.11. ATT_READ_MULTIPLE_VARIABLE_REQ

The ATT_READ_MULTIPLE_VARIABLE_REQ PDU is used to request that the server read two or more values of a set of attributes that have a variable or unknown value length and return their values in an ATT_READ_MULTIPLE_VARIABLE_RSP PDU.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x20 = ATT_READ_MULTIPLE_VARIABLE_REQ PDU
Set Of Handles	4 to (ATT_MTU-1)	A set of two or more attribute handles

Table 3.27: Format of ATT_READ_MULTIPLE_VARIABLE_REQ PDU

The attribute handles in the Set Of Handles parameter shall all be valid handles.

The server shall respond with an ATT_READ_MULTIPLE_VARIABLE_RSP PDU if all attributes have sufficient permissions to allow reading.

Note: The attribute values for the attributes in the Set Of Handles parameters do not have to all be the same size.

Note: The attribute handles in the Set Of Handles parameter do not have to be in attribute handle order; they are in the order that the values are required in the response.

If the client has insufficient authorization to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Authorization*.

If the client has insufficient security to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Authentication*.

If the client has an encryption key size that is too short to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Encryption Key Size Too Short*.

If the client has not enabled encryption, and encryption is required to read any of the attributes, then an ATT_ERROR_RSP PDU shall be sent with the error code *Insufficient Encryption*.

If any of the handles are invalid, then an ATT_ERROR_RSP PDU shall be sent with the error code *Invalid Handle*.

If any of the attribute values cannot be read due to permissions, then an ATT_ERROR_RSP PDU shall be sent with the error code *Read Not Permitted*.

If an ATT_ERROR_RSP PDU is sent, the Attribute Handle In Error parameter in the ATT_ERROR_RSP PDU (see Section 3.4.1.1[attribute-protocol--att-.html#UUID-eefd3e8d-9b16-3af8-1fb4-fa90f52262e8]) shall be set to the handle of the first attribute causing the error.

3.4.4.12. ATT_READ_MULTIPLE_VARIABLE_RSP

The ATT_READ_MULTIPLE_VARIABLE_RSP PDU is sent in reply to a received ATT_READ_MULTIPLE_VARIABLE_REQ PDU and contains the lengths and values of the attributes that have been read.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x21 = ATT_READ_MULTIPLE_VARIABLE_RSP
Length Value Tuple List	4 to (ATT_MTU-1)	A list of Length Value Tuples

Table 3.28: Format of ATT_READ_MULTIPLE_VARIABLE_RSP PDU

The Length Value Tuple List shall be a concatenation of Length Value Tuples for each of the attribute handles in the request in the order that they were requested. If the Length Value Tuple List is longer than (ATT_MTU-1) octets, then it shall be truncated after (ATT_MTU-1) or, if that would be within the Value Length field of a Length Value Tuple, at the start of the Length Value Tuple.

Note: The ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) can be used to read the remaining octets of a long attribute value.

The Value Length field in a Length Value Tuple shall be set to the length of the Attribute Value field. The Attribute Value field in a Length Value Tuple shall be set to the value of the attribute being read.

Value Length	Attribute Value
2 octets	(Value Length) octets

Table 3.29: Format of the Length Value Tuple

Note: If a Length Value Tuple is truncated, then the amount of Attribute Value will be less than the value of the Value Length field. The client must therefore not use the Value Length to determine the amount of the Attribute Value actually included in the PDU.

3.4.5. Writing attributes

3.4.5.1. ATT_WRITE_REQ

The ATT_WRITE_REQ PDU is used to request the server to write the value of an attribute and acknowledge that this has been achieved in an ATT_WRITE_RSP PDU.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x12 = ATT_WRITE_REQ PDU
Attribute Handle	2	The handle of the attribute to be written
Attribute Value	0 to (ATT_MTU-3)	The value to be written to the attribute

Table 3.30: Format of ATT_WRITE_REQ PDU

The Attribute Handle shall be set to a valid handle.

The Attribute Value shall be set to the new value of the attribute.

If the attribute value has a variable length, then the attribute value shall be truncated or lengthened to match the length of the Attribute Value parameter.

Note: If an attribute value has a variable length and if the Attribute Value parameter is of zero length, the attribute value will be fully truncated.

If the attribute value has a fixed length and the Attribute Value parameter length is less than or equal to the length of the attribute value, the octets of the attribute value parameter length shall be written; all other octets in this attribute value shall be unchanged.

The server shall respond with an ATT_WRITE_RSP PDU if the handle is valid, the attribute has sufficient permissions to allow writing, and the attribute value has a valid size and format, and it is successful in writing the attribute.

If the attribute value has a variable length and the Attribute Value parameter length exceeds the maximum valid length of the attribute value then the server shall respond with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the attribute value has a fixed length and the requested attribute value parameter length is greater than the length of the attribute value then the server shall respond with an ATT_ERROR_RSP PDU with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the client has insufficient authorization to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to write the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be written due to permissions then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Write Not Permitted* (0x03).

If the attribute value cannot be written due to an application error then an ATT_ERROR_RSP PDU shall be sent with an error code defined by a higher layer specification.

3.4.5.2. ATT_WRITE_RSP

The ATT_WRITE_RSP PDU is sent in reply to a valid ATT_WRITE_REQ PDU and acknowledges that the attribute has been successfully written.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x13 = ATT_WRITE_RSP PDU

Table 3.31: Format of ATT_WRITE_RSP

The ATT_WRITE_RSP PDU shall be sent after the attribute value is written.

3.4.5.3. ATT_WRITE_CMD

The ATT_WRITE_CMD PDU is used to request the server to write the value of an attribute, typically into a control-point attribute.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x52 = ATT_WRITE_CMD PDU
Attribute Handle	2	The handle of the attribute to be set
Attribute Value	0 to (ATT_MTU-3)	The value of be written to the attribute

Table 3.32: Format of ATT_WRITE_CMD PDU

The attribute handle parameter shall be set to a valid handle.

The attribute value parameter shall be set to the new value of the attribute.

If the attribute value has a variable length, then the attribute value shall be truncated or lengthened to match the length of the attribute value parameter.

Note: If an attribute value has a variable length and if the attribute value parameter is of zero length, the attribute value will be fully truncated.

If the attribute value has a fixed length and the attribute value parameter length is less than or equal to the length of the attribute value, the octets up to the attribute value parameter length shall be written; all other octets in this attribute value shall be unchanged.

If the attribute value has a variable length and the attribute value parameter length exceeds the maximum valid length of the attribute value then the server shall ignore the command.

If the attribute value has a fixed length and the requested attribute value parameter length is greater than the length of the attribute value then the server shall ignore the command.

No ATT_ERROR_RSP or ATT_WRITE_RSP PDUs shall be sent in response to this command. If the server cannot write this attribute for any reason the command shall be ignored.

3.4.5.4. ATT_SIGNED_WRITE_CMD

This command shall not be used on an Enhanced ATT bearer.

The ATT_SIGNED_WRITE_CMD PDU is used to request the server to write the value of an attribute with an authentication signature, typically into a control-point attribute.

Parameter	**Size (Octets)**	**Description**
Attribute Opcode	1	0xD2 = ATT_SIGNED_WRITE_CMD PDU
Attribute Handle	2	The handle of the attribute to be set
Attribute Value	0 to (ATT_MTU - 15)	The value to be written to the attribute
Authentication Signature	12	Authentication signature for the Attribute Opcode, Attribute Handle and Attribute Value parameters

Table 3.33: Format of ATT_SIGNED_WRITE_CMD

The attribute handle parameter shall be set to a valid handle.

The attribute value parameter shall be set to the new value of the attribute.

The attribute signature shall be calculated as defined in Section 3.3.1[attribute-protocol--att-.html#UUID-1d6830c8-e41c-3a31-e431-089d814c5cd1].

For example, if the variable length message m to be signed is ‘D212001337’, SignCounter is 0x00000001 and key is 0x611B64EBFBCD1FD372EC9196DF425E50, then message to be signed (M) by the CMAC function is the octet sequence ‘D21200133701000000’.

The padding(M) is 0x0000000137130012D280000000000000, resultant CMAC is 0xF20F903C931E87F159B64F012574B4D0 and Authentication Signature is the octet sequence ‘01000000F1871E933C900FF2’.

The final signed message is ‘D21200133701000000F1871E933C900FF2’.

If the attribute value has a variable length, then the attribute value shall be truncated or lengthened to match the length of the attribute value parameter.

Note: If an attribute value has a variable length and if the attribute value parameter is of zero length, the attribute value will be fully truncated.

If the attribute value has a fixed length and the attribute value parameter length is less than or equal to the length of the attribute value, the octets up to the attribute value parameter length shall be written; all other octets in this attribute value shall be unchanged.

If the attribute value has a variable length and the attribute value parameter length exceeds the maximum valid length of the attribute value then the server shall ignore the command.

If the attribute value has a fixed length and the requested attribute value parameter length is greater than the length of the attribute value then the server shall ignore the command.

If the authentication signature verification fails, then the server shall ignore the command.

No ATT_ERROR_RSP PDU or ATT_WRITE_RSP PDU shall be sent in response to this command. If the server cannot write this attribute for any reason the command shall be ignored.

3.4.6. Queued writes

The purpose of queued writes is to queue up writes of values of multiple attributes in a first-in first-out queue and then execute the write on all of them in a single atomic operation.

Each client's queued values are separate; the execution of one queue shall not affect the preparation or execution of any other client's queued values. Each client has a single queue regardless of how many ATT bearers are currently established.

3.4.6.1. ATT_PREPARE_WRITE_REQ

The ATT_PREPARE_WRITE_REQ PDU is used to request the server to prepare to write the value of an attribute. The server will respond to this request with an ATT_PREPARE_WRITE_RSP PDU, so that the client can verify that the value was received correctly.

A client may send more than one ATT_PREPARE_WRITE_REQ PDU to a server, which will queue and send a response for each handle value pair.

A server may limit the number of prepared writes that it can queue. A higher layer specification should define this limit.

After an ATT_PREPARE_WRITE_REQ PDU has been issued, and the response received, any other attribute command or request can be issued from the same client to the same server.

Any actions on attributes that exist in the prepare queue shall proceed as if the prepare queue did not exist, and the prepare queue shall be unaffected by these actions. A subsequent execute write will write the values in the prepare queue even if the value of the attribute has changed since the prepare writes were started.

The Attribute Protocol makes no determination on the validity of the Part Attribute Value or the Value Offset. A higher layer specification determines the meaning of the data.

Each ATT_PREPARE_WRITE_REQ PDU will be queued even if the attribute handle is the same as a previous ATT_PREPARE_WRITE_REQ PDU. These will then be executed in the order received, causing multiple writes for this attribute to occur.

If all ATT bearers belonging to the same client are lost while a number of pending prepare write values have been queued, the queue will be cleared and no writes will be executed.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x16 = ATT_PREPARE_WRITE_REQ PDU
Attribute Handle	2	The handle of the attribute to be written
Value Offset	2	The offset of the first octet to be written
Part Attribute Value	0 to (ATT_MTU-5)	The value of the attribute to be written

Table 3.34: Format of ATT_PREPARE_WRITE_REQ PDU

The Attribute Handle parameter shall be set to a valid handle.

The Value Offset parameter shall be set to the offset of the first octet where the Part Attribute Value parameter is to be written within the attribute value. The Value Offset parameter is based from zero; the first octet has an offset of zero, the second octet has an offset of one, etc.

The server shall respond with an ATT_PREPARE_WRITE_RSP PDU if the handle is valid, the attribute has sufficient permissions to allow writing at this time, and the prepare queue has sufficient space.

Note: The Attribute Value validation is done when an ATT_EXECUTE_WRITE_REQ PDU is received. Hence, any *Invalid Offset* (0x07) or *Invalid Attribute Value Length* (0x0D) errors are generated when an ATT_EXECUTE_WRITE_REQ PDU is received.

If the client has insufficient authorization to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authorization* (0x08).

If the client has insufficient security to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Authentication* (0x05).

If the client has an encryption key size that is too short to write the requested attribute then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Encryption Key Size Too Short* (0x0C).

If the client has not enabled encryption, and encryption is required to write the requested attribute, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Insufficient Encryption* (0x0F).

If the server does not have sufficient space to queue this request then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Prepare Queue Full* (0x09).

If the handle is invalid, then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Handle* (0x01).

If the attribute value cannot be written then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Write Not Permitted* (0x03).

The server shall not change the value of the attribute until an ATT_EXECUTE_WRITE_REQ PDU is received.

If an ATT_PREPARE_WRITE_REQ PDU was invalid, and therefore an ATT_ERROR_RSP PDU has been issued, then this prepared write will be considered to have not been received. All existing prepared writes in the prepare queue shall not be affected by this invalid request.

3.4.6.2. ATT_PREPARE_WRITE_RSP

The ATT_PREPARE_WRITE_RSP PDU is sent in response to a received ATT_PREPARE_WRITE_REQ PDU and acknowledges that the value has been successfully received and placed in the prepare write queue.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x17 = ATT_PREPARE_WRITE_RSP PDU
Attribute Handle	2	The handle of the attribute to be written
Value Offset	2	The offset of the first octet to be written

Parameter	Size (octets)	Description
Part Attribute Value	0 to (ATT_MTU-5)	The value of the attribute to be written

Table 3.35: Format of ATT_PREPARE_WRITE_RSP PDU

The attribute handle shall be set to the same value as in the corresponding ATT_PREPARE_WRITE_REQ PDU.

The value offset and part attribute value shall be set to the same values as in the corresponding ATT_PREPARE_WRITE_REQ PDU.

3.4.6.3. ATT_EXECUTE_WRITE_REQ

The ATT_EXECUTE_WRITE_REQ PDU is used to request the server to write or cancel the write of all the prepared values currently held in the prepare queue from this client. This request shall be handled by the server as an atomic operation.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x18 = ATT_EXECUTE_WRITE_REQ PDU
Flags	1	0x00 – Cancel all prepared writes 0x01 – Immediately write all pending prepared values

Table 3.36: Format of ATT_EXECUTE_WRITE_REQ PDU

When the flags parameter is set to 0x01, all pending prepare write values that are currently queued shall be written in the order they were received in the corresponding ATT_PREPARE_WRITE_REQ PDUs. The queue shall then be cleared and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

When the flags parameter is set to 0x00 all pending prepare write values shall be discarded for this client. The queue shall then be cleared, and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

Note: If multiple ATT bearers were used to send the prepare write requests, then the order that writes sent on different ATT bearers are executed is not specified and is not reported to the client. In addition, if the ATT_EXECUTE_WRITE_REQ PDU is sent before the client has received responses to all the prepare write requests, the requests for which the client has not yet received a response might form part of a subsequent write rather than this one; this is also not reported to the client.

If there are no pending prepared write values, then no values are written, and an ATT_EXECUTE_WRITE_RSP PDU shall be sent.

If the prepared Attribute Value exceeds the maximum valid length of the attribute value then all pending prepare write values shall be discarded for this client, the queue shall then be cleared, and an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Attribute Value Length* (0x0D).

If the prepare Value Offset is greater than the current length of the attribute value then all pending prepare write values shall be discarded for this client, the queue shall be cleared and then an ATT_ERROR_RSP PDU shall be sent with the Error Code parameter set to *Invalid Offset* (0x07).

If the pending prepared write values cannot be written, due to an application error, the queue shall be cleared and then an ATT_ERROR_RSP PDU shall be sent with a higher layer specification defined error code. The Attribute Handle In Error parameter shall be set to the attribute handle of the attribute from the prepare queue that caused this application error. The state of the attributes that were to be written from the prepare queue is not defined in this case.

3.4.6.4. ATT_EXECUTE_WRITE_RSP

The ATT_EXECUTE_WRITE_RSP PDU is sent in response to a received ATT_EXECUTE_WRITE_REQ PDU.

Parameter	**Size**	**Description**
Attribute Opcode	1	0x19 - ATT_EXECUTE_WRITE_RSP PDU

Table 3.37: Format of ATT_EXECUTE_WRITE_RSP PDU

The ATT_EXECUTE_WRITE_RSP PDU shall be sent after the attributes are written. In case an action is taken in response to the write, an indication may be used once the action is complete.

3.4.7. Server initiated

3.4.7.1. ATT_HANDLE_VALUE_NTF

A server can send a notification of an attribute's value at any time.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x1B = ATT_HANDLE_VALUE_NTF PDU
Attribute Handle	2	The handle of the attribute
Attribute Value	0 to (ATT_MTU-3)	The current value of the attribute

Table 3.38: Format of ATT_HANDLE_VALUE_NTF PDU

The attribute handle shall be set to a valid handle.

The attribute value shall be set to the current value of attribute identified by the attribute handle.

If the attribute value is longer than (ATT_MTU-3) octets, then only the first (ATT_MTU-3) octets of this attributes value can be sent in a notification.

Note: For a client to get a long attribute, it must use the ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) following the receipt of this notification.

If the attribute handle or the attribute value is invalid, then this notification shall be ignored upon reception.

3.4.7.2. ATT_HANDLE_VALUE_IND

A server can send an indication of an attribute’s value.

Parameter	**Size (octets)**	**Description**
Attribute Opcode	1	0x1D = ATT_HANDLE_VALUE_IND PDU
Attribute Handle	2	The handle of the attribute
Attribute Value	0 to (ATT_MTU-3)	The current value of the attribute

Table 3.39: Format of ATT_HANDLE_VALUE_IND PDU

The attribute handle shall be set to a valid handle.

The attribute value shall be set to the current value of attribute identified by the attribute handle.

If the attribute value is longer than (ATT_MTU-3) octets, then only the first (ATT_MTU - 3) octets of this attributes value can be sent in an indication.

Note: For a client to get a long attribute, it must use the ATT_READ_BLOB_REQ PDU (see Section 3.4.4.5[attribute-protocol--att-.html#UUID-e5c23eef-bf66-5877-e1be-ff53b032d9bd]) following the receipt of this indication.

The client shall send an ATT_HANDLE_VALUE_CFM PDU in response to an ATT_HANDLE_VALUE_IND PDU. No further indications to this client shall occur until the confirmation has been received by the server.

If the attribute handle or the attribute value is invalid, the client shall send an ATT_HANDLE_VALUE_CFM PDU in response and shall discard the handle and value from the received indication.

3.4.7.3. ATT_HANDLE_VALUE_CFM

The ATT_HANDLE_VALUE_CFM PDU is sent in response to a received ATT_HANDLE_VALUE_IND PDU and confirms that the client has received an indication of the given attribute.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x1E = ATT_HANDLE_VALUE_CFM PDU

Table 3.40: Format of ATT_HANDLE_VALUE_CFM PDU

3.4.7.4. ATT_MULTIPLE_HANDLE_VALUE_NTF

A server can send a notification of two or more attributes' values at any time.

Parameter	Size (octets)	Description
Attribute Opcode	1	0x23 = ATT_MULTIPLE_HANDLE_-VALUE_NTF
Handle Length Value Tuple List	8 to (ATT_MTU-1)	A list of Handle Length Value Tuples

Table 3.41: Format of ATT_MULTIPLE_HANDLE_VALUE_NTF PDU

The Handle Length Value Tuple List shall be a concatenation of Handle Length Value Tuples for each of the attributes being notified.

The server shall not truncate a Handle Length Value Tuple.

The Attribute Handle field in a Handle Length Value Tuple shall be set to the handle of the attribute being notified. The Value Length field in a Handle Length Value Tuple shall be set to the length of the Attribute Value field. The Attribute Value field in a Handle Length Value Tuple shall be set to the value of the attribute being notified.

Attribute Handle	Value Length	Attribute Value
2 octets	2 octets	(Value Length) octets

Table 3.42: Format of the Handle Length Value Tuple

If an attribute handle or an attribute value is invalid, then the client shall ignore that attribute when receiving this notification.

3.4.8. Attribute Opcode summary

Table 3.43[attribute-protocol--att-.html#UUID-2cdddfc5-98bd-4ed9-76e7-bcce33836208_table-idm13358910852784] gives a summary of the Attribute Protocol PDUs.

Attribute PDU Name	Attribute Opcode	Parameters
ATT_ERROR_RSP	0x01	Request Opcode in Error, Attribute Handle In Error, Error Code
ATT_EXCHANGE_MTU_REQ	0x02	Client Rx MTU
ATT_EXCHANGE_MTU_RSP	0x03	Server Rx MTU
ATT_FIND_INFORMATION_REQ	0x04	Starting Handle, Ending Handle
ATT_FIND_INFORMATION_RSP	0x05	Format, Information Data
ATT_FIND_BY_TYPE_VALUE_REQ	0x06	Starting Handle, Ending Handle, Attribute Type, Attribute Value
ATT_FIND_BY_TYPE_VALUE_RSP	0x07	Handles Information List

Attribute PDU Name	Attribute Opcode	Parameters
ATT_READ_BY_TYPE_REQ	0x08	Starting Handle, Ending Handle, UUID
ATT_READ_BY_TYPE_RSP	0x09	Length, Attribute Data List
ATT_READ_REQ	0x0A	Attribute Handle
ATT_READ_RSP	0x0B	Attribute Value
ATT_READ_BLOB_REQ	0x0C	Attribute Handle, Value Offset
ATT_READ_BLOB_RSP	0x0D	Part Attribute Value
ATT_READ_MULTIPLE_REQ	0x0E	Handle Set
ATT_READ_MULTIPLE_RSP	0x0F	Value Set
ATT_READ_BY_GROUP_TYPE_REQ	0x10	Start Handle, Ending Handle, UUID
ATT_READ_BY_GROUP_TYPE_RSP	0x11	Length, Attribute Data List
ATT_WRITE_REQ	0x12	Attribute Handle, Attribute Value
ATT_WRITE_RSP	0x13	*none*
ATT_WRITE_CMD	0x52	Attribute Handle, Attribute Value

Attribute PDU Name	Attribute Opcode	Parameters
ATT_PREPARE_WRITE_REQ	0x16	Attribute Handle, Value Offset, Part Attribute Value
ATT_PREPARE_WRITE_RSP	0x17	Attribute Handle, Value Offset, Part Attribute Value
ATT_EXECUTE_WRITE_REQ	0x18	Flags
ATT_EXECUTE_WRITE_RSP	0x19	*none*
ATT_READ_MULTIPLE_VARIABLE_REQ	0x20	Set Of Handles
ATT_READ_MULTIPLE_VARIABLE_RSP	0x21	Length Value Tuple List
ATT_MULTIPLE_HANDLE_VALUE_NTF	0x23	Handle Length Value Tuple List
ATT_HANDLE_VALUE_NTF	0x1B	Attribute Handle, Attribute Value
ATT_HANDLE_VALUE_IND	0x1D	Attribute Handle, Attribute Value
ATT_HANDLE_VALUE_CFM	0x1E	*none*
ATT_SIGNED_WRITE_CMD	0xD2	Attribute Handle, Attribute Value, Authentication Signature

Table 3.43: Attribute Protocol summary

3.4.9. Attribute PDU response summary

Table 3.44[attribute-protocol--att-.html#UUID-191d4479-95b1-b04e-88b1-229f52d4fb60_table-idm13358910905292] gives a summary of the Attribute PDU Method responses that are allowed. Each method indicates the method that should be sent as a successful response, and whether an ATT_ERROR_RSP PDU can be sent in response instead. If an ATT_ERROR_RSP PDU can be sent, then Table 3.44[attribute-protocol--att-.html#UUID-191d4479-95b1-b04e-88b1-229f52d4fb60_table-idm13358910905292] also indicates the error codes that are valid within this ATT_ERROR_RSP PDU for the given method.

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_EXCHANGE_-MTU_REQ	ATT_EXCHANGE_-MTU_RSP	Yes	*Request Not Supported* (0x06)
ATT_FIND_-INFORMATION_-REQ	ATT_FIND_-INFORMATION_-RSP	Yes	*Invalid Handle* (0x01), *Attribute Not Found* (0x0A)
ATT_FIND_BY_-TYPE_VALUE_REQ	ATT_FIND_BY_-TYPE_VALUE_RSP	Yes	*Invalid Handle* (0x01), *Request Not Supported* (0x06), *Attribute Not Found* (0x0A)

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_READ_BY_-TYPE_REQ	ATT_READ_BY_-TYPE_RSP	Yes	*Invalid Handle* (0x01), *Database Out of Sync* (0x12), *Request Not Supported* (0x06), *Attribute Not Found* (0x0A), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_READ_REQ	ATT_READ_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_READ_-BLOB_REQ	ATT_READ_-BLOB_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Request Not Supported* (0x06), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Invalid Offset* (0x07), *Attribute Not Long* (0x0B), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	**Successful Response PDU**	**ATT_ERROR_-RSP Allowed**	**Valid Error Codes**
ATT_READ_-MULTIPLE_REQ	ATT_READ_-MULTIPLE_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Request Not Supported* (0x06), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_READ_BY_-GROUP_TYPE_-REQ	ATT_READ_BY_-GROUP_TYPE_-RSP	Yes	*Invalid Handle* (0x01), *Request Not Supported* (0x06), *Attribute Not Found* (0x0A), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Unsupported Group Type* (0x10), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_READ_-MULTIPLE_-VARIABLE_REQ	ATT_READ_-MULTIPLE_-VARIABLE_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Request Not Supported* (0x06), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Read Not Permitted* (0x02), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	**Successful Response PDU**	**ATT_ERROR_-RSP Allowed**	**Valid Error Codes**
ATT_WRITE_REQ	ATT_WRITE_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Request Not Supported* (0x06), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Write Not Permitted* (0x03), *Invalid Attribute Value Length* (0x0D), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)
ATT_WRITE_CMD	*none*	No	*none*
ATT_SIGNED_-WRITE_CMD	*none*	No	*none*

Attribute PDU Method	Successful Response PDU	ATT_ERROR_-RSP Allowed	Valid Error Codes
ATT_PREPARE_-WRITE_REQ	ATT_PREPARE_-WRITE_RSP	Yes	*Invalid Handle* (0x01), *Database Out Of Sync* (0x12), *Request Not Supported* (0x06), *Insufficient Authorization* (0x08), *Insufficient Authentication* (0x05), *Write Not Permitted* (0x03), *Prepare Queue Full* (0x09), *Insufficient Encryption* (0x0F), *Encryption Key Size Too Short* (0x0C), *Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF)

Attribute PDU Method	**Successful Response PDU**	**ATT_ERROR_-RSP Allowed**	**Valid Error Codes**
ATT_EXECUTE_-WRITE_REQ	ATT_EXECUTE_-WRITE_RSP	Yes	*Application Error* (0x80 to 0x9F), *Common Profile and Service Error Codes* (0xE0 to 0xFF), *Invalid Offset* (0x07), *Invalid Attribute Value Length* (0x0D)
ATT_HANDLE_-VALUE_NTF	*none*	No	*none*
ATT_HANDLE_-VALUE_IND	ATT_HANDLE_-VALUE_CFM	No	*none*
ATT_MULTIPLE_-HANDLE_VALUE_-NTF	*none*	No	*none*

Table 3.44: Attribute request and response summary

4. Security considerations

The Attribute Protocol can be used to access information that may require both authorization and an authenticated and encrypted physical link before an attribute can be read or written.

If such a request is issued when the client has not been authorized to access this information, the server shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authorization* (0x08). The authorization requirements for access to a given attribute are not defined in this Part. Each device implementation will determine how authorization occurs. Authorization procedures are defined in GAP, and may be further refined in a higher layer specification.

If such a request is issued when the physical link is unauthenticated, the server shall send an ATT_ERROR_RSP PDU with the Error Code parameter set to *Insufficient Authentication* (0x05). A client wanting to read or write this attribute can then request that the physical link be authenticated, and once this has been completed, send the request again.

The Attribute Protocol can be used to notify or indicate the value of an attribute that may require an authenticated and encrypted physical link before an attribute notification or indication is performed. A server wanting to notify or indicate this attribute can then request that the physical link be authenticated, and once this has been completed, send the notification or indication.

The list of attributes that a device supports is not considered private or confidential information, and therefore the ATT_FIND_INFORMATION_REQ PDU shall always be permitted. This implies that the error code *Insufficient Authorization* (0x08) or *Insufficient Authentication* (0x05) shall not be used in an ATT_ERROR_RSP PDU for an ATT_FIND_INFORMATION_REQ PDU.

For example, an attribute value may be allowed to be read by any device, but only written by an authenticated device. An implementation should take this into account, and not assume that just because it can read an attribute's value, it will also be able to write the value. Similarly, just because an attribute value can be written, does not mean that an attribute value can also be read. Each individual attribute could have different security requirements.

When a client accesses an attribute, the order of checks that are performed on the server will have security implications. A server shall check authentication and authorization requirements before any other check is performed.

Note: For example, if the authentication and authorization requirement checks are not performed first then the size of an attribute could be determined by sending repeated ATT_READ_BLOB_REQ PDUs for an attribute that a client does not have access to, because either the error code *Invalid Offset* (0x07) or *Insufficient Authentication* (0x05) would be returned.

5. References

[1] Core Specification Supplement, Part B, Common Profile and Service Error Codes

Appendix A. Changes to PDU names

Previous versions of this specification used different names for the PDUs defined in Section 3.4[attribute-protocol--att-.html#UUID-9a2dc3de-5db8-cf7c-b394-3f13af7a11fa]. Table A.1[attribute-protocol--att-.html#UUID-4158738b-a430-2251-0ba8-b433a01bbd52_table-idm13358911224772] shows the previous and current names of these PDUs.

Previous name	**Current name**
Error Response	ATT_ERROR_RSP
Exchange MTU Request	ATT_EXCHANGE_MTU_REQ
Exchange MTU Response	ATT_EXCHANGE_MTU_RSP
Execute Write Request	ATT_EXECUTE_WRITE_REQ
Execute Write Response	ATT_EXECUTE_WRITE_RSP
Find By Type Value Request	ATT_FIND_BY_TYPE_VALUE_REQ
Find By Type Value Response	ATT_FIND_BY_TYPE_VALUE_RSP
Find Information Request	ATT_FIND_INFORMATION_REQ
Find Information Response	ATT_FIND_INFORMATION_RSP
Handle Value Confirmation	ATT_HANDLE_VALUE_CFM
Handle Value Indication	ATT_HANDLE_VALUE_IND
Handle Value Notification	ATT_HANDLE_VALUE_NTF

Previous name	Current name
Prepare Write Request	ATT_PREPARE_WRITE_REQ
Prepare Write Response	ATT_PREPARE_WRITE_RSP
Read Blob Request	ATT_READ_BLOB_REQ
Read Blob Response	ATT_READ_BLOB_RSP
Read by Group Type Request	ATT_READ_BY_GROUP_TYPE_REQ
Read by Group Type Response	ATT_READ_BY_GROUP_TYPE_RSP
Read By Type Request	ATT_READ_BY_TYPE_REQ
Read By Type Response	ATT_READ_BY_TYPE_RSP
Read Multiple Request	ATT_READ_MULTIPLE_REQ
Read Multiple Response	ATT_READ_MULTIPLE_RSP
Read Request	ATT_READ_REQ
Read Response	ATT_READ_RSP
Signed Write Command	ATT_SIGNED_WRITE_CMD
Write Command	ATT_WRITE_CMD
Write Request	ATT_WRITE_REQ
Write Response	ATT_WRITE_RSP

Table A.1: Changes to PDU names

In this section