The default value of ATT_MTU is 23 octets for LE. The exchange MTU request is used by the client if it wants to use an ATT_MTU bigger than this value. A bigger value will finally be used after the request and response messages are exchanged and if both the client and the server support that bigger value. This is shown in Figure 13.11.

This is especially useful where a client may have bigger receive buffers compared to the server. For example, if the client is a mobile phone and the server is a thermometer. The mobile phone may send an MTU size of 300 octets in the Exchange MTU Request. If the thermometer can support an Rx MTU of only 40 bytes, it will send an Exchange MTU Response with 40 bytes. After these messages are exchanged, both the thermometer and mobile phone will use an MTU of 40 bytes. If the thermometer sent an Error Response, then both will use the default MTU of 23 bytes.

13.6.2 Primary Service Discovery

The Primary Service Discovery procedure is used by the client to discover the primary services on the server. It can either use the Discover All Primary Services sub-procedure or Discover Primary Services by Service UUID sub-procedure.

13.6.2.1 Discover All Primary Services

This subprocedure is used by the client to discover all the primary services on the server. It uses the ATT Read By Group Type Request with the following parameters:

- Attribute Type: UUID for <<Primary Service>>.
- Starting Handle: 0x0001.
- Ending Handle: 0xFFFF.

The server either sends the ATT Read By Group Type Response or an Error Response. If the server sends the Read By Group Type Response, then it contains the following parameters:

- Length: Size of each attribute data.

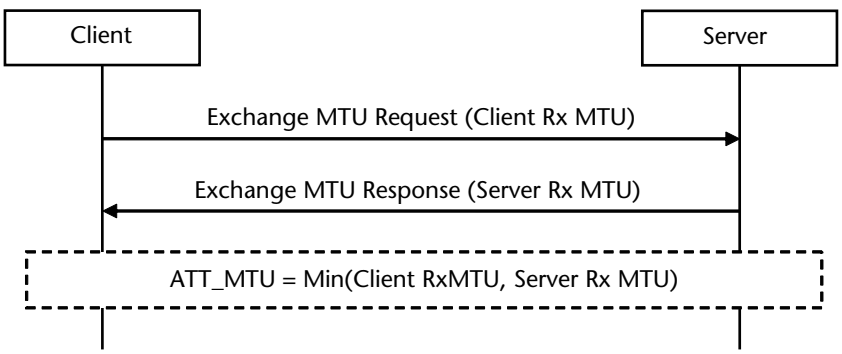

Figure 13.11 Exchange MTU.

- Attribute Data List: This contains the following:
  - Attribute Handle: Handle of the service declaration.
  - End Group Handle: Handle of last attribute within the service definition.
  - Attribute Value: Service UUID of a service supported by the server.

It is possible that not all attributes are returned in one response. In that case, the End Group Handle in the response would be less than the Ending Handle given in the request. The client can then send another Read By Group Type Request by modifying the Starting Handle to the received End Group Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.12. The client sends the Read By Group Type Request message to the server by setting the Starting Handle to 0x0001 and Ending Handle to 0xFFFF to get the attribute handles within the entire possible range

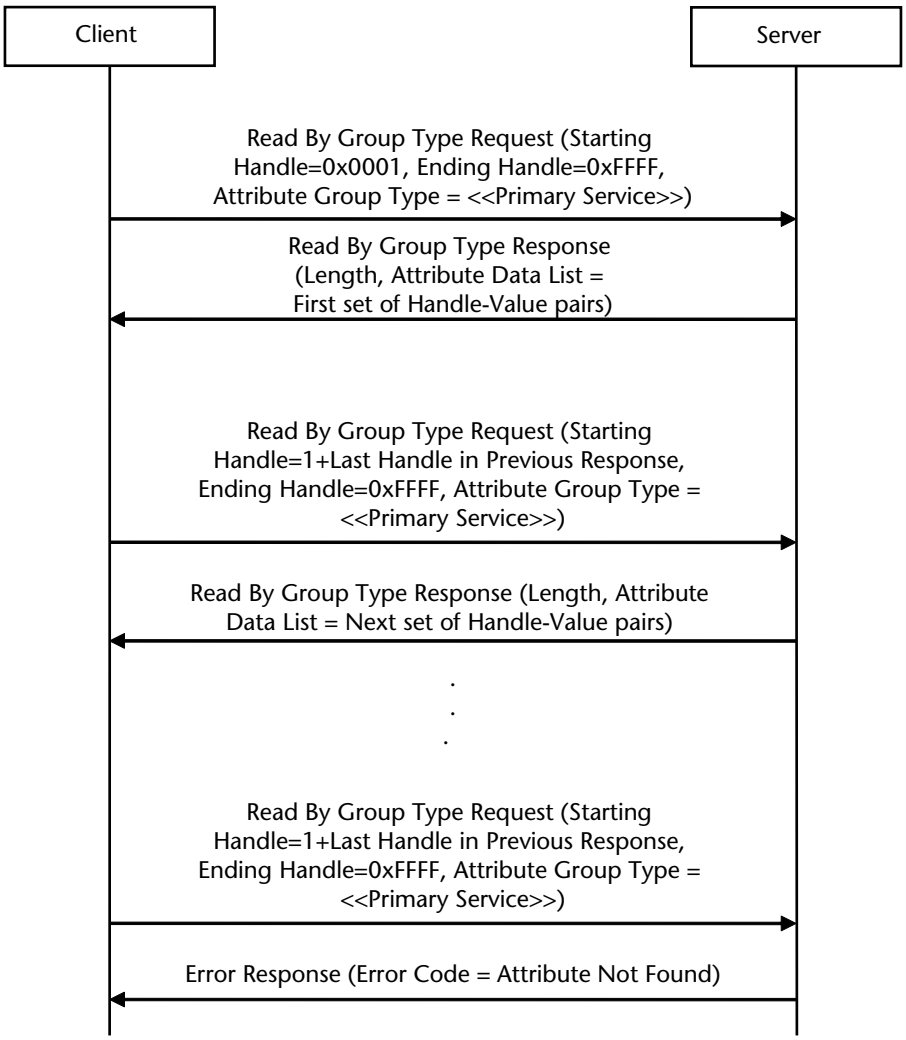

Figure 13.12 Discover all primary services.

of attribute handles. The server responds with the first set of Handle-Value pairs depending on the number of pairs that can be accommodated within one Read By Group Type Response.

The client then sends the next Read By Group Type Request by adjusting the Starting Handle to retrieve the next set of Handle-Value Pairs. The server responds with the next set of Handle Value pairs till no more remain as per the Starting Handle specified by the client. At that time the server returns an Error Response. This indicates to the client that all the handles have been retrieved.

Figure 13.13 shows a message sequence chart of the air logs captured when doing a Discover All Primary Services procedure with an LE server. A few things may be noted:

- In the first Read By Group Type Request, the client tries to read all handles from 1 to 65535.
- The server responds with three handle value pairs:
  - Handles 1 to 7 (Service 1).
  - Handles 16 to 19 (Service 2).
  - Handles 80 to 82 (Service 3) (This got truncated in the screen shot).
- The client increments the Last Handle received in previous response by 1 and does a second Read By Group Type Request, this time reading from 83 to 65535
- The server responds with two handle value pairs:

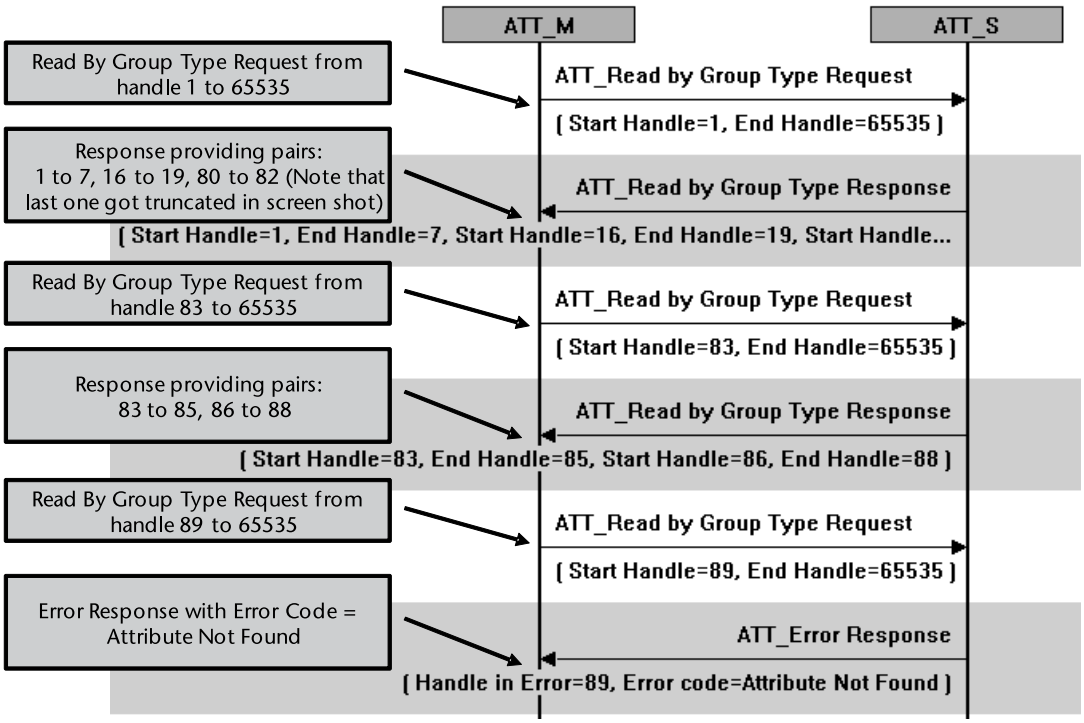

Figure 13.13 MSC generated from air capture of discover all primary services.

  - Handles 83 to 85 (Service 4).
  - Handles 86 to 88. (Service 5).
- The client increments the Last Handle received in previous response by 1 and does a third Read By Group Type Request, this time reading from 89 to 65535.
- The server responds with an Error Response with the Error Code = Attribute Not Found.

This indicates to the client that it has received all handle value pairs. The Discover All Primary Services procedure is considered completed by the client. It was able to retrieve a total of five services from the server.

13.6.2.2 Discover Primary Service By Service UUID

This subprocedure is used by the client to discover a primary service on the server if the Service UUID is known. As shown in Figure 13.10, the Service UUID is a part of the service declaration.

It uses the ATT Find By Type Value Request with the following parameters:

- Attribute Type: UUID for <<Primary Service>>.
- Attribute Value: Service UUID of the Primary service to search (16-bit or 128-bit).
- Starting Handle: 0x0001.
- Ending Handle: 0xFFFF.

The server either sends the ATT Find By Type Value Response or an Error Response.

If the server sends the Find By Type Value Response, then it contains the following parameters:

- List of Attribute Handle Ranges.
  - Starting Handle of service definition.
  - Ending Handle of the service definition.

It is possible that not all attributes are returned in one response. In that case, the End Group Handle in the response would be less than the Ending Handle given in the request. The client can then send another Find By Type Value Request by modifying the Starting Handle to the received Ending Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.14. The client sends the Find By Type Value Request message to the server by setting the Starting Handle to 0x0001, Ending Handle to 0xFFFF, and Attribute Value to the Service UUID to be searched. The server responds with the first set of Handle Information List containing the Starting Handle and Ending Handle of a particular service definition.

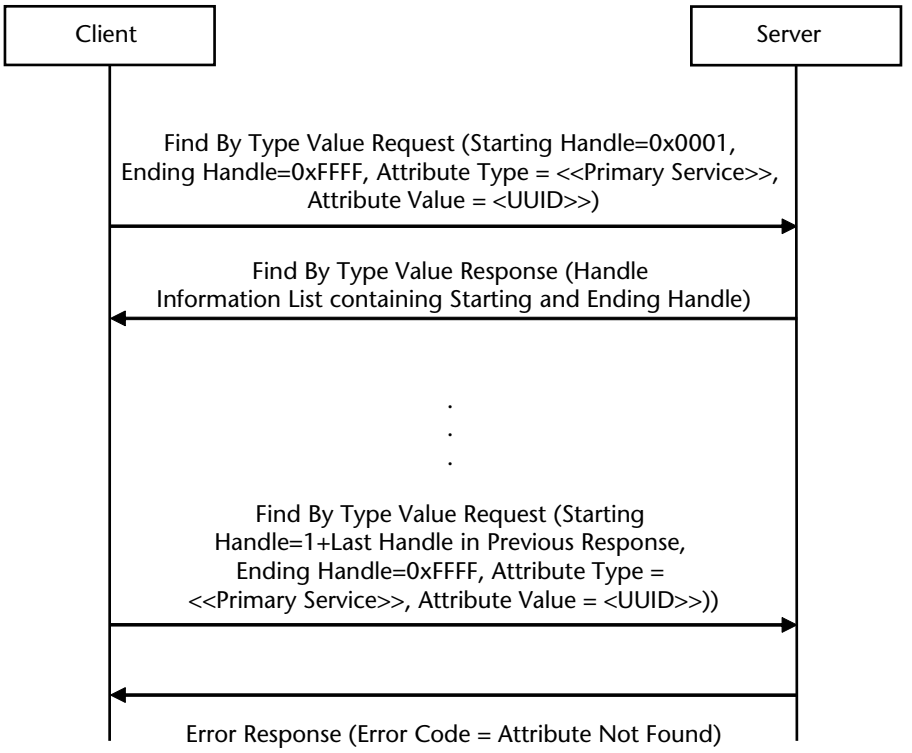

Figure 13.14 Discover primary service by service UUID.

The client then sends the next Find By Type Value Request by adjusting the Starting Handle to retrieve the next set of Handle Information List. The server responds with the next set of Handle Information List until none remain per the Starting Handle specified by the client. At that time the server returns an Error Response. This indicates to the client that all the handles have been retrieved.

13.6.3 Relationship Discovery

The Relationship Discovery procedure is used by the client to discover service relationships of the services. This category contains one subprocedure: Find Included Services.

13.6.3.1 Find Included Services

This subprocedure is used by the client to discover the include service declarations within a service definition on the server. As shown in Figure 13.10, a service definition may contain zero or more include service declarations to specify which other services are included by the service.

It uses the ATT Read By Type Request with the following parameters:

- Attribute Type: UUID for <<Include>>.
- Starting Handle: Starting Handle of the specified service.

- Ending Handle: Ending Handle of the specified service.

The Starting Handle and the Ending Handle specify the values returned in the Primary Service Discovery Procedure. The server either sends the ATT Read By Type Response or an Error Response.

If the server sends the Read By Type Response, then it contains the following parameters:

- Length: Size of the handle-value pairs.
- Attribute Data List: Handle-value pairs containing Attribute Handle and Attribute Value. The Attribute Value contains:
  - Attribute Handle: Handle of the included service declaration.
  - End Group Handle: Handle of last attribute within the included service declaration.
  - UUID: Service UUID.
    - If the Service UUID is 16-bit UUID, then it is also returned in the response.
    - If the UUID is 128-bit, then the ATT Read Request is used later on with the Attribute Handle parameter to retrieve the 128-bit UUID.

Not all handle-value pairs are necessarily returned in one response. In that case, the Attribute Handle in the response would be less than the Ending Handle given in the request. The client can then send another Read By Type Request by modifying the Starting Handle to the last received Attribute Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.15. The client sends the Read By Type Request message to the server by setting the Starting Handle and Ending Handle to the attribute handles that were received in the primary service discovery procedure and the Type set to <<Include>>. The server responds with the first set of Handle-Value pairs of the included services.

The client then sends the next Read By Type Request by adjusting the Starting Handle to retrieve the next set of Handle-Value Pairs. The server responds with the next set of Handle Value pairs until none remain per the Starting Handle specified by the client. At that time the server returns an Error Response. This indicates to the client that all handles for the included services have been retrieved.

Figure 13.16 extends the sample air logs that were shown in Figure 13.13. During Discover All Primary Services, the client found five services on the server. This example shows that the client tries to do a Find Included Services for each of these five services using the Read By Type Request. Each of the requests returns an Error Response. This means that there are no included services on the server.

13.6.4 Characteristic Discovery

The Characteristic Discovery procedure is used by the client to discover the characteristic definitions included in a service on the server. As shown in Figure 13.10, a service may contain zero or more characteristic definitions. This category contains

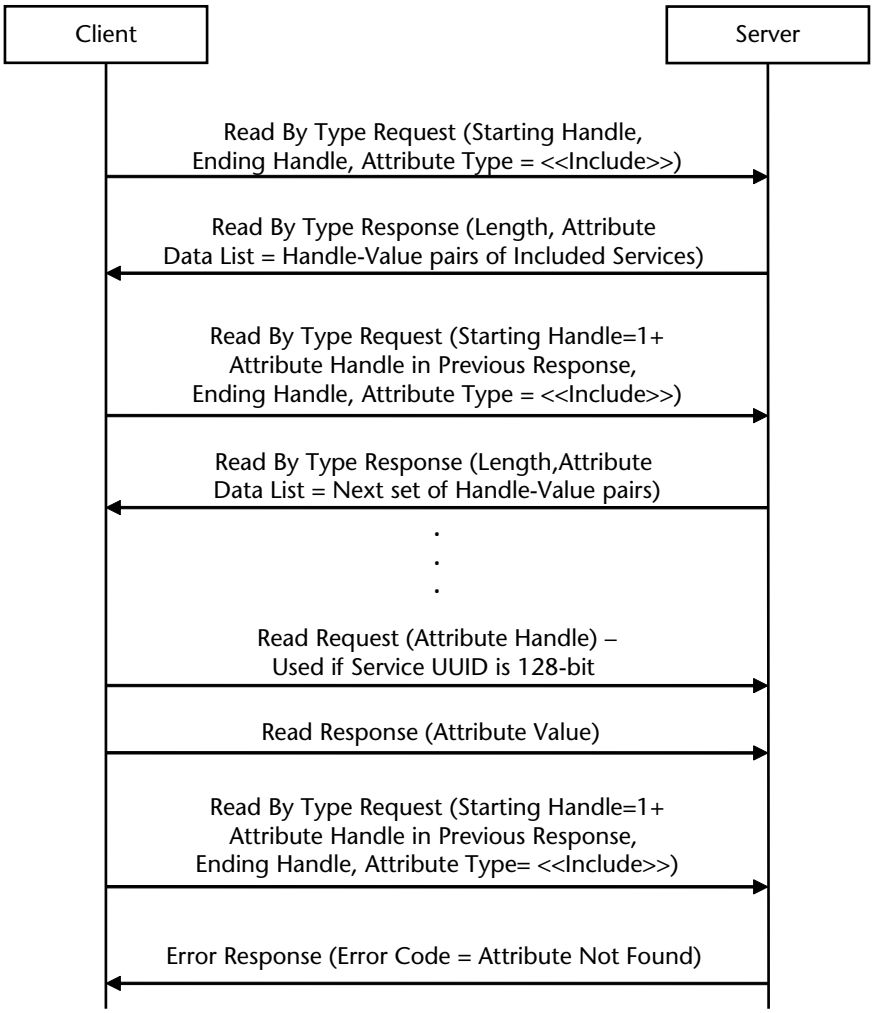

Figure 13.15 Find included services.

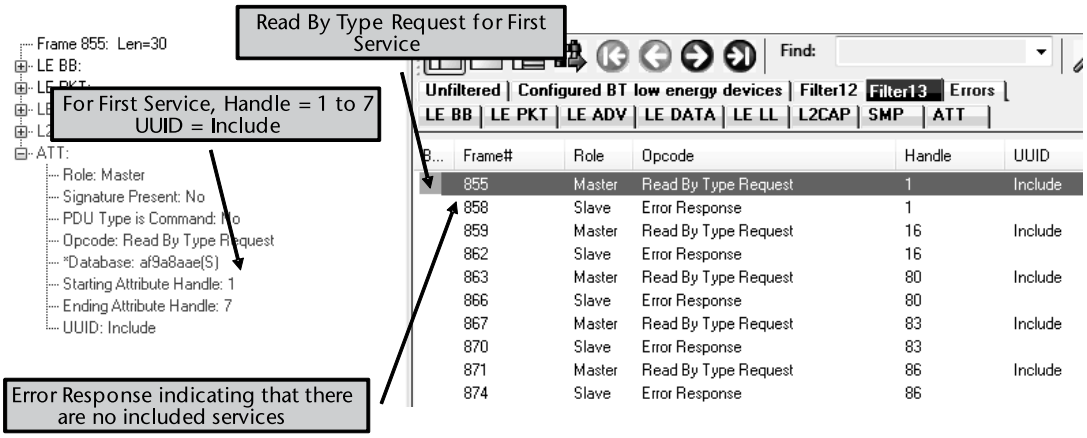

Figure 13.16 Example air log of find included services.

two subprocedures: Discover All Characteristics of a Service and Discover Characteristics by UUID.

13.6.4.1 Discover All Characteristics of a Service

This subprocedure is used by the client to discover all characteristic declarations within a service definition. This is quite similar to the Find Included Services subprocedure. The main difference is that in this case the Attribute Type parameter is set to <<Characteristic>> while in the Find Included Services subprocedure it was set to <<Include>>.

This subprocedure uses the ATT Read By Type Request with the following parameters:

- Attribute Type: UUID for <<Characteristic>>.
- Starting Handle: Starting Handle of the specified service.
- Ending Handle: Ending Handle of the specified service.

The Starting Handle and the Ending Handle specify the values returned in the Primary Service Discovery Procedure. The server either sends the ATT Read By Type Response or an Error Response.

If the server sends the Read By Type Response, then it contains the following parameters:

- Length: Size of the handle-value pairs.
- Attribute Data List: handle-value pairs containing Attribute Handle and Attribute Value.
  - Attribute Handle: Handle of the characteristic declaration.
  - Attribute Value: The Attribute Value contains Characteristic Properties, Characteristic Value Handle, Characteristic UUID.

It is possible that not all handle-value pairs are returned in one response. In that case, the Attribute Handle in the response would be less than the Ending Handle given in the request. The client can then send another Read By Type Request by modifying the Starting Handle to the last received Attribute Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.17. The client sends the Read By Type Request message to the server by setting the Starting Handle and Ending Handle to the attribute handles received in the primary service discovery procedure and the Type set to <<Characteristic>>. The server responds with the first set of Handle-Value pairs of the characteristic declarations in the service definition.

The client then sends the next Read By Type Request by adjusting the Starting Handle to retrieve the next set of Handle-Value Pairs. The server responds with the next set of Handle Value pairs until no more remain as per the Starting Handle specified by the client. At that time the server returns an Error Response. This in-

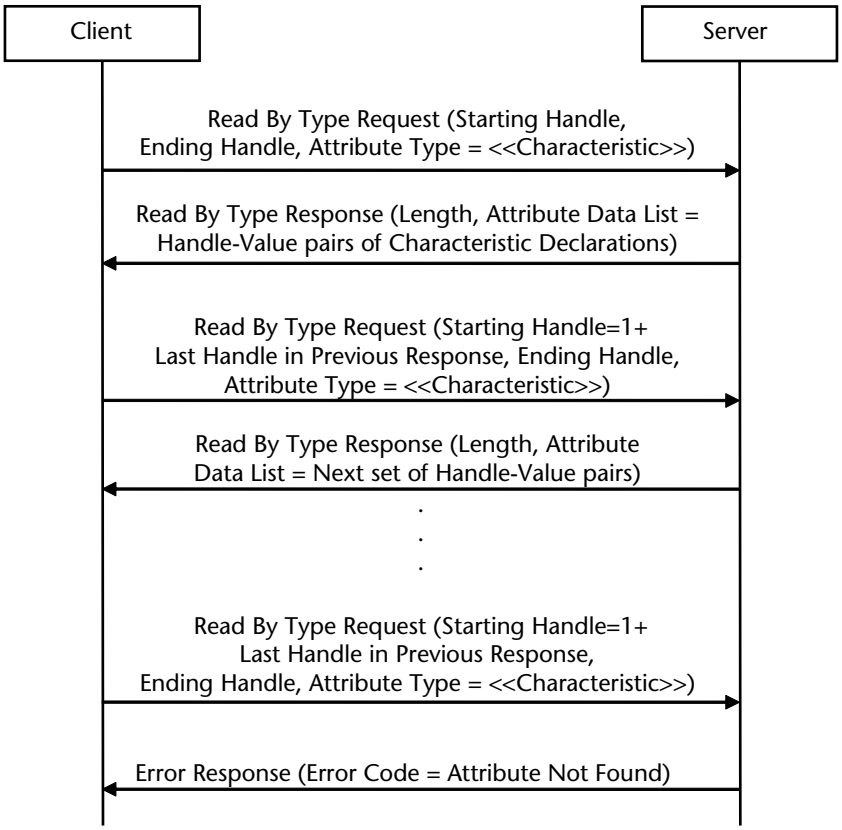

Figure 13.17 Discover all characteristics of a service.

dicates to the client that all handles for the characteristic declarations have been retrieved.

Figure 13.18 extends the sample air logs that were shown in Figure 13.13 and Figure 13.16. During Discover All Primary Services, the client found five services on the server. The first service had the starting handle as 1 and the ending handle as 7. This example shows that the client tries to do a Discover All Characteristics of a Service procedure for the first service.

The right side of the figure shows the following:

- Frame #875: First Read By Type Request by the client with UUID = Characteristic, Handle = 1 to 7.
- Frame #878: Response by the server with the first set of characteristics (Handles 2 and 4).
- Frame #879: Second Read By Type Request by the client with UUID = Characteristic, Handle = 5 to 7.
- Frame #882: Read By Type Response with the second set of characteristics (Handle 6).
- Frame #883: Third Read By Type Request by the client with UUID = Characteristic, Handle = 7.

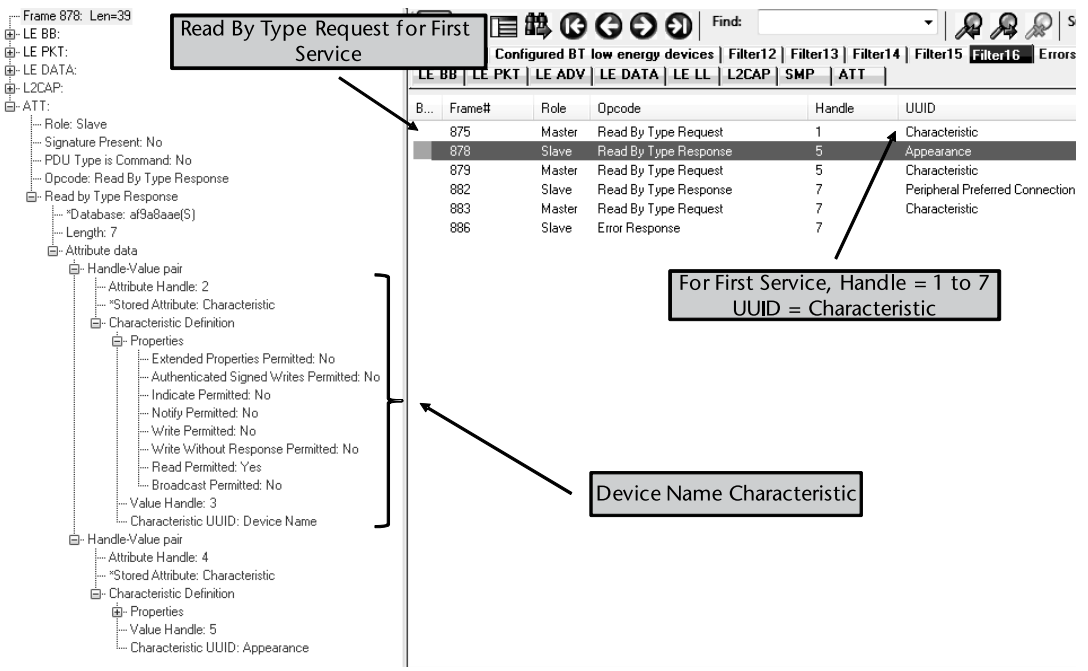

Figure 13.18 Example air log of discover all characteristics of a service.

- Frame #886: Error Response by the server to indicate that there are no more characteristics in this service.

The left side of the figure shows the characteristics received in the first response from the server. The following may be observed:

- The first characteristic is Device Name with Attribute Handle 2.
- The Attribute value is stored at Attribute Handle 3 (Indicated by Value Handle).
- Read operation is permitted on the characteristic. All other operations are not permitted.

13.6.4.2 Discover Characteristics by UUID

This subprocedure is used by the client to discover the characteristic declarations within a service definition when the characteristic UUID is known. This subprocedure is quite similar to the Discover All Characteristics of a Service subprocedure. In this case, once the characteristics are received from the server, they are checked to see if the UUID matches the UUID that was requested. If it matches, then the characteristic is considered to be found. In both the cases, whether the characteristic is found or not, the procedure continues to search the remaining characteristics until all characteristics have been searched.

It uses the ATT Read By Type Request with the following parameters:

- Attribute Type: UUID for <<Characteristic>>.

- Starting Handle: Starting Handle of the specified service.
- Ending Handle: Ending Handle of the specified service.

The Starting Handle and the Ending Handle specify the values that were returned in the Primary Service Discovery Procedure. The server either sends the ATT Read By Type Response or an Error Response.

If the server sends the Read By Type Response, then it contains the following parameters:

- Length: Size of the handle-value pairs.
- Attribute Data List: handle-value pairs containing Attribute Handle and Attribute Value.
  - Attribute Handle: Handle of the characteristic declaration.
  - Attribute Value: The Attribute Value contains Characteristic Properties, Characteristic Value Handle, Characteristic UUID.

The Attribute Value in each of the handle-value pairs is checked to see if it matches the Characteristic UUID that was requested. If it matches, then the characteristic is considered to be found.

It is possible that not all handle-value pairs are returned in one response. In that case, the Attribute Handle in the response would be less than the Ending Handle given in the request. The client can then send another Read By Type Request by modifying the Starting Handle to the last received Attribute Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.19. The client sends the Read By Type Request message to the server by setting the Starting Handle and Ending Handle to the attribute handles received in the primary service discovery procedure and the Type set to <<Characteristic>>. The server responds with the first set of Handle-Value pairs of the characteristic declarations in the service definition. The client checks those handle-value pairs to see if Characteristic UUID matches the one that was provided. If it matches then the characteristic is considered to be found.

The client then sends the next Read By Type Request by adjusting the Starting Handle to retrieve the next set of Handle-Value Pairs. The server responds with the next set of Handle Value pairs till no more remain per the Starting Handle specified by the client. At that time the server returns an Error Response. This indicates to the client that all handles for the characteristic declarations have been retrieved.

13.6.5 Characteristic Descriptor Discovery

The Characteristic Discovery procedure is used by the client to discover the characteristic descriptors of a characteristic. As shown in Figure 13.10, the characteristic descriptor is an optional component of the characteristic definition. This category contains one subprocedure: Discover All Characteristic Descriptors.

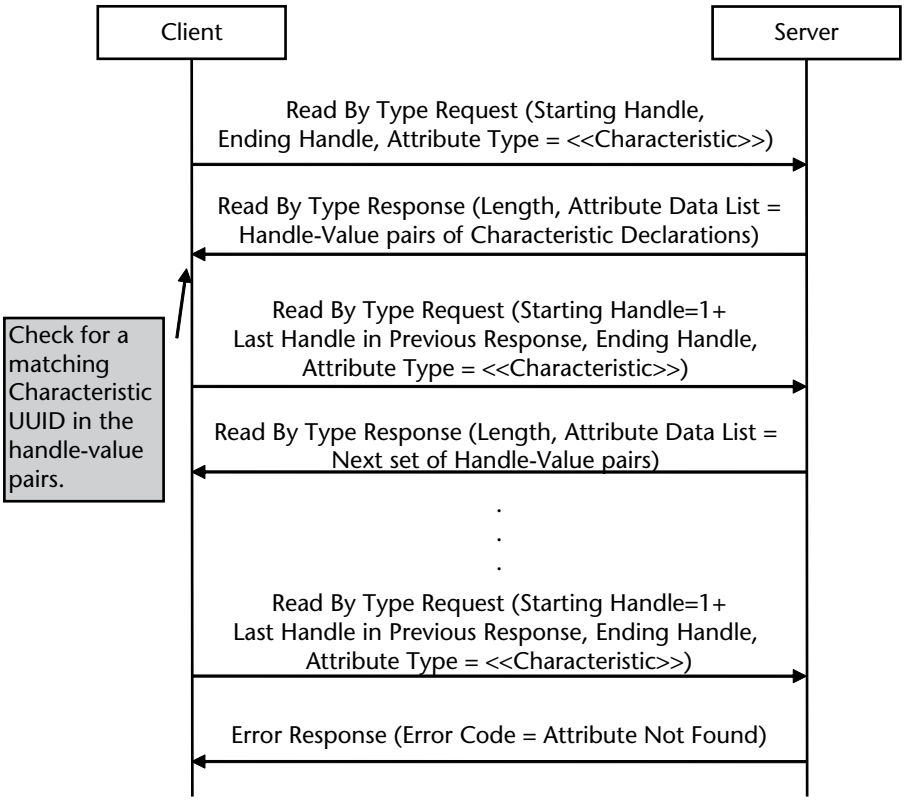

Figure 13.19 Discover characteristics by UUID.

13.6.5.1 Discover All Characteristic Descriptors

This subprocedure is used by the client to discover all the characteristic descriptors within a characteristic definition.

It uses the ATT Find Information Request with the following parameters:

- Starting Handle: Starting Handle of the specified characteristic value + 1.
- Ending Handle: Ending Handle of the specified characteristic.

The server either sends the ATT Find Information Response or an Error Response. If the server sends the Find Information Response, then it contains the following parameters:

- Format: Specifies whether the handles are 16-bit or 128-bit.
- Attribute Data List: handle-value pairs containing Attribute Handle and Attribute Value.
  - Attribute Handle: Handle of the characteristic descriptor declaration.
  - Attribute Value: Characteristic Descriptor UUID.

It is possible that not all handle-value pairs are returned in one response. In that case, the Attribute Handle in the response would be less than the Ending

Handle given in the request. The client can then send another Find Information Request by modifying the Starting Handle to the last received Attribute Handle + 1. This subprocedure is completed when the server returns an Error Response with the error code set to Attribute Not Found.

This is shown in Figure 13.20. The client sends the Find Information Request message to the server by setting the Starting Handle and Ending Handle. The server responds with the first set of Handle-Value pairs of the characteristic descriptors in the characteristic definition.

The client then sends the next Find Information Request by adjusting the Starting Handle to retrieve the next set of Handle-Value Pairs. The server responds with the next set of Handle Value pairs until none remain per the Starting Handle specified by the client. At that time the server returns an Error Response. This indicates to the client that all handles for the characteristic descriptors have been retrieved.

13.6.6 Characteristic Value Read

The Characteristic Value Read procedure is used by the client to read a Characteristic Value from the server. This category contains four sub-procedures:

- Read Characteristic Value.
- Read Long Characteristic Values.
- Read Using Characteristic UUID.
- Read Multiple Characteristic Values.

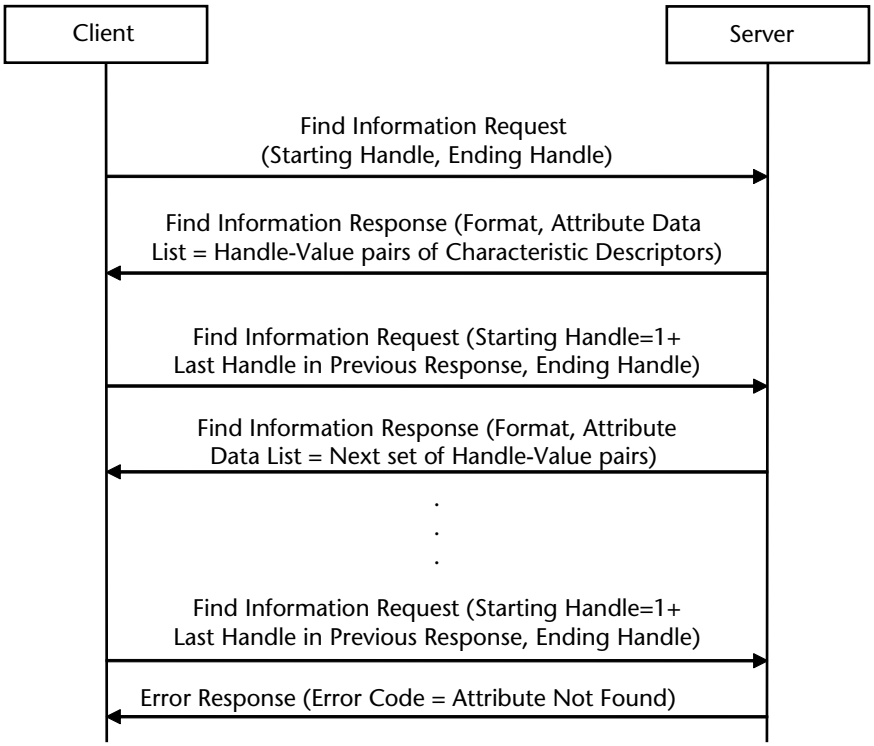

Figure 13.20 Discover all characteristic descriptors.

13.6.6.1 Read Characteristic Value

This subprocedure is used by the client to read a Characteristic Value from the server. It requires a Characteristic Value Handle as an input. It uses the ATT Read Request with the following parameter:

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).

The server either sends the ATT Read Response or an Error Response. The server can send an Error Response if the read operation is not permitted on the Characteristic Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Characteristic Value are set by the profile.

If the server sends the Read Response, then it contains the following parameter:

- Attribute Value: Characteristic Value.

If the length of the characteristic value is more than ATT_MTU-1, then only ATT_MTU-1 bytes are returned and the remaining bytes may be read using the Read Long Characteristic Value procedure. This is shown in Figure 13.21.

13.6.6.2 Read Long Characteristic Value

This subprocedure is very similar to the Read Characteristic Value Procedure explained in previous section. It is used when the length of the Characteristic Value is longer than that can be sent in a single Read Response message.

It uses the ATT Read Blob Request with the following parameters:

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).
- Value Offset: Offset from where the Attribute Value is to be read.

The server either sends the ATT Read Blob Response or an Error Response.

The server can send an Error Response if the read operation is not permitted on the Characteristic Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Characteristic

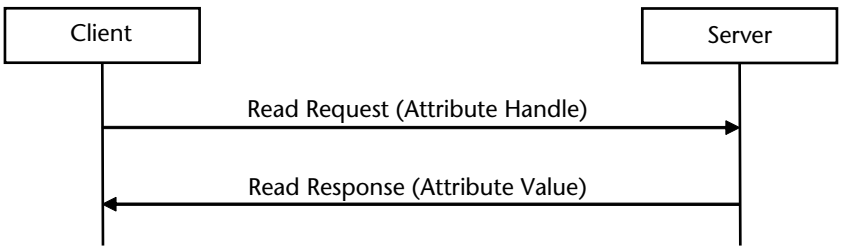

Figure 13.21 Read characteristic value.

Value are set by the profile. An error is also returned if the Characteristic Value is shorter than ATT_MTU-1 bytes.

If the server sends the Read Blob Response, then it contains the following parameter:

- Part Attribute Value: Characteristic Value bytes starting at the specified Value Offset.

It is possible to read the first part of the Attribute Value using Read Request and the remaining parts using Read Blob Request. This subprocedure is shown in Figure 13.22.

13.6.6.3 Read Using Characteristic UUID

This subprocedure is used by the client to read a Characteristic Value from the server when it knows the Characteristic UUID but does not know the Characteristic Value Handle.

It uses the ATT Read By Type Request with the following parameters:

- Starting Handle: Starting Handle.
- Ending Handle: Ending Handle.
- Attribute Type: Characteristic UUID.

The Starting Handle and Ending Handle are typically the handle range of the service in which the characteristic is located. The server either sends the ATT Read By Type Response or an Error Response.

If the server sends the Read By Type Response, then it contains the following parameters:

- Attribute Handle: Characteristic Value Handle.
- Attribute Value: Characteristic Value.

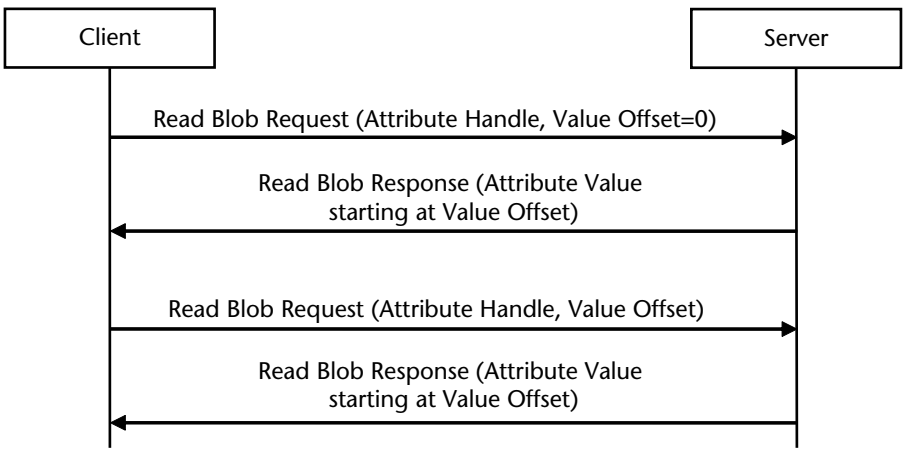

Figure 13.22 Read long characteristic value.

This is shown in Figure 13.23.

13.6.6.4 Read Multiple Characteristic Values

This subprocedure is used by the client to read multiple Characteristic Values from the server when it knows the Characteristic Value Handles. This is very similar to the Read Characteristic Value subprocedure.

It uses the ATT Read Multiple Request with the following parameters:

- Set Of Handles: List of Characteristic Value Handles for which the Values are to be retrieved.

The server either sends the ATT Read Multiple Response or an Error Response. The server can send an Error Response if the read operation is not permitted on any of the Characteristic Value or if the authentication, authorization or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Characteristic Value are set by the higher layer profile or application.

If the server sends the Read Multiple Response, then it contains the following parameters:

- Set Of Value: List of Characteristic Values.

This is shown in Figure 13.24.

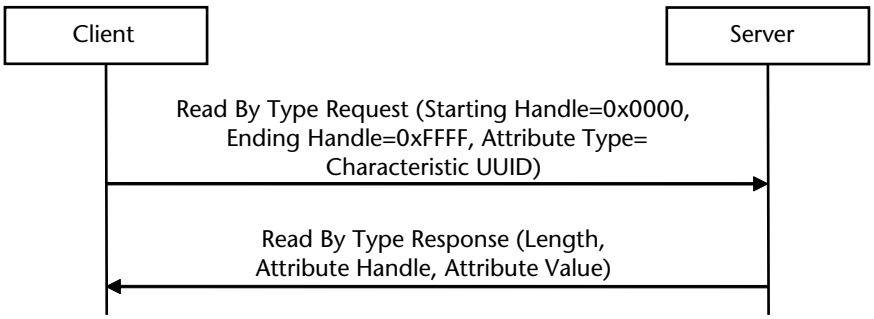

Figure 13.23 Read using characteristic UUID.

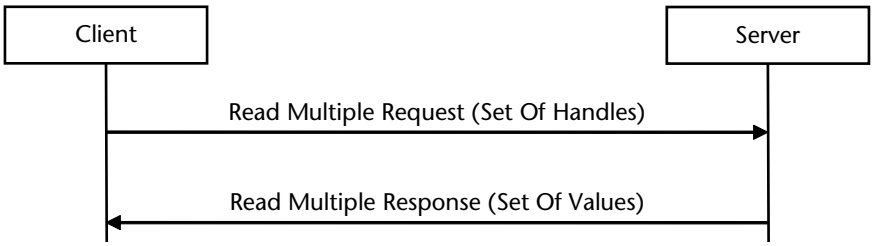

Figure 13.24 Read multiple characteristic values.

13.6.7 Characteristic Value Write

The Characteristic Value Write procedure is used by the client to write a Characteristic Value to the server.This category contains five sub-procedures:

- Write Without Response.
- Signed Write Without Response.
- Write Characteristic Value.
- Write Long Characteristic Value.
- Reliable Writes.

13.6.7.1 Write Without Response

This subprocedure is used by the client to write a Characteristic Value to the server when it does not need an acknowledgment that the write has been successful. This is the simplest procedure that a client can use to write a Characteristic Value to the server.

It uses the ATT Write Command with the following parameters:

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).
- Attribute Value: Characteristic Value to be written.

Since this is an ATT command, there is no response from the server. (Note that the server responds to ATT requests but not ATT commands). This is shown in Figure 13.25.

13.6.7.2 Signed Write Without Response

This subprocedure is used by the client to write a Characteristic Value to the server when it does not need an acknowledgment that the write has been successful. It is similar to the Write Without Response procedure. The main enhancement is that the Attribute Value is signed by the client with an authentication signature. This authentication signature is verified by the server before writing the value. It uses the ATT Write Command with the following parameters:

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).

Figure 13.25 Write without response.

- Attribute Value: Characteristic Value that is authenticated by signing the value.

Since this is an ATT command, there is no response from the server. (Note that the server responds to ATT requests but not ATT commands).

This subprocedure requires the client and server to share a bond (defined in GAP profile). The server verifies the authentication signature to check the authenticity of the client before writing the value. This is shown in Figure 13.26.

13.6.7.3 Write Characteristic Value

This subprocedure is used by the client to write a Characteristic Value to the server. In this subprocedure the client receives an acknowledgment from the server once the value is written. It uses the ATT Write Request with the following parameters:

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).
- Attribute Value: Characteristic Value to be written.

The server either sends the ATT Write Response or an Error Response.

The server can send an Error Response if the write operation is not permitted on the Characteristic Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Characteristic Value are set by the higher layer profile or application. If the server sends the Write Response, then it indicates that the value has been successfully written. This is shown in Figure 13.27.

13.6.7.4 Write Long Characteristic Value

This subprocedure is used by the client to write a Characteristic Value to the server when the length of the Characteristic Value is longer than that can be written in a single Write Request. It splits the Characteristic Value into parts and uses multiple ATT Prepare Write Requests to queue the Characteristic Value parts on the server. Then it uses the Execute Write Request to request the server to write all the queued parts.

It uses the ATT Prepare Write Request with the following parameters to queue the Characteristic Value to write after splitting the Characteristic Value into several parts.

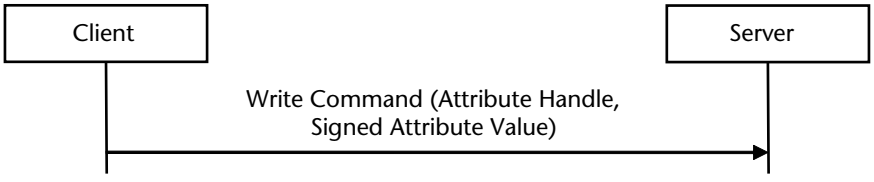

Figure 13.26 Signed write without response.

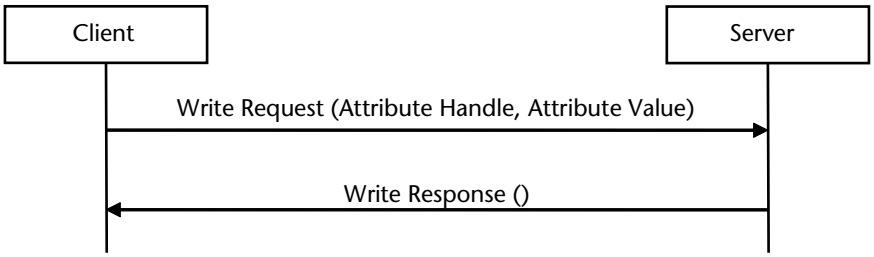

Figure 13.27 Write characteristic value.

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).
- Value Offset: The offset of the first octet to be written.
- Part Attribute Value: The octets of the Characteristic Value starting at Value Offset.

Once all the parts are queued, the subprocedure uses the ATT Execute Write Request with the following parameters to write all the queued parts.

- Flags = 0x01 to write all the queued parts.

The server can send an Error Response if the write operation is not permitted on the Characteristic Value or if the authentication, authorization, or encryption size is insufficient.

This is shown in Figure 13.28. The client splits the Characteristic Value into several parts as per the ATT_MTU and then sends the subsequent parts in the Prepare Write Request after adjusting the Value Offset. The server queues the various parts that it receives. After sending all the parts, the client finally sends the Execute Write Request to write all the queued parts.

One point to note is that, even though the Prepare Write Response from the server contains the same parameters sent in the Prepare Write Request, it is not mandatory for the client to verify that the server has sent back the same values. There is a possibility that some values may have been corrupted during sending from the client to the server (the client does not cross check the received values to see if those were corrupted). The next section describes the Reliable Writes subprocedure where it is mandatory for the client to check if the server returned the same values in response that it had sent in request.

13.6.7.5 Reliable Writes

This subprocedure is used by the client to write a Characteristic Value to the server when it requires assurance that the correct Characteristic Value is going to be written by the server. It is also used when multiple values are to be written together in one single operation. For example, this procedure is used to ensure that a set of values is written by a client together in a particular order without any other values being written by another client in between.

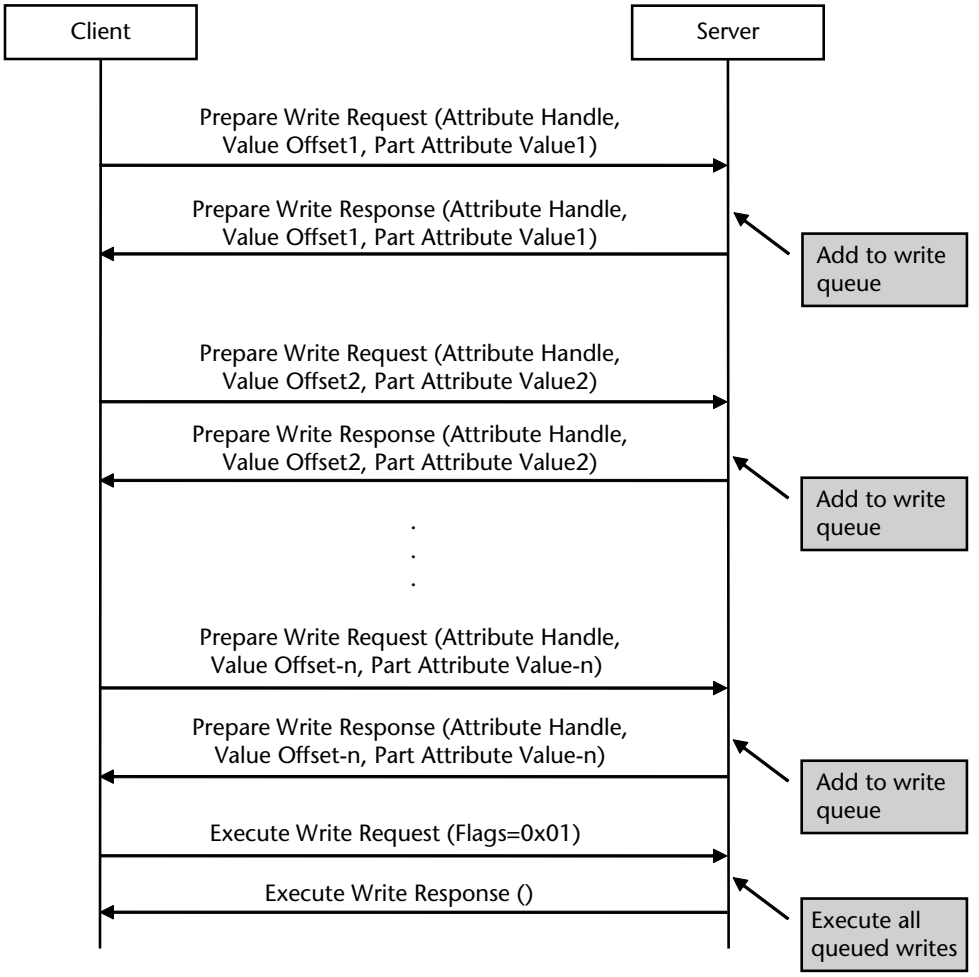

Figure 13.28 Write long characteristic value.

This procedure requires the server to send back the same Characteristic Value which the client had sent to it so that the client can verify that the server had received the value correctly. It uses the ATT Prepare Write Request with the following parameters to queue the Characteristic Value to write after splitting the Characteristic Value into several parts.

- Attribute Handle: Characteristic Value Handle (This handle was retrieved in the Characteristic Discovery Procedure).
- Value Offset: The offset of the first octet to be written.
- Part Attribute Value: The octets of the Characteristic Value starting at Value Offset.

The subsequent ATT Prepare Write Request messages may also send different Attribute Handles if several Characteristic Values are to be written together. It then

uses the ATT Execute Write Request with the following parameters to write all the queued parts.

- Flags = 0x01 to write all the queued parts.

The server can send an Error Response if the write operation is not permitted on any of the Characteristic Values or if the authentication, authorization, or encryption size is insufficient. The server can also send an Error Response if the queue is full. This procedure is shown in Figure 13.29. The client sends the first Characteristic Value to be written in the Prepare Write Request. The server queues the value and also returns the same value in Prepare Write Response. The client verifies the received value with the value it sent to check if the server indeed received the value correctly. After that the client may either send other parts of the same Characteristic Value or another Characteristic Value in the subsequent Prepare Write Requests. Once it has sent all the Characteristic Values, it finally sends the Execute Write Request. On receipt of the Execute Write Request, the

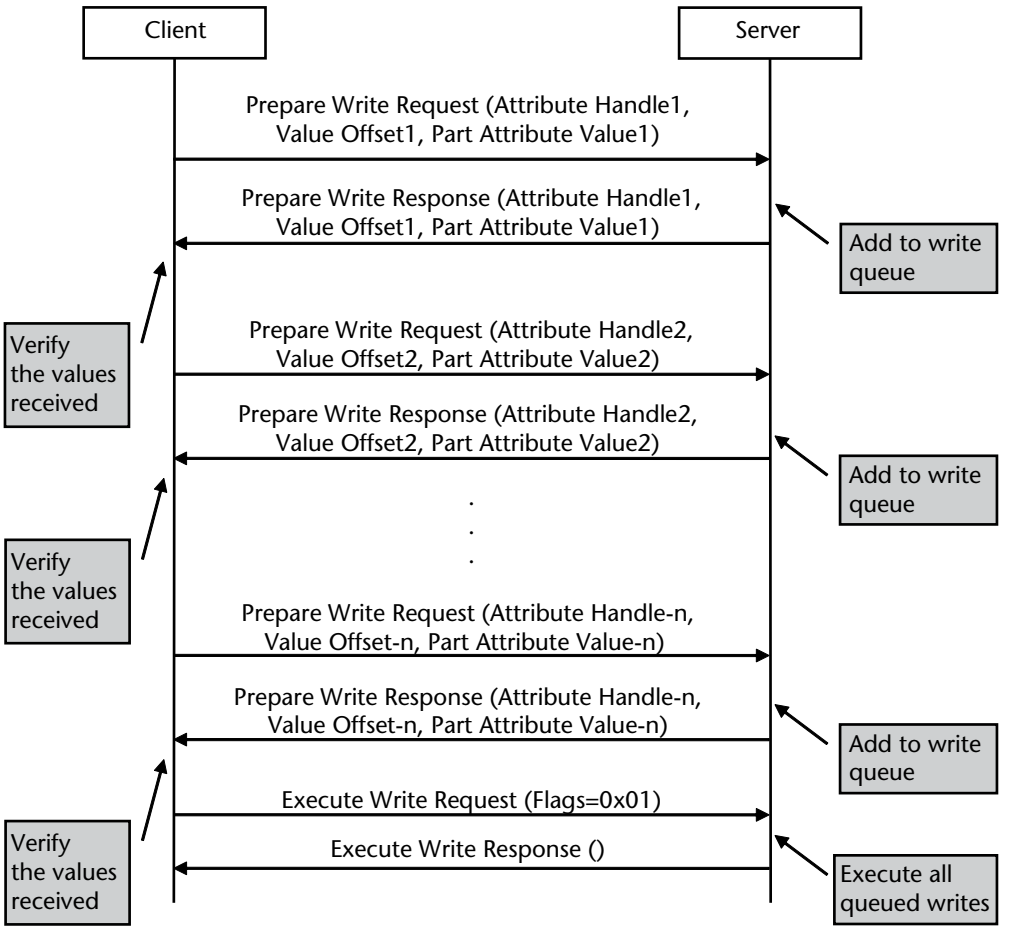

Figure 13.29 Reliable writes.

server writes all the values that it had queued up in the same order in which it had received the requests.

13.6.8 Characteristic Value Notification

The Characteristic Value Notification procedure is used by the server to notify a Characteristic Value to the client. Whether a Characteristic Value can be notified is defined by the profile. This category contains one subprocedure: Notifications.

13.6.8.1 Notifications

This subprocedure is used by the server to notify a Characteristic Value to the client without expecting any acknowledgment from the client. It uses the ATT Handle Value Notification with the following parameters:

- Attribute Handle: Characteristic Value Handle that is to be notified.
- Attribute Value: Characteristic Value.

Since this is an ATT notification, there is no response from the client. (Note that the client responds to ATT indications but not ATT notifications). This is shown in Figure 13.30.

13.6.9 Characteristic Value Indication

The Characteristic Value Indication procedure is used by the server to indicate a Characteristic Value to the client. Whether a Characteristic Value can be indicated is defined by the profile. This category contains one sub-procedure: Indications.

13.6.9.1 Indications

This subprocedure is used by the server to indicate a Characteristic Value to the client. The client responds with a confirmation. It uses the ATT Handle Value Indication with the following parameters:

- Attribute Handle: Characteristic Value Handle that is to be indicated.
- Attribute Value: Characteristic Value.

The client responds with a Handle Value Confirmation to confirm that it has received the indication. This is shown in Figure 13.31.

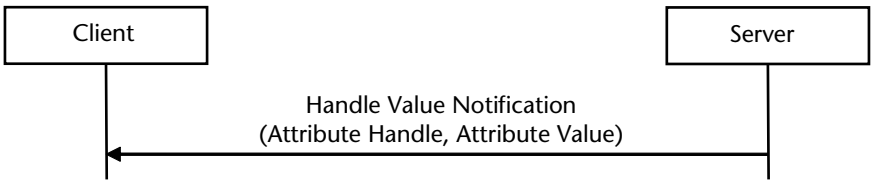

Figure 13.30 Notifications.

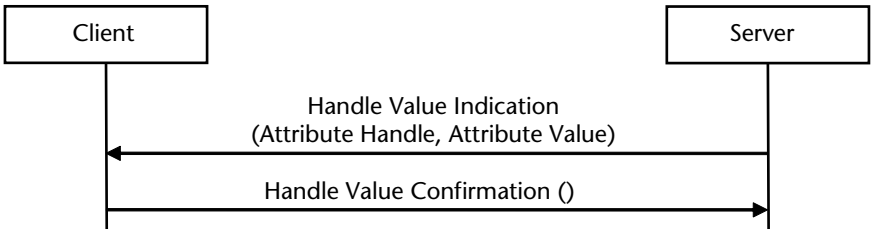

Figure 13.31 Indications.

13.6.10 Characteristic Descriptors

The Characteristic Descriptors procedure is used by the client to read and write the Characteristic Descriptors on the server. All these subprocedures require the handle of the Characteristic Descriptor Declaration. This handle may be retrieved using the Characteristic Descriptor Discovery Procedure which was described earlier.

This category contains four subprocedures:

- Read Characteristic Descriptors.
- Read Long Characteristic Descriptors.
- Write Characteristic Descriptors.
- Write Long Characteristic Descriptors.

13.6.10.1 Read Characteristic Descriptors

This subprocedure is used by the client to read a Characteristic Descriptor from the server. The client needs to have the Attribute Handle of the Characteristic Descriptor Declaration to use this subprocedure. It uses the ATT Read Request with the following parameters:

- Attribute Handle: Characteristic Descriptor Handle. (This handle was retrieved in the Characteristic Descriptor Discovery Procedure.)

The server either sends the ATT Read Response or an Error Response. The server can send an Error Response if the read operation is not permitted on the Attribute Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Attribute Value are set by the higher layer profile or application.

If the server sends the Read Response, then it contains the following parameter:

- Attribute Value: Profile specific value that is stored in the Characteristic Descriptor Declaration.

If the length of the characteristic value is more than ATT_MTU-1, then only ATT_MTU-1 bytes are returned and the remaining bytes may be read using the Read Long Characteristic Descriptor procedure. This is shown in Figure 13.32.

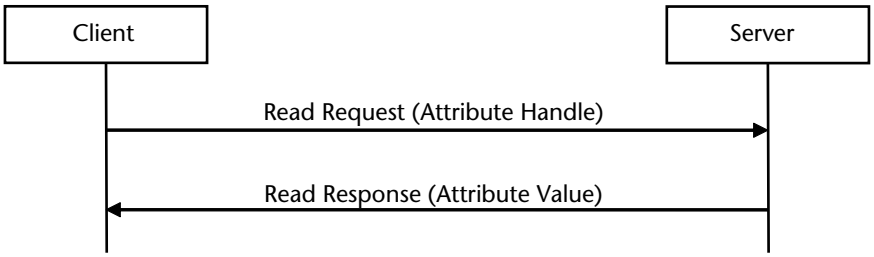

Figure 13.32 Read characteristic descriptor.

13.6.10.2 Read Long Characteristic Descriptor

This subprocedure is very similar to the Read Characteristic Descriptor Procedure explained in previous section. It is used when the length of the Attribute Value is longer than that can be sent in a single Read Response message. It uses the ATT Read Blob Request with the following parameters:

- Attribute Handle: Characteristic Descriptor Handle (This handle was retrieved in the Characteristic Descriptor Discovery Procedure.)
- Value Offset: Offset from where the Attribute Value is to be read.

The server either sends the ATT Read Blob Response or an Error Response. The server can send an Error Response if the read operation is not permitted on the Attribute Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Attribute Value are set by the higher layer profile or application. An error is also returned if the Attribute Value is shorter than ATT_MTU-1 bytes.

If the server sends the Read Blob Response, then it contains the following parameter:

- Part Attribute Value: Attribute Value byte starting at the specified Value Offset.

It is possible to read the first part of the Attribute Value using Read Request and the remaining parts using Read Blob Request. This is shown in Figure 13.33.

13.6.10.3 Write Characteristic Descriptors

This subprocedure is used by the client to write an Attribute Value into a Characteristic Descriptor on the server. In this subprocedure the client receives an acknowledgment from the server once the value is written.

It uses the ATT Write Request with the following parameters:

- Attribute Handle: Characteristic Descriptor Handle (this handle was retrieved in the Characteristic Descriptor Discovery Procedure).
- Attribute Value: Attribute Value to be written into the Characteristic Descriptor.

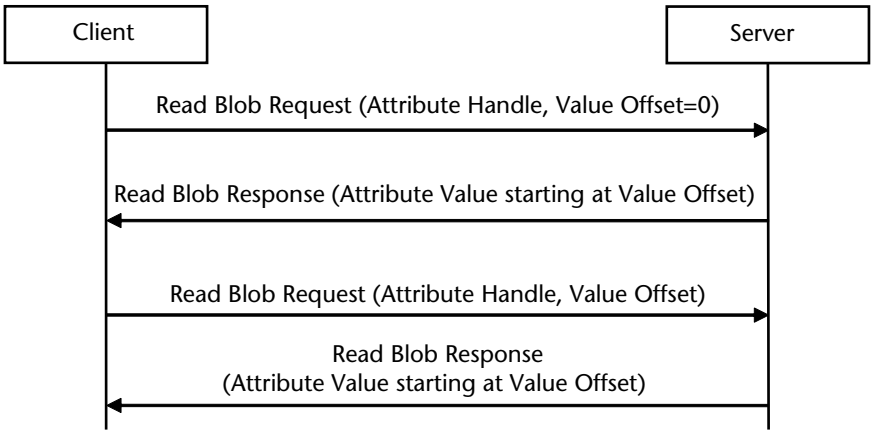

Figure 13.33 Read long characteristic descriptor.

The server either sends the ATT Write Response or an Error Response. The server can send an Error Response if the write operation is not permitted on the Attribute Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Attribute Value are set by the profile. An Error Response may also be returned if the value being written is invalid or of the incorrect size.

If the server sends the Write Response, then it indicates that the value has been successfully written.

This is shown in Figure 13.34.

13.6.10.4 Write Long Characteristic Descriptors

This subprocedure is used by the client to write an Attribute Value into a Characteristic Descriptor on the server when the length of the Attribute Value is longer than that can be written in a single Write Request. It splits the Attribute Value into parts and uses multiple ATT Prepare Write Requests to queue the Attribute Value parts on the server. Then it uses the Execute Write Request to request the server to write all the queued parts. It uses the ATT Prepare Write Request with the following parameters to queue the Attribute Value to write after splitting the Attribute Value into several parts:

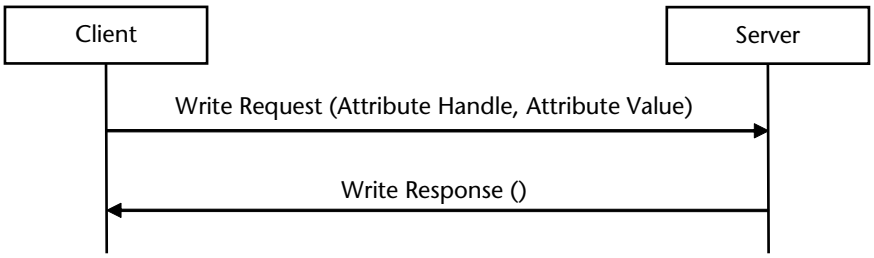

Figure 13.34 Write characteristic descriptor.

- Attribute Handle: Characteristic Descriptor Handle (this handle was retrieved in the Characteristic Descriptor Discovery Procedure).
- Value Offset: The offset of the first octet to be written.
- Part Attribute Value: The octets of the Attribute Value starting at Value Offset.

It then uses the ATT Execute Write Request with the following parameters to write all the queued parts.

- Flags = 0x01 to write all the queued parts.

The server can send an Error Response if the write operation is not permitted on the Attribute Value or if the authentication, authorization, or encryption size is insufficient. Note from Figure 13.10 that the permissions for the Attribute Value are set by the higher layer profile or application. An Error Response may also be returned if the value being written is invalid or of the incorrect size.

This is shown in Figure 13.35. The client splits the Attribute Value into several parts as per the ATT_MTU and then sends the subsequent parts in the Prepare Write Request after adjusting the Value Offset. The server queues the various parts that it receives. After sending all the parts, the client finally sends the Execute Write Request to write all the queued parts.

One point to note is that, even though the Prepare Write Response from the server contains the same parameters that were sent in the Prepare Write Request, it is not mandatory for the client to verify if the server has sent back the same values.

13.7 Timeouts

All GATT procedures use a timeout mechanism to ensure that the procedure does not wait infinitely for a response from the remote side. If a timeout happens, it is assumed that the link has gone down and no further GATT procedures are performed. Further GATT procedures are performed only after a new ATT bearer has been established.

13.8 GATT Service

A server may expose a GATT service in addition to the profile specific services. If the GATT service is exposed by the server then it is exposed as a primary service with service UUID set to <<Generic Attribute Profile>>. GATT specification defines only one characteristic that can be contained in this service:

- Service Changed Characteristic.

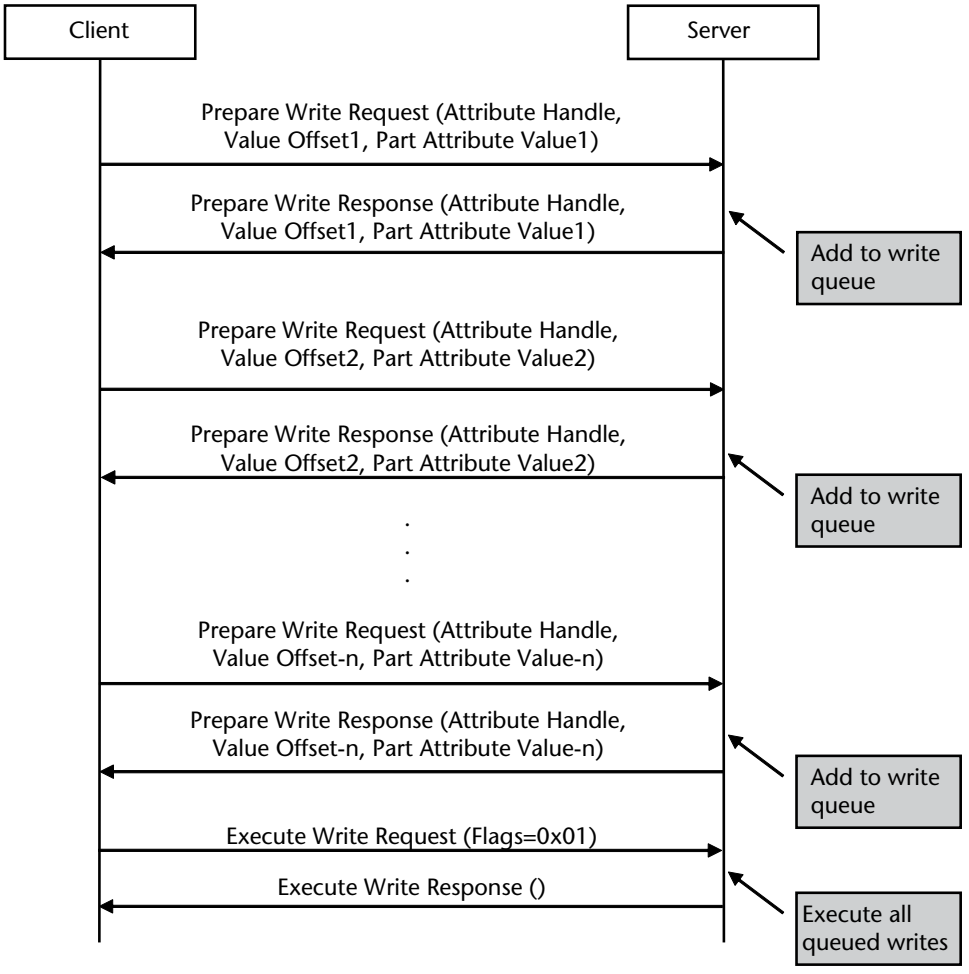

Figure 13.35 Write long characteristic descriptor.

13.8.1 Service Changed Characteristic

The service changed characteristic is used by the server to indicate to the clients that the services on the server have changed. This can happen when new services are added, services are removed, or modified on the server. This characteristic is implemented by the server only if the list of services supported by the device can change over the lifetime of a device. Otherwise this characteristic is not implemented. The list of services supported by a device can change, for example, after a firmware upgrade.

The clients that are bonded to the server are informed about the changed services when they reconnect to the server. This was shown in Figure 13.9. The definition of the service changed characteristic is shown in Figure 13.36.

The Characteristic Value Declaration contains two handles 0xAAAA and 0xBBBB. These indicate the starting and ending handle of the attribute handles that have changed.

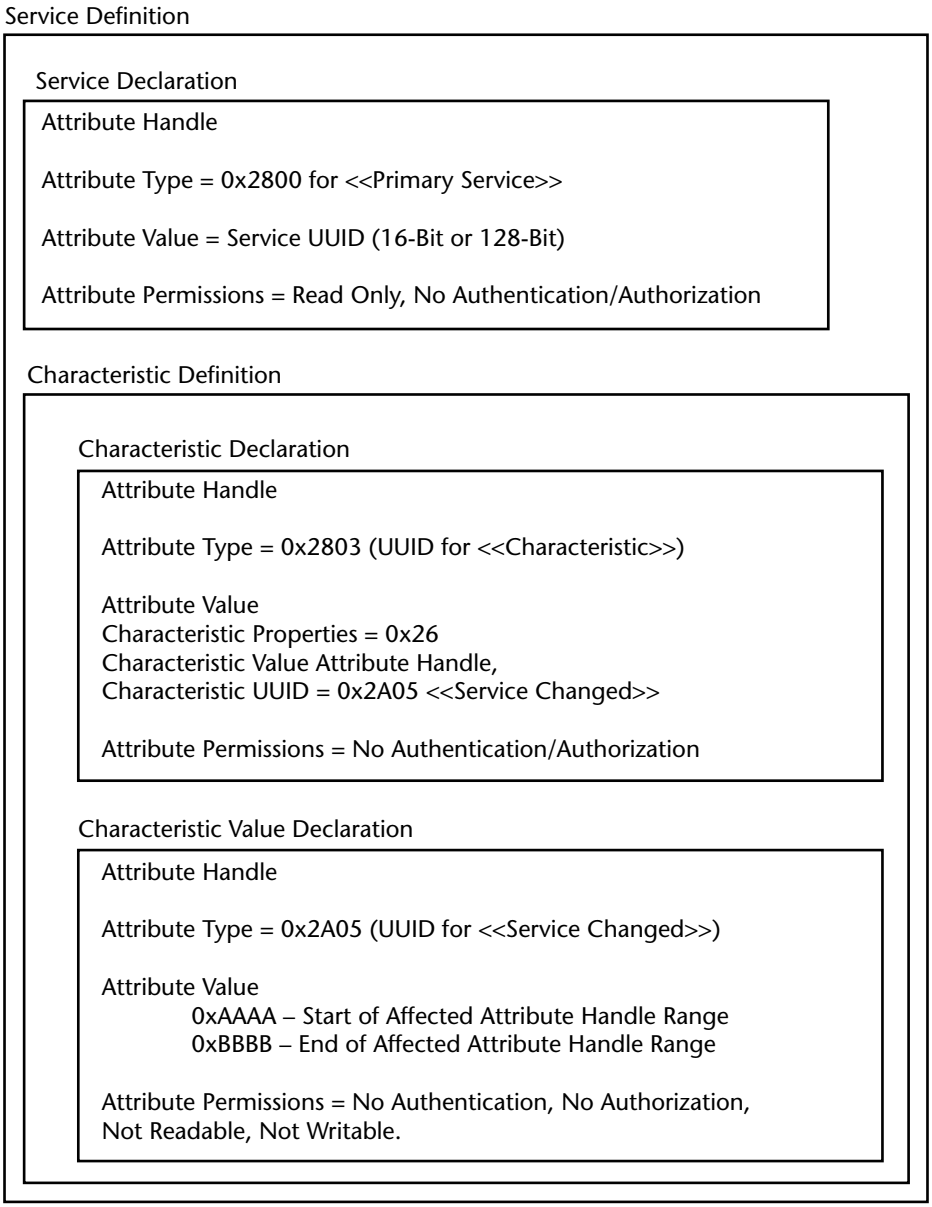

Figure 13.36 Service changed service.

The <<Service Changed>> characteristic is implemented as a control-point attribute. It is configured to be indicated. The Characteristic Properties field within the Characteristic Declaration is set to 0x26. This is a bit mask which has the following bits set:

- 0x20 – Indicate: This characteristic value can be indicated.
- 0x02 – Read: This value can be read.
- 0x04 – Write Without Response: This value can be written using the Write Without Response procedure.