Table 9.1 (continued)

Group	*Commands*	*Events*
Connection State	HCI_LE_Connection_Update	LE_Connection_Update_Complete_Event
	HCI_LE_Remote_Connection_Parameter_Request_Reply	*LE_Remote_Connection_Parameter_Request_Event*
	HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply	
	LE_Set_Data_Length	*LE_Data_Length_Change_Event*
	LE_Read_Suggested_Default_Data_Length_Command	
	LE_Write_Suggested_Default_Data_Length_Command	
Physical Links	HCI_LE_Set_Host_Channel_Classification	
Link Information	HCI_Read_Transmit_Power_Level	
	HCI_Read_RSSI	
	HCI_LE_Read_Advertising_Channel_Tx_Power	
	HCI_LE_Read_Channel_Map	
Authentication and Encryption	HCI_LE_Encrypt	Encryption_Change_Event
	HCI_LE_Long_Term_Key_Request_Reply	Encryption_Key_Refresh_Complete_Event
	HCI_LE_Long_Term_Key_Request_Negative_Reply	LE_Long_Term_Key_Requested_Event
	HCI_LE_Rand	
	HCI_LE_Start_Encryption	
	HCI_Write_Authenticated_Payload_Timeout	*Authenticated_Payload_Timeout_Expired_Event*
	HCI_Read_Authenticated_Payload_Timeout	
	HCI_LE_Read_Local_P-256_Public_Key	*LE_Read_Local_P-256_Public_Key_Complete_Event*
	LE_Generate_DHKey	*LE_Generate_DHKey_Complete_Event*
Testing	HCI_LE_Receiver_Test	
	HCI_LE_Transmitter_Test	
	HCI_LE_Test_End	

This section explains the various LE commands very briefly. The Bluetooth specification may be referred for a detailed explanation including the parameters of each command and event and the significance of each parameter.

Some of the HCI commands and events are explained in further details below.

9.2.1 Device Setup

The device setup commands are used to initialize the controller and put it in a known state. Generally these are among the first commands sent to the controller.

There is only one command in this group for LE:

- HCI_Reset: This command is used to reset the controller. It also resets the link layer and puts it in the standby state with default values for all parameters for which default values are defined. Once the reset of the controller is complete, it sends back a Command_Complete_Event event back to the host.

9.2.2 Controller Flow Control

The controller flow control commands and events are used to control the data flow from the host to the controller. The buffers in the controller can either be separate for BR/EDR and LE or combined.

- If the buffers are separate, then the HCI_LE_Read_Buffer_Size command returns the number of LE buffers and HCI_Read_Buffer_Size returns the number of BR/EDR buffers.
- If the buffers are shared, then HCI_LE_Read_Buffer_Size returns 0 as the length of ACL data packets and HCI_Read_Buffer_Size is used to read the number of shared buffers.

Some of the commands and events in this group are:

- HCI_Read_Buffer_Size: This command is used at the time of initialization to find out the number of ACL data packet buffers in the controller, size of each buffer etc. The host can send as many data packets to the controller as the number of buffers reported by the controller. Then it has to wait for a Number_Of_Completed_Packets_Event to find out how many packets were processed. This gives an indication of how many buffers in the controller got freed up. After that it can send those many more ACL data packets.
- HCI_LE_Read_Buffer_Size: This command is used to find out the total number of LE ACL data buffers in the controller and the maximum size of each buffer.
- Number_Of_Completed_Packets_Event: This event is sent by the controller to indicate the number of packets that it has processed (either transmitted or flushed). This indicates to the host that those buffers have been emptied and the host can send more data to the controller. This provides a flow control mechanism between the host and the controller.

The Host to Controller Data Flow Control was explained in Chapter 3 for BR/EDR. The Data Flow Control for LE is also similar. This is shown in Figure 9.5.

During initialization the host sends the HCI_LE_Read_Buffer_Size command to the controller to get information about the LE buffers in the controller. The controller returns the following parameters:

1. Size of each LE ACL buffer (HC_LE_ACL_Data_Packet_Length).
2. Number of LE ACL buffers (HC_Total_Num_LE_ACL_Data_Packets).

If the controller returns the first parameter as 0, then it means that the controller is using shared ACL buffers for BR/EDR and LE. Subsequently, the HCI_Read_Buffer_Size command can be used to find the length and number of these shared buffers. Once the host has information on the number of ACL buffers, it knows

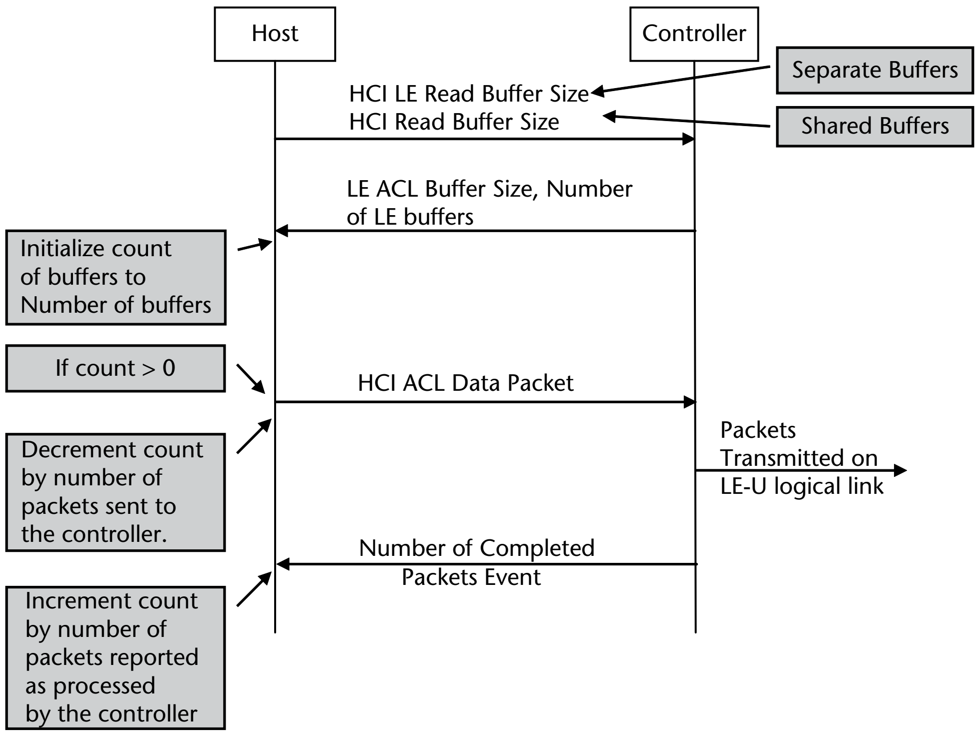

Host
Controller
Separate Buffers
HCI LE Read Buffer Size
HCI Read Buffer Size
Shared Buffers
LE ACL Buffer Size, Number of LE buffers
Initialize count of buffers to Number of buffers
If count > 0
HCI ACL Data Packet
Decrement count by number of packets sent to the controller.
Packets Transmitted on LE-U logical link
Number of Completed Packets Event
Increment count by number of packets reported as processed by the controller

Figure 9.5 Host to controller data flow control.

that at any given time, it can have a maximum of that many packets outstanding (to be processed) on the controller side. For example if the Number of ACL buffers was four, then the host can have a maximum of four packets outstanding on the controller side.

The host maintains a count of the number of ACL buffers available in the controller. It initializes this value to the number of ACL buffers it received in the response to Read Buffer Size command. After that, every time it sends a packet to the controller, it decreases this count by one. Once the controller has completed processing one or more packets, it sends that count in the Number of Completed Packets event. The host knows that some additional buffers have been freed up on the controller side and it increases its count by the number of packets reported in the Number of Completed Packets event.

9.2.3 Host Flow Control

The host flow control commands and events allow the flow control to be used towards the host. In most implementations host flow control is not used. This may still be used if the host has a slow processor or limited memory space.

The following commands are included in this category:

- Host_Buffer_Size_Command: This command is used by the host to provide the controller information about the data buffers present in the host.

- Set_Controller_To_Host_Flow_Control_Command: This command is used to turn the flow control on and off for the data flowing from the controller to the host.
- Host_Number_Of_Completed_Packets_Command: This command is used by the host to indicate to the controller the number of packets that it has processed. The controller can use this information to send additional packets to the host.
- Data_Buffer_Overflow_Event: This event is used by the controller to indicate that its data buffers have overflowed because the host has sent more packets than the number of free buffers in the controller.
- Set_Event_Mask: This command is used to configure which events are generated by the controller for the host.

The following commands for the white list are also included in this category:

- HCI_LE_Read_White_List_Size: This command is used to read the total number of white list entries that can be stored in the controller.
- HCI_LE_Clear_White_List: This command is used to clear the whole white list.
- HCI_LE_Add_Device_To_White_List: – This command is used to add a device to the white list.
- HCI_LE_Remove_Device_From_White_List: This command is used to remove a device from the white list.

The white lists were explained in Chapter 8 and some of the use case scenarios of white lists are shown at the end of this chapter.

As explained in Chapter 8, specifications 4.2 allowed the address resolution to happen in the Controller instead of the Host. In order to perform this resolution, the Host needs to provide details of the peer address to the Controller so that the Controller can perform the address resolution independently. The Controller stores this information in the resolving list. The commands related to resolving list are:

- HCI_LE_Add_Device_To_Resolving_List: This command is used to add a peer device to the resolving list. The information added includes the identity address of the peer device and the identity resolution key (IRK) of the local and peer device.
- HCI_LE_Remove_Device_From_Resolving_List: This command is used remove a peer device from the resolving list.
- HCI_LE_Clear_Resolving_List: This command is used to remove all devices from the resolving list.
- HCI_LE_Read_Resolving_List_Size: This command is used to read the total number of peer devices entered into the resolving list.
- HCI_LE_Read_Peer_Resolvable_Address: This command is used to read the resolvable address of the peer that is currently being used by the controller. The peer's address may change over time because the private addresses keep

changing. This command provides the identity address as an input parameter and receives back the currently used resolvable address in the response.

- HCI_LE_Read_Local_Resolvable_Address: This command is used to read the local resolvable private address being used by the controller for a particular peer device.
- HCI_LE_Set_Address_Resolution_Enable: This command is used to enable or disable address resolution in the Controller. The default value on reset is address resolution disabled.

Typical usage of these commands is shown in Figure 9.6.

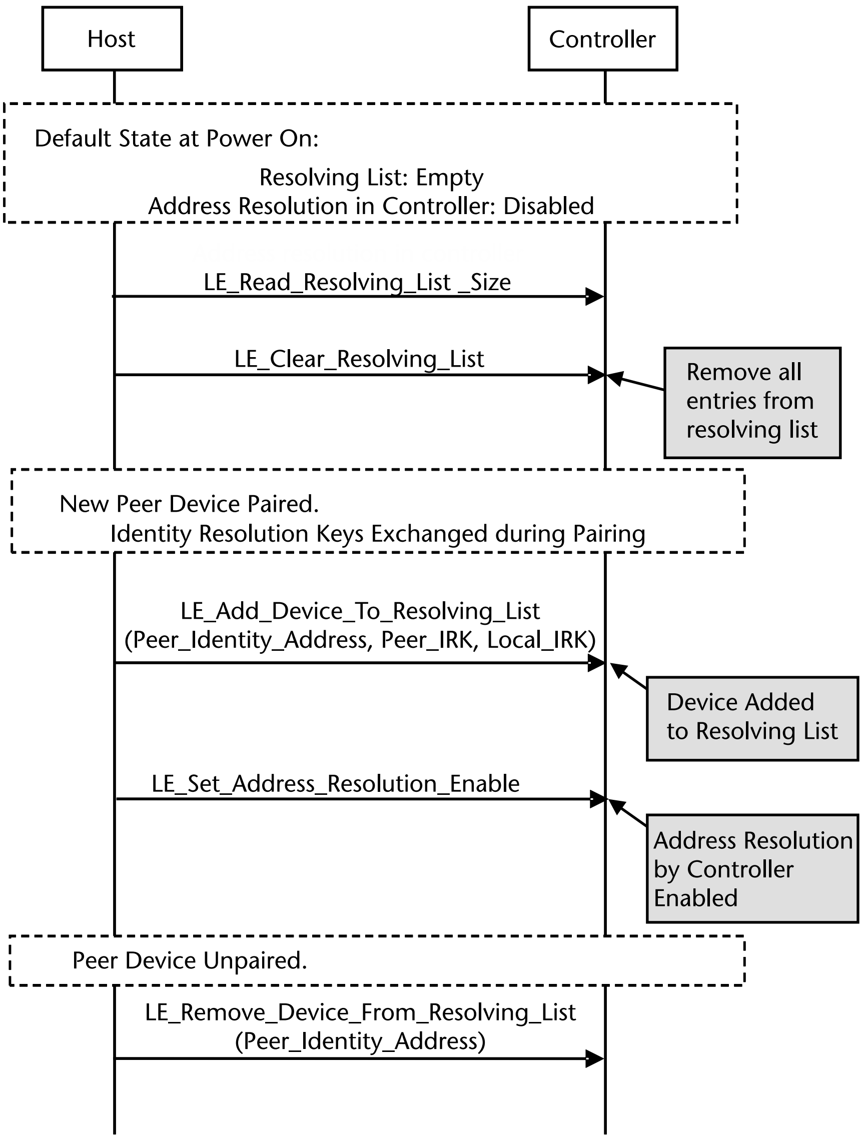

Host
Controller
Default State at Power On:
Resolving List: Empty
Address Resolution in Controller: Disabled
LE_Read_Resolving_List _Size
LE_Clear_Resolving_List
Remove all entries from resolving list
New Peer Device Paired.
Identity Resolution Keys Exchanged during Pairing
LE_Add_Device_To_Resolving_List (Peer_Identity_Address, Peer_IRK, Local_IRK)
Device Added to Resolving List
LE_Set_Address_Resolution_Enable
Address Resolution by Controller Enabled
Peer Device Unpaired.
LE_Remove_Device_From_Resolving_List (Peer_Identity_Address)

Figure 9.6 Typical usage of resolving lists.

9.2.4 Controller Information

The controller information commands are used by the host to find out the information about the controller.

Some of the commands in this category are:

- HCI_Read_Local_Version_Information: This command reads the values of version information from the controller. It can be used to detect whether the controller supports LE functionality or not.
- HCI_Read_Local_Supported_Command: This command reads a bit mask indicating which of the HCI commands are supported by the controller.
- HCI_Read_Support_Features_Command: This command reads a bit mask indicating which of the features are supported by the controller.
- HCI_LE_Read_Support_Features_Command: This command reads a bit mask indicating which of the LE features are supported by the controller. At present it contains only one bit indicating whether the controller supports encryption or not. It may be extended in further versions of the specification.
- HCI_LE_Read_Supported_States: This command is used to retrieve the set of states and state combinations that the controller supports. For example this command is used to know whether directed advertising state is supported.

As explained in Chapter 8, specifications 4.2 allowed LE data packet extensions. This provided support of packet size of up to 251 bytes. The following HCI command was added to allow the Host to read the maximum supported payload length:

- HCI_LE_Read_Maximum_Data_Length: This command is used to read the maximum size of transmit and receive PDUs at the link layer and the maximum time duration the Controller supports for transmission or reception of a link layer PDU.

9.2.5 Remote Information

The remote information commands and events allow the device to find out information about the remote device's configuration.

Some of the commands and events in this category are:

- HCI_LE_Read_Remote_Used_Features: This command requests a list of used features from the remote device. At present the used features bit mask only provides information whether encryption is supported on the remote side or not.
- LE_Read_Remote_Used_Fetures_Complete: This event is generated once the reply from the remote side is received. It reports back to the host the bitmap of features that the remote side supports.

9.2.6 Controller Configuration

The controller configuration commands are used by the host to configure the controller.

Some of the commands in this category are:

- HCI_LE_Set_Advertise_Enable: This command is used to request the controller to start or stop advertising.
- HCI_LE_Set_Advertising_Data: This command is used to set the data used in advertising packets. A maximum of 31 bytes of advertising data can be included.
- HCI_LE_Set_Advertising_Parameters: This command is used to set various parameters related to advertising like the type of advertising, which channels to advertise on, minimum and maximum advertising interval, etc.
- HCI_LE_Set_Random_Address: This command is used to set the random address.
- HCI_LE_Set_Scan_Response_Data: This command is used to provide data in scanning packets.
- HCI_Read_LE_Host_Support: This bit is used to read the current setting of the following two bits in the controller:
  - LE_Supported_Host.
  - Simultaneous LE and BR/EDR to same device capable host.
- HCI_Write_LE_Host_Support: This bit is used to indicate that the host supports LE and whether it supports simultaneous LE and BR/EDR links to the same device. The local host sets these bits in the controller to indicate to remote devices about its LE capabilities:
  - LE_Supported_Host.
  - Simultaneous LE and BR/EDR to same device capable host.

As explained in Chapter 8, specifications 4.2 allowed the private address to be generated periodically. The following command was introduced to support:

- HCI_LE_Set_Resolvable_Private_Address_Timeout: This command is used to specify the time a particular resolvable private address is used before a new one is generated. The default value is 15 minutes, though it can be as low as every 1 second to as high as every 11.5 hours, depending on the level of privacy protection required.

9.2.7 Device Discovery

The device discovery commands and events allow the device to discover other devices in the vicinity.

Some of the commands and events in this group are:

- HCI_LE_Set_Scan_Enable: This command is used to enable and disable scanning. Scanning is used to discover devices in the vicinity.

- HCI_LE_Set_Scan_Parameters: This command is used to set the parameters like scan type, scan interval, etc.
- HCI_LE_Advertising_Report_Event: This event is used to indicate to the host that an advertising report has been received. Multiple reports can be sent in one single event.

Specifications 4.2 introduced another event:

- HCI_LE_Direct_Advertising_Report_Event: This event indicates that a directed advertisement has been received from a peer device that is using a resolvable private address of the local device.

9.2.8 Connection Setup

The connection setup commands and events allow a device to make a connection to another device.

Some of the commands and events in this group are:

- HCI_LE_Create_Connection: This command is used to create a connection to an Advertiser which is connectable.
- LE_Connection_Complete_Event: This event is received on both the Master and Slave side once a connection has been created.
- HCI_Disconnect: This command is used to terminate an existing connection.
- Disconnection_Complete_Event: This event is used to indicate that a connection has been terminated.

Specifications 4.2 introduced an enhanced connection complete event:

- HCI_LE_Enhanced_Connection_Complete_Event: This event indicates that a new connection has been created and is received by Hosts of both the local side and peer side on connection establishment. This event contains additional information about the local and peer resolvable private addresses.

9.2.9 Connection State

The connection state commands and events allow the host to configure the link.

Some of the commands and events in this group are:

- HCI_LE_Connection_Update: This command is used to change the parameters related to a connection, like connection interval, supervision timeout, etc.
- LE_Connection_Update_Complete_Event: This event is used to indicate to the host that the parameters related to a connection have been changed.

Specifications 4.1 introduced additional commands and events related to connection parameters update:

- HCI_LE_Remote_Connection_Parameter_Request_Event: This event is used to indicate to the Host that the remote device is requesting a change to the connection parameters.
- HCI_LE_Remote_Connection_Parameter_Request_Reply: This command is used by the Host to reply to HCI_LE_Remote_Connection_Parameters_Request_Event to accept the changes to the connection parameters provided by the remote device.
- HCI_LE_Remote_Connection_Parameter_Request_Negative_Reply: This command is used by the Host to reply to HCI_LE_Remote_Connection_Parameters_Request_Event to reject the changes to the connection parameters provided by the remote device.

Specifications 4.2 introduced commands and events to support LE data packet extensions. These are:

- HCI_LE_Write_Suggested_Default_Data_Length_Command: This command is used by the Host to specify its preferred values for maximum transmission size and maximum transmission time for all new connections that will be established in the future.
- HCI_LE_Read_Suggested_Default_Data_Length_Command: This command is used by the Host to read its preferred values for maximum transmission size and maximum transmission time to be used for all new connections.
- HCI_LE_Set_Data_Length: This command is used by the Host to suggest the preferred maximum transmission packet size and length for a particular connection. In comparison to the previous command, this one is used to specify the data length for a particular connection. The Controller may use a different value than what is suggested by the Host based on certain parameters.
- HCI_LE_Data_Length_Change_Event: This event is used by the Controller to indicate to the Host about a change in the maximum transmission size or maximum transmission time.

9.2.10 Physical Links

The physical link commands and events allow the configuration of a physical link.
There is only one command related to LE in this group:

- HCI_LE_Set_Host_Channel_Classification: This command is used by the host to provide a channel map to the controller. The channel map indicates to the controller which of the 37 data channels it can use.

9.2.11 Link Information

The link information commands and events allow the host to read information about a link.

Some of the commands in this group are:

- HCI_Read_Transmit_Power_Level: This command is used by the host to get the value of the transmit power level used to transmit for a given connection handle.
- HCI_Read_RSSI: This command reads the Received Signal Strength Indicator (RSSI) value from a controller.
- HCI_LE_Read_Advertising_Channel_Tx_Power: This command is used to read the transmit power level that is used for advertising channel packets.
- HCI_LE_Read_Channel_Map: This command is used to read the current channel map used for the specified connection handle.

9.2.12 Authentication and Encryption

The authentication and encryption commands and events allow authentication of a remote device and then encryption of a link.

Some of the commands and events in this group are:

- HCI_LE_Encrypt: This command is used by the host to request the controller to encrypt some data. The host provides the key and data to be encrypted. The controller encrypts the data and returns the encrypted data in a command complete event.
- HCI_LE_Long_Term_Key_Requested_Event: This event is used by the controller to request the Long Term Key from the host.
- HCI_LE_Long_Term_Key_Request_Reply: This command is used by the host to provide the Long Term Link Key.
- HCI_LE_Long_Term_Key_Request_Negative_Reply: This command is used by the host to refuse to provide the Long Term Link Key.
- HCI_LE_Rand: This command is used by the host to request to the controller to generate an 8 octet random data and send to the host.
- HCI_LE_Start_Encryption: This command is used to request the controller to encrypt a connection.
- Encryption_Change_Event: This event is used to indicate that the change of encryption mode has been completed.
- Encryption Key_Refresh_Complete_Event: This event is used to indicate to the host that the encryption key was refreshed on the given connection handle.

As described in Chapter 8, specifications 4.1 introduced the support of having a maximum time interval wherein an authenticated packet must be received from the peer device. The commands and events related to this feature are:

- HCI_Write_Authenticated_Payload_Timeout: This command is used to set the authenticated payload timeout for a particular connection.
- HCI_Read_Authenticated_Payload_Timeout: This command is used to read the Authenticated Payload Timeout for a particular connection.

- Authenticated_Payload_Timeout_Expired_Event: This event is used to indicate that a packet containing a valid message integrity check (MIC) was not received from the peer side in the specified timeout. This command can be used in conjunction with the LL_Ping request (refer to Section 8.12.8) to solicit a MIC-enabled packet from the remote device

Specifications 4.2 introduced the secure connections feature and upgraded LE pairing to use FIPS-approved algorithms, including AES-CMAC and P-256 elliptic curve. The additional HCI commands and events to support this higher security are:

- HCI_LE_Read_Local_P-256_Public_Key: This command is used to return the local P-256 public key from the Controller. The Controller generates a new P-256 public/private key pair and returns it to the Host.
- HCI_LE_Read_Local_P-256_Public_Key_Complete_Event: This event is generated once the Controller finishes the generation of the P-256 key pair. It returns the local P-256 public key.
- LE_Generate_DHKey: This command is used to initiate the generation of a Diffie-Hellman key. The P-256 key of the peer device is provided as an input to generate the Diffie-Hellman key.
- LE_Generate_DHKey_Complete_Event: This event is used to indicate that the Diffie-Hellman key has been generated by the controller. It returns the DHKey.

9.2.13 Testing

The testing commands allow the controller to be put in a special test mode so that testing can be performed. These consist of commands to test the receiver and the transmitter.

Some of the commands in this group are:

- HCI_LE_Receiver_Test: This command is used to test receiver functionality of the device. The remote tester generates reference test packets which are received by the receiver.
- HCI_LE_Transmitter_Test: This command is used to request the controller to start transmitting test reference packets. The remote tester can read and verify these packets.
- HCI_LE_Test_End: This command is used to stop any test which is in progress.

9.2.14 Usage of White Lists

The commands to reset and set the devices in the white list were described earlier. Once a white list has been set, the following commands are used to enable the use of white list:

- The HCI_LE_Set_Advertising_Parameters command takes the parameter Advertising_Filter_Policy which is used to select whether to allow scan and connect requests from all devices or just the devices present in White List.
- The HCI_LE_Set_Scan_Parameters command takes the parameter Scanning_Filter_Policy which is used to select whether to accept advertising packets from all devices or just devices present in the white list.
- The HCI_LE_Create_Connection command takes the parameter Initiator_Filter_Policy which is used to determine which Advertiser to connect to.

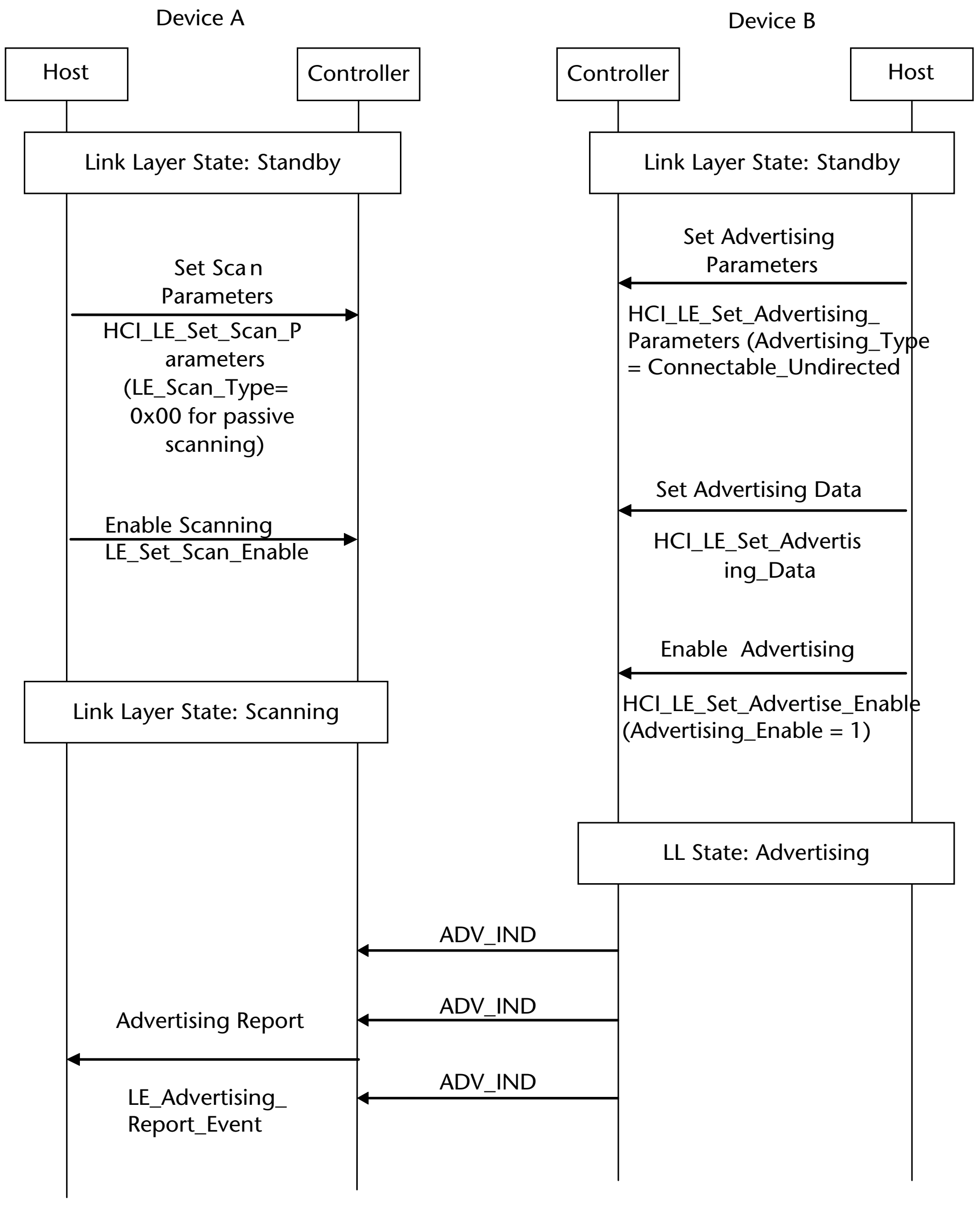

Device A
Device B
Host
Controller
Controller
Host
Link Layer State: Standby
Link Layer State: Standby
Set Scan Parameters
HCI_LE_Set_Scan_Parameters (LE_Scan_Type= 0x00 for passive scanning)
Set Advertising Parameters
HCI_LE_Set_Advertising_Parameters (Advertising_Type = Connectable_Undirected
Enable Scanning
LE_Set_Scan_Enable
Set Advertising Data
HCI_LE_Set_Advertising_Data
Enable Advertising
HCI_LE_Set_Advertise_Enable (Advertising_Enable = 1)
Link Layer State: Scanning
LL State: Advertising
ADV_IND
ADV_IND
Advertising Report
LE_Advertising_Report_Event
ADV_IND

Figure 9.7 Typical sequence for passive scanning.

9.3 Practical Sequence Diagrams

This section provides sequence diagrams for some of the commonly used procedures. The interaction of the link layer with the host, along with the corresponding HCI Commands and Events, is also shown to provide the complete view on how each of the procedures are used in practical scenarios. In the interest of simplicity,

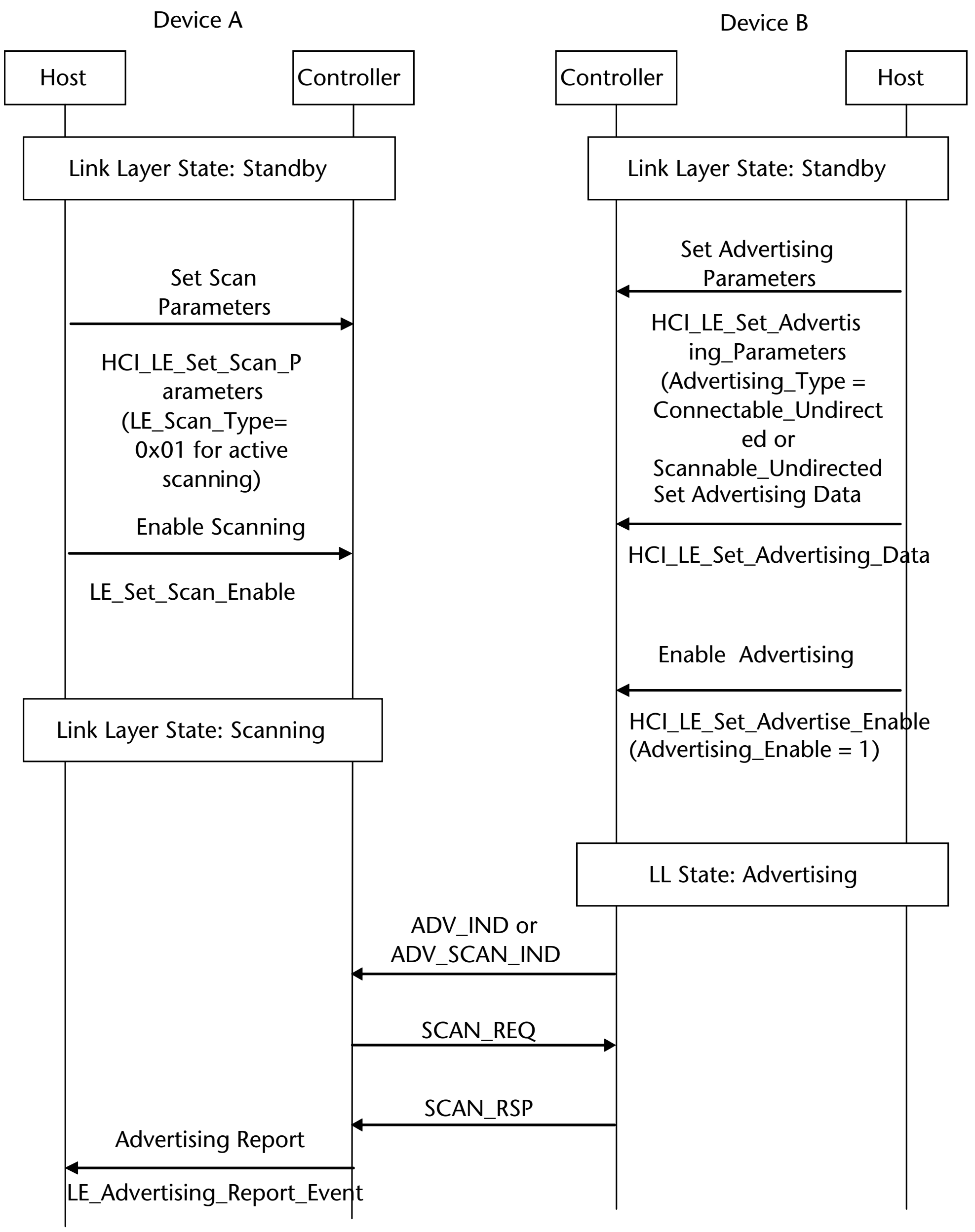

Device A
Device B
Host
Controller
Controller
Host
Link Layer State: Standby
Link Layer State: Standby
Set Scan Parameters
HCI_LE_Set_Scan_Parameters (LE_Scan_Type= 0x01 for active scanning)
Enable Scanning
LE_Set_Scan_Enable
Set Advertising Parameters
HCI_LE_Set_Advertising_Parameters (Advertising_Type = Connectable_Undirected or Scannable_Undirected
Set Advertising Data
HCI_LE_Set_Advertising_Data
Enable Advertising
HCI_LE_Set_Advertise_Enable (Advertising_Enable = 1)
Link Layer State: Scanning
LL State: Advertising
ADV_IND or ADV_SCAN_IND
SCAN_REQ
SCAN_RSP
Advertising Report
LE_Advertising_Report_Event

Figure 9.8 Typical sequence for active scanning.

the command status and command complete events are not shown. These will also be generated when these commands are sent out in practice.

9.3.1 Passive Scanning

A typical sequence for passive scanning is shown in Figure 9.7.

9.3.2 Typical Sequence for Active Scanning

A typical sequence for active scanning is shown in Figure 9.8.

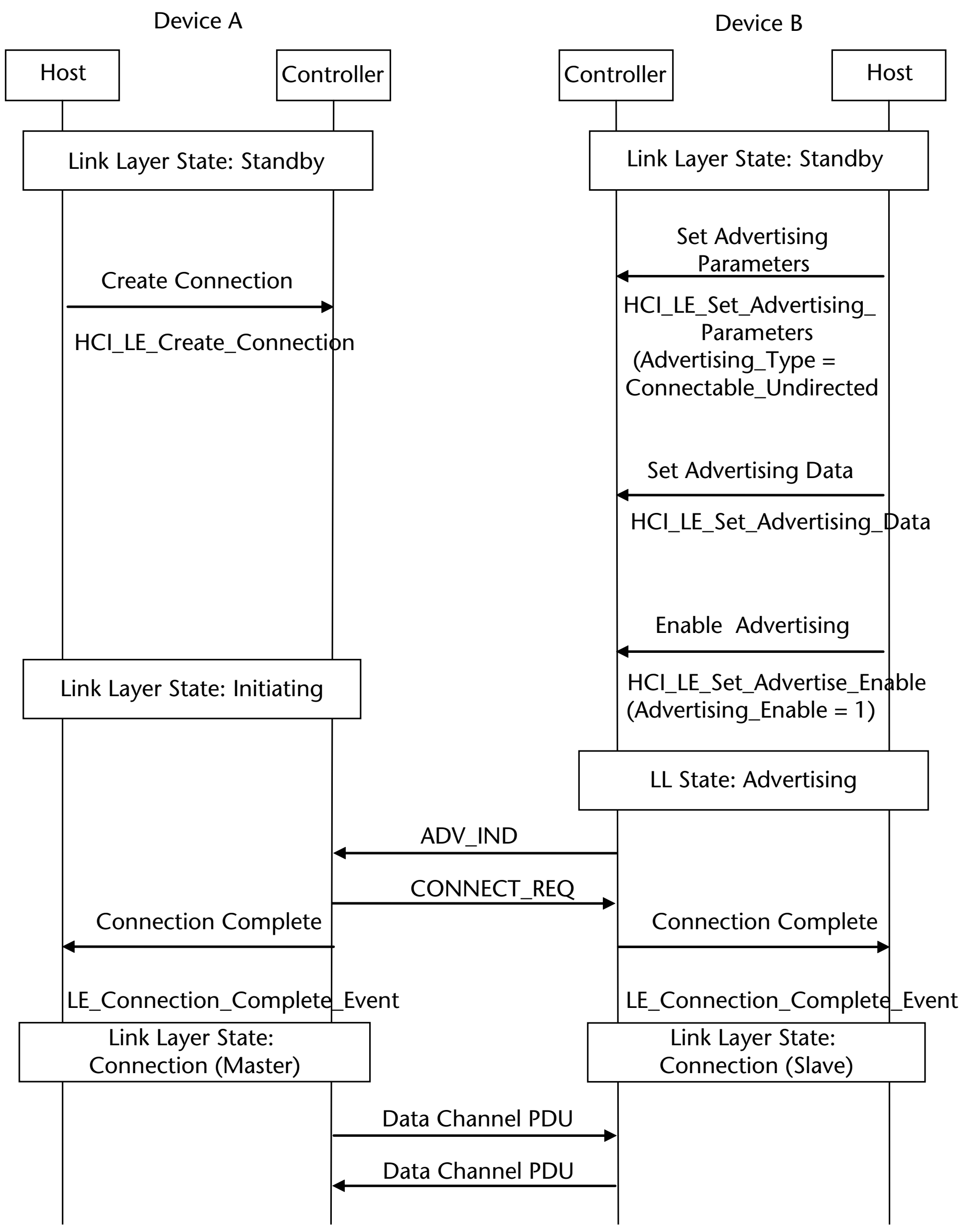

Device A
Device B
Host
Controller
Controller
Host
Link Layer State: Standby
Link Layer State: Standby
Set Advertising Parameters
Create Connection
HCI_LE_Set_Advertising_Parameters (Advertising_Type = Connectable_Undirected
HCI_LE_Create_Connection
Set Advertising Data
HCI_LE_Set_Advertising_Data
Enable Advertising
Link Layer State: Initiating
HCI_LE_Set_Advertise_Enable (Advertising_Enable = 1)
LL State: Advertising
ADV_IND
CONNECT_REQ
Connection Complete
Connection Complete
LE_Connection_Complete_Event
LE_Connection_Complete_Event
Link Layer State: Connection (Master)
Link Layer State: Connection (Slave)
Data Channel PDU
Data Channel PDU

Figure 9.9 Typical sequence for connection establishment.

9.3.3 Connection Establishment

A typical sequence for connection establishment is shown in Figure 9.9.

9.3.4 Setting up White list

Figure 9.10 shows a typical sequence for setting up a white list and then various scenarios in which a white list may be used.

9.4 Summary

The host controller interface provides a communication interface between the upper layers and lower layers. LE specification has reused the HCI layer and enhanced it to add support for LE related commands and events. This chapter explained many of the LE related commands and events. It also provided sequence diagrams for some of the typical use cases of LE.

This chapter completes the information about the LE lower layers. Subsequent chapters will focus on the LE upper layers and profiles.

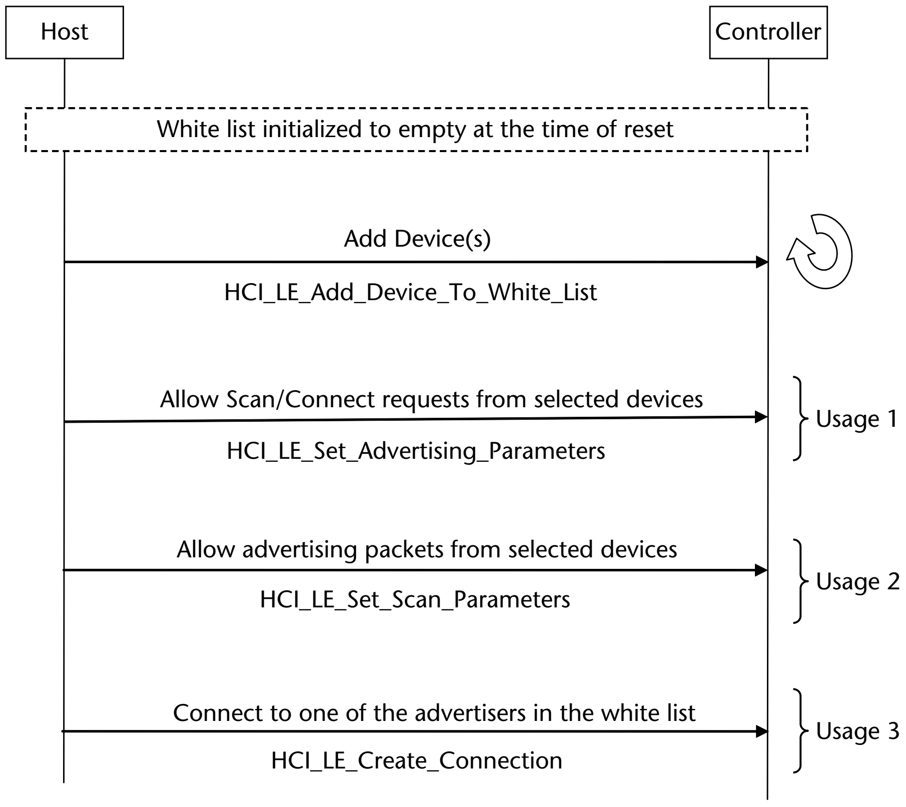

Host
Controller
White list initialized to empty at the time of reset
Add Device(s)
HCI_LE_Add_Device_To_White_List
Allow Scan/Connect requests from selected devices
HCI_LE_Set_Advertising_Parameters
Usage 1
Allow advertising packets from selected devices
HCI_LE_Set_Scan_Parameters
Usage 2
Connect to one of the advertisers in the white list
HCI_LE_Create_Connection
Usage 3

Figure 9.10 Usage of White Lists.

Bibliography

Bluetooth Core Specification 4.0 http://www.bluetooth.org.

CHAPTER 10

Logical Link Control and Adaptation Protocol (L2CAP)

10.1 Introduction

The Logical Link Control and Adaptation Protocol (L2CAP) layer acts as an interface between the higher layer protocols and the lower layers. The position of L2CAP in the LE protocol stack is shown in Figure 10.1.

L2CAP for BR/EDR was explained in detail in Chapter 4. LE reuses the L2CAP functionality of BR/EDR and simplifies the functionality in a major way to make it suitable for LE devices. A significant part of the L2CAP functionality is not needed for LE devices. That functionality has been removed in order to keep the implementation small and simple.

The Bluetooth specification defines the HCI interface as an optional interface. In case of systems which include the HCI interface, the L2CAP layer uses the HCI layer to send data by encapsulating it into HCI ACL Data packets. In the systems in which the HCI layer is not present, the L2CAP layer invokes the functionality of Link Layer directly (maybe through some API mechanism) to send the packets.

Before going further, it will be useful to read the sections related to L2CAP in Chapter 4 because those are broadly applicable to LE as well. This chapter will provide details on the LE specific modifications to L2CAP.

10.2 PDU and SDU

An SDU (Service Data Unit) is a packet that contains data originating from the upper layers (For example Attribute protocol). The L2CAP entities transfer this SDU transparently from the upper layer of one side to the upper layer of the other side.

A PDU (Protocol Data Unit) is a packet containing L2CAP protocol information and may contain data from upper layers as well (SDU). So an SDU will be encapsulated into an L2CAP PDU before transmitting to the remote side. The remote side L2CAP will extract the header information and provide the SDU to the upper layers on the remote side. All SDUs are encapsulated into one or more L2CAP PDUs. The PDU and SDU are illustrated in Figure 10.2.

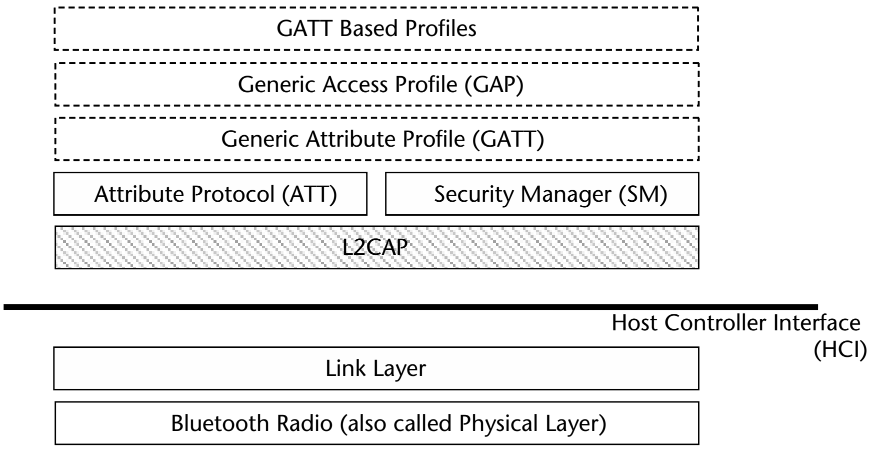

GATT Based Profiles
Generic Access Profile (GAP)
Generic Attribute Profile (GATT)
Attribute Protocol (ATT)
Security Manager (SM)
L2CAP
Host Controller Interface (HCI)
Link Layer
Bluetooth Radio (also called Physical Layer)

Figure 10.1 L2CAP in LE protocol stack.

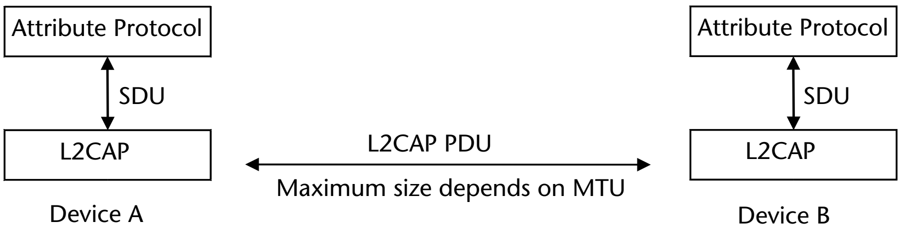


Figure 10.2 L2CAP PDU and SDU.

10.3 Basic Assumptions

The L2CAP protocol is designed with the following basic assumptions about the controller:

1. The packets are delivered in the correct sequence by the controllers on both sides. This means that the packet which was transmitted first by the host on the transmitter side will be received first by the host on the receiver side.
2. Only one LE-U logical link exists between the two devices.
3. The controllers provide a degree of reliability by including error detection, and retransmission mechanisms.
4. The controllers provide flow control mechanisms for data going over the air as well as data going over the HCI transport layer. This ensures that data does not get overwritten at any stage.

10.4 Maximum Transmission Unit (MTU)

The MTU is used to inform the other side the maximum size of SDU it is capable of accepting. The minimum MTU for LE is 23 octets. This means that an LE device will certainly receive a packet of 23 octets, but, it may receive a bigger packet as well. It is possible that the two devices which are connected may have different

MTU sizes. The sender always keeps the MTU size of the receiver in mind when sending a packet and does not exceed that MTU size. In case the sender transmits a packet that exceeds the MTU size of the receiver, the receiver sends back a Command_Reject to indicate that it cannot accept this packet.

The minimum supported MTU for LE is much smaller than that of BR/EDR. In the case of BR/EDR, the minimum supported MTU is 48 octets if extended flow specification is not supported and 672 if extended flow specification is supported. The smaller MTU size of 23 octets in the case of LE helps to keep the packet sizes of LE smaller, thus, leading to savings of buffer space needed for transmit/receive and the power consumed in exchanging the packets.

10.5 L2CAP Features

Similar to BR/EDR, L2CAP provides the abstraction of channels to layers on top of it. It provides the following features:

1. Fixed Channel Identifiers.
2. Fragmentation and Defragmentation of data.
3. Multiplexing and Demultiplexing of various channels over a shared logical link (LE-U).

10.5.1 Fixed Channel Identifiers

As explained in Chapter 4, L2CAP is based on the concept of channels. A channel identifier (CID) is a local name representing a logical channel end point on the device. All channels going over an LE physical link are mapped to a single LE-U logical link.

In the case of BR/EDR, CID 0x0001 is fixed for signaling and CID 0x0002 is fixed for connectionless data. The remaining CIDs are dynamically allocated. So, for example, if SDP layer wants to create a connection, L2CAP dynamically assigns a CID to it.

In the case of LE, the CID assignment is simplified by using only fixed CIDs. The fixed CIDs for LE are shown in Table 10.1. CID 0x0005 is fixed as the L2CAP signaling channel. Only two protocols can use the services of L2CAP layer in the case of LE, viz Attribute Protocol and Security Manager Protocol. Both of these are assigned fixed CIDs. This is very much simplified as compared to BR/EDR where the CIDs for various protocols are allocated dynamically.

All these channels are available as soon as the LE-U logical link is setup. The higher layers can start sending data to L2CAP for these channels that is transmitted to the remote device. So ATT protocol can send data over CID 0x0004 as soon as

Table 10.1 CID Name Space for LE

CID	*Description*
0x0004	Attribute Protocol. Attribute Protocol will be covered in detail in later chapters.
0x0005	LE L2CAP Signaling Channel.
0x0006	Security Manager Protocol (SMP). SMP will be covered in detail in later chapters.

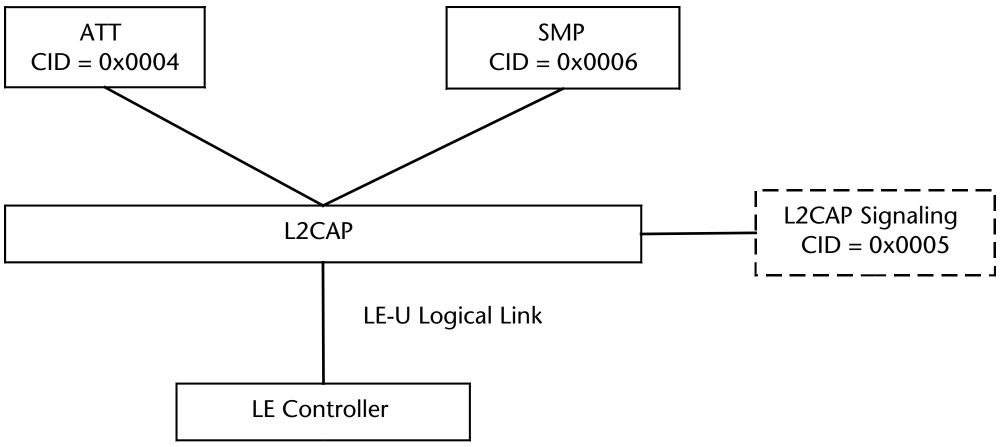


Figure 10.3 Channel multiplexing.

the link layer creates a connection to the remote device. This is much simpler than, for example, SDP sending data over L2CAP in the case of BR/EDR. In the case of BR/EDR the Connect and Configure signals need to be exchanged between the two devices to establish an L2CAP connection before any data can be sent. This is one of the significant enhancements made in LE toward faster connection setup time. The number of PDUs exchanged between the two devices to setup the connection before transferring higher layer data is reduced to zero in the case of L2CAP.

Specifications 4.1 defined additional CID name space to support credit based flow control. This is shown in Table 10.2

10.5.2 Fragmentation and Defragmentation of Data

L2CAP allows higher layer protocols to send bigger chunks of data to it even though the LE PDUs at the link layer are restricted to much smaller sizes. In such cases, L2CAP fragments the data according to the size of the ACL buffers that are present for LE in the controller before sending it to the controller for transmission. At the time of reception, L2CAP re-assembles the data to reconstruct the original packet that was sent by the higher layers. It then sends the reconstructed packet to the higher layers.

10.5.3 Channel Multiplexing

As shown in Table 10.1, three L2CAP channels are defined in the case of LE:

1. ATT with CID = 0x0004.
2. SMP with CID = 0x0006.

Table 10.2 Additional CID Name Space for LE Specifications 4.1 Onwards

CID	*Description*
0x0020–0x003E	As per SIG assigned numbers page
0x0040–0x007F	Dynamically allocated during credit based connection mechanisms

3. L2CAP's own signaling channel with CID = 0x0005.

L2CAP performs the multiplexing and demultiplexing of these three channels on top of the shared LE-U logical link. This means that it accepts data from these three channels and multiplexes it on the shared link at the time of transmitting. At the time of receiving, it demultiplexes the data and provides the data to the appropriate higher layer entity.

As shown in Table 10.3, additional L2CAP channels were defined starting from specifications 4.1. Channels are assigned numbers with CID 0x0020–0x003E and dynamically allocated channel numbers with CID 0x0040–0x007F.

10.6 Data Packets

As mentioned in Chapter 4, L2CAP supports five modes of operation for BR/EDR.

- Basic L2CAP Mode (used in LE as well);
- Flow Control Mode;
- Retransmission Mode;
- Enhanced Retransmission Mode;
- Streaming Mode.

Out of these five modes of operation, only the Basic Mode is used for LE. The PDU that is exchanged in the Basic L2CAP Mode is also referred to as a B-Frame. The format of B-Frame is shown in Figure 10.4.

The Length field indicates the size of the information payload. It can go up to 65535 bytes and is used during recombination when the different fragments are reassembled at the receiver side to reconstruct the whole packet. The Channel ID is 0x0004 for ATT and 0x0006 for SMP. The Information Payload contains the higher layer data. So in the case of LE it contains the data that is sent by ATT or SMP.

As compared to BR/EDR, L2CAP supports only the Basic L2CAP mode. This simplifies the design of the L2CAP layer to a large extent while still not compromising

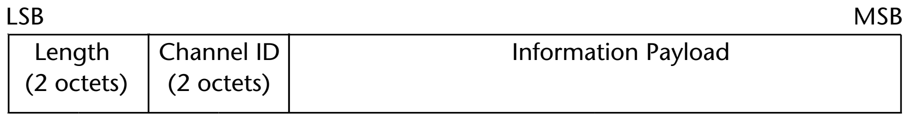


Figure 10.4 LE L2CAP data packet format (B-Frame).

Table 10.3 L2CAP Parameters for ATT and SMP

Parameter	*Value*
MTU	23
Flush Timeout	0xFFFF (Infinite)
Quality of Service	Best Effort
Mode	Basic Mode

on the functionality required by the layers that sit on top of L2CAP(ATT and SMP). This is another major step towards decreasing the cost and power consumption of LE devices.

Specifications 4.1 defined an additional mode of operation for LE, known as LE credit-based flow control mode.

The LE credit-based flow control mode is used for L2CAP data on connection oriented channels. It uses a credit based scheme that will be explained in Sections 10.8 and 10.9. It is not used for signaling packets.

The LE connection-oriented channels and LE credit-based flow control mode are heavily used by Internet protocol support profile (IPSP) to allow Bluetooth devices to connect to the Internet through gateways. This will be explained in detail in Chapter 15.

10.7 L2CAP Parameters

Table 10.2 shows the L2CAP Parameters that are used by both ATT and SMP.

A Flush Timeout of 0xFFFF means that the baseband continues to do retransmissions until the link is dropped if the packet is not acknowledged.

LE requires only Best Effort service. This means that there is no guarantee that the data will be received by the remote side. (In the case of BR/EDR, L2CAP also supports Guaranteed QoS option which guarantees a specific amount of bandwidth for the particular L2CAP channel. This option is not supported in the case of LE.)

10.8 L2CAP Signaling

As shown in Table 10.1 LE uses channel 0x0005 as the Signaling channel. The signaling channel is available as soon as the lower layer logical transport is set up and L2CAP traffic is enabled. The commands on the signaling channel are in the form of requests and responses.

LE simplifies the procedure for sending commands on the signaling channel. While on the BR/EDR signaling channel (0x0001) multiple commands can be sent within a single PDU, in the case of LE signaling channel (0x0005) only one command can be sent per PDU. This makes the logic for decoding the packets simpler on the receiving L2CAP entity.

The PDUs that contain L2CAP signaling messages are known as C-Frames (Control Frames). These are used only on the L2CAP Signaling channel. The format of C-frames is shown in Figure 10.5.

The code identifies the type of command that is being sent on the Signaling channel. The commands that are allowed on LE Signaling channel are shown in Table 10.4. This list is much smaller than the number of commands that are supported on the BR/EDR signaling channel. This allows the LE software to be simpler in terms of the number of commands that it needs to process. The LE credit-based flow control request, LE credit-based flow control response, and LE flow control credit commands were introduced in the specifications 4.1 to support credit-based flow control for connection-oriented channels. These are shown in italics in Table 10.3

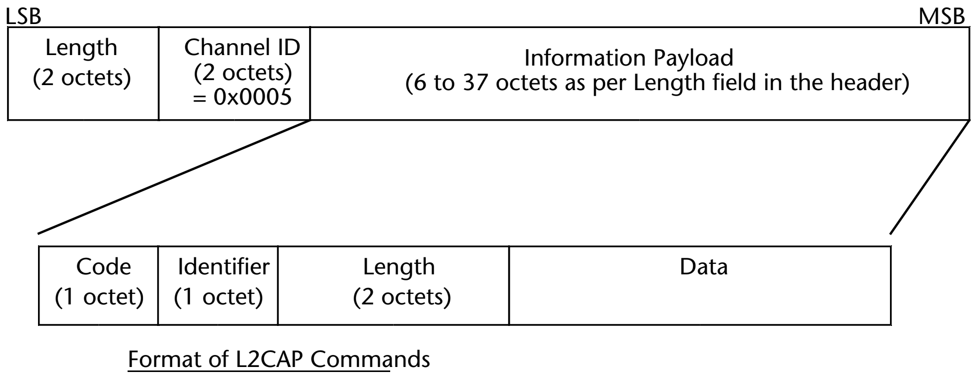


Figure 10.5 L2CAP PDU on signaling channel (C-Frame).

Table 10.4 L2CAP Signaling Commands

Code	*Command*	*Direction*
0x01	Command Reject	Both Directions
0x12	Connection Parameter Update Request	Slave to Master
0x13	Connection Parameter Update Response	Master to Slave
0x14	*LE Credit Based Connection Request*	*Both Directions*
0x15	*LE Credit Based Connection Response*	*Both Directions*
0x16	*LE Flow Control Credit*	*Both Directions*

The identifier field is used to match the responses with the requests. This field is set by the requesting device and the responding device uses the same value while responding. When the requesting device receives the response back, it can identify which request that response is for. The length field indicates the size of the Data field. The Data field is variable in length and the size depends on the command that is being sent.

10.8.1 Command Reject

The command reject PDU is sent as a response if the command code was not identified or the length was incorrect. It contains a reason field to indicate why the packet was rejected. The reason can be any of the following:

- Command not understood;
- Signaling MTU exceeded;
- Invalid CID in request.

10.8.2 Connection Parameter Update Request

This packet is sent by the LE Slave to the LE Master to request a set of new connection parameters. The connection parameters include:

- Interval Min: The minimum value for the connection event interval.
- Interval Max: The maximum value for the connection event interval.
- Slave Latency: This defines the Slave latency of the connection in number of connection events. For example if Slave Latency is 4, then it will listen to a packet from the Master on every 4th anchor point. If it is 0, then the Slave will listen to a packet from the Master on every anchor point.
- Timeout Multiplier: The connection supervision timeout can be calculated from this field as follows:
  - Connection Supervision Timeout = Timeout Multipler * 10 ms.

If the Master decides to accept this request, then it sends the new set of connection parameters to the Slave using the link layer connection update procedure. The connection update procedure at the link layer level was explained in Chapter 8. If the Master decides not to accept the parameters, then it rejects the request using the Command Reject.

It may be noted that this command can only be sent from the LE Slave to the LE Master and not the other way round. If the LE Master needs to change the connection parameters, then it can directly use the link layer procedure.

10.8.3 Connection Parameter Update Response

This packet is sent by the LE Master to the LE Slave in response to the connection parameter update request. If the Master accepts the parameters sent by the Slave, it also sends the connection parameter update to the controller using the HCI_LE_Connection_Update command so that the controller can start the link layer procedure to update the connection. A typical sequence for updating the connection parameters is shown in Figure 10.6.

10.8.4 LE Credit-Based Connection Request

This command is used to create and configure an L2CAP channel between two devices. The parameters of this command are:

- LE Protocol Service Multiplexer (PSM): This specifies the PSM value of the higher layer protocol for which the L2CAP connection is being created. It could either be a fixed value assigned by Bluetooth SIG or a dynamic value assigned by GATT.
- Source CID: This is the channel identifier that will be used to receive packets. The sender will send all packets to this CID once the channel is established.
- Maximum Transmission Unit (MTU): This indicates the maximum SDU size that L2CAP can accept. An SDU may be split across several PDUs
- Maximum PDU Size (MPS): This indicates the maximum payload size that L2CAP can accept.
- Initial Credits: This indicates the number of LE-frames that the peer device can send.

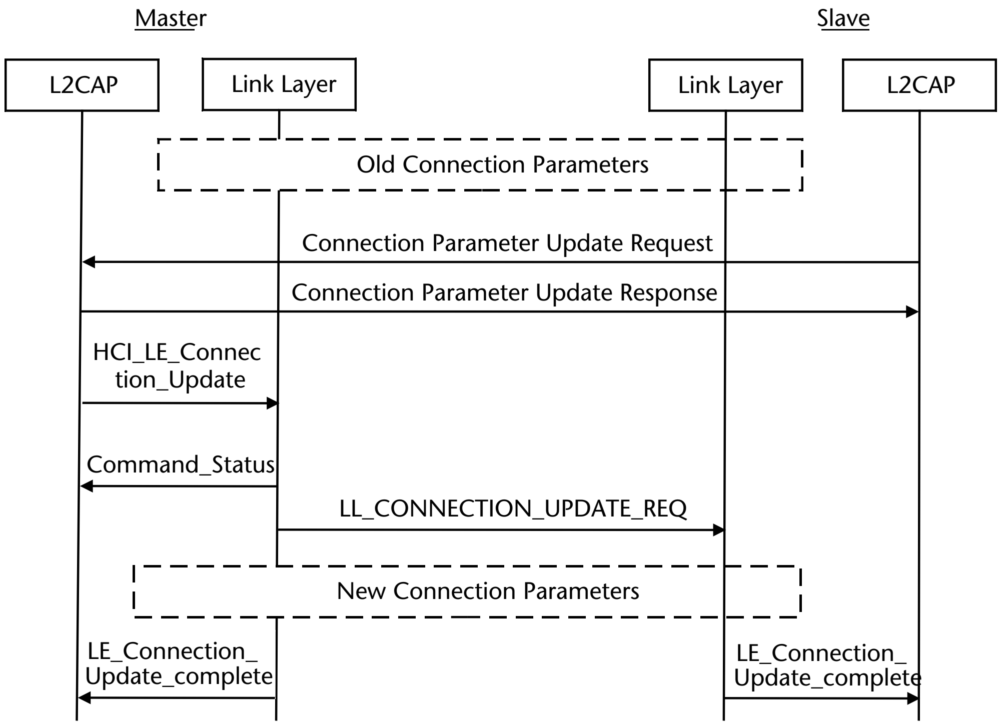

Master
Slave
L2CAP
Link Layer
Link Layer
L2CAP
Old Connection Parameters
Connection Parameter Update Request
Connection Parameter Update Response
HCI_LE_Connec tion_Update
Command_Status
LL_CONNECTION_UPDATE_REQ
New Connection Parameters
LE_Connection_ Update_complete
LE_Connection_ Update_complete

Figure 10.6 Typical connection parameter update sequence.

10.8.5 LE Credit-Based Connection Response

This packet is used to respond to an LE credit-based connection request. The parameters of this packet are:

- Destination CID: This specifies the channel end point of the device that is sending the response.
- Maximum Transmission Unit: This indicates the maximum SDU size that L2CAP can accept.
- Maximum PDU Size: This indicates the maximum payload size that L2CAP can accept.
- Initial Credits: This indicates the number of LE-frames that the peer device can send.
- Result: This field indicates the outcome of the connection request. A value of 0x0000 indicates success.

10.8.6 LE Flow Control Credit

This packet is sent by a device when it is capable of accepting additional LE-frames. For example, once it has processed 5 frames, it may send this packet to indicate 5 credits. The parameters of this packet are the CID (which indicates the source channel endpoint on which additional LE-frames can be sent) and the credits (which represents the number of additional LE-frames that can be sent to this device).

Once the receiver is ready to take more packets, it will send the credits to the transmitter.

10.9 Credit-Based Flow Control

One of the enhancements made in specifications 4.1 is the introduction of connection-oriented channels and the usage of credit-based flow control for these channels.

An LE information frame (also called an LE-frame PDU) is used to exchange data with the peer entity. The format of an LE-frame is shown in Figure 10.7

The length field specifies the length of the information payload.

The Channel ID identifies the destination channel end point.

The L2CAP SDU length field is only present in the first PDU and specifies the length of the entire SDU (higher layer protocol packet), which will be transported in L2CAP packets. For subsequent L2CAP packets that contain the same SDU, this field is absent.

The credit-based flow control used in LE is very similar to the credit-based flow control that was described in Chapter 4 for RFCOMM over BR/EDR. At the time of connection establishment, the remote entity provides initial credits. This is the maximum number of LE-frames that can be sent to the remote entity. One credit represents permissions to send one LE-frame. Whenever an LE-frame is sent to the remote entity, the number of credits is reduced by 1. If the number of credits reaches 0 at any time, no further packets are sent until more credits are received from the remote entity. Once the remote entity processes these packets and is capable of receiving more packets, it sends an equivalent number of credits using the LE flow control credits packet.

The credit-based flow control provides a simple yet effective mechanism for flow control between peer devices. It is particularly useful when the receiving entity has large buffers and is capable of storing or processing multiple packets. The transmitter does not need to wait for a response from each packet; rather, it can send a set of packets (determined by the credits received). While this is happening, the transmitter can send additional packets without interrupting the flow of packets if more credits are received from the remote side.

The credit-based flow control mechanism is also useful in scenarios where the transmitter is a fast device (e.g., a smartphone), while the receiver is a slow or memory-constrained device (e.g., a sensor). The sensor can control the pace at which the smartphone sends packets to it.

The LE connection-oriented channels and LE credit-based flow control modes are heavily used by the IPSP in order to allow Bluetooth devices to connect to the internet through gateways. This will be discussed in detail in Chapter 15.

LSB MSB

Length (2 octets)	Channel ID (2 octets)	L2CAP SDU Length (2 octets)	Information Payload (Up to 65533 octets)

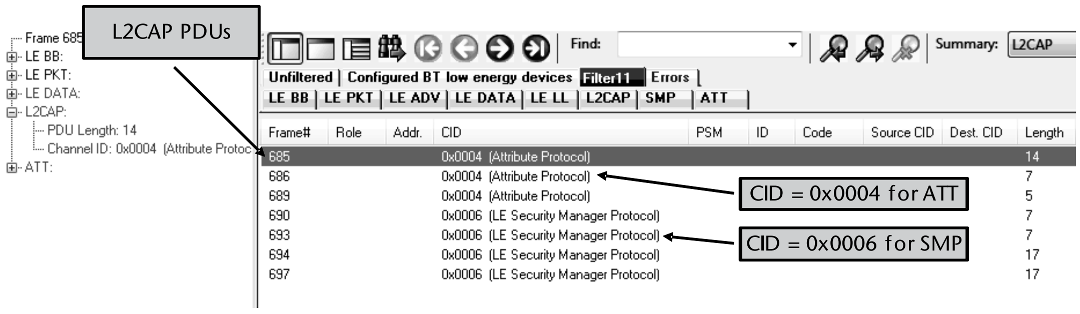
Figure 10.7 Format of an LE information frame (LE-frame).

10.10 Practical Examples

Figure 10.8 shows a practical example of the L2CAP PDUs being exchanged between two devices. The following points may be observed:

- The CID used for ATT is 0x0004 and CID used for SMP is 0x0006.
- There are no CONNECT and CONFIGURE requests as in the case of BR/EDR. In the case of L2CAP, once the connection is established at the link layer level, data PDUs can be exchanged.

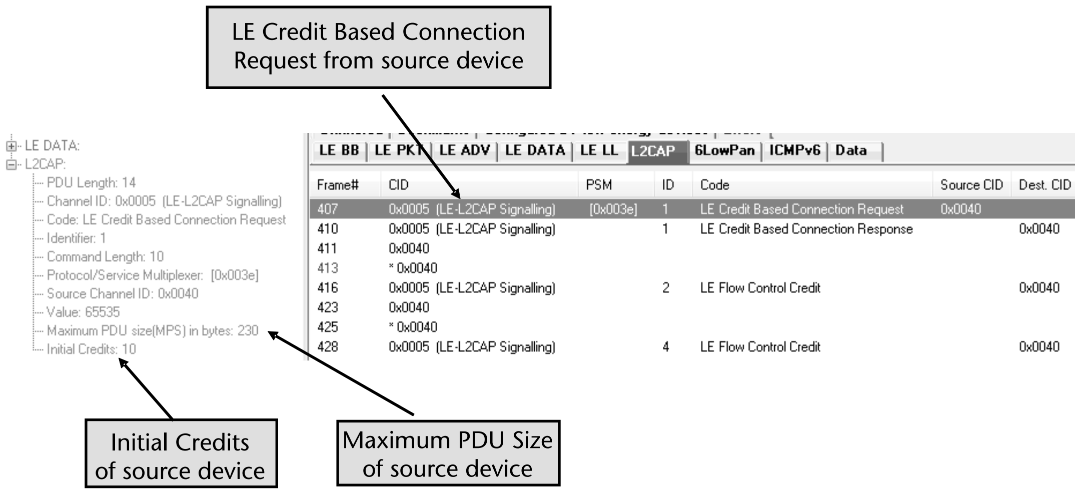
Figures 10.9, 10.10, and 10.11 show another practical example of L2CAP credit-based flow control. The LE credit-based connection request is sent by the source device shown in Figure 10.11. During this request, the source device provides an initial number of credits (in this example, 10 credits). This means that the destination would be allowed to send up to 10 packets once the connection is established.


L2CAP PDUs
CID = 0x0004 for ATT
CID = 0x0006 for SMP
Find:
Summary: L2CAP
Unfiltered | Configured BT low energy devices | Filter11 | Errors
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | SMP | ATT

```
Frame 685
LE BB:
LE PKT:
LE DATA:
L2CAP:
  PDU Length: 14
  Channel ID: 0x0004 (Attribute Protoc
ATT:
```

Frame#	Role	Addr.	CID	PSM	ID	Code	Source CID	Dest. CID	Length
685			0x0004 (Attribute Protocol)						14
686			0x0004 (Attribute Protocol)						7
689			0x0004 (Attribute Protocol)						5
690			0x0006 (LE Security Manager Protocol)						7
693			0x0006 (LE Security Manager Protocol)						7
694			0x0006 (LE Security Manager Protocol)						17
697			0x0006 (LE Security Manager Protocol)						17

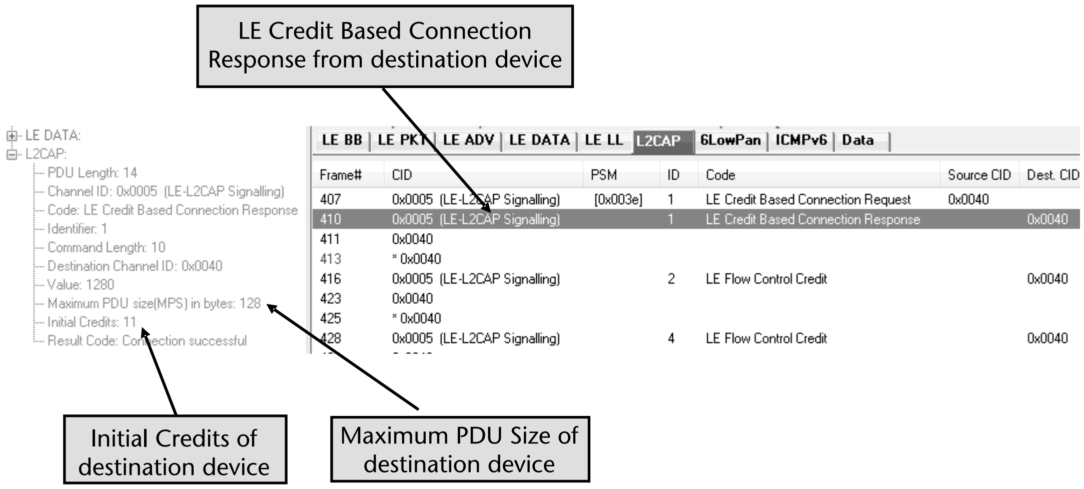
Figure 10.8 Example of L2CAP exchanges.


LE Credit Based Connection Request from source device
Initial Credits of source device
Maximum PDU Size of source device
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | 6LowPan | ICMPv6 | Data

```
LE DATA:
L2CAP:
  PDU Length: 14
  Channel ID: 0x0005 (LE-L2CAP Signalling)
  Code: LE Credit Based Connection Request
  Identifier: 1
  Command Length: 10
  Protocol/Service Multiplexer: [0x003e]
  Source Channel ID: 0x0040
  Value: 65535
  Maximum PDU size(MPS) in bytes: 230
  Initial Credits: 10
```

Frame#	CID	PSM	ID	Code	Source CID	Dest. CID
407	0x0005 (LE-L2CAP Signalling)	[0x003e]	1	LE Credit Based Connection Request	0x0040	
410	0x0005 (LE-L2CAP Signalling)		1	LE Credit Based Connection Response		0x0040
411	0x0040					
413	* 0x0040					
416	0x0005 (LE-L2CAP Signalling)		2	LE Flow Control Credit		0x0040
423	0x0040					
425	* 0x0040					
428	0x0005 (LE-L2CAP Signalling)		4	LE Flow Control Credit		0x0040

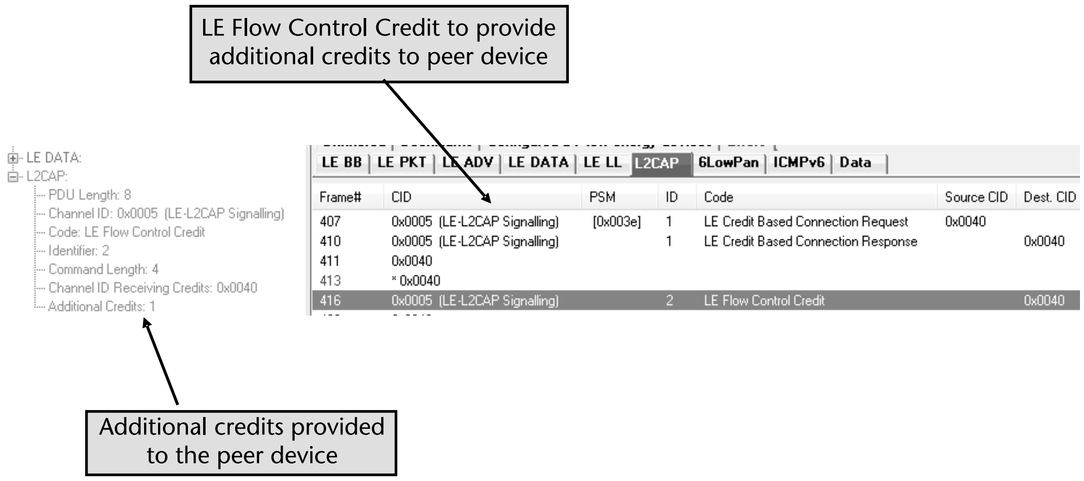
Figure 10.9 LE credit-based connection request.


LE Credit Based Connection Response from destination device
Initial Credits of destination device
Maximum PDU Size of destination device
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | 6LowPan | ICMPv6 | Data

```
⊞- LE DATA:
⊟- L2CAP:
   PDU Length: 14
   Channel ID: 0x0005 (LE-L2CAP Signalling)
   Code: LE Credit Based Connection Response
   Identifier: 1
   Command Length: 10
   Destination Channel ID: 0x0040
   Value: 1280
   Maximum PDU size(MPS) in bytes: 128
   Initial Credits: 11
   Result Code: Connection successful
```

Frame#	CID	PSM	ID	Code	Source CID	Dest. CID
407	0x0005 (LE-L2CAP Signalling)	[0x003e]	1	LE Credit Based Connection Request	0x0040	
410	0x0005 (LE-L2CAP Signalling)		1	LE Credit Based Connection Response		0x0040
411	0x0040					
413	* 0x0040					
416	0x0005 (LE-L2CAP Signalling)		2	LE Flow Control Credit		0x0040
423	0x0040					
425	* 0x0040					
428	0x0005 (LE-L2CAP Signalling)		4	LE Flow Control Credit		0x0040

Figure 10.10 LE credit-based connection response.

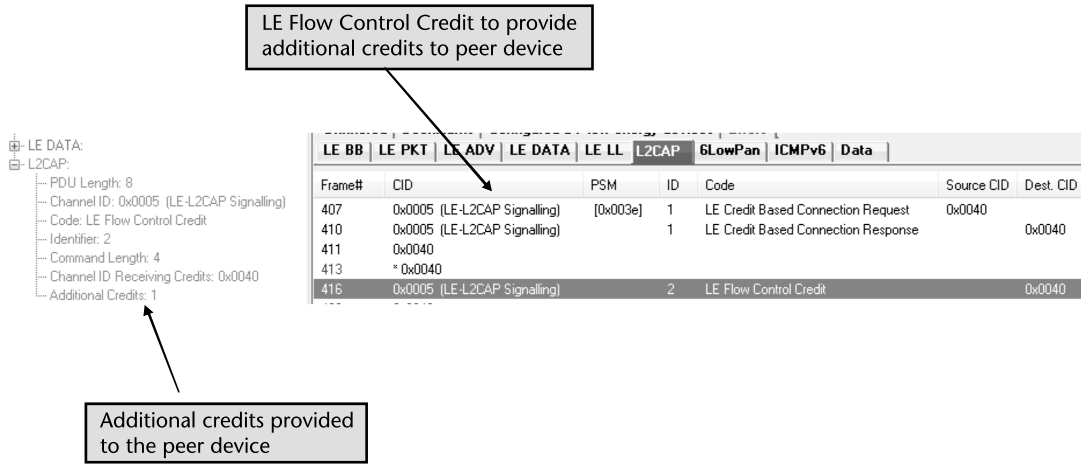

LE Flow Control Credit to provide additional credits to peer device
Additional credits provided to the peer device
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | 6LowPan | ICMPv6 | Data

```
⊞- LE DATA:
⊟- L2CAP:
   PDU Length: 8
   Channel ID: 0x0005 (LE-L2CAP Signalling)
   Code: LE Flow Control Credit
   Identifier: 2
   Command Length: 4
   Channel ID Receiving Credits: 0x0040
   Additional Credits: 1
```

Frame#	CID	PSM	ID	Code	Source CID	Dest. CID
407	0x0005 (LE-L2CAP Signalling)	[0x003e]	1	LE Credit Based Connection Request	0x0040	
410	0x0005 (LE-L2CAP Signalling)		1	LE Credit Based Connection Response		0x0040
411	0x0040					
413	* 0x0040					
416	0x0005 (LE-L2CAP Signalling)		2	LE Flow Control Credit		0x0040

Figure 10.11 LE flow control credit.

- The LE credit-based connection response is sent by the destination device in Figure 10.11. During this response, the destination device provides an initial number of credits (in this example, 11 credits). This means that the source device would be allowed to send up to 11 packets once the connection is established.
- The LE flow control credit is used to send additional credits to the peer device. As show in Figure 10.12, one additional credit is sent to the peer device. This means that the peer device can send one additional packet in addition to the ones that it could send based on the credits that it received in the past.


LE Flow Control Credit to provide additional credits to peer device
Additional credits provided to the peer device
LE BB | LE PKT | LE ADV | LE DATA | LE LL | L2CAP | 6LowPan | ICMPv6 | Data

```
⊞- LE DATA:
⊟- L2CAP:
   ├─ PDU Length: 8
   ├─ Channel ID: 0x0005  (LE-L2CAP Signalling)
   ├─ Code: LE Flow Control Credit
   ├─ Identifier: 2
   ├─ Command Length: 4
   ├─ Channel ID Receiving Credits: 0x0040
   └─ Additional Credits: 1
```

Frame#	CID	PSM	ID	Code	Source CID	Dest. CID
407	0x0005 (LE-L2CAP Signalling)	[0x003e]	1	LE Credit Based Connection Request	0x0040	
410	0x0005 (LE-L2CAP Signalling)		1	LE Credit Based Connection Response		0x0040
411	0x0040					
413	* 0x0040					
416	0x0005 (LE-L2CAP Signalling)		2	LE Flow Control Credit		0x0040

Figure 10.12 Flow control chart.

10.11 Summary

The L2CAP layer provides data services to SMP and ATT protocols. Since there are only two layers on top of it, L2CAP layer use fixed CIDs for signaling leading to a simpler implementation.

In the case of LE, the L2CAP layer is simplified in a major way. It includes only three signaling commands: Command Reject, Connection Parameter Update Request, and Response. Even these commands don't need to be sent in the beginning to start data exchange between two layers. As soon as the link layer connection is established, the upper layers can provide data to L2CAP to send to the remote device.

The next two chapters will focus on Security Manager and Attribute Protocol. These are the two entities which use the services provided by L2CAP.

Bibliography

Bluetooth Core Specification 4.0 http://www.bluetooth.org.