- connSlaveLatency;
- connSupervisionTimeout.

If the parameters are not acceptable to the receiving device, it may respond with either an alternative set of parameters or LL_REJECT_IND_EXT PDU. The sequence diagram for this procedure is shown in Figure 8.30.

8.12.8 LE Ping Procedure

In the networking world, a ping command is used to check the reachability of a remote host and whether or not it can accept and respond to requests. It is a very powerful diagnostic tool and can be used to check connectivity to a remote device and certain other metrics.

The specifications 4.1 introduced the ping procedure in order to verify the presence of a remote link layer. Besides this, it can also be used to check message integrity by requesting the remote ACL layer to send a data packet containing a valid message integrity check (MIC).

This procedure may be initiated by the Master or the Slave by sending the LL_PING_REQ PDU. The remote link layer responds with a LL_PING_RSP PDU. If the remote link layer does not support the ping command, it sends an LL_UNKNOWN_RSP PDU. The sequence diagram for this procedure is shown in Figure 8.31.

8.12.9 Data Length Update Procedure

The data length ppdate procedure is used to inform the remote link layer about the following parameters:

- Maximum receive data channel PDU payload length (connMaxRxOctets);

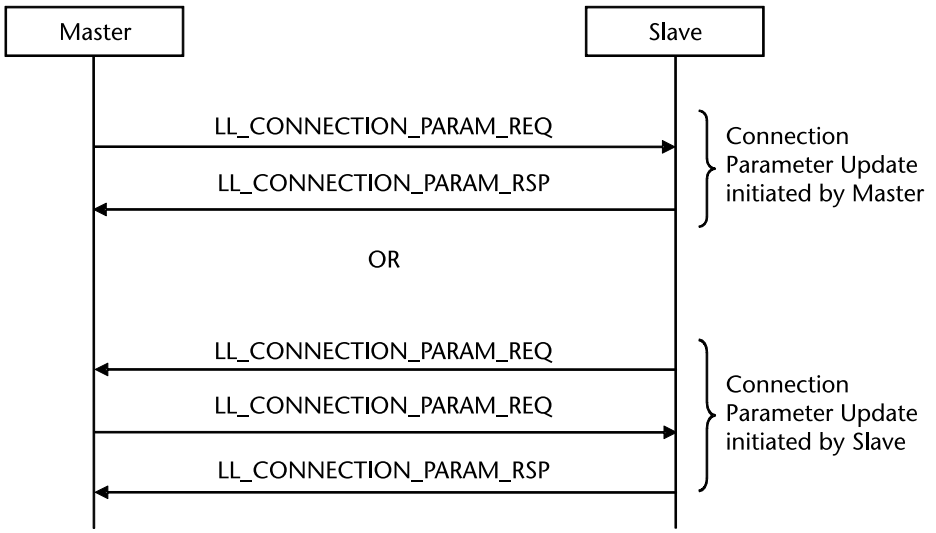

Figure 8.30 Connection parameters request procedure.

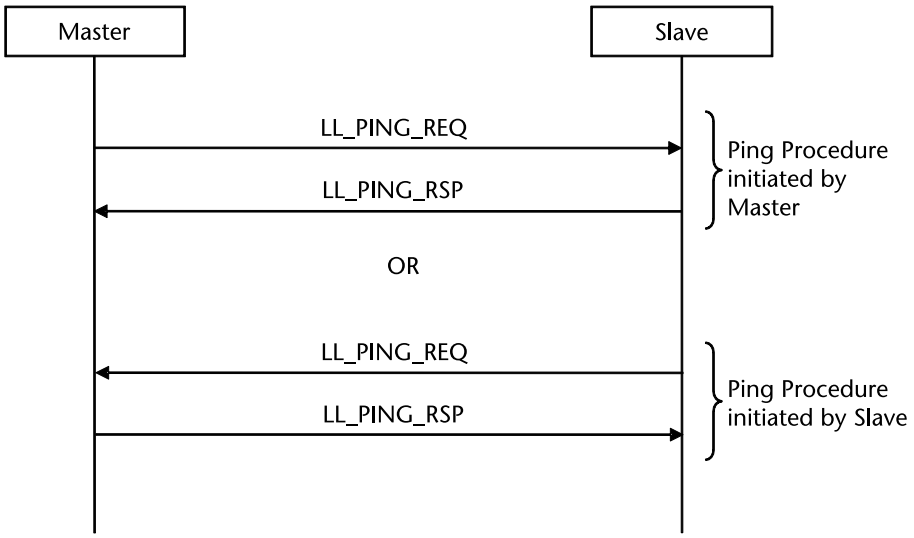

Figure 8.31 LE ping procedure.

- PDU time (connMaxRxTime);
- Maximum transmit data channel PDU payload length (connMaxTxOctets);
- PDU time (connMaxTxTime).

While responding, the remote entity provides its own parameters, which could, for example, have changed because of the request that was just received.

This procedure may be initiated by the Master or Slave and allows the initiating device to inform the remote entity whenever any of these parameters change. For example, if the data buffers are getting full, it may inform the remote entity that the maximum receive data channel PDU payload length has decreased. Once the buffers are freed up, it may increase the maximum receive data channel PDU payload length again. The sequence diagram for this procedure is shown in Figure 8.32.

8.13 Management of Link Layer Procedures

In order to have a smooth interaction between peer link layers, certain rules have been defined regarding timeouts and the handling of collisions. These are explained in the following sections.

8.13.1 Procedure Response Timeout

Since LE is a wireless protocol, it's possible that the peer devices may go out of range at any time and possibly come back in range within a certain time interval. Besides this, it's possible that the remote device may stop responding (i.e., if there is a bug and the device hangs or the battery dies).

The link layer defines a procedure response timeout of 40 seconds. If a response to a link layer PDU is not received within this timeout, the connection is considered to have been lost and the Host is notified of this connection.

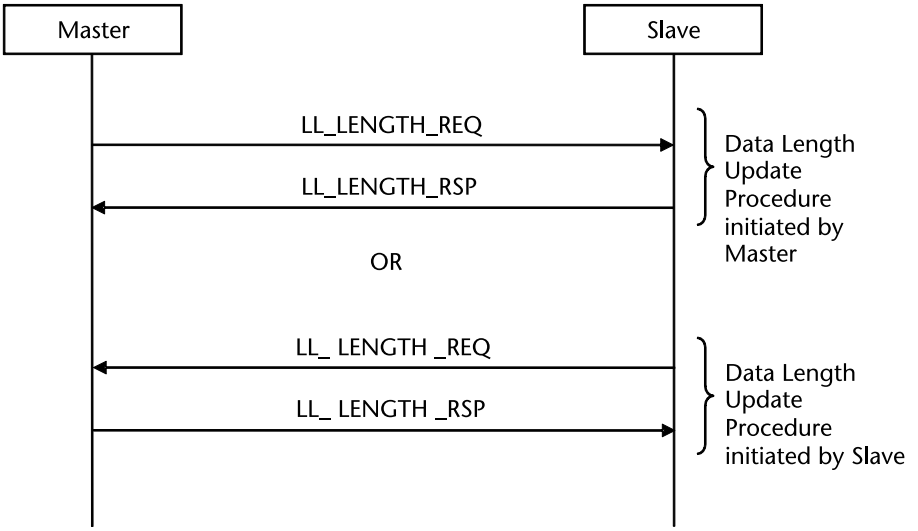

Figure 8.32 Data length update procedure.

Please note that this is quite similar to the Link Supervision Timeout used in BR/EDR to monitor link loss. A Link Supervision Timeout Event is indicated to the Host if no packets are received from the remote side for that duration. The default link supervision timeout is 20 seconds for BR/EDR as compared to 40 seconds timeout used for link layer procedures.

8.13.2 Procedure Collisions

It is possible that the Master and Slave initiate the same procedure or procedures that update the same link layer parameters. In that case, the procedure initiated by the Master gets priority. Therefore, if the Master has initiated or is in the process of initiating a similar procedure as the one initiated by the Slave, the Master will reject the procedure initiated by the Slave.

8.13.3 LE Authenticated Payload Timeout

Specifications 4.1 introduced the support of having a maximum time interval within which an authenticated packet must be received from the peer device. An authenticated packet is a packet containing a valid MIC.

If a packet is not received during this interval and the timeout is about to expire, the link layer can send a ping request (LE_Ping procedure) in order to receive an authenticated packet from the remote side.

The default value of authenticated payload timeout is 30 seconds.

8.14 Link Layer Privacy 1.2

Since many of the LE devices are supposed to be carried or worn by people (like shoes, watch, heart rate sensor, etc.), tracking those devices would allow a person

to be tracked by tracking the transmissions from his or her devices. This could compromise a person's privacy.

Take, for example, a fitness band which is worn on the user's wrist. Whenever it has to transmit data to a mobile phone, it begins advertisement. The mobile phone will receive these advertisement packets, initiate a connection, and transfer the data. These advertisement packets can be picked up by sniffers or other malicious devices to determine the user's location. Thus, tracking the advertisement packets over a period of time could be used as a mechanism with which to track the user.

The privacy feature is used to prevent the tracking of devices over a period of time. This is done by changing the Bluetooth device address frequently. Instead of sending out the real address in advertisement packets, a pseudo-random value that changes over time is inserted. The information on how to resolve this random address to know the real address of the device is known only to the peer device because it is exchanged when the devices are paired. (As explained briefly in Chapter 3, pairing is the process of associating two devices with each other and creating a trust relationship between the two by exchange security keys. It will be covered at length in Chapter 11.) During pairing, identity resolution keys (IRK) are exchanged between the two devices, and only the device that has an IRK can resolve a pseudorandom address to the real address. The pseudorandom addresses change at frequent intervals that are known to both the devices.

This is an optional feature and was introduced in specification 4.0. The processing related to privacy was done by the upper layers of the protocol stack (usually running in the Host).

One of the key enhancements introduced in specifications 4.2 is privacy at the link layer level. This will be described in the following sections.

8.14.1 Address Resolution in the Controller Instead of the Host

Consider the previous example of a user wearing a fitness band. The address of the fitness band keeps on changing in order to maintain privacy. When the mobile phone receives the advertisement packet from the fitness band, the link layer (running in the Controller) can ascertain whether the packet is coming from the fitness band by resolving that address. Therefore, the link layer passes on the information to the Host for only the selected devices instead of all devices. In specifications 4.0, the private addresses were generated and resolved by the Host. In specifications 4.2, the private addresses are generated and resolved by the Controller without involving the Host. The Host provides the Controller with information called device identity information; this allows the Controller to resolve the addresses.

This can lead to a huge power savings in case there are several devices around and the Host is not interested in transmissions from all the other devices.

8.14.1.1 Device Identity and Resolving List

The Host provides the device identity information to the Controller. Thereafter, the Controller can resolve the addresses on its own without any involvement from the Host.

A device identity contains the peer device's identity address and the local and peer's key pair needed to resolve identities. These keys are called identity resolution keys (IRK) and will be discussed in Chapter 11.

The Host and the Controller refer to the peer device by the identity address when communicating with each other. This means that the Host refers to the peer device by providing the device identity address (i.e., while creating an LE connection). If the Controller is able to resolve the peer device, it sends the device identity address in the events (i.e., when providing an LE advertising report event).

A resolving list is a set of device identities for all bonded devices. The Host maintains this list and provides it to the Controller. It may add or remove device identities at any stage using the HCI commands LE_Add_Device_To_Resolving_List and LE_Remove_Device_From_Resolving_List.

There may be cases where the Controller cannot store all the device identities. In such cases, the Host may provide a subset of the resolving list to the controller.

8.14.2 Better Privacy

The private address of the device is used within advertisement packets and is used by the peer device for creating connections. This address changes at periodic intervals. The peer device resolves this address before initiating a connection.

8.14.2.1 Private Address Generation Interval

The link layer uses a timer to generate private addresses at periodic intervals. The private address is generated under the following two conditions: the Link Layer is reset or the timer expires.

Once the new address is generated, the timer is restarted. The specification recommends the timer to be 15 minutes.

8.14.2.2 Privacy in Different States

The link layer uses the private address during advertising, scanning, and initiating state.

Continuing the previous example, when the fitness band wants to create a connection with the mobile phone, it sends its private address in the advertisement packets (ADV_IND or ADV_DIRECT_IND). The mobile phone then resolves this address, and it may send its own private address in the scanning and initiating requests while also using the private address of the fitness band.

8.15 Device Filtering and White List

One of the important power savings mechanisms introduced by LE is device filtering. With the use of device filtering, the link layer can be restricted to respond to only a certain set of devices. This is done through white lists that are maintained by the link layer. This is nothing but the set of devices that the link layer responds to (advertisers, scanners or initiators).

The transmissions from devices that are not in the white list are simply ignored. This reduces the number of transmissions made by the link layer thereby reducing the power consumption of the controller. In addition to this, it also reduces the communication the controller has with the host thereby reducing both the host's and controller's power consumption.

Besides power savings, the white list also eases the host processing in use cases where a host may wish to establish a connection with any one of the devices amongst a list of devices (for example, if those devices provide similar data). At the time of LE connection, the host can specify a list of devices to connect to using white list. The controller connects to one of the available devices. The address to which the connection was established is returned in the Peer_Address parameter of LE_Connection_Complete Event. White List is very useful in such a scenario because the host does not have to repeatedly attempt making connections to each of the devices till the time it is able to successfully make a connection. This eases the host processing and lowers power consumption by reducing the communication between the host and the controller. The white list is initialized to empty by the controller at the time of reset. There are three filter policies that make use of the white lists. These are described below.

8.15.1 Advertising Filter Policy

This policy determines how the link layer of the Advertiser processes scan and connection requests.

There are four possible modes and one of these can be selected by the host by using the HCI command HCI_LE_Set_Advertising_Parameters.

- Process scan and connect requests only from devices in white list.
- Process scan and connect requests from all device (White list not in use). This is the default on reset.
- Process scan request from all devices but connect request from only devices in the white list.
- Process connect request from all devices but scan request from only devices in the white list

This policy is used only when the link layer is not using connectable directed advertising. In the case of connectable directed advertising, the Advertiser accepts the scan or connect request only from the device which it addresses in the advertising events. So this policy is not needed.

8.15.2 Scanner Filter Policy

This policy determines how the Scanner's link layer processes advertising packets. There are two possible modes and one of these can be selected by the host by using the HCI command HCI_LE_Set_Scan_Parameters.

- Process advertising packets only from devices in the white list.

- Process all advertising packets. (White list is not in use). This is the default on reset.

As per specifications version 4.0, a connectable directed advertising packet that did not contain the scanner's address was ignored. This simply meant that all of the packets that were not meant for the scanner were ignored. This is not useful if the scanner is using a resolvable private address because whether the packet is meant for the scanner or not would be known only after the address is resolved.

Specifications version 4.2 provided support for extended scanner filter policies. With this, two new scanner filter policies were added:

- Process advertising packets only from devices in the White List (same as 4.0) and do not ignore a connectable directed advertising packet if the initiator address (InitA field in ADV_DIRECT_IND PDU) is a resolvable private address.
- Process all advertising packets (same as 4.0) and do not ignore a connectable directed advertising packet if the initiator address (InitA field in ADV_DIRECT_IND PDU) is a resolvable private address.

The default on reset is still the same as specifications 4.0—process all advertising packets and the White List is not in use.

This mechanism has strengthened the Link Layer Privacy. Now the scanner can enable the filter policy and both the advertiser and the scanner can use the resolvable private address of the scanner instead of using a public address. The filter policy would enable the resolving of the address before deciding to accept or discard the advertising PDU.

8.15.3 Initiator Filter Policy

This policy determines how the Initiator's link layer processes advertising packets. There are two possible modes and one of these is selected at the time of creation of a connection using the HCI_LE_Create_Connection command.

- Process connectable advertising packets from all devices in the white list.
- Process connectable advertising packets from a specific single device specified by the host. (White list is not in use). This is the default on reset.

8.16 Practical Examples

Figure 8.33 shows an air log capture of the transactions happening between the link layers of two devices to support the Proximity profile. These procedures are carried out after a connection has been established between the two devices.

The sequence of procedures that are carried out is as follows:

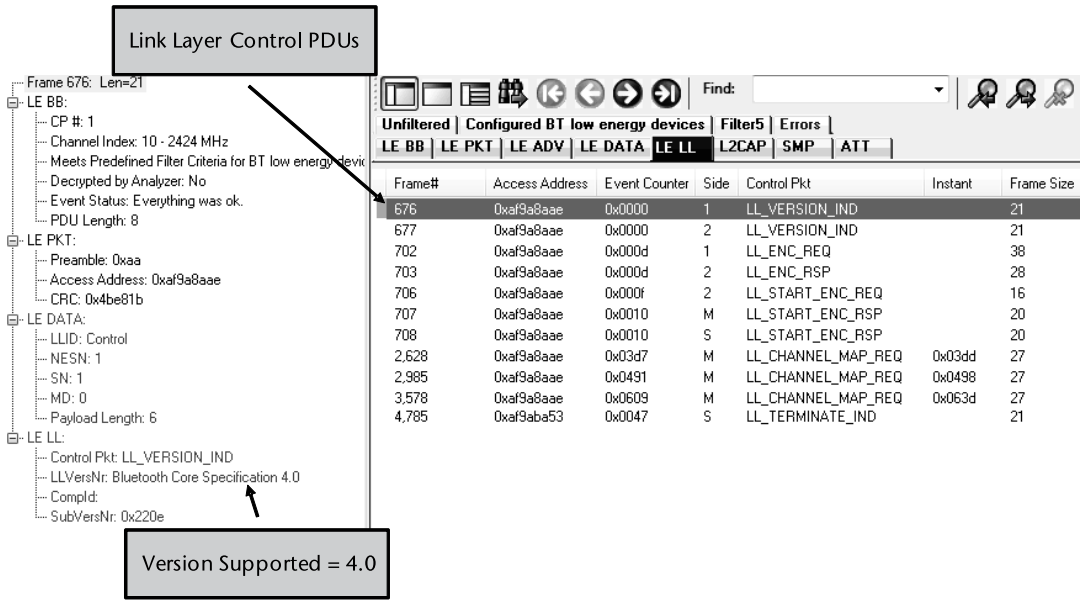

Figure 8.33 Practical example of link layer transactions.

- Frames #676 and #677: The link layer of the Master and Slave exchange information about the versions that they support.
- Frames #702 to #708: These frames are used to start encryption on the link.
- Frames #2628, #2985, #3578: The link layer of the Master provides updated Channel Map to the Slave.
- Frame #4785: The Slave requests the link to be terminated.

8.17 Summary

As the name suggests, the link layer is responsible for the maintenance of the link. This includes establishing the link, selecting the frequencies, supporting different topologies and disconnecting the link. LE uses a very simple architecture for link layer with just five states. It imposes several restrictions in order to make the link layer state machine very simple, thereby reducing both the silicon cost of implementation as well as the power consumption.

LE uses dedicated channels for advertisement and data. The PDUs that can be exchanged between the link layers of the two devices were explained in detail in this chapter.

The next chapter will focus on HCI interface. The HCI interface is used by the upper layers to interface with the link layer.

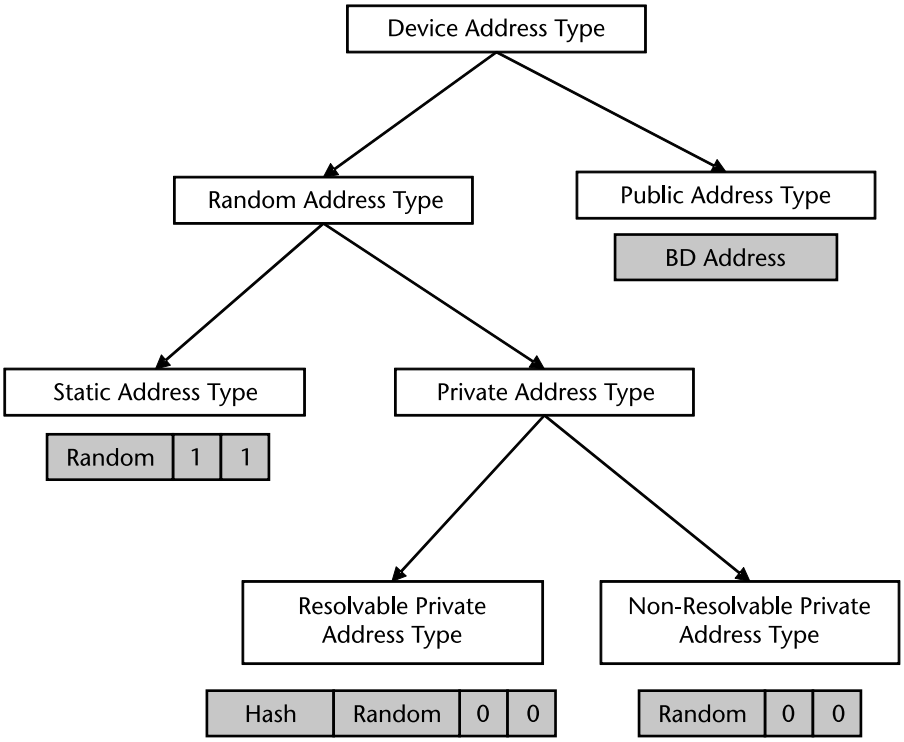

Figure 8.34 Format of various types of device addresses.

Bibliography

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

Bluetooth Low Energy Training and Marketing information from the Bluetooth SIG website. http://www.bluetooth.org.

CHAPTER 9

Host Controller Interface and Commands

9.1 Introduction

The host controller interface (HCI) provides a standard method of communication between the upper and lower layers of the protocol stack. In many implementations, the upper layers generally reside on a host and the lower layers reside on a separate Bluetooth controller chip. The HCI interface provides a communication mechanism between the host and the Bluetooth controller. The position of HCI interface in the LE protocol stack is shown in Figure 9.1.

The HCI was explained in detail in Chapter 3. LE reuses the specification of the HCI layer for BR/EDR and extends it with commands related to Low Energy. The LE controllers implement a reduced set of HCI commands and events that are only related to Low Energy while Dual mode controllers may implement both BR/EDR and LE commands.

Since the HCI layer is reused from the BR/EDR specification, it provides following major advantages:

1. All the code written for the HCI layer in BR/EDR can be reused with LE. This means the code to transmit commands, receive events, transmit, and receive data packets can be completely reused for LE. Only the support for LE specific commands and events needs to be added.
2. If the controller supports dual mode, then it is fully backward compatible with the BR/EDR controller. This means that a BR/EDR controller can be replaced with a dual mode controller without compromising on any existing BR/EDR functionality.
3. No software change is needed when replacing a BR/EDR controller with dual mode controller.

Before going further, it will be useful to read the sections related to Host Controller Interface in Chapter 3 because those are broadly applicable to LE as well. This chapter will provide details on the LE specific parts only.

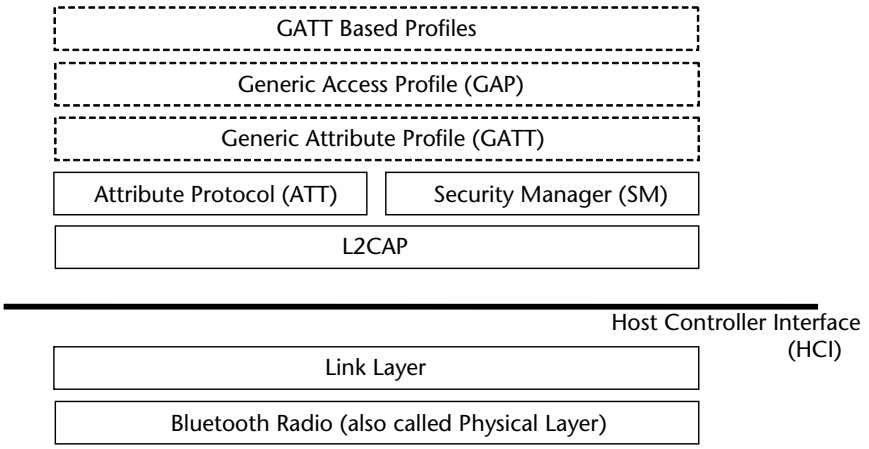

Figure 9.1 HCI in LE protocol stack.

9.1.1 HCI Packet Types

The different packet types that can be exchanged between the host and controller on the HCI interface were explained in Chapter 3. These are described here briefly for ease of reference. The format of HCI Command Packets and HCI ACL Data Packets is the same in the case of LE. The format of HCI Event Packets is slightly enhanced in the case of LE. All event packets are returned with the same event code (LE Meta Event) and a subevent code is used to identify the exact LE event. LE interface does not support Synchronous (SCO/eSCO) packets.

9.1.2 HCI Command Packets

The format of HCI Command packets was explained in Chapter 3. It is shown again in Figure 9.2. It consists of a 16-bit OpCode followed by an 8-bit Parameter Total Length field. The Parameter Total Length field specifies the total length of all parameters that are contained in the remaining packet measured in octets. This is followed by the command parameters. For the LE commands, the OGF (Opcode Group Field) is set to 0x08. The OGF occupies upper 6 bits of the opcode.

9.1.3 HCI Event Packet

The format of HCI Events Packets is shown in Figure 9.3. In the case of LE, all the events are encapsulated into one single event code called the LE Meta Event. The event code of LE Meta Events is 0x3E.

The Subevent code is used to identify the exact LE event that was generated by the controller. The remaining parameters depend on the type of the Subevent code.

9.1.4 HCI ACL Data Packet

The format of HCI ACL data packet is shown in Figure 9.4. All the data fields were explained in Chapter 3 and are valid for LE as well.

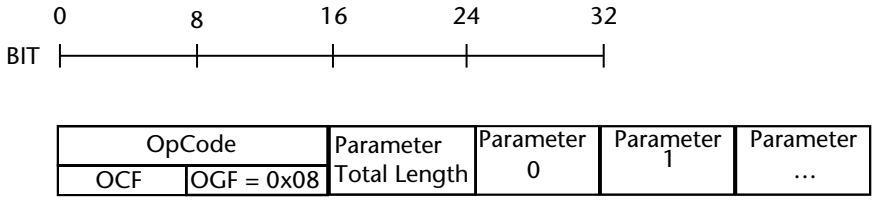

Figure 9.2 HCI command packet format.

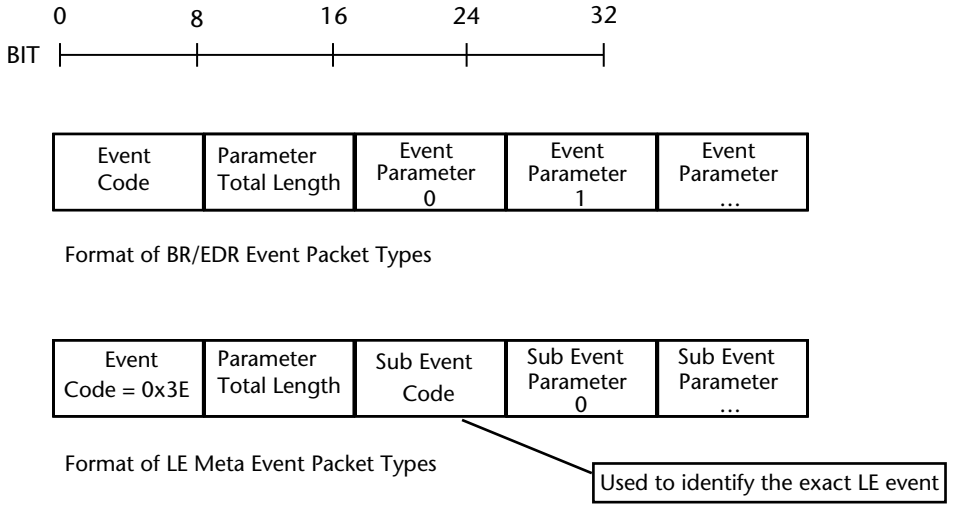

Figure 9.3 HCI event packet and LE meta event packet formats.

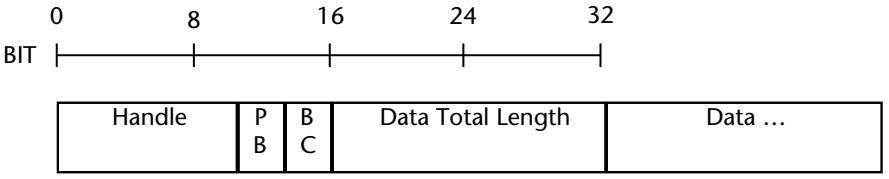

Figure 9.4 HCI ACL data packet.

9.2 HCI Commands and Events

The HCI commands are classified into several groups. The LE related commands in different groups are shown in Table 9.1. The commands that are applicable only to LE start with the HCI_LE prefix. The commands applicable to BR/EDR controllers as well as LE controllers start with HCI prefix. Similarly, the events specific to LE only start with an LE prefix while the events applicable to both BR/EDR and LE controllers do not have any prefix. LE specification tries to reuse the same HCI commands as BR/EDR whereever possible for LE functionality as well.

Note: the HCI commands and events that have been introduced in specifications 4.1 and 4.2 have been highlighted in italics in Table 9.1.

Table 9.1 LE Related HCI Commands and Events

Group	*Commands*	*Events*
Device Setup	HCI_Reset	
Controller Flow Control	HCI_Read_Buffer_Size	Number_Of_Completed_Packets_Event
	HCI_LE_Read_Buffer_Size	
Host Flow Control	HCI_Host_Buffer_Size	Data_Buffer_Overflow_Event
	HCI_Set_Event_Mask	
	HCI_Set_Controller_To_Host_Flow_Control	
	HCI_Host_Number_Of_Completed_Packets	
	HCI_LE_Add_Device_To_White_List	
	HCI_LE_Clear_White_List	
	HCI_LE_Read_White_List_Size	
	HCI_LE_Remove_Device_From_White_List	
	HCI_LE_Set_Event_Mask	
	HCI_LE_Add_Device_To_Resolving_List	
	HCI_LE_Remove_Device_From_Resolving_List	
	HCI_LE_Clear_Resolving_List	
	HCI_LE_Read_Resolving_List_Size	
	HCI_LE_Read_Peer_Resolvable_Address	
	HCI_LE_Read_Local_Resolvable_Address	
	HCI_LE_Set_Address_Resolution_Enable	
Controller Information	HCI_Read_Local_Version_Information	
	HCI_Read_Local_Supported_Commands	
	HCI_Read_Local_Supported_Features	
	HCI_LE_Read_Local_Supported_Features	
	HCI_LE_Read_Supported_States	
	HCI_LE_Read_Maximum_Data_Length	
Remote Information	HCI_Read_Remote_Version_Information	Read_Remote_Version_Information_Complete
	HCI_LE_Read_Remote_Used_Features	LE_Read_Remote_Used_Features_Complete
Controller Configuration	HCI_LE_Set_Advertise_Enable	
	HCI_LE_Set_Advertising_Data	
	HCI_LE_Set_Advertising_Parameters	
	HCI_LE_Set_Random_Address	
	HCI_LE_Set_Scan_Response_Data	
	HCI_Read_LE_Host_Support	
	HCI_Write_LE_Host_Support	
	HCI_LE_Set_Resolvable_Private_Address_Timeout	
Device Discovery	HCI_LE_Set_Scan_Enable	LE_Advertising_Report_Event
	HCI_LE_Set_Scan_Parameters	*LE_Direct_Advertising_Report_Event*
Connection Setup	HCI_Disconnect	Disconnect_Complete_Event
	HCI_LE_Create_Connection_Cancel	LE_Connection_Complete_Event
	HCI_LE_Create_Connection	*LE_Enahanced_Connection_Complete_Event*