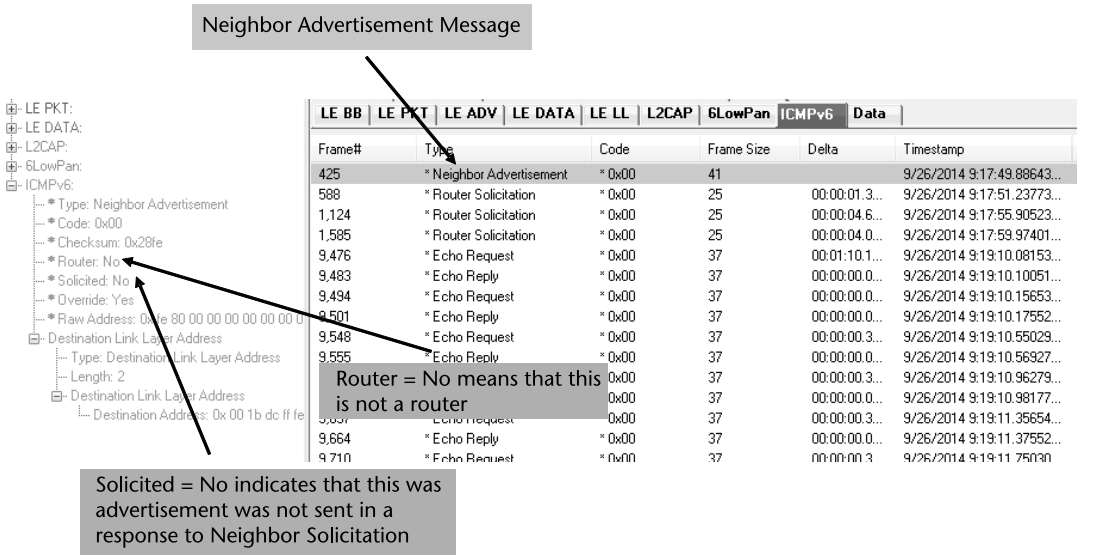

Figure 15.10 IPv6 neighbor advertisement message.

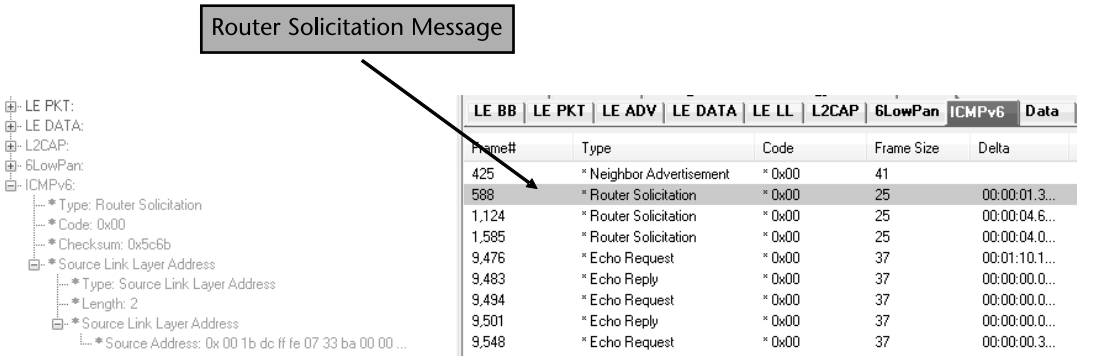

Figure 15.11 Caption IPv6 router solicitation message.

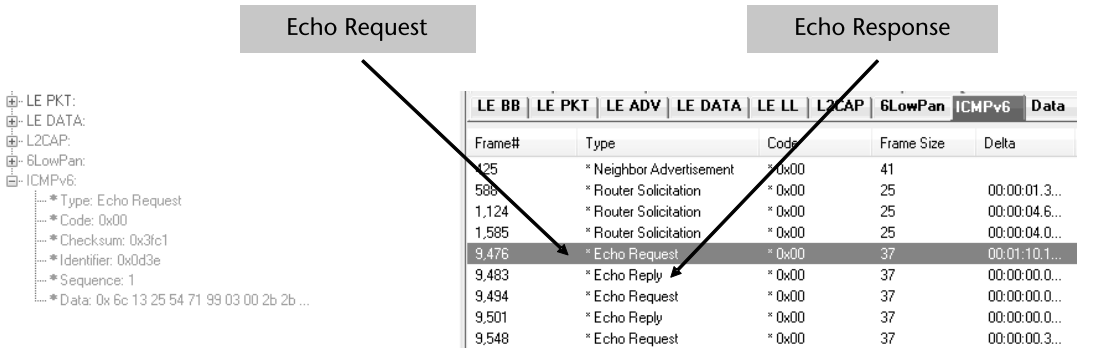

Figure 15.12 Caption IPv6 echo request and echo response messages.

15.17 Other Services and Profiles

There are several other profiles and services that LE specifies for use in different applications. These include the following:

1. Alert Notification Service (ANS): This service exposes information such as count of new alerts, count of unread alerts, different types of alerts etc. This information can be passed, for example, from a mobile phone to a watch so that the user can see this information on the watch itself instead of pulling out the mobile phone from the pocket or briefcase.
2. Alert Notification Profile (ANP): This profile enables a client to receive different types of alerts from a server device.
3. Phone Alert Status Service (PASS): This service exposes the phone alert status and ringer settings. It could be used, for example, to display an alert on a watch when there is an incoming call on the mobile. Besides this, it could also be used to set the phone to silent mode or some other mode from the watch itself.
4. Phone Alert Status Profile (PASP): This profile enables a client to receive the phone alert from a server.
5. HID Service: The HID service exposes data like HID reports which can be used between HID hosts and HID devices. For example, an LE-based keyboard or mouse could use this service to send reports to a computer whenever a user presses a key on the keyboard or moves the mouse.
6. HID over GATT (HOGP): This profile enables HID devices to interact with HID hosts using the GATT profile over an LE transport. (Note that BR/EDR specification also defines an HID profile which defines HID over a BR/EDR transport).

15.18 Practical Examples

This section explains a practical use of the Proximity profile with the help of air logs. It shows the communication between a dual mode device and an LE sensor supporting proximity profile.

Step 1: Discover All Services of the LE sensor

As a first step, the dual mode device discovers the services of the LE sensor using GATT procedure "Discover All Services". This procedure was explained in Chapter 13. The resulting list of services is shown in Figure 15.13. It shows that the LE sensor contains the five services shown in Table 15.1.

Step 2: Discover All Characteristics of the Alert Level Service

Once all the services are discovered, the dual mode device goes on to discover all characteristics of each of those services using the GATT "Discover All Characteristics of a Service" procedure. Figure 15.14 shows the characteristics of the Alert Level Service. It contains one characteristic as follows:

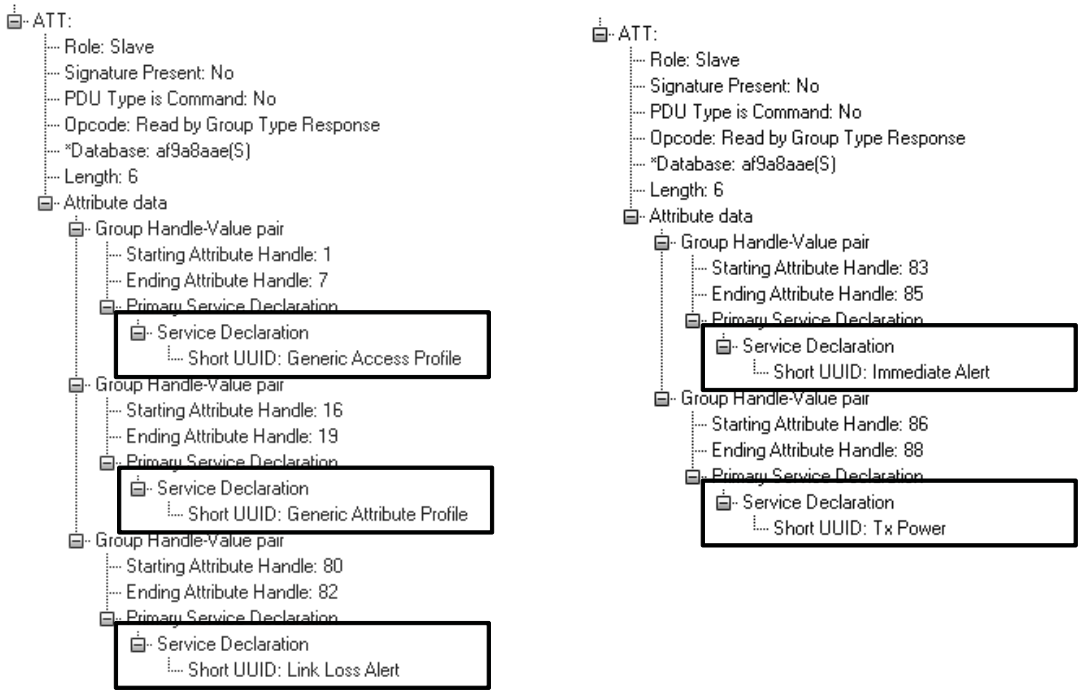

Figure 15.13 Discover all services.

Table 15.1 Services Supported by LE Sensor

S. No.	*Service Name*	*Starting Attribute Handle*	*Ending Attribute Handle*
1	Generic Access Profile	1	7
2	Generic Attribute Profile	16	19
3	Link Loss Alert	80	82
4	Immediate Alert	83	85
5	Tx Power	86	88

- Alert Level Characteristic: Value Handle 85.

Step 3: Read or Write the Characteristics

The final step executed by the profile is to write the alert level characteristic. Figure 15.15 shows two operations:

1. The top part (a) shows writing Alert Level to high. Once this is written, the device starts buzzing.
2. The bottom part (b) shows writing Alert Level to None. Once this is written, the device stops buzzing.

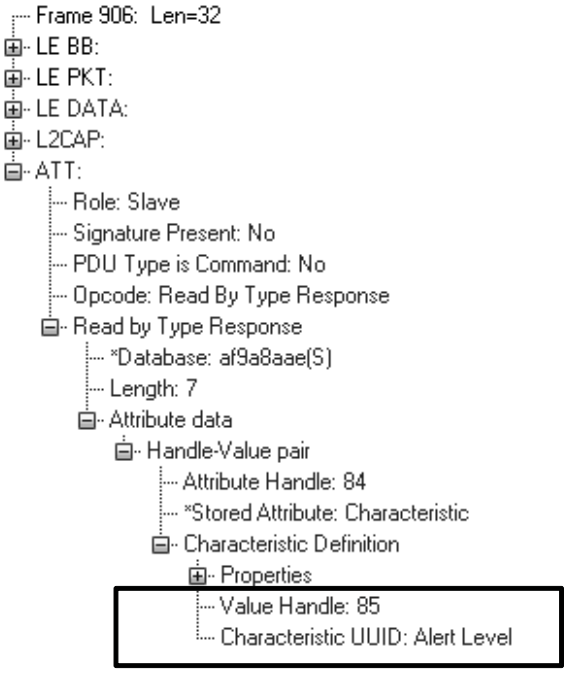

Figure 15.14 Discover all characteristics of the alert level service.

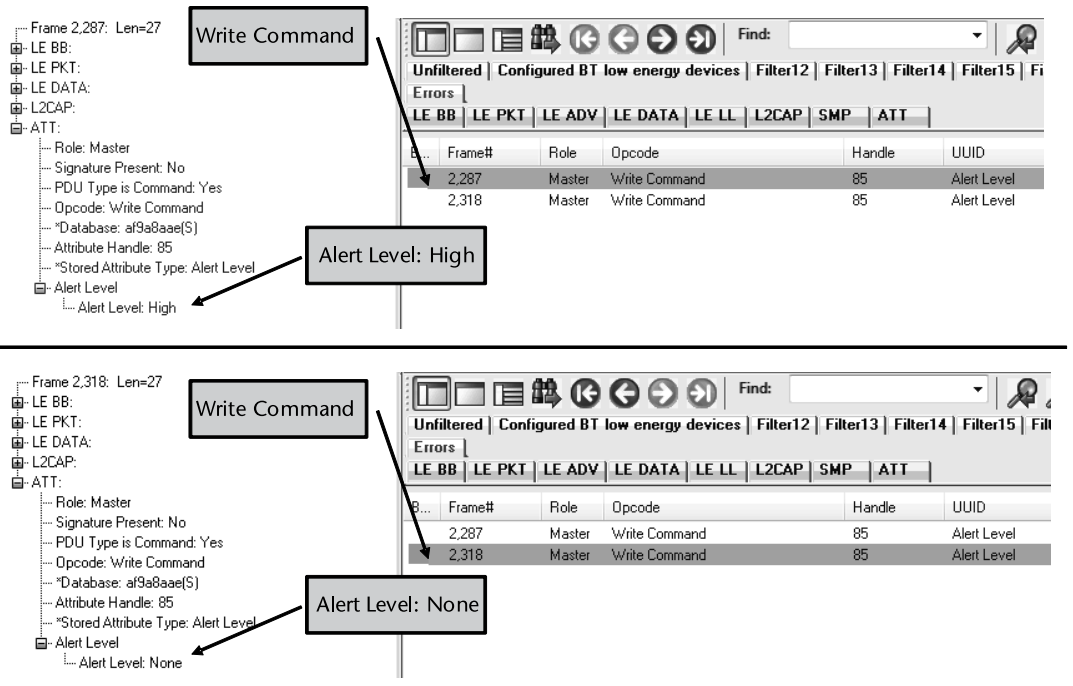

Figure 15.15 Read or write alert level characteristic.

15.19 Summary

The GATT-based LE profiles are much simpler than most of the BR/EDR profiles. As of writing of this book, already 20 profiles have been defined by the Bluetooth SIG. Each profile is expected to cater to some specific use case scenario. The use cases of LE were explained in Chapter 1. This chapter provided details on how those use cases get implemented using the GATT-based LE profiles.

Bibliography

Bluetooth Core Specification 4.0 http://www.bluetooth.org.
Bluetooth Profiles Specifications http://www.bluetooth.org.
Bluetooth Assigned Numbers, https://www.bluetooth.org/assigned-numbers.
IEEE Std 11073-20601™- 2008 Health Informatics, Personal Health Device Communication, Application Profile, Optimized Exchange Protocol, version 1.0 or later.
RFC 7668 IPv6 over Bluetooth Low Energy https://tools.ietf.org/html/rfc7668
RFC 4861 Neighbor Discovery for IP versoin 6 (IPv6) https://tools.ietf.org/html/rfc4861
RFC 6775 Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs) https://tools.ietf.org/html/rfc6775

CHAPTER 16

Developing LE Applications

16.1 Introduction

In Chapter 5, we looked at some practical exercises that can be done on an Ubuntu system using the BlueZ stack to understand the various Bluetooth operations.

In this chapter, we will look at how to build an LE application based on the BlueZ stack. We will start with basic operations like setting up the development environment, enabling the Bluetooth adapter, advertising, and scanning. After that various GATT operations like discovering services, characteristics and reading and writing characteristics will be explained. Finally this chapter will explain the steps needed to make a real world application—an application to find lost keys.

Broadly this chapter will show examples of the various concepts that were explained in previous chapters and how various components come together to make a complete real world application. Besides the various procedures, this chapter will also explain what happens inside the protocol stack when those procedures are executed. In particular, this chapter will go into the transactions happening at the ATT level as well as the HCI level.

The BlueZ stack running in an Ubuntu environment has been selected for the examples due to several reasons:

1. This environment is easy to setup. The Ubuntu operating system can be downloaded and is pretty straightforward to install.
2. This environment is quite inexpensive. Only a couple of PCs are needed along with Bluetooth dongles and application development can be started.
3. The latest versions of BlueZ provide in-built support for LE. This means that the developer can directly focus on writing the application instead of first looking at how to bring up the various protocol stack layers.
4. BlueZ is open source. This means that the source code is available for reference at any time to see how exactly the different components are implemented. It can, for example, be used to see how the *gatttool* interfaces with the BlueZ stack to carry out various GATT operations.
5. Once the developer has made some enhancements to the BlueZ stack it can also be contributed back to the community so that others can also benefit from those.

6. BlueZ provides a powerful *hcidump* tool which can be used for analysis of the various procedures and what happens internally when those procedures are invoked. BlueZ also provides several utilities and sample examples which can be used for reference.
7. Even though the end environment may not be Linux- or BlueZ-based, these examples and sample source code will be very useful in understanding how a typical LE implementation works. This can help in quick ramp-up and result in speedier application development in the target environment.
8. BlueZ is a powerful environment to use as a peer device for testing since it supports both the LE only and dual mode roles. For example, if the developer is developing an LE device (Bluetooth Smart), then the BlueZ environment can be used as a Bluetooth Smart Ready device for testing that LE device. On the other hand if the developer is developing a Bluetooth Smart Ready device or application, the BlueZ environment can be used as a Bluetooth Smart device.

This chapter will also explain some of the tips and tricks that can be useful while developing and debugging an LE application. Some of the tools that can be used for assistance are also explained.

16.2 Ingredients

You will need the following:

1. Two PCs running any flavor of Linux (Preferably a current one to ensure it includes support for LE). For the purpose of examples, Ubuntu 12.10 is used here though any other Linux system will serve the purpose provided it's not too old. These two PCs will be referred to by the following names:
   a. *lepc*: This is the PC that will run the LE side of the sample applications.
   b. *dualpc*: This is the PC that will run the Dual mode side of the sample applications.
2. A couple of LE devices. If you search the internet you may find development kits from some vendors which can be used for developing LE applications. Some LE devices have also started appearing in the market and these are expected to grow rapidly. For the purpose of examples in this chapter, we will use a couple of PTS dongles and attach them via the USB interface to each of the Ubuntu PCs. The PTS dongle is available for purchase from the Bluetooth SIG website.

16.2.1 Installing hcidump

BlueZ comes with a very useful tool called *hcidump* which can be used to find out the packets being exchanged on the HCI interface. By default this tool is not installed on the Ubuntu system. The message shown in Figure 16.1 is received on invoking hcidump if this tool is not installed.

There are two methods for installing hcidump:

```
dualpc# hcidump
The program 'hcidump' is currently not installed. You can install it by
typing:
apt-get install bluez-hcidump
```

Figure 16.1 Invoking hcidump.

Method 1: The following command may be given to download and install it.

```
apt-get install bluez-hcidump
```

This should install hcidump on the system.

Method 2: The source code of the hcidump command may be downloaded from the bluez website (Reference [3]) and built. This method is explained in detail here.

The commands needed to build hcidump are shown in Figure 16.2. Some of the output of the commands is removed for ease of understanding.

```
dualpc# gunzip bluez-hcidump-2.3.tar.gz
dualpc#

dualpc# tar xvf bluez-hcidump-2.3.tar
bluez-hcidump-2.3/
.
.
.
bluez-hcidump-2.3/README
dualpc#

dualpc# cd bluez-hcidump-2.3
dualpc#
dualpc# ./configure
checking for ...
.
.
.
dualpc#

dualpc# make
make -no-print-directory all-am
  CC     src/bpasniff.o
.
.
.
dualpc#

dualpc# make install
test -z ...
.
.
.
dualpc#
```

Figure 16.2 Building hcidump.