Developer Study Guide: An introduction to Bluetooth Low Energy Development

Profile Design

Version:	5.1
Last updated:	14th February 2019

Contents

Revision History

Version	Date	Author	Changes
1.0.0	1st May 2013	Matchbox	Initial version
2.0.0	1st September 2014	Martin Woolley, Bluetooth SIG	Replaced Button Click custom service with the Proximity Monitoring Service. Introduced the Time Monitoring Service. Described implementation of link loss, link loss alert level change and immediate alert use cases and included Arduino code.
2.0.4	2nd February 2015	Martin Woolley, Bluetooth SIG	Fixed some minor issues with the contents of this document; unpackaging instructions incorrectly assumed the Arduino BLE library would be in a zip file, which it is not. The circuit diagram for the suggested circuit had an error.
3.0.0	7th July 2016	Martin Woolley, Bluetooth SIG	Replaced Arduino Uno with Arduino/Genuino 101 Introduced use of Bluetooth Developer Studio in the design of the lab profile, the generation of skeleton code and testing of the peripheral device.
3.0.1	18th November	Martin Woolley, Bluetooth SIG	Added further information on setting up the Arduino development environment.
3.1.0	3rd October	Martin Woolley, Bluetooth SIG	Android code retired to the legacy folder and designated 'Android 4'.

			New Android lab and solution based on Android Studio instead of Eclipse and the newer Android 5+ APIs created.
3.2.0	16th December 2016	Martin Woolley, Bluetooth SIG	iOS Objective-C resources retired and replaced with new lab and solution written in Swift.
4.0.0	7th July 2017	Martin Woolley, Bluetooth SIG	Added a new lab, with associated solution source code, which explains how to use the Apache Cordova SDK to create a cross platform mobile application which uses Bluetooth Low Energy.
5.0.0	4th December 2017	Martin Woolley Bluetooth SIG	Arduino Primo now supported as well as the “end of life” Arduino 101. Added a thermistor to the electronics circuit and the Health Thermometer service to the Bluetooth profile. Indications used to deliver periodic temperature measurements. Split Arduino lab document into three documents; LE Theory, Profile Design and Profile Implementation and Testing - Arduino.
5.0.2	15th June 2018	Martin Woolley Bluetooth SIG	Removed use of Bluetooth Developer Studio.
5.0.3	17th December 2018	Martin Woolley Bluetooth SIG	Name changed to “Developer Study Guide: An introduction to Bluetooth Low Energy Development”

5.1.0	14th February 2019	Martin Woolley Bluetooth SIG	Added Zephyr to the list of server devices and associated labs. Included depiction of the final, custom profile at the end of this document.

Exercise 3 – Designing the Bluetooth Profile

Requirements

Before we can start designing a profile, we need some requirements to design for. The following is a list of behaviours we want our device to be able to exhibit or make possible in a client application connected to it.

Table 1 - Requirements

	Feature	Expected Behaviour
1	Peripheral device allows a maximum of one client to connect to it.	A client device like a smartphone can connect to a peripheral device like an Arduino using Bluetooth low energy.
2	Client can set an alert level of 0, 1 or 2 to be used by the peripheral device when indicating that the Bluetooth link has been lost.	The peripheral device connected circuit must flash a corresponding LED (0=green, 1=yellow, 2=red) four times as an acknowledgement of having received the new alert level from the smartphone and the selected alert value must be stored by the peripheral device for subsequent use in link loss situations.
3	Client must be able to indicate to the peripheral device that it should make a noise and flash its lights so that it can be easily located if lost or hidden.	Client will send the peripheral device an alert level value of 0, 1 or 2 and the peripheral should respond as follows, depending on the alert level value: 0 - flash all 3 LEDs in unison 3 times. buzzer is silent. 1 - flash all 3 LEDs and beep 4 times. 2 - flash all 3 LEDs and beep 5 times.
4	Proximity monitoring	The connected client application will track its distance from the peripheral device using the received signal strength indicator (RSSI) and classify it as near (green), middle distance (yellow) or far away (red). The peripheral device must allow the client to communicate its proximity classification and signal strength as measured at the client. The peripheral device's LED of the colour corresponding to the proximity classification should be illuminated.
5	Link Lost / Disconnected	If the connection between client and peripheral device is lost then if the alert level set in feature (2) is greater than 0, the peripheral device circuit should make a noise and flash all LEDs for an extended period of time. The duration of the alert made should be 30 seconds if the alert level set in feature (2) is 2 or 15 seconds if it was set to 1.
6	Temperature monitoring	The peripheral device must be able to measure the ambient temperature once per second and to communicate measurements to the connected client. The client must be able to enable or disable this behaviour.

Terminology Reminder

Our completed Bluetooth system will consist of two major parts; a device (#1) running some software which will display proximity information and allow users to initiate the flashing of lights and sounding of the buzzer in our custom electronic circuit and a device (#2) which is connected electrically to that circuit and which controls its state based on communication it receives over Bluetooth from the other device.

Device #1 will usually be referred to as a client or sometimes as a Central mode device. It will often be an application running on a smartphone but could be implemented some other way, such as in a web browser.

Device #2 will sometimes be referred to as a peripheral device, sometimes as a server and sometimes as a GAP Peripheral. Which is used will depend on the context. They all refer to the same thing and are all equally valid in a given context.

“Client” and “Server” come from GATT. “Peripheral” and “Central” come from GAP. “Peripheral device” is informal. Re-read the LE Basic Theory guide if any of this is not making sense.

Profile Design

So, now we have an understanding of the requirements our peripheral device must satisfy, we can consider what its Bluetooth profile could look like.

Standard vs Custom

A fundamental question we should answer at each step is whether or not there is already a standard service, characteristic or descriptor defined by the Bluetooth SIG which will meet our requirement. If there is, we should use it. If there is not we should consider the availability and suitability of custom services/characteristics/descriptors designed and shared by others. If none exist, we need to design our own custom profile items.

With our lab requirements, the good news is that most of them will be well serviced using a standard profile called the Proximity Profile. Look for Proximity Profile (search for “PXP”) on this page:

https://www.bluetooth.com/specifications/adopted-specifications

and download the PDF specification. Don’t worry, there’s no need to read the whole specification but a glance at what it contains will give you an appreciation of what profile specifications look like and in particular, Section 2 defines device roles and the GATT services that are involved. The definition of the Proximity Profile looks like this, comprising three GATT services as shown:

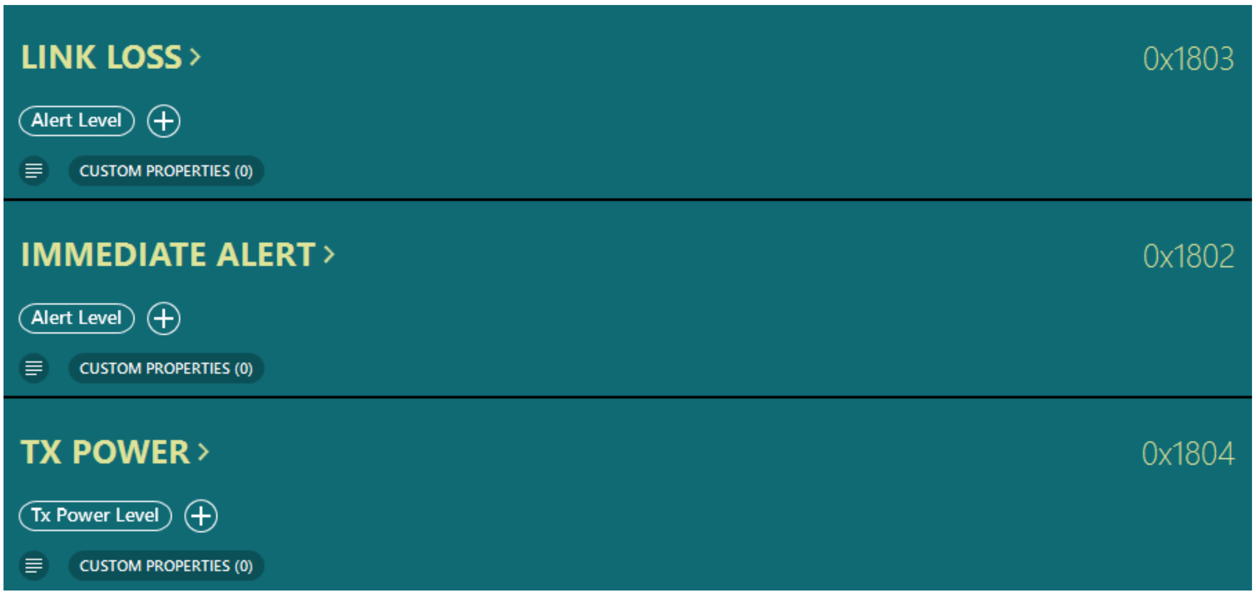

LINK LOSS >
0x1803
Alert Level
CUSTOM PROPERTIES (0)
IMMEDIATE ALERT >
0x1802
Alert Level
CUSTOM PROPERTIES (0)
TX POWER >
0x1804
Tx Power Level
CUSTOM PROPERTIES (0)

Figure 6 – The Proximity Profile GATT services and characteristics

Our peripheral device will play the part of a Proximity Reporter. Let's review the services and assess how they help us to meet our requirements. Note that you'll find the full specification for each of these services at the same page from which the Proximity Profile specification was linked if you'd like to review any of them.

Service	Description	Requirements
Link Loss Service	The specification says "When this service is instantiated in a device and the connection is lost without any prior warning, the device shall start alerting to the current link loss alert level".	Supports requirement (2) and (5). Custom code will be needed to control the lights and buzzer in response to link loss events.
Immediate Alert Service	The specification says "When the Alert Level characteristic is written the device shall start alerting to the written alert level."	Supports requirement (3). Custom code will be needed to control the lights and buzzer in response to the connected client writing to this service's Alert Level characteristic.
TX Power Service	"The specification says "The Tx Power Level characteristic returns the current transmit power level when read".	Useful in calculating an estimate of the distance from the Proximity Reporter device, using a path loss formula but probably not useful in meeting the requirements of this lab.

Implementing the Proximity Profile in an appropriate way will allow us to immediately meet requirements (2), (3) and (5). We'll make our peripheral device a GAP Peripheral meaning that it will need to advertise and be discovered by a scanning smartphone which will act as a GAP Central. Since the peripheral device will be a GAP Peripheral, this means requirement (1) is automatically met.

For requirement (4) Proximity Monitoring, we need some way of being able to communicate to the peripheral device the RSSI and 'proximity band' (green|yellow|red) as determined by the client. This is distinct data and therefore warrants new characteristics in a new service. Have a think about what you think a new service, introduced to meet this requirement could look like. Our solution is on the next page so don’t scroll down yet! Think about its characteristic or characterstics, how the value part of any characteristic would break down into one or more fields, about data types and operations. Then go to the next page and see how we approached this.

Here’s the design we came up with to satisfy requirement (4):

Service: Proximity Monitoring

Characteristic: Client Proximity

Field: uint8 client_proximity_band

Field: uint8 rssi_at_client

Properties: Write Without Response

To meet requirement (4) therefore, the idea is to add a custom service to the Proximity Profile called the Proximity Monitoring service. It will contain a single characteristic called Client Proximity, whose value part will break down into two unsigned, single octet fields, the first of which will contain the proximity band as reported by the client (1=near, 2=middle distance, 3=far) and the second of which will contain the RSSI value measured by the client. Properties of the Client Proximity characteristic indicate that it supports the Write Without Response operation and this can be used by the client to send proximity monitoring data to the peripheral device.

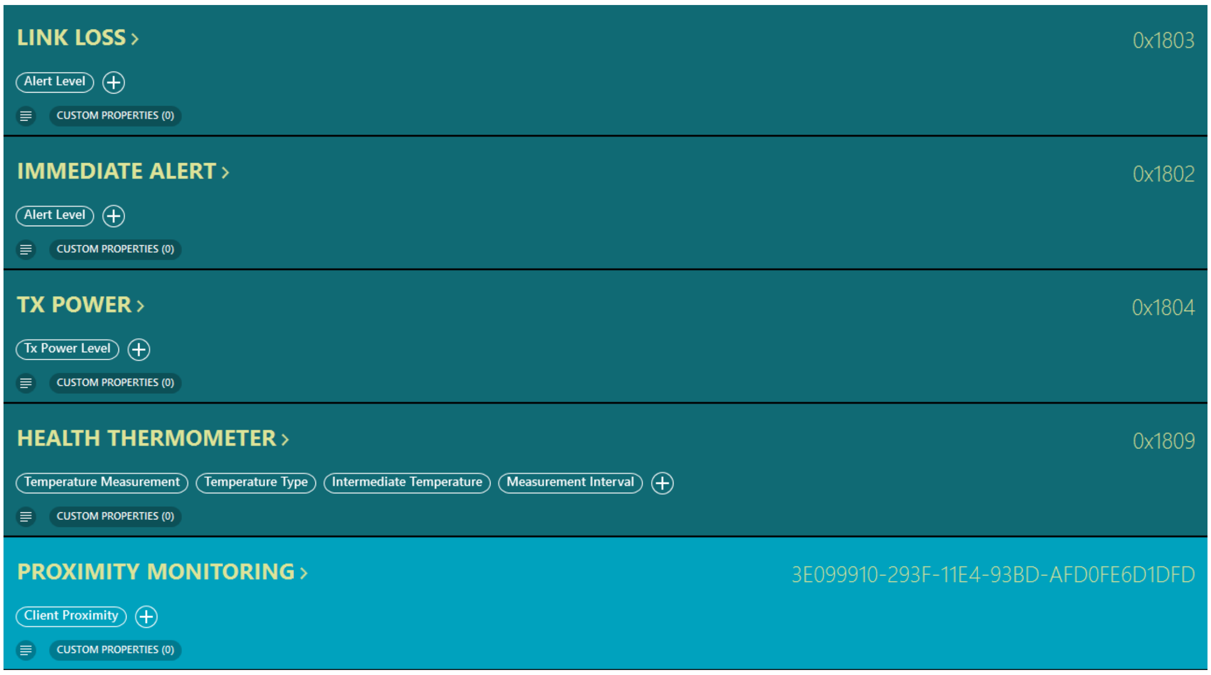
For requirement (6) Temperature Monitoring, you may have noticed that amongst the adopted specifications on the web page referenced above, there’s a service called the Health Thermometer service. The service includes a characteristic called Temperature Measurement and it’s ideal for our purposes, supporting both Celsius and Fahrenheit temperature measurements. The Temperature Measurement characteristic can transmit temperature data to a connected client using *indications*. Indications are like notifications in that they are sent by the GATT server to the connected GATT client, but notifications do not require the client to send back a message confirming receipt of the data whereas indications do. The server cannot send another indication relating to a given characteristic while it is waiting for a response to a previous one, although it may time out and give up waiting.

UUIDs

Services, characteristics and descriptors all need UUIDs to uniquely identify their type. For the adopted items, their UUIDs have already been allocated by the Bluetooth SIG. For custom items, we’ll manually allocate their UUID values now. You could use any tool you like (e.g. Linux can generate UUIDs from the command line) but to ensure all parts of the Bluetooth LE Developer Study Guide work together and keep things simple for you, here are the UUID values we’ll be using:

Proximity Monitoring service	3E09**9910**-293F-11E4-93BD-AFD0FE6D1DFD
Client Proximity characteristic	3E09**9911**-293F-11E4-93BD-AFD0FE6D1DFD

That’s it! You’ve designed a custom profile for your peripheral device! It should contain the services and characteristics shown in Figure 7, below.



LINK LOSS › 0x1803

Alert Level

CUSTOM PROPERTIES (0)

IMMEDIATE ALERT › 0x1802

Alert Level

CUSTOM PROPERTIES (0)

TX POWER › 0x1804

Tx Power Level

CUSTOM PROPERTIES (0)

HEALTH THERMOMETER › 0x1809

Temperature Measurement | Temperature Type | Intermediate Temperature | Measurement Interval

CUSTOM PROPERTIES (0)

PROXIMITY MONITORING › 3E099910-293F-11E4-93BD-AFD0FE6D1DFD

Client Proximity

CUSTOM PROPERTIES (0)

Figure 17 - The final profile design