How to Implement a Bluetooth Low Energy Proximity Profile Device and Application

February 23, 2012

Abstract

This paper provides information about Bluetooth Low Energy support for Windows 8 Beta. It provides guidelines to create a device implementation of the Proximity Profile which is compatible with the Windows implementation and provides high level guidance about how to create a proximity application. It assumes that the reader is familiar with the Windows Bluetooth Low Energy Functions and the Bluetooth Low Energy specification for the Proximity Profile.

This information applies to the following operating systems:
    Windows 8 Consumer Preview

References and resources discussed here are listed at the end of this paper.

The current version of this paper is maintained on the Web at:
    How to Implement a Bluetooth Low Energy Proximity Profile Device and Application

Disclaimer: This document is provided “as-is”. Information and views expressed in this document, including URL and other Internet website references, may change without notice. Some information relates to pre-released product which may be substantially modified before it’s commercially released. Microsoft makes no warranties, express or implied, with respect to the information provided here. You bear the risk of using it. Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2012 Microsoft. All rights reserved.
Document History

<table>
<thead>
<tr>
<th>Date</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 23, 2012</td>
<td>First publication</td>
</tr>
</tbody>
</table>

Contents

Introduction .................................................................................................................. 3
Bluetooth Low Energy .................................................................................................... 3
Proximity Profile ......................................................................................................... 4
  Proximity Reporter ................................................................................................. 4
  Proximity Monitor ................................................................................................. 4
Windows 8 support for Bluetooth Low Energy .............................................................. 5
  Power Efficiency Implications ................................................................................. 5
Implementation Details .............................................................................................. 6
  Connection Establishment ....................................................................................... 6
Path Loss Detection ..................................................................................................... 6
  Monitoring the connection state ........................................................................... 6
Glossary ....................................................................................................................... 8
Resources ..................................................................................................................... 8
Introduction

Windows 8 introduces support for the Bluetooth Low Energy technology. One of the described use cases for this technology is for proximity detection. This paper describes the specifics of this profile, the specifics of the Windows 8 support for Bluetooth Low Energy and how a developer can implement a proximity scenario using this technology on Windows 8.

Bluetooth Low Energy

The Bluetooth Low Energy introduces a new physical layer that shares the same frequency space as Bluetooth Basic Rate. Profiles that are developed on this technology are organized into what’s called the Generic Attribute Profile (or GATT).

Each profile defines the use of one or more services to create a use case or scenario. Compliant service implementations are constructed from characteristics organized in a way that conforms to the established schema defined on the Bluetooth Special Interest Group developer website.

http://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

The diagram below illustrates the way objects are structured inside a typical GATT service.

![GATT Service Declaration Diagram](image-url)

*Figure 1 - GATT Service Declaration*
Proximity Profile

The Proximity Profile defines 2 roles with the intent to allow devices to detect their proximity. The 2 roles are called

- The Proximity Reporter
- The Proximity Monitor

![Proximity Monitor and Proximity Reporter Diagram](image.png)

**Figure 2 - Role Relationship (Source: Figure 2.1 from the Bluetooth Proximity Profile Specification)**

The following sections describe the 2 roles in more details.

**Proximity Reporter**

The proximity reporter is required to be a GATT Server. It support the following GATT Services:

- Link Loss Service (mandatory)
- Immediate Alert Service (optional)
- Tx Power Service (optional)

**Proximity Monitor**

The proximity Monitor is the GATT client. It should create and maintain a connection to the proximity Reporter as well as monitor the Radio Signal Strength Information (or RSSI) of the connection to calculate the signal’s path loss. If the optional “Tx Power Service” is available on the Proximity Reporter, it can also use this additional information to normalize the RSSI value by subtracting the RSSI from the Tx Power Level.
Windows 8 support for Bluetooth Low Energy

When a Bluetooth Low Energy device is paired with a Windows 8 machine, the device becomes part of the system and Windows will provide Device Objects to represent both the device and the primary services reported by the device.

![Device Object structure](image)

**Figure 3 - Device Object structure of the Windows 8 Bluetooth Low Energy Implementation**

Each device and their primary services are represented as Device Objects in the system and these Device Objects can be queried and managed using the Device Installation Functions (such as SetupDiEnumDeviceInfo, SetupDiGetDeviceProperty, etc.)


In addition to the standard device functions, Windows 8 provides a set of APIs which allows for the development of Bluetooth GATT client applications. These APIs are documented here


These functions allows for the enumeration of Services and their objects (included services, characteristics and their descriptors) as well as read and write capabilities.

**Power Efficiency Implications**

The Windows 8 support for Bluetooth Low Energy has a strong focus on power efficiency for the platform. This includes power usage minimization for the local Bluetooth radio adapter as well as ensuring a maximum CPU idle resiliency. For this purpose, the following rules are defined:
1. Windows will not automatically connect to a device unless an application has registered a CharacteristicValueChangedEvent via a call to the BluetoothGATTRegisterEvent function.

2. If any other Bluetooth Low Energy API is called without specifying the \texttt{BLUETOOTH\_GATT\_FLAG\_FORCE\_READ\_FROM\_CACHE} and the device is not already connected, Windows will enter a fast connection scanning mode for a short period of time in an attempt to establish a connection to the device.

3. Windows does not actively monitor the RSSI value of the link by polling the local Bluetooth radio controller.

**Implementation Details**

Provided the implications of the power efficient implementation, specific requirements are put on device implementations to ensure that Windows implementation can be compatible with the device implementation.

The sections below examine the device side requirements to work around this power efficiency implication as well as describe a technique by which the connection state can be monitored.

**Connection Establishment**

Windows 8 will automatically connect to a device when an application has a CharacteristicValueChangedEvent registered via the BluetoothGATTRegisterEvent function. However, the basic definition of the services included in the Proximity Profile does not contain any indicatable or notifiable characteristics. A device can add a service that contains an indicatable or notifiable characteristic to the services included in the Proximity Profile. This means that a proximity device must support at least one Indicatable or Notifiable characteristic value and an application must register a CharacteristicValueChangedEvent via a call to the BluetoothGATTRegisterEvent API for the connection to be automatically established.

**Path Loss Detection**

For power efficiency reasons as described above, Windows 8 does not expose the RSSI value of Bluetooth connections. As a result, the application cannot use the RSSI value to calculate the connection path loss. It is recommended that the device tie its proximity to the link loss event.

**Monitoring the connection state**

An application can monitor the connection state of LE devices in a similar way to BR/EDR connection. This can be achieved by monitoring device change notification from the Bluetooth radio. This can be done via the RegisterDeviceNotification API.

Specifically, an application can listen for GUID_BLUETOOTH_HCI_EVENT and look at the connectionType member of the BTH_HCI_EVENT_INFO structure that is present in the event buffer. The code sample below illustrates a Windows Message handler procedure that looks for Low Energy connection events.

```c
LRESULT
CALLBACK
WndProc(
    HWND Hwnd,
    UINT Msg,
    WPARAM Wparam,
    LPARAM Lparam
)
{
    switch (Msg) {

    case WM_DEVICECHANGE:

        ULONG eventCode = (ULONG)Wparam;
        PDEV_BROADCAST_HANDLE pHdrHandle =
            (PDEV_BROADCAST_HANDLE)Lparam;

        switch (eventCode){
            case DBT_CUSTOMEVENT:
                if (pHdrHandle->dbch_eventguid ==
                    GUID_BLUETOOTH_HCI_EVENT) {

                    PBTH_HCI_EVENT_INFO pCxnEvent =
                        (PBTH_HCI_EVENT_INFO)pHdrHandle->dbch_data;

                    if (HCI_CONNECTION_TYPE_LE ==
                        pCxnEvent->connectionType) {

                        printf("LE device [%04x%08x] is [%s]\n",
                            GET_NAP(pCxnEvent->bthAddress),
                            GET_SAP(pCxnEvent->bthAddress),
                            pCxnEvent->connected ?
                                "Connected" : "Disconnected");
                    }
                }
```

February 23, 2012
© 2012 Microsoft. All rights reserved.
}  
  break;
}  
break;
}  
return DefWindowProc(Hwnd, Msg, Wparam, Lparam);
}  

Note that the constant and structures relevant to Bluetooth used in this example in present in bthdef.h.

Glossary

This section contains definitions for terms used throughout this document.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ATT</td>
<td>Attribute protocol</td>
</tr>
<tr>
<td>GATT</td>
<td>Generic Attribute Profile</td>
</tr>
<tr>
<td>SIG</td>
<td>Bluetooth Special Interest Group</td>
</tr>
<tr>
<td>Tx</td>
<td>Transmission</td>
</tr>
</tbody>
</table>

Resources

**Bluetooth Low Energy Functions**


**Bluetooth SIG Developer Portal**

http://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx

**Bluetooth SIG Proximity Profile Specification**


**Device Installation Functions**

RegisterDeviceNotification Function