QPP Integration Guide for Android

Version 0.1

Table of Contents

1. Introduction

The QPP (Quintic Private Profile) is used to transfer the raw data between BLE devices.
The libQBlueQPP library acts as QPP client role, which is used by application to transfer and receive the raw data between BLE devices.

Features:

- Transmit free raw data between BLE devices. Single free raw data package maximum length is 20bytes, minimal is 1byte.

2. Overview

The QPP client diagram consists of three parts:

App Layer:

- Send connection requests to BluetoothGatt, and configure API layer.
- Send data to API layer.
- Receive data from API layer.

API Layer:

- Receive data from App layer and deliver the data received to BluetoothGatt.
- Receive data from BluetoothGatt and deliver the data received to App layer.

BluetoothGatt Layer:

- Receive request from API layer.
- Update value to API layer.

The QPP client diagram is shown in Figure 1

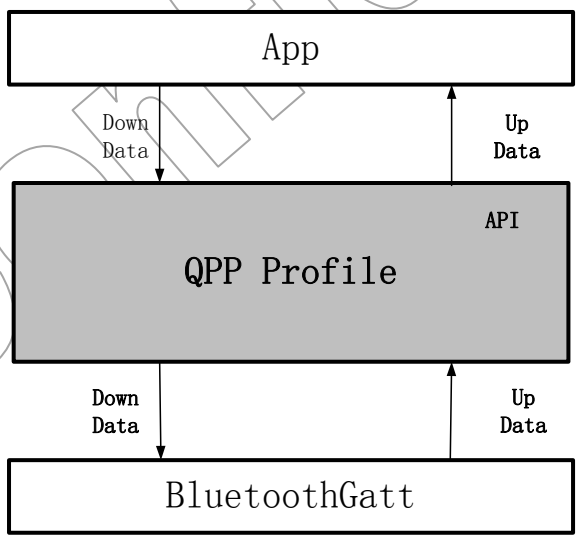

Figure 1 QPP Client Diagram

3. Flowchart

The QPP client general flowchart is the following:

- Scan BLE devices around.
- Establish a connection with the device which is built-in QPP profile server.
- Discover services and characteristics.
- Register user's special UUIDs (including QPP service UUID and write characteristic UUID), here you'd call the method: qppEnable.
- User receives data in the onQppReceiveData function, or sends data by the QppSendData function.

QPP TX flowchart is shown in Figure 2:

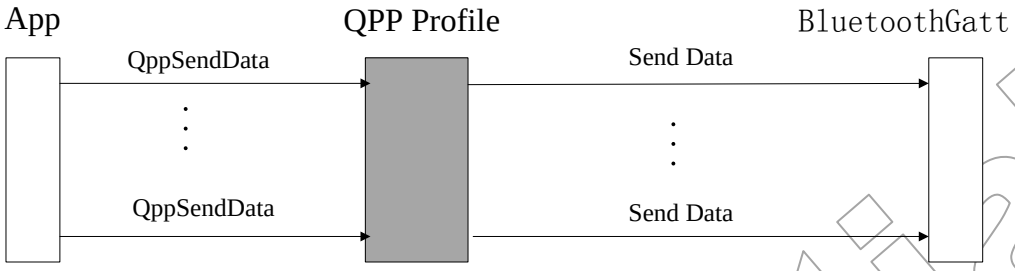

Figure 2 QPP Client TX flowchart

QPP RX flowchart is shown in Figure 3:

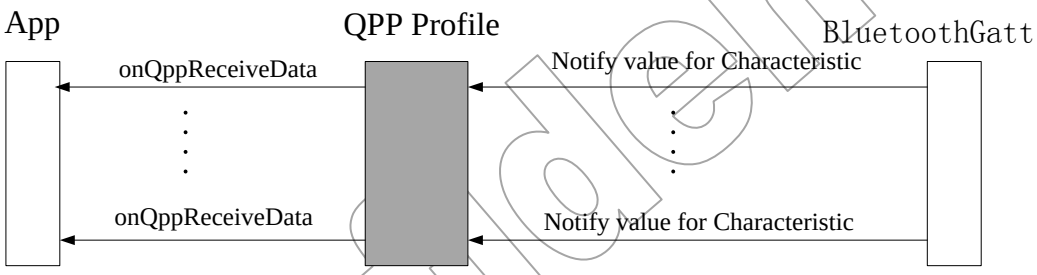

Figure 3 QPP Client RX flowcharts

4. API and Callback Description

There are one public class *QppApi* and one interface *iQppCallback* in the libQblueQpp library. The class *QppApi* defines APIs. The interface *iQppCallback* declares callbacks. There are five functions relevant: three API functions and two callback functions. These API functions are responsible to enable register service's UUIDs, transfer data. These callback functions are used to receive data, get QPP service status.

4.1 Class QppApi

4.1.1 General Definition

```
public class QppApi {
    public static boolean qppEnable(BluetoothGatt bluetoothGatt, String qppServiceUUID, String
  writeCharUUID);
    public static boolean qppSendData(BluetoothGatt bluetoothGatt, byte[] qppData);
    public static boolean setQppNextNotify(BluetoothGatt bluetoothGatt, boolean EnableNotifyChara);
    public static void   updateValueForNotification(BluetoothGatt bluetoothGatt,
    BluetoothGattCharacteristic characteristic);
    public static void setCallback(iQppCallback mCb);
};
```

4.1.2 API Description

public static boolean qppEnable()

Function public static boolean qppEnable(BluetoothGatt bluetoothGatt, String qppServiceUUID, String writeCharUUID);

Brief Register customer's UUIDs, in order to support customer's devices using customized QPP UUIDs.

Parameters:

In	bluetoothGatt	Android BluetoothGatt client handler
In	qppServiceUUID	UUID for QPP service in string
In	writeCharUUID	UUID for write Characteristic in string

Returns:

True The service is found and bluetoothGatt is not null.
False The service is not found or bluetoothGatt is null.

Note:

The qppServiceUUID must match the QPP UUID on the device side.

public static boolean qppSendData()

Function public static boolean qppSendData(BluetoothGatt bluetoothGatt, byte[] qppData);

Brief Send raw data to QPP Profile.

Parameters:

In	bluetoothGatt	Android BluetoothGatt client handler
In	qppData	Data to send, the length should not be larger than 20bytes

Returns:

True Argument is valid and sends data is successful.
False Argument is invalid or sends data is failed.

public static boolean setQppNextNotify ()

Function public static boolean setQppNextNotify(BluetoothGatt bluetoothGatt, boolean EnableNotifyChara);

Brief Enable characteristics notification.

Parameters:

In	bluetoothGatt	Android BluetoothGatt client handler
In	EnableNotifyChara	'true' to enable and 'false' to disable

Returns:

True set characteristics is successful.
False set characteristics is failed.

public static boolean updateValueForNotification ()

Function public static void updateValueForNotification(BluetoothGatt bluetoothGatt, BluetoothGattCharacteristic characteristic);

Brief Notify libQblueQpp that data have been received.

Parameters:

In	bluetoothGatt	Android BluetoothGatt client handler
In	characteristic	Notify characteristic

Returns:

None.

Note:

This function should be invoked in BluetoothGattCallback. onCharacteristicChanged.

public void boolean setCallback ()

Function public static void setCallback(iQppCallback mCb);

Brief Set callback function handler.

Parameters:

In	mCb	iQppCallback object

Returns:

None.

4.2 Interface iQppCallback

4.2.1 General Definition

```
public interface iQppCallback {
    void onQppReceiveData(BluetoothGatt bluetoothGatt, String qppUUIDForNotifyChar, byte[]
    qppData);
}
```

4.2.2 API Description

void onQppReceiveData()

Function void onQppReceiveData(BluetoothGatt bluetoothGatt, String qppUUIDForNotifyChar, byte[] qppData);

Brief Process the data that received from QPP Profile.

Parameters:

In	bluetoothGatt	Android BluetoothGatt client handler

In	qppUUIDForNotifyChar	UUID for notify characteristics.
Out	qppData	The received data from the notify characteristics.

Returns:

None.

5. Integration Note

5.1 Initialize

5.1.1 Add **'QppApi.qppEnable'** method

The method is used by the application to register user's UUIDs in order to support customer's devices using customized QPP UUIDs. The qppServiceUUID must match the QPP UUID on the device side. Then profile discovery the service, characteristic from bluetoothGatt and enable notification characteristics to bluetoothGatt. The parameter bluetoothGatt is a connected BluetoothGatt.

Add this method in following function:

```
private final BluetoothGattCallback mGattCallback = new BluetoothGattCallback(){
{
      … …
      public void onServicesDiscovered(BluetoothGatt bluetoothGatt, int status) {
            if(QppApi.qppEnable(bluetoothGatt, uuidQppService, uuidQppCharWrite))
                  isInitialize = true;
      }
      … …
 }
```

5.2 Rx Data

5.2.1 Add '*QppApi.setQppNotify()*' method

This method is to enable the QPP notification characteristics.
Add this method in following function:

```
public void onDescriptorWrite(BluetoothGatt bluetoothGatt, BluetoothGattDescriptor
                                    descriptor, int status)
{
    QppApi.setQppNextNotify(bluetoothGatt, true);
    /// user code
}
```

5.2.2 Add '*QppApi.updateValueForNotification*' method

This method is to update value for notification characteristic.
Add this method in following function:


```
public void onCharacteristicChanged(BluetoothGatt bluetoothGatt,
BluetoothGattCharacteristic characteristic)
{
      QppApi.updateValueForNotification(bluetoothGatt, characteristic);
      /// user code
}
```

5.2.3 Receive data chapter 4.2.1 **onQppReceiveData()**.

5.3 Tx Data

5.3.1 Call ***QppApi .qppSendData()*** to write data

```
public void onCharacteristicWrite(BluetoothGatt bluetoothGatt,
                                  BluetoothGattCharacteristic characteristic,int status)
{
     handlersend.postDelayed(runnableSend,20);
}

private Handler handlersend = new Handler( );

final Runnable runnableSend = new Runnable( )
{
     public void run ( )
     {
          QppApi.qppSendData(bluetoothGatt, qppDataSend);
     }
};
```

6. Example code

There is one example project named as *'QPP Demo'* in QBlue SDK which shows how to use the lib *'libQBlueQPP.jar'* to transfer raw data between QN902x device and QPP client.

Release History

REVISION	CHANGE DESCRIPTION	DATE
0.1	Initial release	2014-07-25

