QPP Integration Guide for iOS

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.

2. Overview

The QPP Diagram consists of three parts:

App Layer:

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

API Layer:

- Receive data from App layer and pass the data received to CoreBluetooth.
- Receive data from CoreBluetooth and pass the data received to App layer.

CoreBluetooth Layer:

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

The QPP Diagram is shown in Figure 1

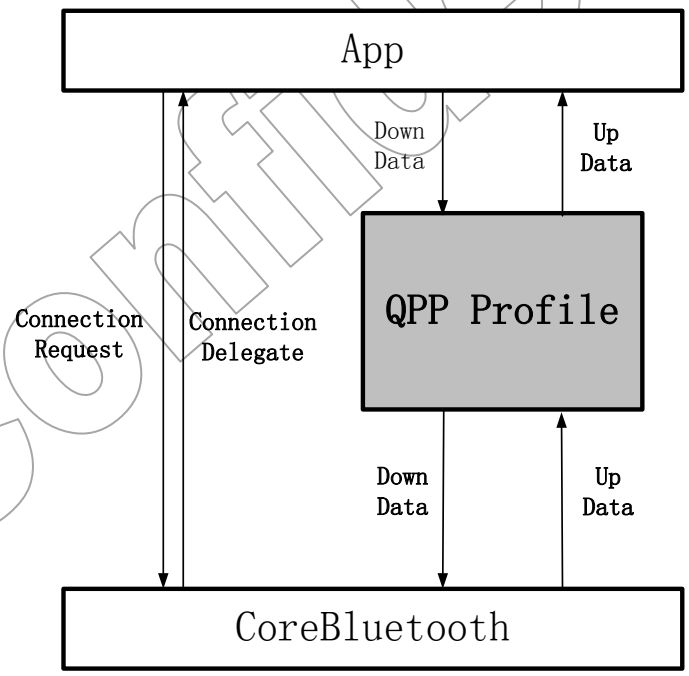

Figure 1 QPP Diagram

3. Flowchart

The QPP general flowchart is the following:

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

QPP TX flowchart is shown in Figure 2:

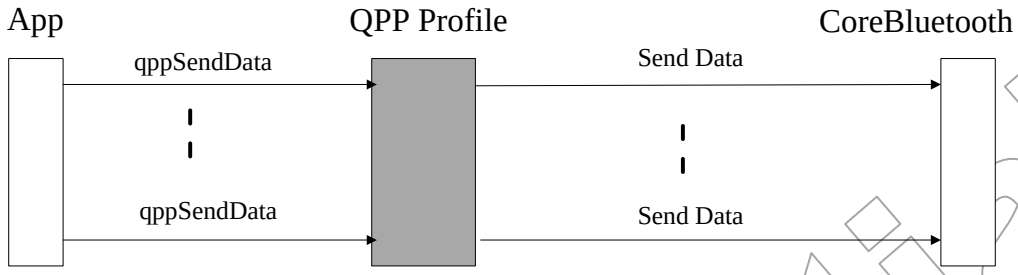

Figure 2 QPP TX flowchart

QPP RX flowchart is shown in Figure 3:

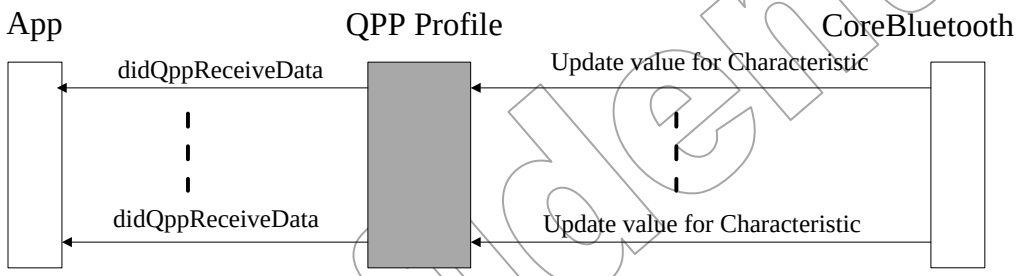

Figure 3 QPP RX flowchart

4. API and Delegate Description

These functions consist of two API functions and one delegate function. API functions implement to register user's UUIDs and to transfer data, delegate function used to receive data.

4.1 qppRegUUIDs()

Prototype:

-(void)qppRegUUIDs : (NSString *)qppServiceUUID
withWrChar : (NSString *)writeCharUUID

Parameters:

in	qppServiceUUID	UUID for QPP service in string
in	writeCharUUID	UUID for write Characteristic in string

Returns:

None.

Description: 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. The method is called before discovery procedure.

4.2 qppSendData()

Prototype:

```
-(void)qppSendData : (CBPeripheral *)aPeripheral
            withData : (NSData*)qppData;
```

Parameters:

in	aPeripheral	The peripheral must be built-in QPP profile server
in	qppData	The raw data

Returns:

None.

Description: The function is used by application to send raw data to QPP Profile.

4.3 didQppReceiveData()

Prototype:

```
-(void)didQppReceiveData : (CBPeripheral *)aPeripheral
              withCharUUID : (CBUUID *)qppUUIDForNotifyChar
                  withData : (NSData *)qppData;
```

Parameters:

Out	aPeripheral	The data received is from the peripheral.
Out	qppUUIDForNotifyChar	The UUID for notify characteristics.
Out	qppData	The data received is from the notify characteristics.

Returns:

None.

Description: The function is used by application to process the data received from QPP Profile.

4.4 Integration Note

4.4.1 Please insert the "***bleDidUpdateCharForQppService***" delegate method in the ***didDiscoverCharacteristicsForService*** delegate. The delegate is to update write characteristic and notify characteristic for OTA service.

```
- (void) peripheral : (CBPeripheral *)aPeripheral
      didDiscoverCharacteristicsForService : (CBService *)service error : (NSError *)error
{
      /// for quintic profile delegate
      [bleUpdateForQppDelegate bleDidUpdateCharForQppService  : aPeripheral
                                                      withService : aService
                                                            error : error];

      /// user code
      ……
}
```

4.4.2 Please insert the "***bleDidUpdateValueForQppChar***" delegate method in the "***didUpdateValueForCharacteristic***" delegate. The delegate is to update value for notification characteristic.

```
- (void) peripheral:(CBPeripheral *)aPeripheral
didUpdateValueForCharacteristic:(CBCharacteristic *)characteristic error:(NSError *)error
{
      for (CBService *aService in aPeripheral.services)
      {
            [bleUpdateForQppDelegate
                  bleDidUpdateValueForQppChar : (CBPeripheral*)aPeripheral
                              withService : (CBService *)aService
                                withChar : (CBCharacteristic *)characteristic
                                   error : (NSError *)error];

            /// user code
            ……
      }
}
```

5. Example code

There is an example iOS project in QBlue SDK, and it can be found in "\Quintic Corporation\QBlue-X.X.X\Projects\iOS\Qpp\". It shows how to use the libQBlueQPP library to implement transfer raw data between QN902x device and QppDemo.

The sent file should be put into a folder "Documents" which is located in the example iOS application by some tools.

Release History

REVISION	CHANGE DESCRIPTION	DATE
0.1	Initial release	2014-05-19

