Programming Guidelines of
A2 Version Bluetooth API Interface

Documented By: Yang Liang

Audited By: _____

Approved By: _____

Issued Date: _____

Programming Guidelines of A2 Version Bluetooth API Interface		File No.:	
		Issued Edition: 0.2	
Product Name:		**Product Model:**	

Revision History		
Edition	**Description**	**Date**
V0.1	Preliminary Programming Guidelines	2013-1-17
V0.2	Revision of Information Download Procedures	2013-1-22
V0.3	Adding the special description of the kitchen scale disposal	2013-7-9
V0.4	Adding the special description of the BLE version body fat analyzer	2013-7-17

I. Introduction of Functions of Interface Bank

The functions of this interface bank is regulated and realized by LFHardwareConnector and the corresponding protocol LFHardwareConnectorDelegate. The interface bank is used to proceed to Bluetooth 4.0 communication with Lifesense Health Equipments, including device search, device connection and data interaction with the devices, and then cut off Bluetooth connection with the devices after the communication completes. All of the communication process starts from device search. The devices can proceed to communication only when the device is searched. This interface bank focuses on Bluetooth communication with the devices. You may not bother to know how to set up connection, how to proceed to data transmission and when to cut off connection. If you plan to develop apps that can communicate with Lifesense Bluetooth 4.0 Health equipments, you may select this interface bank. You may only be responsible for data processing and leave the communication task to the interface bank.

II. Communication Stage Instruction

Generally speaking, there are two stages for the interface and the equipments to communicate. One is pairing stage, the other is receiving data stage. The normal equipments all need to finish the pairing stage to gain the necessary information of the equipments. After that, they can connect with the interface to get the measurement data. However, for the kitchen scale- like equipment and the BLE version body fat analyzer, the pairing stage is not necessary, it can connect with the interface directly to receive the measurement data.

III. Simple Configuration to Utilize object of LFHardwareConnector Class to Search Devices

You may visit the public singleton object via shareConnector class method. After this singleton object's been configured, the instance method startScanning can be used to open its device search function. The searched device will submit the searched device object to the proxy object through hardwareConnectorDiscoveredSensor method in the protocol. The following codes in List1 shows the example of this process.

List1:

```
/*Make the class comply with LFHardwareConnectorDelegate protocol*/
@interface ViewContronller:UIViewController<LFHardwareConnectorDelegate>

@end
```

```
@implementation


-(void)viewDidLoad
{
    [super viewDidLoad];
    LFHardwareConnector * hardwareConnector = [LFHardwareConnctor shareConnctor];
    hardwareConnector.delegate = self;
    /*Set the connector to search the weighing scale devices*/
    [hardwareConnector    enableSensorType: LF_SENSOR_TYPE_WEIGHT_SCALE];
    /*Set the connector to search the devices under pair-up status only*/
    [hardwareConnctor setConnectorState:CONNECTOR_PAIRING_STATE];
    [hardwareConnector startScanning];
}

#pragram mark LFHardwareConnectorDelegate Method
-(void)hardwareConnectorDiscoveredSensor:(LFHardwareSensor*)sensor
{
    /*According to the current setting, this proxy method will only return the weighing scale devices under pair-up status*/
    NSLog(@"The device's modelNumber :%@\n",sensor.modelNumber);
    NSLog(@"The device is    preparePair state is :%d\n",sensor.preparePair);
}

@end
```

IV. Proceed to pair-up procedure with the searched equipments

After a device under pair-up status is searched, you may use the instance method of LFHardwareConnector class pairWithHardwareSensor to request pair-up communication with this device (NOTE: the parameters of pairWithHardwareSensor must be the searched device object). The result of pair-up communication will be informed to the proxy object through hardwareConnectorPairedSensor:withState mehod in the protocol. The following codes in List2 demonstrates this process:

List2

```
-(void)viewDidLoad
{
    [super viewDidLoad];
    LFHardwareConnector * hardwareConnector = [LFHardwareConnctor shareConnctor];
    hardwareConnector.delegate = self;
```

```
        /*Set the connector to search weighing scale devices*/
        [hardwareConnector    enableSensorType: LF_SENSOR_TYPE_WEIGHT_SCALE];
        /*Set the connector to search the devices under pair-up status as well as under data transmission status*/
        [hardwareConnctor setConnectorState:CONNECTOR_ANY_STATE];
        [hardwareConnector startScanning];
    }
    #pragram mark LFHardwareConnectorDelegate Method
    -(void)hardwareConnectorDiscoveredSensor:(LFHardwareSensor*)sensor
    {
        if (sensor.preparePair)
        {
          /*Request pair-up with the searched devices*/
          [[LFHardwareConnector shareConnector]pairWithHardwareSensor:sensor];

        }
    }
   -(void)hardwareConnectorPairedSensor:(LFHardwareSensor*)sensor withState:(BOOL)state
    {
        if (state)
       {
          /*Store the complete equipment information of the successfully pair-up devices to the local database*/
          [[SQLiteOperation shareOperate] insertNewSensorRecord:sensor];

          /*Other handling operations*/
       }else
       {
          /*Failure to pair up with the devices*/
       }
    }
```

V. Procedures of acquiring measurement data from the paired devices

For those searched devices that are under data transmission status, pairSignature attribute in the device object can be used to check whether information of this device is stored in the local drive so as to judge whether this device is paired. If yes, you may write the device information in the local drive onto the searched device object and then put the device object into the connection queue through enqueueHardwareSensor method so as to proceed to data communication with the devices and acquire measurement data. (NOTE: the parameters of enqueueHardwareSensor method must be the searched

device object) The following codes in List3 demonstrates the procedures of acquiring measurement data.

List3

```
-(void)viewDidLoad
 {
      [super viewDidLoad];
      LFHardwareConnector * hardwareConnector = [LFHardwareConnctor shareConnctor];
      hardwareConnector.delegate = self;
      /*Set the connector to search weighing scale devices*/
      [hardwareConnector    enableSensorType: LF_SENSOR_TYPE_WEIGHT_SCALE];
      /*Set the connector to search the devices both under waiting status and under data transmission status*/
      [hardwareConnctor setConnectorState:CONNECTOR_ANY_STATE];
      [hardwareConnector startScanning];
 }
 #pragram mark LFHardwareConnectorDelegate Method
 -(void)hardwareConnectorDiscoveredSensor:(LFHardwareSensor*)sensor
 {
     if (!sensor.preparePair)
     {
      /*Look up the device information stored in the local drive*/
      LFHardwareSensor*localSensor=
                          [[SQLiteOperation shareOperate]selectSensorWith:sensor.pairSingnature];

    if(localSensor)
    {
       /*Comprehensive device information in the local drive, fill the corresponding information in the searched device object*/
          sensor.deviceId = localSensor.deviceId;
          sensor.deviceSn= localSensor.deviceSn;
          sensor.sensorName= localSensor.sensorName;
          sensor.pairSignature = localSensor.pairSignature;
          sensor.password= localSensor.password;
          sensor.supportDownloadInfoFeature = sensor.supportDownloadInfoFeature;
          sensor.hardwareVersion= localSensor.hardwareVersion;
          sensor.softwareVersion= localSensor.softwareVersion;
          sensor.firmwareVersion= localSensor.firmwareVersion;
          sensor.manufactureName= localSensor.manufactureName;
          sensor.maxUserQuantity= localSensor.maxUserQuantity;
          sensor.systemId = localSensor.systemId;
          /*Put the device into the connecting queue*/
```

```
            [[LFHardwareConnector shareConnector]enqueueHardwareSensor:sensor];
        }else
        {
         /*No device information stored in the local drive. The device is not yet paired*/
        }
      }else
     {
       /*The searched device is under waiting status*/
     }
     }
 -(void)hardwareConnectorReceiveWeightMeasurementData:
                                                    (WeightData*)weightData
  {
      /*Receive data object of weight measurement and store the data in the local data base*/
      [[SQLiteOperation    shareOperate]insertNewWeightRecord:weightData];
      /*Other handling*/

    }
```

VI. Procedures of downloading the specific setting information to the device

For the paired devices that are under data transmission status, you may utilize the attribute value of supportDownloadInfoFeature and enumerated value of enumerated LF_DOWNLOAD_INFO_TYPE of LFHardwareSensor object of device class for calculation so as to judge whether this device supports certain kind of download information. The type of information you plan to download must be included in the supported download sphere of the device. When you have a certain kind or several kinds of download information and wish to update the devices, you may utilize setOfDownloadInfoTypesForSensor method in LFHardwareConnectorDelegate protocol to inform the interface bank of download category. Then the interface bank will acquire the corresponding download information object sequentially through the method hardwareConnectorGetDownloadInfoForSensor:ofInfoType. You may realize this method to seal the required download information contents in the download information object and submit to the interface bank. The interface bank will inform you the download results through hardwareConnectorDidDownloadInfoForSensor: ofInfoType: withResult method.

1. Download the set device name to the device.

You may set the device name and then choose to download the name to the device

when successfully connected to the device so as to complete naming the device. The following codes in List4 demonstrates the device name setting procedures.

```
List4
    -(void)viewDidLoad
     {
      [super viewDidLoad];
     LFHardwareConnector * hardwareConnector = [LFHardwareConnctor shareConnctor];
        hardwareConnector.delegate = self;
        /*set the connector to search weighing scale devices*/
        [hardwareConnector    enableSensorType: LF_SENSOR_TYPE_WEIGHT_SCALE];
        /*set the connector to search the devices both under waiting status and under data transmission status*/
        [hardwareConnctor setConnectorState:CONNECTOR_ANY_STATE];
        [hardwareConnector startScanning];
    }
  #pragram mark LFHardwareConnectorDelegate Method
  -(void)hardwareConnectorDiscoveredSensor:(LFHardwareSensor*)sensor
    {
        if (!sensor.preparePair)
        {
         /*Look up the stored device information in the local data base*/
         LFHardwareSensor*localSensor= [[SQLiteOperationshareOperate]
                                                selectSensorWith:sensor.pairSingnature];

         if(localSensor)
         {
            /*Comprehensive device information in the local drive. Fill the corresponding information in the searched device object*/
            sensor.deviceId = localSensor.deviceId;
            sensor.deviceSn= localSensor.deviceSn;
            sensor.pairSignature = localSensor.pairSignature;
            sensor.sensorName= localSensor.sensorName;
            sensor.password= localSensor.password;
            sensor.supportDownloadInfoFeature = sensor.supportDownloadInfoFeature;
            sensor.hardwareVersion= localSensor.hardwareVersion;
            sensor.softwareVersion= localSensor.softwareVersion;
            sensor.firmwareVersion= localSensor.firmwareVersion;
            sensor.manufactureName= localSensor.manufactureName;
            sensor.maxUserQuantity= localSensor.maxUserQuantity;
```

```
            sensor.systemId = localSensor.systemId;
            /*Place the device in the connection queue*/
            [[LFHardwareConnector shareConnector]enqueueHardwareSensor:sensor];
        }
      }
    }

-(NSSet*)setOfDownloadInfoTypesForSensor:
                                        (LFHardwareSensor*)sensor
    {
        NSSet *downloadTypes = nil;
       /*Judge whether this device supports downloading device setting
information */

if(sensor.supportDownloadInfoFeature&LF_DOWNLOAD_TYPE_SENSORINFO)
        {
            /*This device supports downloading device setting information*/

            /*There is device setting information to be downloaded to the
device*/
            downloadTypes = [NSSet setWithObject:
             [NSNumber

numberWithUnsignedInteger:LF_DOWNLOAD_TYPE_SENSORINFO]];

        }
       return downloadTypes;
    }
  -(id)hardwareConnectorGetDowanloadInfoForSensor:
                                            (LFHardwareSensor*)sensor
      ofInfoType:(LF_DOWNLAOD_INFO_TYPE)infoType
      {
          switch(infoType)
                {
                    case LF_DOWNLOAD_TYPE_SENSORINFO:
                        {
                            /*Acquire the required setting information of this
device for download, e.g. Device name*/
                            SensorDownloadInfo *info =
                        [[[SensorDownloadInfo alloc]init]autorelease];
```

```
                                    info.sensorName = newSensorName;
                                    return info;
                                }
                            }
                        return nil;

                }

-(void)hardwareConnectorDidDownloadInfoForSensor:
                                        (LFHardwareSensor*)sensor
                        ofInfoType:(LF_DOWNLOAD_INFO_TYPE)infoType
                                        withResult:(BOOL)result;
            {
                if（infoType==LF_DOWNLOAD_TYPE_SENSORINFO）
                    {
                        if(result)
                            {
                                /*The setting information of the device is successfully downloaded to the device*/
                            }
                        else
                            {
                                /*The setting information of the device fails to be downloaded to the device*/
                            }
                    }

            }
```