|  |
| --- |
|  |
| SR Smartphone API – SR BLE Module |
| Version 01.01.02 |



**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version #** | **Remark** | **Date** | **Done By** |
| 01.01.01 | Initial Draft | 8th May 2015 | Paresh Vaghela |
| 01.01.02 | Format changes | 10th May 2015 | Heer Hirpara |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[1. Purpose 4](#_Toc419047599)

[2. Scope 4](#_Toc419047600)

[3. Definition(s) and Abbreviation(s) 4](#_Toc419047601)

[4. Reference(s) 4](#_Toc419047602)

[5. Market Study & Research 5](#_Toc419047603)

[6. BLE Communication Flow in Mobile 6](#_Toc419047604)

[7. Proposed Framework for SR API 9](#_Toc419047605)

[7.1 List of API 10](#_Toc419047606)

[8. BLE callbacks (functions) 11](#_Toc419047607)

[9. Summary 14](#_Toc419047608)

[10. FAQ 15](#_Toc419047609)

# Purpose

The purpose of this document is to provide overview of SR Smartphone API for SR Module.

# Scope

This document is intended for design team, development team, testing team, marketing team.

# Definition(s) and Abbreviation(s)

|  |  |
| --- | --- |
| **Name** | **Abbreviation** |
| SRP | SecuRemote Portal |
| BLE | Bluetooth Low Energy |
| SR App | SecuRemote® Smart Phone Application |
| SR Device | Device which uses SecuRemote® technology |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Table 1

# Reference(s)

|  |  |
| --- | --- |
| **Document No.** | **Title** |
|  |  |
|  |  |
|  |  |
|  |  |

Table 2

# Market Study & Research

We have explored various vendors like BlueGiga, Nordic, BlueRadio etc. mobile’s framework or libraries, which supports BLE communication.

We have not found out any of these, which can give direct support of BLE Communication. They are providing sample apps and open source codes.

Reason – Mobile has its own framework, which will be used to interact with any BLE Devices.

Apple - The Core Bluetooth framework provides the classes needed for your iOS and Mac apps to communicate with devices that are equipped with Bluetooth low energy wireless technology. For example, your app can discover, explore, and interact with low energy peripheral devices, such as heart rate monitors and digital thermostats.

So each mobile developer needs to follow the guidelines and process defined by mobile’s native frameworks.

Yes we can customize our wrapper framework above native ones.

# BLE Communication Flow in Mobile

Initialize BLE Central Manger

Scan Devices, which offers SR Smart Configurations

Connect to specific Device

Verify Services and Desired Characteristics are present

Read Characteristics to Get Current Values

Write Characteristics to Make Configuration Changes

Disconnect from Device

**Understanding Central and Peripheral Devices in Bluetooth**

**Central** - A central is like the “boss”. It wants information from a bunch of its workers in order to accomplish a particular task.

**Peripheral** - A peripheral is like the “worker”. It gathers and publishes data to that is consumed by other devices.

Client - Central Server - Peripheral

Wants Data

Has Data

**How Centrals communicate with Peripherals**

Advertising is the primary way that peripherals make their presence known via Bluetooth LE.

The job of central is to scan for these advertising packets, identify any peripherals it finds relevant, and connect to individual devices for more information.

**The Structure of Peripheral Data**

Once the central connects to peripheral, it needs to choose the data it is interested in. In Bluetooth LE, data is organized into concepts called “services” and “characteristics”.

A **service** is a collection of data and associated behaviors describing a specific function or a feature of a device. For example Device Information etc.

A **characteristic** provides further details about a peripheral’s service. For example S/W version, H/W version, manufacture name etc.

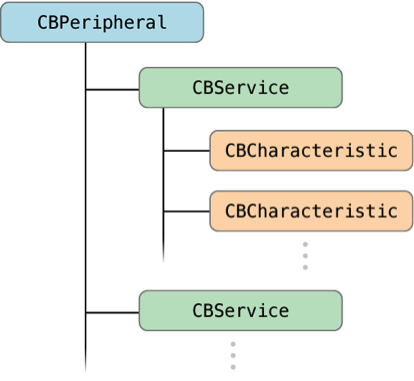


Fig: 1. Peripheral with Service & Characteristics

# Proposed Framework for SR API

Smart Phone

SR Smart Framework

Mobile Core Bluetooth Framework

BLE Commands

BLE Commands

## List of API

Here is the list of APIs or methods we can prepare for our SR Module.

NOTE – This is only draft version; we will define and decide return type and parameters of APIs later.

Methods or APIs: -

1) **FindSRDevices** – This function will find out SR Devices those are nearby mobile’s range

2) **ConnectSRDevice** – This function will connect to desired SR Device

3) **GetServicesSRDevice** – This function will find out all the services defined for selected SR Device

4) **GetCharacteristicsSRDevice** – This function will find out all characteristics for selected service

5) **WriteValueforCharacteristic** – This function will write data/value at desired characteristic. Here we can put our SR Smart Security and Encrypt/Decrypt algorithms.

6) **ReadValueforCharacteristic** - This function will read or get data from specified characteristic. Here we can put our SR Smart Security and Encrypt/Decrypt algorithms.

7) **MakeCharacteristicNotifiable** – This function will set notification for specific characteristic. When you subscribe to a characteristic’s value, you receive a notification from the peripheral when the value changes.

8) **GetSRDeviceInfo** – This function will get all the details like SR#, Security Token, Security Level, Crypto Keys etc.

9) **DisconnectSRDevice** – This function will be used to disconnect from SR Device.

10) **ReadRSSIValue** – This function will get the current RSSI value of SR Device.

11) **ISSRDeviceConnected** – This function will return Boolean value if SR Device is connected or disconnected.

12) **SetConfigurationParams** – This function will set various configuration parameters like connection time out, scan time interval, UUIDs of services and characteristics etc.

# BLE callbacks (functions)

Developer needs to set several callbacks (Android) or delegates (iOS) for transferring/receiving data in their source code apart from SR API.

**For Examples:**

#pragma mark - CBCentralManagerDelegate

 // Method called whenever you have successfully connected to the BLE peripheral

- (void)centralManager:(CBCentralManager \*)central didConnectPeripheral:(CBPeripheral \*)peripheral

{

}

// CBCentralManagerDelegate –

// This is called with the CBPeripheral class as its main input parameter. This contains most of the information there is to know about a BLE peripheral.

- (void)centralManager:(CBCentralManager \*)central didDiscoverPeripheral:(CBPeripheral \*)peripheral advertisementData:([NSDictionary](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSDictionary_Class/) \*)advertisementData RSSI:([NSNumber](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSNumber_Class/) \*)RSSI

{

}

// method called whenever the device state changes.

- (void)centralManagerDidUpdateState:(CBCentralManager \*)central

{

}

#pragma mark - CBPeripheralDelegate

// CBPeripheralDelegate –

// Invoked when you discover the peripheral's available services.

- (void)peripheral:(CBPeripheral \*)peripheral didDiscoverServices:([NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*)error

{

}

// Invoked when you discover the characteristics of a specified service.

- (void)peripheral:(CBPeripheral \*)peripheral didDiscoverCharacteristicsForService:(CBService \*)service error:([NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*)error

{

}

// Invoked when you retrieve a specified characteristic's value, or when the peripheral device notifies your app that the characteristic's value has changed.

- (void)peripheral:(CBPeripheral \*)peripheral didUpdateValueForCharacteristic:(CBCharacteristic \*)characteristic error:([NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*)error

{

}

#pragma mark - CBCharacteristic helpers

// Instance method to get the Hardware Version

- (void) getHardwareVersion:(CBCharacteristic \*)characteristic error:([NSError](http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSError_Class/) \*)error

{

}

// Instance method to get the manufacturer name of the device

- (void) getManufacturerName:(CBCharacteristic \*)characteristic

{

}

// Instance method to get the Model Number of the device

- (void) getModelNumber:(CBCharacteristic \*)characteristic

{

}

# Summary

In Nutshell, We will provide SR Smart Framework wrapper on above Mobile’s core Bluetooth framework that will communicate with mobile and SR Device.

We are going to follow best practices with coding guide-lines and coding structure.

We will prepare detail documentations for customers so they can easily start development using our SR Framework.

# FAQ

Que: 1. How do we manage security of source code in iOS & Android?

Ans: iOS provides framework which is secured. In Android, we are planning to provide jar file as library but it is quite possible to do reverse engineering. We are working on same.

Que: 2. How smartphone API will get Device info which is attached with SR Module (SR #, Security token, crypto keys, ANT parameters)?

Ans: