



密级：公开资料

Android BLE API 使用说明

文档版本：V1.0

深圳市昇润科技有限公司

2016年06月06日

版权所有

版本	修订日期	修订人	审稿人	修订内容
1.0	2016-06-06	贾捷飞	张眼	初版发布

目 录

1. Android BLE SDK 介绍.....	1
2. 工程配置.....	1
3. 代码示例.....	1
3.1 创建 BleService 实例.....	1
3.2 连接设备.....	2
3.3 给从机发送数据.....	2
3.3.1 默认数据发送通道.....	2
3.3.2 其他自定义数据发送通道.....	2
3.4 接收从机数据.....	3
3.4.1 Notify 数据.....	3
3.4.2 Read 数据.....	3
3.5 从机广播数据.....	4
3.5.1 Manufacturer Specific Data.....	4
4. 联系我们.....	5

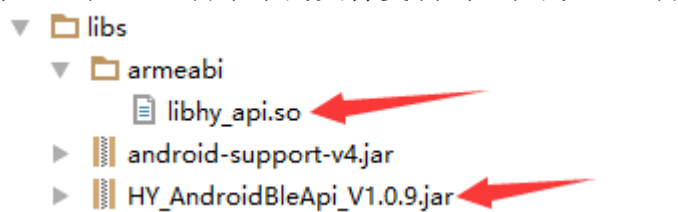
1. Android BLE SDK 介绍

- AndroidBLE_doc_V1.0.9
- BLE_API_DEMO
- libs
- PDF AndroidBLE_API使用说明.pdf

- 1) AndroidBLE_doc_V1.0.9 为 API 文档;
- 2) BLE_API_DEMO 为示例代码;
- 3) libs 目录下的文件是需要添加的库文件。

2. 工程配置

1) 将我司提供的 API 中 libs 目录下的文件复制到工程的 libs 目录下:



2) Android 从 4.3 开始支持 BLE, 所以需在 AndroidManifest.xml 中添加如下配置:

```
<uses-sdk android:minSdkVersion="18" />
```

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

3) 蓝牙功能需要用到一个关键服务 BleService, 需在 AndroidManifest.xml 中添加如下配置:

```
<service android:name="com.ble.ble.BleService"
```

```
android:enabled="true" android:exported="false" />
```

3. 代码示例

3.1 创建 BleService 实例

```
private BleService mLeService;  
private final ServiceConnection conn = new ServiceConnection() {  
  
    @Override  
    public void onServiceDisconnected(ComponentName name) {  
        mLeService = null;  
    }  
}
```

```

@Override
public void onServiceConnected(ComponentName name, IBinder
service) {
    // BleCallBack 是个关键参数, 后面会提到
    mLeService = ((BleService.LocalBinder)
service).getService(mBleCallBack);
    // 必须调用初始化函数
    mLeService.initialize();
}
};
// Activity 绑定服务
bindService(new Intent(this, BleService.class), conn,
BIND_AUTO_CREATE);

```

3.2 连接设备

```
mLeService.connect(mac, false);
```

参数 1: 从机的蓝牙地址;

参数 2: 断线后是否重新连接。

3.3 给从机发送数据

3.3.1 默认数据发送通道

默认的数据发送通道为 1000 服务下的 1001 通道:

```
mLeService.send(mac, value, true);
```

参数 1: 从机的蓝牙地址;

参数 2: 发送的数据, 可以是 16 进制字符串或者 byte 数组

参数 3: 我们的模组分加密跟不加密两种, true 表示加密发送数据, 最多发送 17 字节, false 表示不加密发送数据, 最多可发送 20 字节。

3.3.2 其他自定义数据发送通道

非默认通道可按下面的代码进行发送, 以服务 2000, 通道 2001 为例:

```

BluetoothGatt gatt = mLeService.getBluetoothGatt(mac);
BluetoothGattService service =
gatt.getService(UUID.fromString("00002000-0000-1000-8000-00805f9b
34fb"));
BluetoothGattCharacteristic characteristic =
service.getCharacteristic(UUID.fromString("00002001-0000-1000-800
0-00805f9b34fb"));
byte[] value = {(byte) 0xaa, (byte) 0xbb, (byte) 0xcc};
mLeService.send(gatt, characteristic, value, true);

```

3.4 接收从机数据

从机数据的接收有两种方式，一种是 Notify，另一种是 Read

3.4.1 Notify 数据

这种方式是主机实时被动接收从机数据，需要在 `BleCallBack` 的 `onServicesDiscovered()` 触发后打开 `notify`。我司模组程序默认服务 1000 下的 1002 通道，API 内部代码会打开 `notify`，如果需要打开其他通道的 `notify`，以服务 2000，通道 2002 为例，步骤如下：

```
BluetoothGatt gatt = mLeService.getBluetoothGatt(mac);
BluetoothGattService service =
gatt.getService(UUID.fromString("00002000-0000-1000-8000-00805f9b
34fb"));
BluetoothGattCharacteristic character =
service.getCharacteristic(UUID.fromString("00002002-0000-1000-800
0-00805f9b34fb"));
// 打开 notify, 如要关闭 notify, 参数 3 改为 false 即可
mLeService.setCharacteristicNotification(gatt, character, true);
```

Notify 数据都是在 `BleCallBack` 的 `onCharacteristicChanged(mac, characteristic)` 回调函数中接收：

```
// uuid 区分数据来自哪个通道
UUID charUuid = characteristic.getUuid();
// 接收到的数据
byte[] data = characteristic.getValue();
```

3.4.2 Read 数据

这种方式是主机主动向从机索要数据，主机索要一次，从机返回一次，下面以服务 2000，通道 2003 为例：

```
BluetoothGatt gatt = mLeService.getBluetoothGatt(mac);
BluetoothGattService service =
gatt.getService(UUID.fromString("00002000-0000-1000-8000-
00805f9b34fb"));
BluetoothGattCharacteristic characteristic =
service.getCharacteristic(UUID.fromString("00002003-0000-
1000-8000-00805f9b34fb"));
gatt.readCharacteristic(characteristic);
```

Read 数据都是在 `BleCallBack` 的 `onCharacteristicRead(mac, characteristic, status)` 回调函数中接收：

```
if (status == BluetoothGatt.GATT_SUCCESS) {
// 成功接收到 read 数据
// uuid 区分数据来自哪个通道
UUID charUuid = characteristic.getUuid();
// 接收到的数据
```

```
byte[] data = characteristic.getValue();  
}
```

3.5 从机广播数据

从 1.0.7 开始，支持解析广播数据，app 扫描的时候会得到一个类型为 byte 数组的参数 scanRecord:

```
LeScanRecord record = LeScanRecord.parseFromBytes(scanRecord);
```

可以通过 record 获取广播中的信息，目前支持： Advertise Flags, Local Name, Service UUID, Service Data, Tx Power Level, Manufacturer Specific Data, Slave Connection Interval Range。

record.getBytes () 即可获得原始的广播数据(即扫描时得到的 scanRecord)。

3.5.1 Manufacturer Specific Data

我司的某些产品，会在广播数据中附带一些信息，一般是以厂商数据出现，标准的厂商数据格式是:厂商 id (2 字节，低字节在前，高字节在后)+厂商数据，而且可以有多个厂商数据，我司产品一般只有一条厂商数据，如果想得到原始的厂商数据，可以按照如下步骤:

```
List<byte[]> rawDataList = new ArrayList<byte[]>();  
SparseArray<byte[]> mfrData = record.getManufacturerSpecificData();  
for (int i = 0; i < mfrData.size(); i++) {  
    int id = mfrData.keyAt(i);  
    byte[] data = mfrData.get(id);  
    byte[] rawData = new byte[2 + data.length]; // 单条原始的厂商数据  
    rawData[0] = (byte) (id & 0xff);  
    rawData[1] = (byte) ((id >> 8) & 0xff);  
    System.arraycopy(data, 0, rawData, 2, data.length);  
    rawDataList.add(rawData);  
}
```

4. 联系我们

深圳市昇润科技有限公司

ShenZhen ShengRun Technology Co.,Ltd.

Tel: 0755-86233846 Fax: 0755-82970906

官网地址: www.tuner168.com

阿里巴巴网址: <http://shop1439435278127.1688.com>

E-mail: marketing@tuner168.com

地址: 广东省深圳市南山区西丽镇龙珠四路金谷创业园 B 栋 6 楼 601-602

