Nut Android SDK 文档

修订记录：

|  |  |  |
| --- | --- | --- |
| 时间 | 文档版本 | 备注 |
| 2016/12/29 | 1.0 | 初稿 |
| 2020/08/19 | 1.1 | 支持Beacon特性 |
| 2020/11/06 | 1.2 | 增加API使用示例 |
| 2020/11/26 | 1.3 | 添加DFU使用帮助 |
| 2021/05/24 | 1.4 | 添加常见问题 |

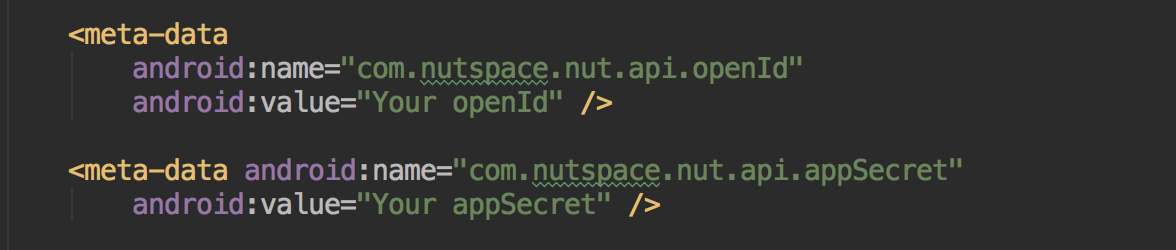
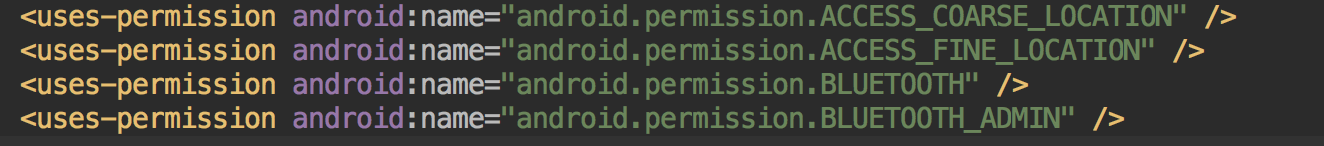
### 开发指南

**一、开发环境**

推荐Android Studio 2.0及以上版本；Android系统版本4.3及以上版本。

**二、配置manifest和填写openId及appSecret**

1、manifest的配置主要包括添加权限，以下权限缺一不可，填写openId和appSecret，代码示例如下：

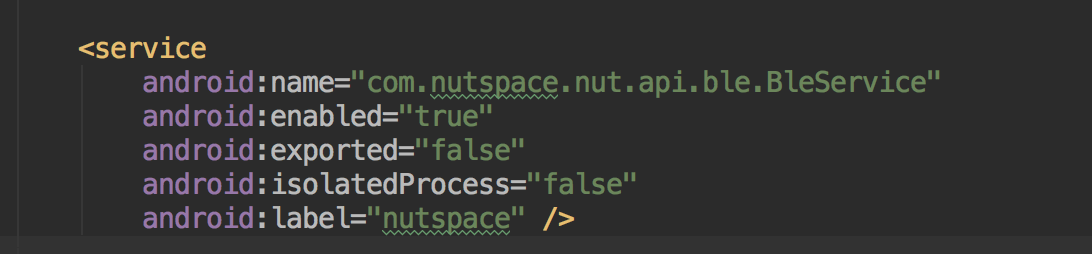


权限说明：

|  |  |
| --- | --- |
| 权限 | 用途 |
| ACCESS\_COARSE\_LOCATION | 扫描Nut蓝牙设备 |
| ACCESS\_FINE\_LOCATION | 扫描Nut蓝牙设备 |
| BLUETOOTH | 蓝牙相关功能 |
| BLUETOOTH\_ADMIN | 蓝牙相关功能 |

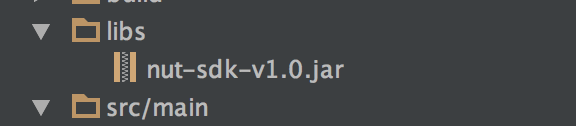
2、配置BleService

在manifest中定义BleService，代码示例如下：



**三、导入和初始化SDK**

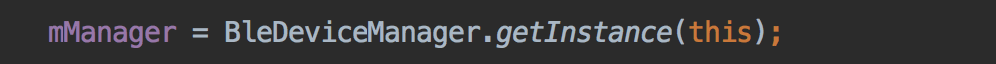
1、将SDK的jar包导入项目中，例如：



2、在组件（Activity或Service）中初始化BleManager并绑定BleService

代码示例如下：

初始化BleManager：

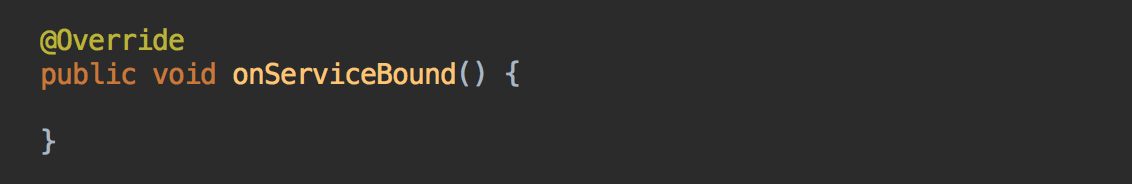


绑定BleService：



注意：绑定bind() 和unbind() 方法必须成对儿出现

绑定成功后会回调方法：

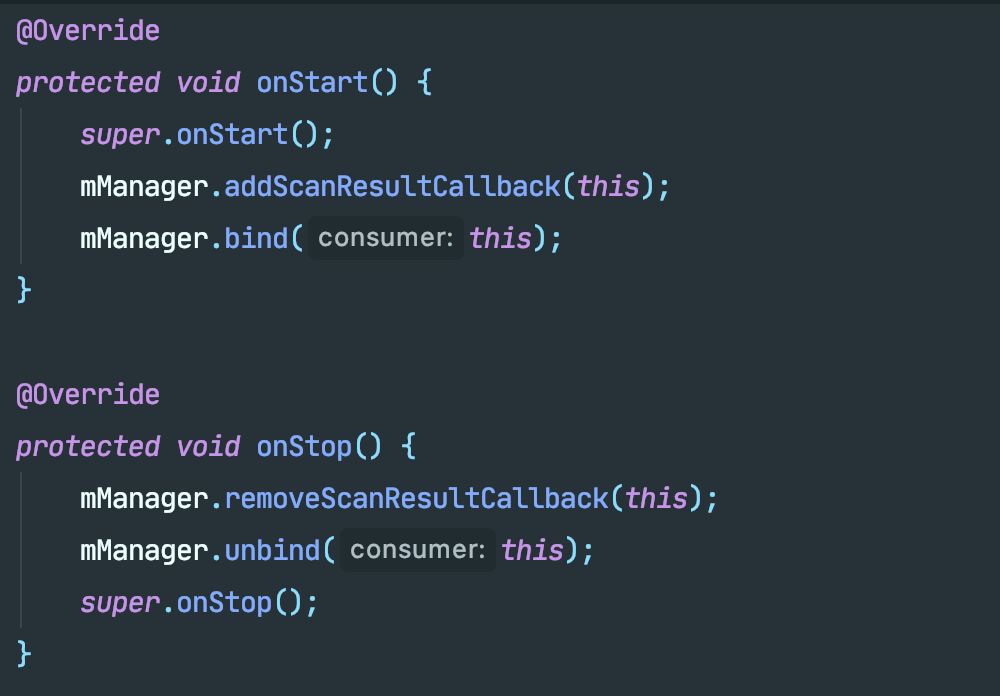


绑定成功之后即可通过调用对象mManager提供的方法使用SDK。

**四、SDK功能使用**

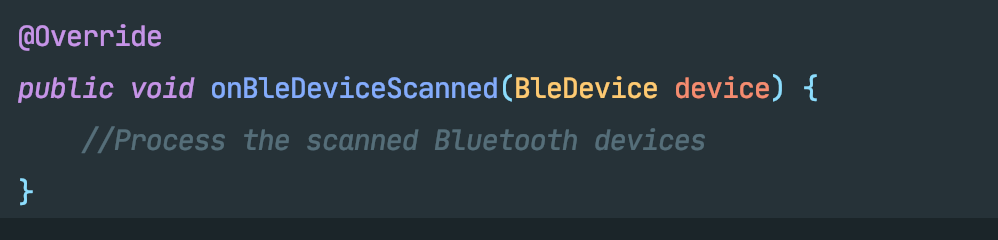
1、扫描蓝牙设备

注册扫描回调接口ScanResultCallback在onBleDeviceScanned()中处理扫描到的设备。



注意：不需要监听扫描事件时，必须调用removeScanResultCallback()方法移除监听事件。

扫描到设备后会回调方法：



并返回包含设备信息的类BleDevice的对象。

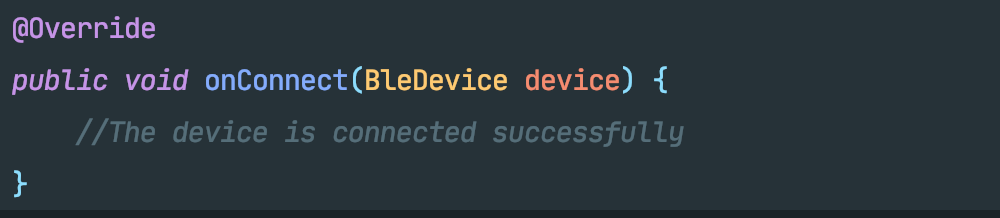
2、连接设备

注册连接回调接口ConnectStateChangedCallback，然后调用连接设备方法

connect(Context context, BleDevice device, boolean isAuto)



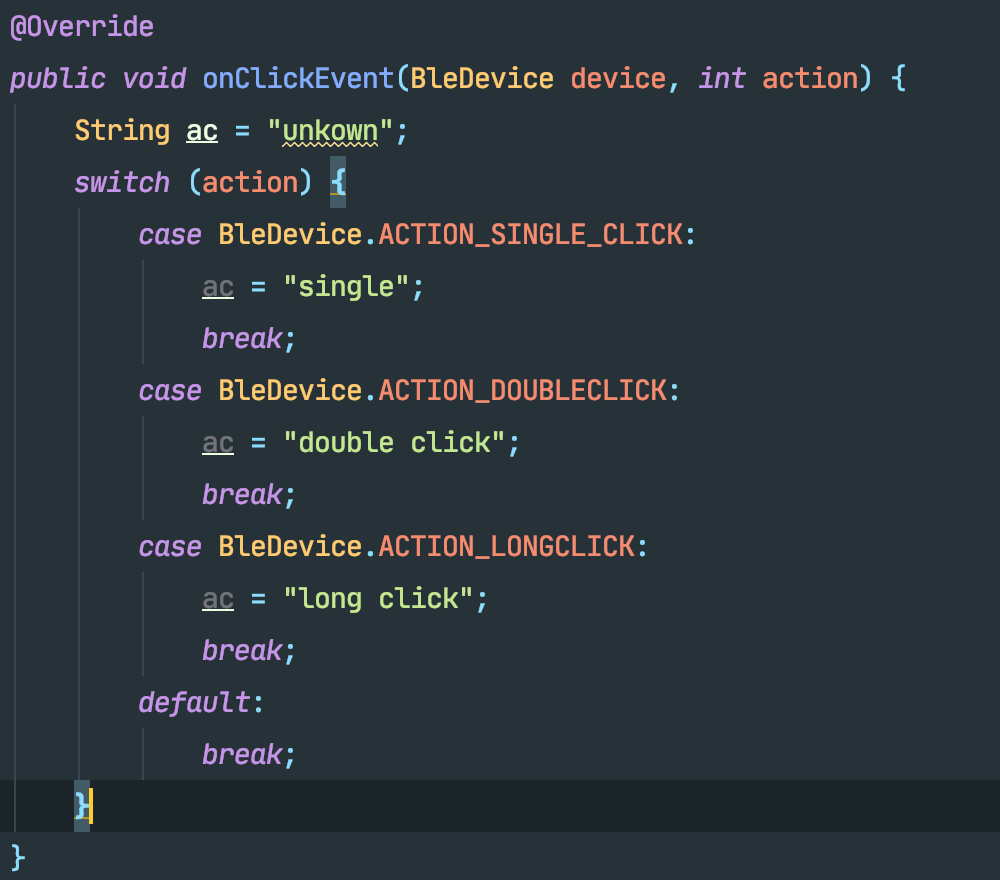
设备连接成功，会从onConnect(BleDevice device)返回设备对象实例，然后可以调用设备的其他方法。



3、设备基本功能和事件回调

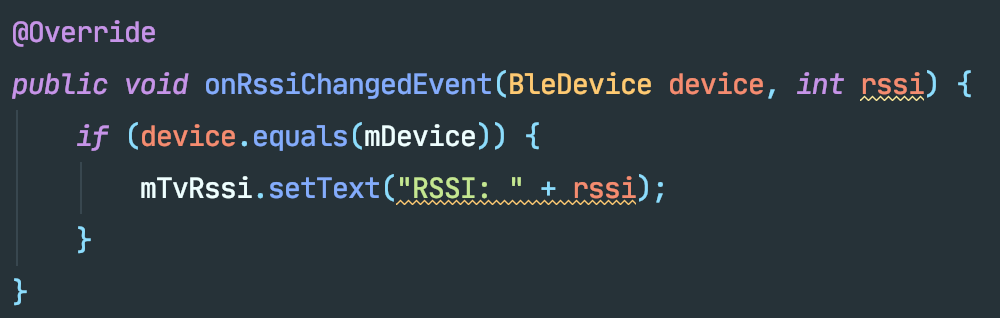
注册设备事件回调接口EventCallback（请参考注册连接回调接口），添加接口实现。

监听设备按钮单击/双击操作（新固件暂不支持长按事件）：



读取设备信号强度值：

mManager.readRssi(device);



读取设备电量值：

mManager.readBattery(this, mDevice);



寻找设备/停止寻找设备：

mManager.changeRingState(mDevice, BleDevice.STATE\_RING);

mManager.changeRingState(mDevice, BleDevice.STATE\_QUIT);



开启和关闭设备断开提醒功能：

mManager.enableAntiLost(mDevice, true/false);

4、Beacon配置功能（需要定制固件才可以支持此API）

4.1启用Beacon功能

硬件开启进入配置模式后，设置Beacon有效的UUID和Major、Minor信息后，设备再次重启会进入Beacon运行模式。

//设置Beacon的UUID信息，UUID的字符串格式需要采用标准格式，例如："10102233-4455-6677-8899-AABBCCDDEEFF"

void setBeaconUUID(BleDevice device, String beaconUUID)

//设置UUID信息完成，result=true表示成功，false表示失败

void onBeaconUUID(BleDevice device, String uuid, boolean result)

//设置Beacon的Major、Minor信息

void setBeaconMajorMinor(BleDevice device, int major, int minor)

//设置Major、Minor信息完成，result=true表示成功，false表示失败

void onBeaconMajorMinor(BleDevice device, int major, int minor, boolean result)

4.2停用Beacon功能

硬件开启进入配置模式后，将Beacon的UUID和Major、Minor设置为无效值，设备再次重启会进入Nut Tag运行模式。

UUID无效值：FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFF

Major无效值：FFFF

Minor无效值：FFFF

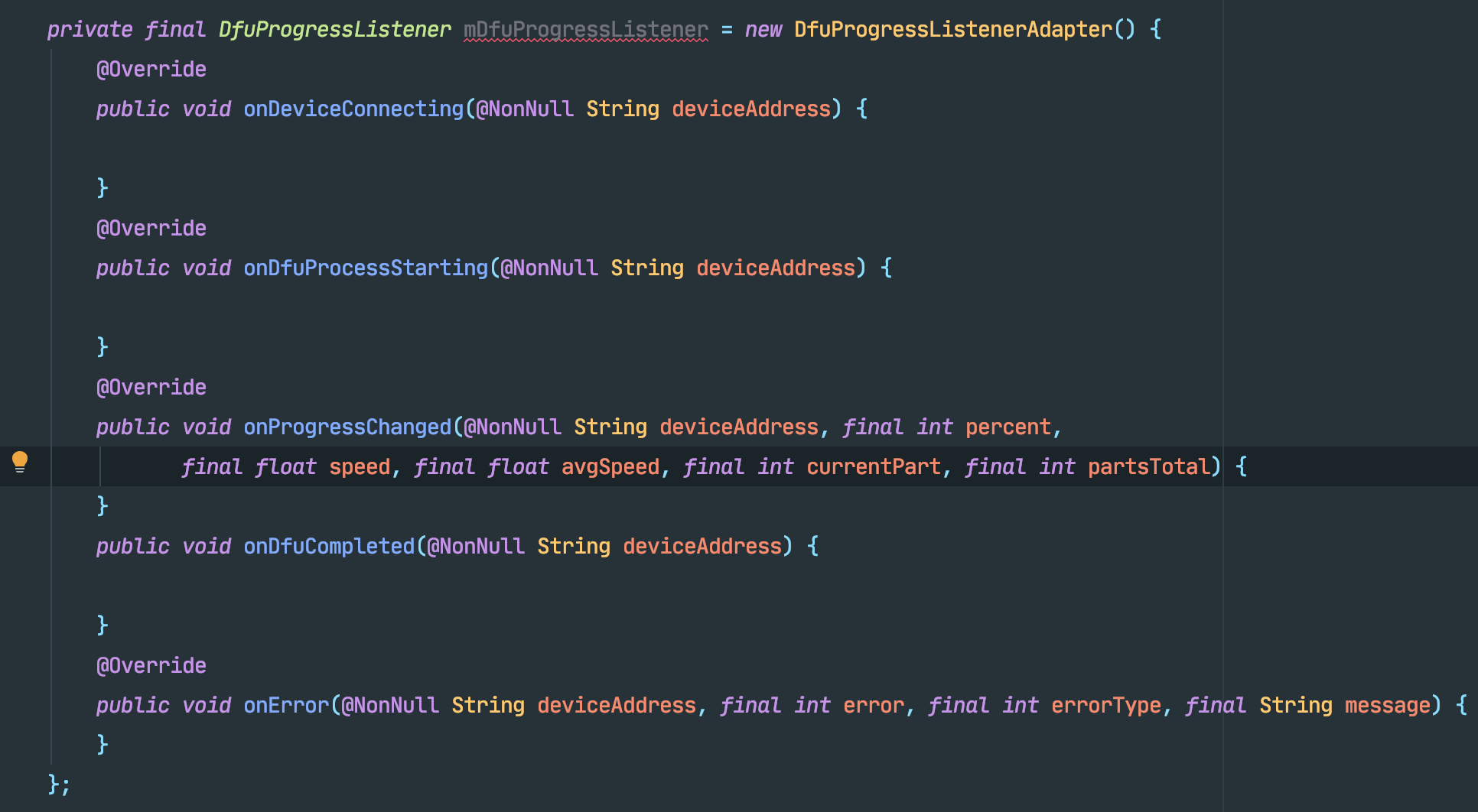
注意：每次更新Beacon信息，需要先调用setBeaconUUID，完成后再调用setBeaconMajorMinor，每次更新Beacon信息，需要UUID和MajorMinor都更新，只更新某一个会导致Beacon信息异常。

5、DFU接口使用

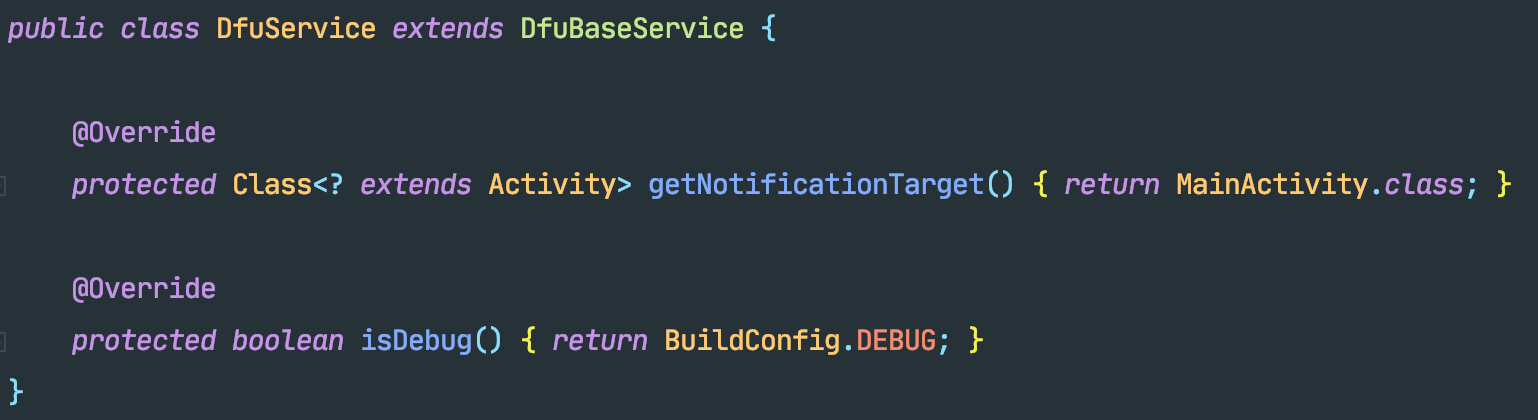
5.1、实现DfuProgressListener接口，DfuProgressListener是处理DFU状态变更、错误处理、固件发送等信息。

注意：需要在onRsume()和onPause()调用registerProgressListener(this, mDfuProgressListener)注册此接口。





5.2、继承DfuBaseService并实现DfuService，需要在getNotificationTarget方法中返回实现DFU处理的Activity。



5.3、初始化DfuServiceInitiator，初始化固件对象，然后开始升级过程。



5.4、设备DFU流程简介

5.4.1、开启DFU模式

当设备进入TAG工作模式后，连接设备然后执行switchToDFUMode，设备会先重启然后进入DFU工作模式，此时设备的蓝牙广播名称为“DfuTarg”，而且设备的Mac地址或DeviceId会递增1。

5.4.2、扫描设备并开始升级

发现处于DFU模式的设备后，调用performDFUUpload开始DFU升级过程。DfuServiceInitiator会自动连接设备、发送固件文件、发送完成后设备会自动重启，此时完成DFU过程。

**注意事项：**

上述功能都是通过Android系统蓝牙读写API实现的功能，由于系统API执行限制，不能并发执行蓝牙读写等操作，每个操作执行需要一定的时间（几百毫秒）发送到外围设备，如果并行执行，Android蓝牙协议栈会直接丢弃API的执行。为了保险起见，可以将操作放入执行队列，每隔1-2s从队列取出并执行蓝牙操作。

6、接口介绍：

|  |  |
| --- | --- |
| 接口 | 功能 |
| ScanResultCallback | 监听扫描设备结果 |
| EventCallback | 监听rssi值变化，呼叫设备，设备按键次数，电池变化 |
| EventCallback | 监听设备连接状态 |
| BeaconResultCallback | 监听设置Beacon配置变化 |

文档中的代码片段参见SDK demo

**六、常见问题**

1、常见错误码

|  |  |
| --- | --- |
| Error Code | Description |
| 0x08(8) | Android BLE error: connection timeout. |
| 0x13(19) | Android BLE error: connection terminate by peer user. |
| 0x16(22) | Android BLE error: connection terminated by local host. |
| 0x03E(62) | Android BLE error: connection fail to establish. |
| 0x85(133) | Android BLE error: Bluetooth gatt error. |
| 0x0101(257) | Android BLE error: no connection to cancel |

2、针对以上错误，App需要调用Nut SDK BleDeviceManager.disconnect()，主动执行设备断开并回收系统资源，然后执行蓝牙扫描，等待扫描到此蓝牙设备，执行BleDeviceManager.connect()方法，重试连接操作。