

# Android 接口编程手册

Bluetooth, Wi-Fi, USB, Serial, NFC

v3.5.3

（注：浏览时请使用 PDF 左侧导航栏）

## 目录

Android 接口编程手册 .....	1
1. 介绍 .....	3
1.1. 初始化 .....	3
1.2. 创建打印机连接 .....	3
1.3. 打印 .....	3
1.4. 关闭打印机 .....	3
2. POSConnect .....	3
2.1. init .....	3
2.2. createDevice .....	4
2.3. exit .....	4
2.4. getUsbDevices .....	4
2.5. getSerialPort .....	5
2.6. openLog .....	5
2.7. getUsbSerialNumber .....	5
2.8. registerUsbReceiver .....	6
2.9. unregisterUsbReceiver .....	6
3. IDeviceConnection .....	6
3.1. connect .....	6
3.2. close .....	7
3.3. sendData .....	7
3.4. readData .....	8
3.5. stopReadLoop .....	8
3.6. getConnectInfo .....	9
3.7. getConnectType .....	9

## 1. 介绍

这个安卓 SDK 接口编程手册介绍了怎么通过蓝牙、USB、Wi-Fi、串口来连接打印机，向打印机发送内容。

### 1.1. 初始化

```
POSConnect.init(appContext)
```

### 1.2. 创建打印机连接

```
val connect = POSConnect.createDevice(POSConnect.DEVICE_TYPE_BLUETOOTH)
connect.connect("12:34:56:78:9A:BC") { code, connectInfo, msg ->
    if (code == POSConnect.CONNECT_SUCCESS) {
        Log.i("tag", "device connect success")
        val printer = POSPrinter(connect)
    } else if (code == POSConnect.CONNECT_FAIL) {
        Log.i("tag", "device connect fail")
    }
}
```

### 1.3. 打印

```
printer.printString("test ~")
```

### 1.4. 关闭打印机

```
connect.close()
```

## 2. POSConnect

连接打印设备的类。

### 2.1. init

SDK 初始化。建议再 application 类 onCreate 方法里面调用。

static void init(Context appContext)

【参数】

➤ appContext  
应用的上下文

2.2. createDevice

根据设备类型，创建设备。

static IDeviceConnection createDevice(int deviceType)

【参数】

➤ deviceType  
设备类型

变量	描述
DEVICE_TYPE_USB	USB 类型
DEVICE_TYPE_BLUETOOTH	蓝牙类型
DEVICE_TYPE_ETHERNET	网络类型
DEVICE_TYPE_SERIAL	串口类型
DEVICE_TYPE_NFC	NFC 类型

【返回值】

连接的对象

2.3. exit

退出打印服务，调用该方法后，如需再使用打印 SDK，需再调用 init 的方法。

static void exit()

2.4. getUsbDevices

获取 USB 路径列表

static List<String> getUsbDevices(Context context)

【参数】

➤ context  
Context

【返回值】

USB 路径列表

## 2.5. getSerialPort

获取串口路径列表

```
static List<String> getSerialPort()
```

【返回值】

串口路径列表

## 2.6. openLog

设置日志开关，日志文件保存在 sdcard\Android\data\[Package Name]\files\logs 目录中

```
void openLog(boolean isOpen)
```

【参数】

➤ isOpen

是否打开日志

【返回值】

void

## 2.7. getUsbSerialNumber

获取 USB 序列号，如果 device 没有权限时，则会弹出权限弹窗，用户同意权限则可以获得 USB 序列号。拒绝则回调 null。

注:获取序列号需一个一个的获取，不能同时多个获取。

```
void getUsbSerialNumber(UsbDevice device, IStrCallback callback)
```

【参数】

➤ device

待获取序列号的 USB 设备

➤ callback

序列号回调，获取成功则返回真实的序列号，失败则返回 null

```
public interface IStrCallback {  
    void receive(String info);  
}
```

【返回值】

void

## 2.8. registerUsbReceiver

注册 USB 事件监听

void registerUsbReceiver(IUsbEventCallback listener)

### 【参数】

➤ listener

监听事件的回调

```
public interface IUsbEventCallback {  
    void receive(int event, UsbDevice device);  
}
```

event 值	描述
USB_ATTACHED	USB 设备已连上
USB_DETACHED	USB 设备已断开

### 【返回值】

void

## 2.9. unregisterUsbReceiver

取消 USB 事件监听

void unregisterUsbReceiver()

### 【返回值】

void

## 3. IDeviceConnection

连接设备的接口类。用于发送数据到打印机或读取打印机返回的数据。Sdk 初始化后可通过 POSConnect.createDevice(deviceType)方式获得。

### 3.1. connect

连接设备

```
void connect(String info, IConnectListener listener);  
boolean connectSync(String info, IConnectListener listener)
```

同步连接，如使用同步连接，所有的发送、接收、断开连接都会强行使用同步的方式。

**【参数】**

➤ info

连接信息。

◆ 设备类型为 DEVICE\_TYPE\_USB 时，info 为 USB 路径名 ("/dev/bus/usb/002/003") 或 vid,pid("8888,9999")

◆ 设备类型为 DEVICE\_TYPE\_BLUETOOTH 时，info 为蓝牙 MAC 地址

◆ 设备类型为 DEVICE\_TYPE\_ETHERNET 时，info 为网络的 IP 地址，或 IP 地址,端口号组合。

例:"192.168.1.100" 或 "192.168.1.100,9100"

◆ 设备类型为 DEVICE\_TYPE\_SERIAL 时，info 为 串口名,串口波特率 组合成的字符串。例如: "/dev/ttyS4,38400"

➤ listener

连接状态监听器。

```
public interface IConnectListener {  
    void onStatus(int code, String connectInfo, String message);  
}
```

■ code

code 值	描述
CONNECT_SUCCESS	连接成功
CONNECT_FAIL	连接失败
CONNECT_INTERRUPT	连接中断
SEND_FAIL	发送失败
USB_ATTACHED	USB 设备已连上
USB_DETACHED	USB 设备已断开
BLUETOOTH_INTERRUPT	蓝牙设备断开

■ connectInfo

连接信息，例如:当使用网络连接时，connectInfo 为传入的 ip 地址。

■ message

提示信息

### 3.2. close

关闭连接

void close()

void closeSync()

使用同步的方式关闭连接。如通过 connectSync 连接，请使用 closeSync 关闭连接。

### 3.3. sendData

该方法用于发送数据到打印机。

void sendData(byte[] data)

void sendData(List<byte[]> datas)

`int sendSync(byte[] data)`

使用同步的方式发送数据，如通过 `connectSync` 连接，都会使用同步的方式发送数据。

**【参数】**

➤ `data`

需发送的字节数组

➤ `datas`

需发送的字节数组集合

**【返回值】**

`void`

### 3.4. readData

该方法用于读取从打印机传回的数据。默认超时时间为 5000ms。如通过 `connectSync` 连接，请使用 `readSync` 读取数据。

`void readData(int timeout, IDataCallback callback);`

`void readData(IDataCallback callback);`

`byte[] readSync(int timeout);`

同步读取，有可能会阻塞线程，请放到子线程里面调用。

`void startReadLoop(IDataCallback callback)`

开启读取数据监听。

**注意:**`startReadLoop` 不可与其它读取方法共用。

**【参数】**

➤ `timeout`

读取超时时间，单位为毫秒，默认为 5000

➤ `callback`

数据回调

```
public interface IDataCallback {  
    void receive(byte[] data);  
}
```

**【返回值】**

`void`

### 3.5. stopReadLoop

停止循环读取数据



```
void stopReadLoop()
```

【返回值】

```
void
```

### 3.6. **getConnectInfo**

获取连接信息。

```
String getConnectInfo();
```

【返回值】

返回连接信息，对应的 `connect` 方法里面的 `info`

### 3.7. **getConnectType**

获取连接类型

```
int getConnectType();
```

【返回值】

返回连接类型。对应 `createDevice` 方法中的 `deviceType`