

# **Android Port Program Manual**

**Bluetooth, Wi-Fi, USB, Serial,NFC**

**v3.5.3**

**(Note: Please use the PDF left navigation bar when browsing)**

Android Port Program Manual .....	1
1. Instruction .....	3
1.1. initialization .....	3
1.2. Create a printer connection .....	3
1.3. Print .....	3
1.4. Close the connection .....	3
2. POSConnect .....	4
2.1. init .....	4
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 .....	7
3.1. connect .....	7
3.2. close .....	8
3.3. sendData .....	8
3.4. readData .....	8
3.5. stopReadLoop .....	9
3.6. getConnectInfo .....	9
3.7. getConnectType .....	9

# 1. Instruction

This Android Port Program Manual describes how to connect the printer through Bluetooth, USB, Wi-Fi and serial ports and send content to the printer.

## 1.1. initialization

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

## 1.2. Create a printer connection

```
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. Print

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

## 1.4. Close the connection

```
connect.close()
```

## 2. POSConnect

This class is used to connect printers. Constant variable are defined in POSConnect class.

### 2.1. init

This function is used for SDK initialization. It is recommended to call in the onCreate function of the application class.

```
static void init(Context appContext)
```

[Parameter]

➤ appContext

Context of application

### 2.2. createDevice

Create a print device by device type

```
static IDeviceConnection createDevice(int deviceType)
```

[Parameter]

➤ deviceType

Device type

Variable	Description
DEVICE_TYPE_USB	USB
DEVICE_TYPE_BLUETOOTH	BLUETOOTH
DEVICE_TYPE_ETHERNET	ETHERNET
DEVICE_TYPE_SERIAL	SERIAL
DEVICE_TYPE_NFC	NFC

[Return]

Connect Object

### 2.3. exit

Exit the print service.

```
static void exit()
```

### 2.4. getUsbDevices

Get USB pathname list

static List<String> getUsbDevices(Context context)

[Parameter]

➤ context

Context

[Return]

USB port pathname list.

## 2.5. getSerialPort

Get serial port path list

static List<String> getSerialPort()

[Return]

Serial port path list

## 2.6. openLog

Set the log switch, Log files are saved in the sdcard\Android\data\[Package Name]\files\logs directory

void openLog(boolean isOpen)

[Parameter]

➤ isOpen

Whether to open the log

[Return]

void

## 2.7. getUsbSerialNumber

Retrieves the USB serial number. If the device does not have permission, a permission dialog will pop up. The user can grant permission to obtain the USB serial number; if denied, the callback will return null.

Note: Serial numbers must be retrieved one at a time; multiple devices cannot be queried simultaneously.

void getUsbSerialNumber(UsbDevice device, IStrCallback callback)

[Parameter]

➤ device

The USB device for which to retrieve the serial number.

➤ callback

The callback for the serial number. Returns the actual serial number on success, or null on failure.

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

[Return]

void

## 2.8. registerUsbReceiver

Register USB receiver event listener

```
void registerUsbReceiver(IUsbEventCallback listener)
```

[Parameter]

➤ listener

Call back for monitoring events

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

event	Description
USB_ATTACHED	USB Attached
USB_DETACHED	USB Detached

[Return]

void

## 2.9. unregisterUsbReceiver

Unregister USB receiver event listener

```
void unregisterUsbReceiver()
```

[Return]

void

## 3. IDeviceConnection

The interface class connecting the device, which is used to send data to the printer or read the data returned by the printer. Available through `POSConnect.createDevice(deviceType)`.

### 3.1. connect

Connect the printer

```
void connect(String info, IConnectListener listener)
```

```
boolean connectSync(String info, IConnectListener listener)
```

Synchronous connection, such as using a synchronous connection, all sending, receiving, and disconnection will be forced to use the synchronous method.

[Parameter]

➤ info

Device Information.

1. If the device type is `DEVICE_TYPE_USB`, Info is the USB path name ("/dev/bus/USB/002/003") or vid, pid ("8888,9999").
2. If the device type is `DEVICE_TYPE_BLUETOOTH`, info is the Bluetooth MAC address.
3. When the device type is `DEVICE_TYPE_ETHERNET`, info is the IP address of the network, or a combination of IP address and port number. eg: "192.168.1.100" or "192.168.1.100,9100".
4. If the device type is `DEVICE_TYPE_SERIAL`, info is a string composed of serial port name and serial port baud rate. eg: "/dev/ttyS4,38400".

➤ listener

Connect the status listener.

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

■ code

code	Description
CONNECT_SUCCESS	Connection successful
CONNECT_FAIL	Connection failed
CONNECT_INTERRUPT	Disconnected
SEND_FAIL	fail in send
USB_ATTACHED	USB Attached
USB_DETACHED	USB Detached
BLUETOOTH_INTERRUPT	Bluetooth Interrupt

■ connectInfo

Connection information, eg: when using a network connection, connectInfo is the incoming ip address.

■ message

prompt information.

## 3.2. close

Close the connection

```
void close()
```

```
void closeSync()
```

Close the connection synchronously. If connected through connectSync, Please use closeSync to close the connection.

## 3.3. sendData

This function is used to send data to the printer.

```
void sendData(byte[] data)
```

```
void sendData(List<byte[]> datas)
```

```
int sendSync(byte[] data)
```

Send data in a synchronous way, such as connecting through connectSync, will use a synchronous way to send data.

[Parameter]

➤ data

Byte array to be sent.

➤ datas

Byte array collection to be sent.

## 3.4. readData

This function is used to read the data returned from the printer. If connected through connectSync, Please use readSync to read data.

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

```
void readData(IDataCallback callback)
```

```
byte[] readSync(int timeout)
```

Synchronous reading may block the UI thread.

```
void startReadLoop(IDataCallback callback)
```

Start reading data monitoring.

**Note:** startReadLoop cannot coexist with other read methods.

[Parameter]

➤ timeout

Read timeout, in milliseconds. The default is 5000.



➤ callback

Data callback

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

### 3.5. stopReadLoop

Stop reading data and monitoring

```
void stopReadLoop()
```

[Return]  
void

### 3.6. getConnectInfo

Get connection information.

```
String getConnectInfo();
```

[Return]  
Return the connection information, corresponding to the info in the connect method

### 3.7. getConnectType

get connection type.

```
int getConnectType();
```

[Return]  
Returns the connection type. Corresponds to the deviceType in the createDevice method.