

Android Port Program Manual

Bluetooth, Wi-Fi, USB, Serial

v2.1.0

Contents

| | |
|---------------------------------------|---|
| Android Port Program Manual..... | 1 |
| Bluetooth, Wi-Fi, USB, Serial..... | 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..... | 3 |
| 2.1. init..... | 4 |
| 2.2. createDevice..... | 4 |
| 2.3. exit..... | 4 |
| 2.4. getUsbDevices..... | 4 |
| 2.5. getSerialPort..... | 5 |
| 3. IDeviceConnection..... | 5 |
| 3.1. connect..... | 5 |
| 3.2. close..... | 6 |
| 3.3. sendData..... | 6 |
| 3.4. readData..... | 6 |
| 3.5. isConnect..... | 6 |

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, 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 | UBS |
| DEVICE_TYPE_BLUETOOTH | BLUETOOTH |
| DEVICE_TYPE_ETHERNET | ETHERNET |
| DEVICE_TYPE_SERIAL | SERIAL |

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

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, IPOSListener listener);
```

[Parameter]

➤ info

Device Information.

1. If the device type is DEVICE_TYPE_USB, info is the USB path name
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
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.

| 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 |

3.2. close

Close the connection

```
void close()
```

3.3. sendData

This function is used to send data to the printer.

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

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

[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.

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

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

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

Synchronous reading may block the UI thread.

[Parameter]

➤ timeout

Read timeout, in milliseconds. The default is 5000.

➤ callback

Data callback

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

3.5. isConnected

Get connection status

```
boolean isConnected()
```

[Return]

Returns a status that the interface were connected (true or false)