SunmiPrinter 开发者文档

目录

简介

商米机器内置了**缓存式热敏打印机**，允许App通过sdk直接打印热敏小票，具有打印机的商米产品有:

手持非金融系列 —— V1、V1s、V2 Pro等
手持金融系列 —— P1、P1-4g等
台式收银机设备 —— T1、T2、T1mini、T2mini等
台式收银秤设备 —— S2等

商米产品的内置打印机一共有 2 种规格:

- 80mm 纸张宽度，带切刀，兼容 58mm， 如T1搭载了这种打印机
- 58mm 纸张宽度，不带切刀，如V1搭载了这种打印机

App 开发者可以使用以下方式调用内置热敏打印机:

- **通过内置打印服务接口调用打印机**——此种方式适用于初次开发打印相关app且对Epson指令没有了解的开发者，通过商米打印服务提供的多个打印接口实现自己需要的打印效果；
- **通过内置虚拟蓝牙设备调用打印机**——此种方式适用于之前有过开发蓝牙、usb打印机经验或是开发者app已经实现蓝牙打印机打印的开发者，仅需稍微改动代码即可实现打印效果；

1. 通过内置打印服务接口调用打印机

1.1. 连接内置打印服务

1.1.1. 通过远程依赖方式

为了方便调用内置打印服务，对使用gradle配置的开发者，我们推荐使用远程依赖库，同时由于我们的库对机型问题做了适配，使用远程依赖方式会自动区分不同机型的接口，不用因为机型不同需要配置不同的AIDL文件而发愁，当用了错误的接口也会有相应异常提示；

集成方式

在根目录下的build.gradle文件中确认已经配置了中央仓库：**mavenCtral**

在app目录下的build.gradle文件中添加依赖和属性配置：

```
dependencies { implementation 'com.sunmi:printerlibrary:1.0.15' }
```

针对V2s和V2s_Plus的设备由于Android版本要求，需要增加包可见性的声明才可以连接服务，所以需要在AndroidManifest.xml中添加：

```
<queries>
    <package android:name="woyou.aidlservice.jiuiv5"/>
</queries>
```

初始化

像绑定一个服务组件一样，这里通过单例调用绑定方法

```
boolean result = InnerPrinterManager.getInstance().bindService(context,
innerPrinterCallback);
InnerPrinterCallback innerPrinterCallback = new InnerPrinterCallback(){

    @Override
    protected void onConnected(SunmiPrinterService service){
        //这里即获取到绑定服务成功连接后的远程服务接口句柄
        //可以通过service调用支持的打印方法
    }

    @Override
    protected void onDisconnected() {
        //当服务异常断开后，会回调此方法，建议在此做重连策略

    }
};
```

result:表示绑定成功或失败

简单调用打印方法

```
try{
    service.printText("要打印的内容\n", new InnerResultCallbcak() {
```

```
                @Override public void onRunResult(boolean isSuccess) throws
RemoteException
                {
                      //返回接口执行的情况(并非真实打印):成功或失败
                }


                @Override
                public void onReturnString(String result)
throws RemoteException
                {
                      //部分接口会异步返回查询数据
                }

                @Override
                public void onRaiseException(int code, String msg) throws
RemoteException
                {
                      //接口执行失败时，返回的异常状态说明
                }

                @Override
                public void onPrintResult(int code, String msg)
Throws RemoteException
                {
                      //事务模式下真实的打印结果返回
                }
          });
     } catch (RemoteException e) {
          //如部分接口只能用于指定机型所以会跑出调用接口异常，如钱箱接口只能用于台式机
     }
```

结束释放

正常调用结束后可以调用如下方法，断开服务

```
InnerPrinterManager.getInstance().unBindService(context,
innerPrinterCallback);
```

类说明

InnerPrinterManager 打印库的管理类，主要用来实现连接、断开打印服务

InnerPrinterCallback 连接远程服务的回调接口，用来获取远程服务连接结果

SunmiPrinterService 商米打印服务接口类，定义了所有的打印方法，通过连接服务后获取此类实例，同AIDL IWoyouService 接口相同

InnerPrinterException 打印异常类，调用方法出现错误或异常； 由于部分机型不具备相应方法功能（如手持机器不具备钱箱功能，当调用开钱箱接口时会抛 出打印异常，相应接口不能使用）

InnerResultCallback 打印方法的回调接口，用来获取接口方法的执行情况

InnerLcdCallback 客显方法的回调接口，用来获取客显方法的执行情况

InnerTaxCallback 税控方法的回调接口，用来获取发送税控的结果

1.1.2. 通过AIDL方式

AIDL 简介

AIDL 是 Android Interface Definition language 的缩写，它是一种 Android 内部进程通信接口的描述语言，通过它我们可以定义进程间的通信接口。商米AIDL提供封装好的常用打印指令，方便开发者快速接入 Sunmi 打印机

AIDL 使用

建立连接可分以下 5 步骤：

1. 在项目中添加AIDL文档资源文件中附带的AIDL文件且不能修改包路径和包名（部分机型还包含java文件）；
2. 在控制打印的代码类中实现ServiceConnection；
3. 调用ApplicationContext.bindService()，并在ServiceConnection实现中进行传递。注意：bindservice是非阻塞调用，意味着调用完成后并没有立即绑定成功，必须以 onServiceConnected回调为准。
4. 在ServiceConnection.onServiceConnected()实现中，你会接收一个IBinder实例(被调用的Service)。调用 **IWoyouService.Stub.asInterface(service)**将参数转换为IWoyouService类型。
5. 现在就可以调用IWoyouService接口中定义的各种方法进行打印了。

AIDL 资源下载

⚠注意：由于机型差异各个机型会有部分接口不同，所以请区分机型使用AIDL资源文件，目前有如下四个资源包：

old（V1）用于商米首款V1机型
——主要包含

IWoyouService.aidl	打印机接口文件
ICallback.aidl	打印机回调接口文件
TransBean.aidl	
TransBean.java	页打印类文件

handheld（手持机 除V1）用于商米手持机型，如V1s、V2、P2等
——主要包含

IWoyouService.aidl	打印机接口文件
ICallback.aidl	打印机回调接口文件
TransBean.aidl	
TransBean.java	页打印类文件
ITax.aidl	税控回调接口文件

desktop（台式机）用于商米台式打印机，如T1、T2等
——主要包含

IWoyouService.aidl	打印机接口文件

ICallback.aidl　　打印机回调接口文件

ITax.aidl　　税控回调接口文件

desktop+lcd（台式+客显）用于商米带客显的打印机，如T1mini、T2mini

——主要包含

IWoyouService.aidl　　打印机接口文件

ICallback.aidl　　打印机回调接口文件

ITax.aidl　　税控回调接口文件

ILcdCallback.aidl　　客显回调接口文件

通用资源

WoyouConsts.java　　定义打印机暴露的常量类，用来设置配置

绑定服务示例

实现ServiceConnection

```
private ServiceConnection connService = new ServiceConnection() {
@Override
public void onServiceDisconnected(ComponentName name) {
    woyouService = null;
}
@Override
public void onServiceConnected(ComponentName name, IBinder service) {
     woyouService = IWoyouService.Stub.asInterface(service);
}
```

绑定打印服务

```
private void Binding(){
    Intent intent=new Intent();
    intent.setPackage("woyou.aidlservice.jiuiv5");
    intent.setAction("woyou.aidlservice.jiuiv5.IWoyouService");
    bindService(intent, connService, Context.BIND_AUTO_CREATE);
}
```

1.2. 接口定义说明

通过以上方式与内置打印服务建立连接后，并获得IWoyouService或SunmiPrinterService对象后即可调用如下的接口实现自己的打印

1.2.1. 打印机初始化及设置

编号	方 法
1	void **printerInit( )** 打印机初始化
2	void **printerSelfChecking(ICallback** callback**)** 打印机自检

1、打印机初始化

函数：void **printerInit( )**

备注：重置打印机的逻辑程序（例如：排版、加粗等样式设置），但不清空缓存区数据，因此未完成的打印作业将在重置后继续

2、打印机自检

函数：void **printerSelfChecking (ICallback** callback**)**

参数：callback → 结果回调

示例：

```
woyouService.printerSelfChecking(callback);
```

1.2.2. 获取设备及打印机信息

编号	方 法
1	String **getPrinterSerialNo( )** 获取打印机板序列号
2	String **getPrinterModal( )** 获取打印机类型接口
3	String **getPrinterVersion( )** 获取打印机固件版本号
4	Build**.MODEL**（常量） 设备名称
5	int **updatePrinterState()** 获取打印机的最新状态
6	String **getServiceVersion()** 获取打印服务版本号

7	int **getPrintedLength(ICallback** callback**)** 获取打印头打印长度
8	int **getPrinterPaper()** 获取打印机当前的纸张规格

1、获取打印机的最新状态

函数： String **updatePrinterState ()**

返回值：

1 → 打印机工作正常
2 → 打印机准备中
3 → 通讯异常
4 → 缺纸
5 → 过热
6 → 开盖
7 → 切刀异常
8 → 切刀恢复
9 → 未检测到黑标
505 →未检测到打印机
507 →打印机固件升级失败

说明： 此接口返回值可适用于所有商米机器判断但部分状态由于硬件配置不会拿到（例如手持机不支持开盖检测）

备注： V1设备暂不支持该接口；除主动获取状态，也可以通过注册广播异步获取，参考附录A

2、获取打印机已打印长度

函数： int **getPrintedLength(ICallback** callback**)**

说明： 目前可获取到上电以来的打印长度，由于台式机和手持机的硬件区别，获取打印结果的返回略有不同，即手持机通过ICallback callback接口获取打印长度，台式机通过返回值直接获取长度。

3、获取打印机当前的纸张规格

函数： int **getPrinterPaper()**

说明： 手持打印机默认为58mm的纸张规格，台式打印机默认为80mm的纸张规格，但可以通过增加挡板并进行打印机配置设置为使用58mm的纸张规格，此接口会返回当前打印机设置的纸张规格；

备注： 目前台式机器T1在v2.4.0版本以上支持此接口；T2、S2在v1.0.5以上支持此接口；其他机型在4.1.2版本后均支持此接口查询纸张规格;

1.2.3. ESC/POS指令

编号	方 法
1	void **sendRAWData** (byte[] data, **ICallback** callback) 打印ESC/POS格式指令

1、打印ESC/POS格式指令

函数： void **sendRAWData(**byte[] data, **ICallback** callback**)**

参数：

data → ESC/POS指令

callback → 结果回调

备注：相关指令，请参考《ESC/POS》指令集。

示例：

```
    woyouService.sendRAWData(new byte[]{0x1B,0x45,0x01},
callback); // 1B,45, 01是字体加粗指令
```

1.2.4 打印样式设置接口说明

编号	方 法
1	void **setPrinterStyle(**int key, int value**)** 设置打印机的样式

1、设置打印机的样式

函数： void **setPrinterStyle(**int key, int value**)**

参数： 参考WoyouConsts.java 打印机样式设置常量类

key →通过常量类接口中的定义设置不同的属性，一般分**ENABLE_XXX**和**SET_XXX**

value →对应属性设置状态或大小

使能ENABLE_XXX属性需要选择ENABLE或DISABLE

设置SET_XXX属性需要设置具体的大小

备注：此接口需要打印服务v4.2.22版本以上支持!

示例：

```
  woyouService.setPrinterStyle( WoyouConsts.ENABLE_ILALIC,
WoyouConsts.ENABLE); //使之后打印文本内容变为斜体
  woyouService.setPrinterStyle(WoyouConsts.SET_LINE_SPACIN
G,123); //使之后打印文本行间距变为123个像素
```

1.2.5. 打印模式切换

编号	方 法 / 指 令
1	int **getPrinterMode**() 获取打印机模式
2	int **getPrinterBBMDistance**() 获取当前黑标模式下设置的走纸距离
3	相关模式设置，请在系统“设置”→”打印设置”中进行设置。

1、获取打印机模式

函数： int **getPrinterMode**()

返回值：

0 → 普通模式

1 → 黑标模式

2 → 标签模式

备注：黑标模式目前支持所有商米T1、T2台式机器；

标签模式目前支持所有商米V2、V2pro手持机器；

2、获取黑标模式打印机自动走纸距离

函数： int **getPrinterBBMDistance**()

返回值： 走纸距离(点行)

备注：仅支持 T1 、T2设备。

1.2.6. 文字打印

编号	方 法 / 指 令
1	void **setAlignment**(int alignment, **ICallback** callback) 设置对齐模式
2	void **setFontName**(String typeface, **ICallback** callback) 设置自定义打印字体
3	void **setFontSize**(float fontsize, **ICallback** callback) 设置字体大小
4	esc/pos指令：字体加粗{0x1B, 0x45, 0x1}、取消加粗{0x1B, 0x45, 0x0} 设置与取消加粗
5	void **printText**(String text, **ICallback** callback) 打印文字
6	void **printTextWithFont**(String text, String typeface, float fontSize, **ICallback** callback) 打印指定字体，大小的文本
7	void **printOriginalText** (String text, **ICallback** callback) 打印矢量文字

1、设置对齐模式

函数： void **setAlignment** (int alignment, **ICallback** callback)

参数：

alignment → 对齐方式：0→居左 , 1→居中, 2→居右

callback → 结果回调

备注：全局方法，对之后执行的打印有影响，打印机初始化时取消相关设置。

示例：

```
woyouService.setAlignment(1, callback);
```

2、设置自定义打印字体

函数： void **setFontName** (String typeface, **ICallback** callback)

参数：

typeface → 指定要使用的自定义字体名称，目前仅支持矢量字体，字体需预置在应用assets目录

callback → 结果回调

说明：此接口可以扩展打印机默认的字体样式，支持开发者使用自定义的字体，但由于各个字体的内宽不一致，其行距和行宽需自行调整

备注：此接口需要v4.14.0以上版本打印服务支持

3、设置字体大小

函数： void **setFontSize** (float fontSize, **ICallback** callback)

参数：

fontSize → 字体大小

callback → 结果回调

备注：全局方法，对之后打印有影响，初始化能取消设置，字体大小是超出标准国际指令的打印方式，调整字体大小会影响字符宽度，每行字符数量也会随之改变，因此按等宽字体形成的排版可能会错乱。

示例：

```
woyouService.setFontSize(36, callback);
```

4、设置与取消加粗

指令：字体加粗{0x1B, 0x45, 0x1}、取消加粗{0x1B, 0x45, 0x0}

备注：请参考本文**1.1.3.3. ESC/POS指令**。

示例：

```
woyouService.sendRAWData(new byte[]{0x1B,0x45,0x0},
callback); // 取消字体加粗指令
```

5、打印文字

函数： void **printText(**String text, in **ICallback** callback**)**

参数：

text → 打印内容，文字宽度**超出一行自动换行排版**，不满一行或超出一行不满一行部分需要在结尾加**强制换行符"\n"**才会即时打印出来，否则会缓存在缓存区

callback → 结果回调

备注：若要修改打印文本的样式（如：对齐方式、字体大小、加粗等），请在调用**printText**方法前设置。

示例：

```
woyouService.setAlignment(1, callback);
woyouService.setFontSize(36, callback);
woyouService.printText("商米科技\n", callback);
```

6、打印指定字体，大小的文本

函数： void **printTextWithFont(**String text, String typeface, float fontSize, **ICallback** callback**)**

参数：

text → 打印内容，文字宽度**超出一行自动换行排版**，不满一行或超出一行不满一行部分需要在结尾加**强制换行符"\n"**才会即时打印出来，否则会缓存在缓存区。

typeface → 自定义字体名称，目前仅支持矢量字体，字体需预置在应用assets目录

fontSize → 字体大小，只对该方法有效

callback → 结果回调

备注：此接口的字体设置效果需要v4.14.0以上版本打印服务支持

示例：

```
woyouService.printTextWithFont("商米\n","",36,callback);
```

7、打印矢量文字

函数： void **printOriginalText(**String text, **ICallback** callback**)**

参数：

text → 打印内容，文字宽度超出一行自动换行排版，不满一行或超出一行不满一行部分需要在结尾加强制换行符"\n"才会即时打印出来，否则会缓存在缓存区

callback → 结果回调

返回值：

备注：文字按矢量文字宽度原样输出，即每个字符不等宽。

示例：

```
woyouService.printOriginalText ("κρχκμνκλρκνκνμρτυφ\n",
callback);
```

1.2.7. 表格打印

编号	方 法 / 指 令
1	void **printColumnsText**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback) 打印表格的一行(不支持阿拉伯字符)
2	void **printColumnsString**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback) 打印表格的一行，可以指定列宽、对齐方式

1、打印表格的一行(不支持阿拉伯字符)

函数： void **printColumnsText**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback)

参数：

colsTextArr → 各列文本字符串数组。

colsWidthArr → 各列宽度数组，以英文字符计算，每个中文字符占两个英文字符，每个宽度大于 0。

colsAlign → 各列对齐方式：0 居左，1 居中，2 居右。

callback → 结果回调。

备注： 三个参数的数组长度应该一致，如果 colsText[i]的宽度大于colsWidth[i]，则文本换行，不支持阿拉伯字符。

示例：

```
woyouService.printColumnsText(new String[]{"商米","商米","商米"}, new int[]{4,4,8}, new int[]{1,1,1},callback);
```

2、打印表格的一行，可以指定列宽、对齐方式

函数： void **printColumnsString**(String[] colsTextArr, int[] colsWidthArr, int[] colsAlign, **ICallback** callback)

参数：

colsTextArr → 各列文本字符串数组。

colsWidthArr → 各列宽度权重即各列所占比。

colsAlign → 各列对齐方式：0 居左，1 居中，2 居右。

callback → 结果回调。

备注： 三个参数的数组长度应该一致，如果 colsText[i]的宽度大于colsWidth[i]，则文本换行。

示例：

```
woyouService.printColumnsString(new String[]{"商米","商米","商米"}, new int[]{1,1,2}, new int[]{1,1,1},callback);
```

1.2.8. 打印图片

编号	方 法
1	void **printBitmap**(Bitmap bitmap, **ICallback** callback) 打印图片
2	void **printBitmapCustom**(Bitmap bitmap, int type, **ICallback** callback) 打印图片(2)

1. 打印图片

函数： void **printBitmap** (Bitmap bitmap, **ICallback** callback)

参数：

bitmap → 图片Bitmap对象

callback → 结果回调

备注： 图片最大像素需要宽x高小于250万，且宽度根据纸张规格设置（58为384像素，80为576像素），如果超过纸张宽度将不显示

示例：

```
woyouService.printBitmap(bitmap,callback);
```

2. 打印图片(2)

函数： void **printBitmapCustom** (Bitmap bitmap,int type **ICallback** callback)

参数：

bitmap → 图片bitmap对象(最大宽度384像素，图片超过1M无法打印)。

type→ 目前有两种打印方式：

0 → 同方法 **printBitmap**()

1 → 阈值200的黑白化图片

2 → 灰度图片

callback → 结果回调

备注： 图片像素分辨率小于200万，且宽度根据纸张规格设置（58为384像素，80为576像素），如果超过纸张宽度将不显示

支持版本：P1-v3.2.0以上、P14g-v1.2.0以上、V1s-v3.2.0以上、V2-v1.0.0以上、T1-v2.4.0以上、T2、S2-v1.0.5以上、T1mini-v2.4.1以上、T2mini-v1.0.0以上

示例：

```
woyouService.printBitmapCustom(bitmap,callback);
```

1.2.9. 一维码与二维码的打印

编号	方 法
1	void **printBarCode**(String data, int symbology, int height, int width, int textPosition, **ICallback** callback) 打印一维条码
2	void **printQRCode**(String data, int modulesize, int errorlevel, **ICallback** callback) 打印QR条码
3	void **print2DCode**(String data, int symbology, int modulesize, int errorlevel, **ICallback** callback) 打印二维条码

1. 打印一维条码

函数： void **printBarCode(**String data, int symbology, int height, int width, int textPosition, **ICallback** callback**)**

参数：

data → 一维码内容

symbology → 条码类型（0 - 8）：

0 → UPC-A
1 → UPC-E
2 → JAN13(EAN13)
3 → JAN8(EAN8)
4 → CODE39
5 → ITF
6 → CODABAR
7 → CODE93
8 → CODE128

height → 条码高度, 取值 1 - 255, 默认：162

width → 条码宽度, 取值 2 - 6, 默认：2

textPosition → 文字位置（0 - 3）：

0 → 不打印文字
1 → 文字在条码上方
2 → 文字在条码下方
3 → 条码上下方均打印

callback → 结果回调

备注：条码种类不同有如下区别

编码	说明
code39	最长打印13个数字
code93	最长打印17个数字
ean8	要求8位数字（最后一位校验位），有效长度8个数字
ean13	有效长度13个数字，其中最后一位为校验位

ITF	要求输入数字，且有效小于14位，必须是偶数位
Codebar	要求0-9及6个特殊字符，最长打印18个数字
UPC-E	要求8位数字（最后一位校验位）
UPC-A	要求12位数字（最后一位校验位）
code128	Code128分三类： A类：包含大写字母、数字、标点等； B类：大小写字母，数字； C类：纯数字，复数字符，若为单数位，最后一个将忽略； 接口默认使用B类编码，若要使用A类、C类编码需在内容前面加“{A”、“{C”，例如：“{A2344A”，”{C123123”，”{A1A{B13B{C12”。

示例：

```
woyouService.printBarCode("1234567890", 8, 162, 2, 2, callback);
```

2. 打印QR条码

函数： void **printQRCode (**String data, int modulesize, int errorlevel, **ICallback** callback**)**

参数：

data → QR码内容

modulesize → QR码块大小，单位:点, 取值 4 至 16

errorlevel → 二维码纠错等级(0 - 3)：

0 → 纠错级别 L (7%)

1 → 纠错级别 M (15%)

2 → 纠错级别 Q (25%)

3 → 纠错级别 H (30%)

callback → 结果回调

备注：普通打印状态下在调用该方法后会直接输出打印，每个二维码块为 4 个像素点（小于 4 扫码解析有可能失败）。最大支持 version19（93*93）的模式。

示例：

```
woyouService.printQrCode("商米科技", 4, 3, callback);
```

3. 打印二维条码

函数： void **print2DCode(**String data, int symbology, int modulesize, int errorlevel, **ICallback** callback)

参数：

data → 二维码内容

symbology → 二维码类型

1 Qr（同**printQRCode**接口）

2 PDF417

3 DataMatrix

modulesize → 二维码有效块大小，根据码类型不同，支持的最佳块大小不同

Qr码	4～16（同**printQRCode**接口）
PDF417	1～4
DataMatrix	4～16

errorlevel → 二维码纠错等级，根据码类型不同，支持等级范围不同

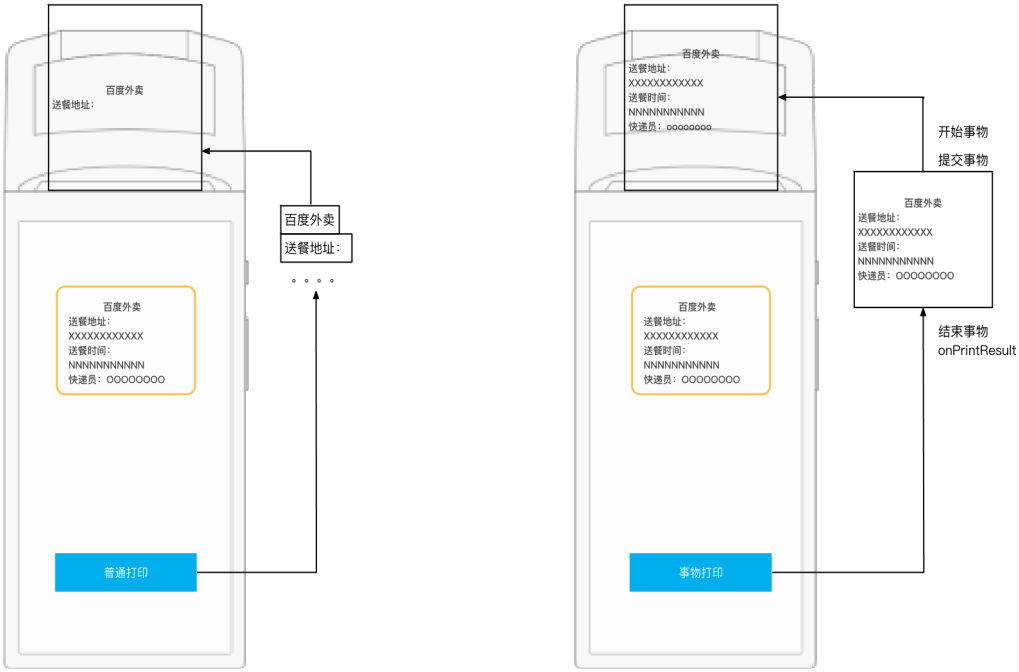
Qr码	0～3（同**printQRCode**接口）
PDF417	0～8
DataMatrix	默认使用ECC200自动纠错 不支持设置

callback → 结果回调。

备注：普通打印状态下在调用该方法后会直接输出打印；此接口在4.1.2版本后支持；

1.2.10. 事务打印

事务打印模式适用于需要控制打印内容并得到打印结果反馈(是否打印出小票)的需求，此模式相当于建立一个事务队列缓冲区，当开发者进入事务打印模式，将开启一个事务队列，可以向其中增加打印方法，此时打印机不会立刻打印内容，当提交事务后，打印机才会依次执行队列中的任务，执行结束将获得此次事务的结果反馈。



事务打印注意事项：

1. 当进入缓冲（事务）打印后，提交打印成功将返回成功结果，但遇到**打印机异常如缺纸、过热**等，将会**丢掉本次提交事务中所有指令任务**，同时**反馈异常**，即当一单任务执行前或执行中打印机异常，则此单不会打出；
2. 当指令打印和缓冲（事务）打印交替使用时，如果打印机异常，不会清除指令打印的内容！
3. 进入事务打印模式后，同样使用**1.2.x.** 中的其它接口方法输出内容，但**不会**立即打印输出，会将输出内容缓存到缓存区，当调用**exitPrinterBuffer()或commitPrinterBuffer()**等方法才会进行打印输出。

4. 事务打印结果回调在**ICallback**方法中的**onPrintResult**(int code, String msg)方法（会有一定的**耗时**，要等**物理打印出纸**，不推荐**单行频繁使用**事务打印，将会**影响打印速度**，推荐**整张小票**使用事务打印），对应返回code如下：
   a) 0 → 打印成功，msg为“Transaction print successful!”；
   b) 1 → 打印失败，msg为“Transaction print failed!”；
5. 整个事务打印伪代码示例如下：

```
enterPrinterBuffer(true) ——进入事务模式，此后所有命令不会立刻输出
printText(/*something*/)
printBitmap(/*bitmap resource*/)
…… 其它打印相关方法——打印一些内容
commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)——提交一次事务，此时打印机将开始打印，当打印成功或失败将在callback中返回
……等待上一次事务的返回
printText(/*something*/)
printBitmap(/*bitmap resource*/)
…… 其它打印相关方法——可以选择等待或不等待上一次事务的返回继续打印内容
commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)——继续提交下一次事务，此时打印机将继续打印
exitPrinterBuffer(true)/exitPrinterBufferWithCallback(true, callback)——退出事物模式时调用，如果在上一次提交后又输入新的数据则会继续打印否则不打印
```

6. 具体方法说明

编号	方 法
1	void **commitPrint**(TransBean[] tranBean, **ICallback** callback) lib 包事务打印专用接口
2	void **enterPrinterBuffer**(boolean clean) 进入事务打印模式
3	void **exitPrinterBuffer**(boolean commit) 退出事务打印模式
4	void **exitPrinterBufferWithCallback**(boolean commit, **ICallback** callback) 退出事务打印模式并回调结果
5	void **commitPrinterBuffer**() 提交事务打印
6	void **commitPrinterBufferWithCallback**(**ICallback** callback) 提交事务打印并回调结果

1. lib 包事务打印专用接口

函数： void **commitPrint** (TranBean[] tranBean, **ICallback** callback)

参数：

tranBean → 打印任务列表

callback → 结果回调

示例：

```
woyouService.commitPrint(tranBean, callback);
```

2. 进入事务模式

函数： void **enterPrinterBuffer(**Boolean clear**)**

参数：

clear → 是否清除缓冲区内容：

true → 清除上一次事务打印未提交的内容

false → 不清除上一次事务打印未提交的内容，下次提交将包含上次的内容

备注：

1、启用并进入事务打印模式，在这个模式下不会立刻打印数据，直到提交事物或退出提交事物，

2、除V1设备均支持事务模式

示例：

```
woyouService.enterPrinterBuffer(false);
```

3. 退出事务模式

函数： void **exitPrinterBuffer(**Boolean commit**)**

参数：

commit → 是否打印出缓冲区内容：

true → 会打印出事务队列中的所有内容

false → 不会打印事务队列中的内容，此内容将保存直到下次提交

备注：除V1设备均支持事务模式

示例：

```
woyouService.exitPrinterBuffer(true);
```

4. 退出事务打印模式并回调结果

函数： void **exitPrinterBuffer(**Boolean commit, **ICallback** callback**)**

参数：

commit → 是否打印出缓冲区内容：

true → 会打印出事务队列中的所有内容

false → 不会打印事务队列中的内容，此内容将保存直到下次提交

callback → 结果回调。

备注：除V1设备均支持事务模式

示例：

```
woyouService.exitPrinterBuffer(true);
```

5. 提交事务打印

函数：void **commitPrinterBuffer()**

备注：

1、将事务队列中的所有内容提交并打印，之后仍然处于事务打印模式；

2、除V1设备均支持事务模式

示例：

```
woyouService.commitPrinterBuffer();
```

6. 提交事务打印并回调结果

函数：void **commitPrinterBufferWithCallbacka(ICallback** callback**)**

参数：

callback → 结果回调

备注：除V1设备均支持事务模式

示例：

```
woyouService.commitPrinterBufferWithCallback(callback);
```

1.2.11. 走纸相关

编号	方 法
1	void **lineWrap(**int n, **ICallback** callback**)** 打印机走纸n行

1. 打印走纸n行

函数：void **lineWrap(**int n, **ICallback** callback**)**

参数：

n → 走纸行数

callback → 结果回调

备注：强制换行，结束之前的打印内容后走纸 **n** 行。

示例：

```
woyouService.printBitmap(3, callback);
```

1.2.12. 切刀（切纸）相关

编号	方 法
1	void **cutPaper(ICallback** callback**)** 切纸
2	int **getCutPagerTimes()** 获取切刀累计次数

1. 切纸

函数： void **cutPaper (ICallback** callback**)**

参数： callback → 结果回调

备注： 由于打印头和切刀有一定距离，调用接口将自动补全这段距离；
仅支持台式机带切刀功能机器。

示例：

```
woyouService.cutPager(callback);
```

2. 获取切刀次数

函数： int **getCutPaperTimes ()**

返回值： 切刀次数

备注： 仅支持台式机带切刀功能机器。

1.2.13. 钱箱相关

编号	方 法
1	void **openDrawer(ICallback** callback**)** 打开钱箱
2	int **getOpenDrawerTimes()** 获取钱箱累计打开次数
3	int **getDrawerStatus()** 获取当前的钱箱状态

1. 打开钱箱

函数： void **openDrawer (ICallback** callback**)**

参数：

callback → 结果回调

备注： 仅支持台式机带钱箱功能机器。

示例：

```
woyouService.openDrawer(callback);
```

2. 获取钱箱累计打开次数

函数： int **getCutPaperTimes ()**

返回值： 钱箱累计打开次数

备注： 仅支持台式机带钱箱功能机器。

3. 获取当前的钱箱状态

函数： int **getDrawerStatus()**

说明： 可以通过此接口在部分具有连接钱箱功能的机型上获取钱箱开关状态

备注： 目前仅对S2、T2、T2mini机器 v4.0.0版本以上支持此接口

1.2.14. 获取全局设置属性

商米机器可以通过设置配置一些打印样式，从而覆盖掉开发者自定义的样式，通过下列接口可以获取当前启用的配置状态；

编号	方 法
1	int **getForcedDouble()** 获取全局字体倍高倍宽状态
2	boolean **isForcedAntiWhite()** 获取全局字体反白样式使能
3	boolean **isForcedBold()** 获取全局字体加粗样式使能
4	boolean **isForcedUnderline()** 获取全局字体下划线样式使能
5	int **getForcedRowHeight**() 获取全局行高设定值
6	int **getFontName**() 获取当前的使用字体
7	int **getPrinterDensity**() 获取打印机浓度

备注：目前除获取打印机浓度接口其他接口仅在手持机V1、V1s、P1 v3.2.0版本以上及P14g v1.2.0以上支持

1.2.15. 顾显(客显）接口说明

mini系列机器具有顾显（客显）功能，可通过如下方法使用；

编号	方 法
1	void **sendLCDCommand**(int flag) 发送控制命令
2	void **sendLCDString**(String string, ILcdCallback callback) 发送单行文本内容
3	void **sendLCDDoubleString**(String topText, String bottomText, ILcdCallback callback) 发送双行文本内容
4	void **sendLCDMultiString**(String[] text, int[] align, ILcdCallback callback) 发送多行文本内容，每行内容根据权重自动分配大小
5	void **sendLCDFillString**(String string, int size, boolean fill, ILcdCallback callback) 发送单行文本，文本可设置字体大小，可设置填充方式
6	void **sendLCDBitmap**(Bitmap bitmap, ILcdCallback callback) 发送bitmap图显示

1. 发送顾显控制命令

函数： void **sendLCDCommand**(int flag)

参数： flag → 1 初始化 2 唤醒LCD 3休眠LCD 4清屏

备注： 仅支持台式机mini系列带顾显功能机器。

2. 发送单行文本内容

函数： void **sendLCDString**(String string, ILcdCallback callback)

参数： string → 显示的文本内容
callback → 异步回调执行结果

备注： 仅支持台式机mini系列带顾显功能机器，文本内容过长将不显示。

3. 发送双行文本内容

函数： void **sendLCDDoubleString**(String topText，String bottomText, ILcdCallback callback)

参数： topText → 显示上半区文本内容
bottomText → 显示下半区文本内容
callback → 异步回调执行结果

备注： 仅支持台式机mini系列带顾显功能机器，文本内容过长将不显示。

4. 发送多行文本内容

函数： void **sendLCDMultiString**(String[] text, int[] align, ILcdCallback callback)

参数： text → 显示各行文本内容的数组，根据数组元素数量确定行数，某行可为空则不显示内容
align → 各行文本所占区域的权重比，此数组元素大小必须同文本数组一致
callback → 异步回调执行结果

备注： 仅支持台式机mini系列带顾显功能机器，需要v4.0.0以上版本支持，文本内容过长将不显示。

5. 发送自定义大小单行文本内容

函数： void **sendLCDFillString**(String string, int size, boolean fill, ILcdCallback callback)

参数： string → 要显示的文本内容
size → 显示文本内容的字体大小，目前默认居中
fill → 是否拉伸字体填充满显示区域
callback → 异步回调执行结果

备注： 仅支持台式机mini系列带顾显功能机器，需要v4.0.0以上版本支持，文本内容过长将不显示。

6. 发送图片显示内容

函数： void **sendLCDBitmap**(Bitmap bitmap, ILcdCallback callback)

参数： bitmap → 要显示的图片内容
callback → 异步回调执行结果

备注： 仅支持台式机mini系列带顾显功能机器，顾显可显示最大图片像素为128*40。

1.2.16.标签打印说明

商米移动V系列产品现已支持打印标签纸的功能（如 V2pro、V2s），如需使用请参考如下说明：

前置条件：

1、设置模式——商米设备的打印机默认是用于打印热敏小票的，如果想使用标签功能请首先进入系统设置，找到内置打印（或商米打印）项，进入后选择打印模式，选择标签热敏模式，选择一次后机器将记录并作为标签打印机使用，如下图：**（若没有找到请更新系统到最新版本）**

开发者可以通过获取打印机模式接口 getPrinterMode()查询当前的打印机模式

2、学习标签纸——在选择了标签热敏模式时，可以看到新增了标签学习项，如果首次使用或者更换了不同类型的标签纸，请进入标签学习并点击标签学习，如下图

这将启动学习标签纸程序，请保证至少有三张标签，学习结束后会通过弹窗提示学习结果；

建议使用标签纸的规格为宽度50mm，高度30mm以上，标签间隙2mm以上，这样将达到最佳定位效果！

标签功能相关接口：

编号	方　法
1	void **labelLocate**() 定位下一张标签
2	void **labelOutput**() 输出标签到切纸口位置

打印标签：

标签接口仅提供了针对标签的定位输出操作，而标签内容需要开发者根据自己的需求，像热敏打印一样构造自己的内容，其中需要注意的是要控制打印内容的高度在标签纸内，否则将造成打印内容超出标签纸，打印到下一张标签或定位下一张标签不准的问题；

如果使用高度是30mm的标签纸，那么支持的高度是240（30mmx8点）像素高度的打印内容，打印机默认行间距是32点行，那么可以打印8行左右的文本内容或者一张384x240高度的图片

举例来说——构造一个简单标签内容如下

```
labelLocate()
//每次发送打印内容前均需定位标签位置

构造打印内容
setPrinterStyle(WoyouConsts.ENABLE_BOLD, WoyouConsts.ENABLE)
//设置字体加粗
linewrap(1, null)
//头部留一个空行
setAlignment(0, null)
//内容居左对齐
printText("商品：　　豆浆\n", null)
//打印内容，需要\n换行
printText("到期时间：　　12-13 14时\n", null)
//打印内容，需要\n换行
printBarCode("{C1234567890123456", 8, 90, 2, 2, null)
//打印条码，这里打印高度为90占据154点行的code128c类型条码（条码会预留上下一个空行）
linewrap(1, null)
//根据实际情况是否留白
```

这将输出一个基本充满标签区域的内容，如下

内容打完后根据需要是否循环继续执行labelLocate和打印内容

如果不需要打印可执行labelOutput()，这将使标签输出到切纸口方便剥离开发者可以根据增加其他API来设计自己需要的标签内容

打印多张标签：
如果只打印一张标签，只需要执行以下操作
labelLocate -> 打印标签 （-> labelOutput）
如果要打印多张标签，那么不用每次发送标签内容后执行labelOutput输出标签，只需要
labelLocate->打印标签1->labelLocate->打印标签2->labelLocate->打印标签3……
->labelLocate->打印标签n （->labelOutput）

备注：当前标签只能通过接口方式使用，暂不支指令集调用

1.3．接口返回说明

由于AIDL回调接口开发者经常实例化错误，针对远程导入库用 InnerResultCallback 替代原回调接口

1.3.1. InnerResultCallback接口方法说明

编号	方　法
1	void **onRunResult**(boolean isSuccess) 指令执行结果
2	void **onReturnString**(String result) 指令执行结果返回String
3	void **onRaiseException**(int code, String msg) 异常信息返回
4	void **onPrinterResult**(int code, String msg) 针对事物打印的反馈

1. 指令执行结果

函数： void **onRunResult** (boolean isSuccess)

参数：

isSuccess → 执行结果：

true → 执行成功

false → 执行失败

备注： 该接口返回处理结果是指命令处理执行结果，而不是打印出纸的处理结果。

2. 指令执行结果返回String

函数： void **onReturnString** (String result)

参数：

result → 执行结果

3. 异常信息返回

函数： void **onRaiseException** (int code, String msg)

参数：

code → 异常代码

msg → 异常描述

备注：见异常信息对照。

4. 针对事物打印的反馈

函数：void **onPrintResult (**int code, String msg**)**

参数：

code → 状态码：

0 →成功

1 →失败

msg → 成功时为null，失败时的异常描述

备注：该接口返回结果是指令处理及物理打印输出后的结果（会有一定的**耗时**，要等**物理打印出纸**）。

1.3.2. Callback对象示例

```
new InnerResultCallback{
        @Override
        public void onRunResult(boolean b) throws RemoteException {
            // 指令执行结果
        }
        @Override
        public void onReturnString(String s) throws RemoteException {
            // 指令执行结果返回String
        }
        @Override
        public void onRaiseException(int i, String s) throws RemoteException {
            // 异常信息返回
        }
        @Override
        public void onPrintResult(int i, String s) throws RemoteException {
             // 针对事物打印的反馈
        }
    }
```

1.3.3. 异常信息对照表

code	msg
-1	"command is not support,index #"; // # 代表第 # 个byte出错
-2	"# encoding is not support"; // # 代表第 # 个byte出错
-3	"oops,add task failed (the buffer of the task queue is 10M),please try later";
-4	"create command failed";
-5	"Illegal parameter";
-6	"param found null pointer"

2. 通过内置虚拟蓝牙调用打印机

2.1. 虚拟蓝牙简介

在蓝牙设备列表中可以看到一个已经配对，且永远存在的蓝牙设备“InnerPrinter”，这是由操作系统虚拟出来的打印机设备，实际并不存在。虚拟蓝牙支持Sunmi《esc/pos》指令。

其中有部分特殊的指令属于sunmi自定义指令，如：

功能	指令
开钱箱指令	byte［5］：0x10 0x14 0x00 0x00 0x00
切刀指令 全部切割	byte［4］：0x1d 0x56 0x42 0x00
切刀指令 切割（左边留一点不切）	byte［4］：0x1d 0x56 0x41 0x00

2.2. 虚拟蓝牙使用

1. 与该蓝牙设备建立连接。
2. 将指令和文本内容拼接转码为Bytes。
3. 发送给InnerPrinter。
4. 底层打印服务驱动打印设备完成打印。

注：**BluetoothUtil** 一个蓝牙工具类，用于连接虚拟蓝牙设备InnerPrinter。

2.2.1. 工具类BluetoothUtil，标准的蓝牙连接工具类

```
public class BluetoothUtil {

    private static final UUID PRINTER_UUID =
                    UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

    private static final String Innerprinter_Address = "00:11:22:33:44:55";

    public static BluetoothAdapter getBTAdapter() {
        return BluetoothAdapter.getDefaultAdapter();
    }

    public static BluetoothDevice getDevice(BluetoothAdapter bluetoothAdapter) {
        BluetoothDevice innerprinter_device = null;
        Set<BluetoothDevice> devices = bluetoothAdapter.getBondedDevices();
        for (BluetoothDevice device : devices) {
            if (device.getAddress().equals(Innerprinter_Address)) {
                innerprinter_device = device;
                break;
            }
        }
        return innerprinter_device;
    }

    public static BluetoothSocket getSocket(BluetoothDevice device) throws IOException {
        BluetoothSocket socket = device.createRfcommSocketToServiceRecord(PRINTER_UUID);
        socket.connect();
        return socket;
    }

    public static void sendData(byte[] bytes, BluetoothSocket socket) throws IOException {
        OutputStream out = socket.getOutputStream();
        out.write(bytes, 0, bytes.length);
        out.close();
    }
}
```

2.2.2. 蓝牙连接打印服务事例

1: Get BluetoothAdapter

```
BluetoothAdapter btAdapter = BluetoothUtil.getBTAdapter();

if (btAdapter == null) {
   Toast.makeText(getBaseContext(),"Please Open Bluetooth!", Toast.LENGTH_LONG).show();
   return;
}
```

2: Get Sunmi's InnerPrinter BluetoothDevice

```
BluetoothDevice device = BluetoothUtil.getDevice(btAdapter);
if (device == null) {
      Toast.makeText(getBaseContext(),"Please Make Sure Bluetooth have       InnterPrinter!",
      Toast.LENGTH_LONG).show();
      return;
}
```

3: Generate a order data ， user add data here

```
byte[] data = null;
```

4: Using InnerPrinter print data

```
BluetoothSocket socket = null; socket = BluetoothUtil.getSocket(device);
BluetoothUtil.sendData(data, socket);
```

2.2.3. 注意事项

需要在App的项目中**添加蓝牙权限声明才能使用蓝牙设备**：

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

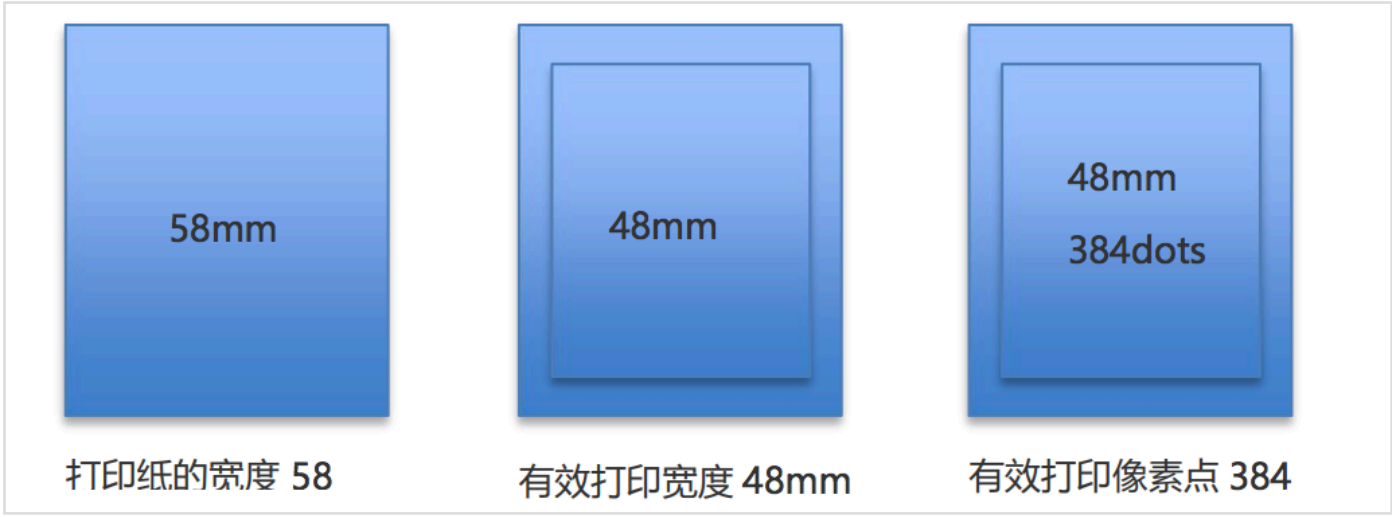
附录A 打印服务广播

通过广播的形式：用户需要建立一个广播接收者来监听广播。

功能	Action
打印机准备中	"woyou.aidlservice.jiuv5.INIT_ACTION"
打印机更新中	"woyou.aidlservice.jiuv5.FIRMWARE_UPDATING_ACITON"
可以打印	"woyou.aidlservice.jiuv5.NORMAL_ACTION"
打印错误	"woyou.aidlservice.jiuv5.ERROR_ACTION"
缺纸异常	"woyou.aidlservice.jiuv5.OUT_OF_PAPER_ACTION"
打印头过热异常	"woyou.aidlservice.jiuv5.OVER_HEATING_ACITON"
打印头温度恢复正常	"woyou.aidlservice.jiuv5.NORMAL_HEATING_ACITON"
开盖子	"woyou.aidlservice.jiuv5.COVER_OPEN_ACTION"
关盖子异常	"woyou.aidlservice.jiuv5.COVER_ERROR_ACTION"
切刀异常1－卡切刀	"woyou.aidlservice.jiuv5.KNIFE_ERROR_ACTION_1"
切刀异常2－切刀修复	"woyou.aidlservice.jiuv5.KNIFE_ERROR_ACTION_2"
打印机固件开始升级	"woyou.aidlservice.jiuv5.FIRMWARE_UPDATING_ACITON"
打印机固件升级失败	"woyou.aidlservice.jiuv5.FIRMWARE_FAILURE_ACITON"
未发现打印机	"woyou.aidlservice.jiuv5.PRINTER_NON_EXISTENT_ACITON"
未检测到黑标	"woyou.aidlservice.jiuv5.BLACKLABEL_NON_EXISTENT_ACITON"

附录B 打印服务F&Q

1. 打印纸张规格说明



注：Sunmi 打印机支持 58mm，80mm 打印纸，本文档以 58mm 打印纸为案例说明打印机的支持参数，80mm 打印纸规格类似；

一张 58 打印纸宽度为 58mm，有效打印宽度为 48mm。有效打印宽度一行为 384 个像素点。V1 纸槽深度 40mm，最多能放直径 40mm 的纸；

2. 商米打印机分辨率是多少?

打印机分辨率为 205DPI，计算公式如下

DPI=384dots/48mm=8dots/1mm=205dots/in=205

3. 如何查询有无打印机?

通过打印机状态查询接口 **updatePrinterState** ，当返回505时表示打印机离线，无法获取打印机

4. 为什么我执行了打印图片却没有打印出来?

首先，打印机是行缓冲的，目前机器对于除二维码外指令接口均以满一行打印输出，不满一行缓冲，所以当像调用printBitmap接口后发现图片没有打印，很可能是由于图片宽度达不到纸张宽度，没有输出，这个时候只要调用linewrap走纸接口即可将缓存的图片打印出来了。

但如果仍然没有打印出图片，那么就要检查一下接口的回调，看是否有失败的异常出现，一般情况下可能由于送的图片过大导致接口调用失败，sunmi 打印机支持最大打印图片分辨率大小 2M(非图片实际大小)，最大可显示宽度根据纸张规格决定。所以开发者想要打印合适的图像时需要自行缩放图片大小；

另外，如果开发者是通过蓝牙十六进制指令发送光栅位图，导致图片打印失败时，需要自行检查指令是否有错误，此时一个字节数据的错误就有可能影响整条数据的实现；

5. 我的条形码内容很长，小票显示不下，我应该选择哪种条码?

由于打印纸张宽度限制（手持机一般384像素点宽、台式机576像素点宽），打印位数较多的条码将不能全部显示，这个时候最好先调整条码的宽度设置，看是否能解决问题；

如果条码内容过于多，超过20位，这种情况一般是使用code128类型条码，请调用**printBarCode**接口选择code128码类型，目前此接口实现了混合编码，即可通过增加{A、{B、{C三种表示动态切换编码类型（A:数字、大写字母、控制字符；B:数字、大小字母、字符;C:双位数字），当是数字时切换为C类型，当是其他字符可选择切换为A、B类型，即可打印长条码，例如打印ABab1212：传入"{BABab{C1212";

混合编码需要打印服务4.0.0版本以上支持，如果开发者打印服务不支持或使用蓝牙方式打印，那么需要引入下列方法获取一组epson指令数组，通过sendRawData接口或蓝牙发送返回的数组即可；

其中，data为直接传入需要打印的条码内容即可；width由于默认的宽度最小值是2个像素，当更小时扫码速度严重下降，但为了打印超过25甚至更多的条码内容，可以选择设置为1；

```
public static byte[] getPrintBarCode(String data, int height, int width, int textposition){
    if(width < 1 || width > 6){width = 2;}
    if(textposition < 0 || textposition > 3){textposition = 0;}
    if(height < 1 || height>255){height = 162;}
    ByteArrayOutputStream buffer = new ByteArrayOutputStream();
    try{
        buffer.write(new byte[]{0x1D,0x66,0x01,0x1D,0x48,(byte)textposition,
                0x1D,0x77,(byte)width,0x1D,0x68,(byte)height,0x0A});
        byte[] barcode = checkCode128Auto(data.getBytes());
        buffer.write(new byte[]{0x1D,0x6B,0x49,(byte)(barcode.length)});
        buffer.write(barcode);
    }catch(Exception e){
        e.printStackTrace();
    }
    return buffer.toByteArray();
}
```

```
public static byte[] checkCode128Auto(byte[] data){
    ByteArrayOutputStream temp = new ByteArrayOutputStream();
    int pos = 0;
    boolean mode_C = true;
    temp.write(0x7b);
    temp.write(0x43);
    while(pos < data.length){
        if(data[pos] == '{' && pos + 1 != data.length){
            switch (data[pos + 1]){
                case 'A':
                case 'B':
                    if(mode_C){mode_C = false;temp.write(data[pos++]);temp.write(data[pos++])
                    }else{pos++;pos++;}
                    break;
                case 'C':
                    if(!mode_C){mode_C = true;temp.write(data[pos++]);temp.write(data[pos++]);
                    }else{pos++;pos++;}
                    break;
                default:
                    break;}
        }else if(pos + 1 == data.length){
            if(mode_C){temp.write(0x7b);temp.write(0x42);mode_C = false;}
            temp.write(data[pos++]);
        }else if(data[pos] < '0' || data[pos] > '9'){
            if(mode_C){temp.write(0x7b);temp.write(0x42);mode_C = false;}
            temp.write(data[pos++]);
        } else if(data[pos+1] < '0' || data[pos+1] > '9'){
            if(mode_C){temp.write(0x7b);temp.write(0x42);mode_C = false;}
            temp.write(data[pos++]);
            temp.write(data[pos++]);
        }else{
            if(!mode_C){temp.write(0x7b);temp.write(0x43);mode_C = true;}
            int left = data[pos] - '0';
            int right = data[pos + 1] - '0';
            int num = left*10 + right;
            if(num < 0 || num > 99){
                return null;
            }else{
                temp.write(num);
            }
            pos++;
            pos++;
        }
    }
    return temp.toByteArray();
}
```

6. 为什么我收不到打印接口回调结果?

首先，如果开发者用的是AIDL方式接入接口，需要注意是否使用匹配的AIDL资源：

P系列和V1S和V2 AIDL文档资源，T系列和S系列AIDL文档资源，Mini系列AIDL文档资源，V系列AIDL文档资源

在确认资源文件正确的前提下，需要注意当我们创建回调对象时，需要用引入Aidl生成的回调类的内部静态类Stub！

~~new ICallback(...)~~ —> new ICallback.Stub()

对于使用远程导入库的开发者，只要遵循文档使用已经封装好的InnerPrinterCallback等类就不会出现收不到结果的问题了。

7. 字符集的选择和设置

背景：由于内置打印机兼容二进制字节流的形式传输，所以通过字节码发送的打印文本内容涉及到字符集的选择和设置；默认的机器设置是多字节、GB18030字符编码；

内置打印机支持的单字节编码类型如下：

```
[参数] [编码] [国家]
0 "CP437"; 美国欧洲标准
2 "CP850"; 多语言
3 "CP860"; 葡萄牙语
4 “CP863"; 加拿大 -法语
5 “CP865"; 北欧
13 "CP857"; 土耳其语
14 "CP737"; 希腊语
15 "CP928";
16 "Windows-1252";
17 "CP866"; 西里尔文
18 "CP852"; 拉丁-中欧语
19 "CP858";
21 "CP874";
33 "Windows-775"; 波罗的海语
34 "CP855"; 西里尔文
36 "CP862"; 希伯来语
37 "CP864";
254 "CP855"; 西里尔文
```

内置打印机支持的多字节编码类型如下：

```
[参数] [编码]
0x00 ll 0x48 "GB18030";
0x01 ll 0x49 "BIG5";
0xFF  “utf-8”;
```

不同的国家可以根据本国或需要修改不同的机器设置使打印机识别接收到的打印内容数据流，比如打印CP437页码表的内容需要发送：

0x1C 0x2E ——设置为单字节编码类型

0x1B 0x74 0x00 ——设置为单字节编码页中CP437代码页表

打印CP866页码表的内容需要发送：

0x1C 0x2E ——设置为单字节编码类型

0x1B 0x74 0x11 ——设置为单字节编码页中CP866代码页表

打印繁体内容需要发送：

0x1C 0x26 ——设置为多字节编码类型

0x1C 0x43 0x01—— 设置为多字节编码页中BIG5编码

打印UTF-8编码内容（使用utf-8编码可以支持所有unicode字符集，可以保证打印所有内容）发送：

0x1C 0x26 ——设置为多字节编码类型

0x1C 0x43 0xFF—— 设置为多字节编码页中UTF8编码

8. 黑标打印说明

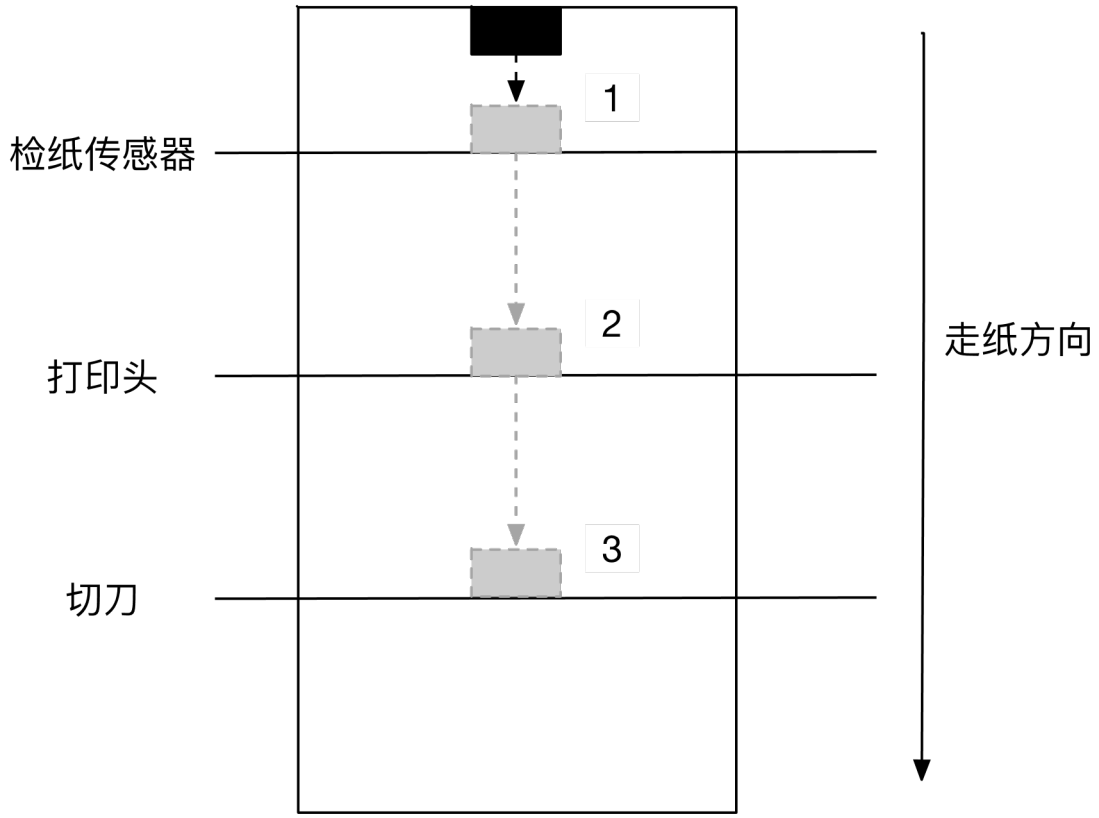
商米 T1 可使用符合规范的黑标打印纸进行打印。

对黑标打印纸的要求：商米 T1 的黑标打印与一般的黑标打印机不同，商米 T1 硬件上并不具备检测黑标的传感器，而是利用了检纸传感器对反射率不超过 6% （红外波长 700-1000nm 范围内）的材料认为是无纸的原理，制造了类似黑标打印的功能。因为原理不同，所以商米 T1 黑标打印模式无法像黑标打印机那样精确的定位黑标位置，存在一定误差）

对黑标位置的要求：黑标需要在纸张水平中间位置才能被检纸传感器器检测到。

商米 T1 黑标实现原理：

检纸传感器与打印头和切刀位置不在一个水平线上，纸张上的黑标会先经过检纸传感器（图 1 位置），再经过打印头（图 2 位置），最后到达切刀位置（图 3 位置）切纸出仓。



在黑标模式下，检纸传感器在检测到没纸后，打印机会自动走 7mm 的距离，如果走完之前检纸传感器还是没检测到有纸，就按照没纸处理。如果走完之前检测到有纸，就按照黑标处理。

当打印内容覆盖黑标位置，会继续打印即当黑标走到图 2 位置，仍然有打印任务，会覆盖黑标继续打印任务，直到打印完成。

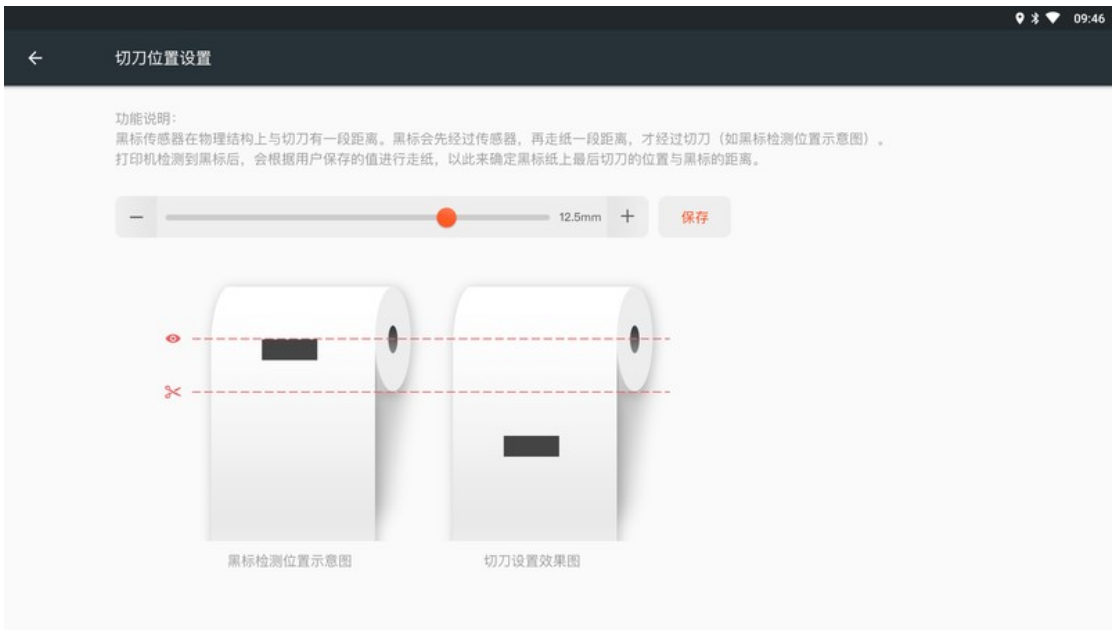
按照此机制，打印最后的位置与检纸传感器需要保持一定距离（打印最后的内容与黑标边沿距离 6mm）才能保证打印内容在 2 个黑标范围内。

黑标模式指令序列：

1. 发送打印内容
2. 输入走纸到黑标的指令{0x1c, 0x28, 0x4c, 0x02, 0x00, 0x42, 0x31}
3. 输入切刀指令{0x1d, 0x56, 0x00}

这样打印会在完成打印内容后，自动检索下一个黑标，并在指定位置切刀。

用户可以在 设置->打印->内置打印管理中对黑标模式和切刀位置进行修改。



9. 一些特殊符号如何打印?

如果有需求打印一些特殊符号如货币：$ ¥ € £ ₣等，有以下几种方法：

最简单的方式是使用我们提供的打印接口 printText，直接调用打印即可

printText("$ ¥ € £ ₣\n", null)

如果是通过蓝牙或者使用esc指令打印的开发者，由于不同的货币符号对应不同国家的页码，打印特殊符号的字符集不一定相同，建议使用我们指定的utf-8字符（机器默认的是GB18030），关于字符集的设置参考FAQ7；

例如还是打印$ ¥ € £ ₣这些符号：

需要发送指令 byte[] data = new byte[]{0x1C, 0x26, 0x1C, 0x43, 0xFF} 设置打印机接收utf-8字符编码

之后发送对应的符号数据 “$¥€ £ ₣”.getBytes(“utf-8”)

即：

sendData(new byte[]{0x1C, 0x26, 0x1C, 0x43, 0xFF}, null)

sendData(“$¥€ £ ₣”.getBytes(“utf-8”))