

SunmiPrinter 开发者文档

目 录

目 录	0
简介	1
1. 连接打印服务	1
1.1. 方式一：AIDL.....	1
1.1.1. AIDL 简介	1
1.1.2. AIDL 使用	1
1.1.3. AIDL 接口定义说明	2
1.2. 方式二：虚拟蓝牙	15
1.2.1. 虚拟蓝牙简介	15
1.2.2. 虚拟蓝牙使用	15
1.3. 方式三：HTML中的JS	18
1.3.1. JS简介	18
1.3.2. HTML 使用	18
2. 状态反馈	19
2.1. 打印机状态反馈	19
2.2. 指令回调说明	21
2.2.1. ICallback接口方法说明.....	21
2.2.2. Callback对象示例	22
2.2.3. 异常信息对照表	22
3. 打印服务参数介绍	23
3.1. 打印机分辨率	23
3.2. 查询有无打印机	23
3.3. 字体说明	23
3.4. 二维码说明	23
3.5. 图片说明	23
3.6. 条码说明	24
3.7. 字符集设置	25
3.8. 黑标打印说明	25
文档更新说明	28

简介

商米的V1、V1s、P1、P1-4G、T1、T1 mini、T2内置了缓存式热敏打印机，允许 App 直接打印热敏小票出来。商米产品的内置打印机一共有 2 种规格：

- 80mm 宽，带切刀。兼容 58mm，T1 搭载了这种打印机。
- 58mm 宽，不带切刀。V1、P1 搭载了这种打印机。

App 开发者可以使用 3 种方式调用内置热敏打印机：

1. 通过AIDL调用打印机
2. 通过内置虚拟蓝牙设备调用打印机
3. H5 Web页面通过JS桥调用打印机

下面以 V1 设备为例，介绍三种调用方式，该方案也适用于 V1s、P1、P1-4G、T1、T1 mini、T2 设备。

1. 连接打印服务

1.1. 方式一：AIDL

1.1.1. 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，同时Sunmi也支持《ESC/POS》指令集。

1.1.2. AIDL 使用

建立连接可分以下 5 步骤：

1. 在项目中添加资源文件中附带的[AIDL文件](#)（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 serviceConnected 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。

绑定服务示例

```

private ServiceConnection connService = new ServiceConnection() {
@Override
public void onServiceDisconnected(ComponentName name) {
    Toast.makeText(this, "service disconnected", Toast.LENGTH_LONG).show();
    setButtonEnable(false);
    woyouService = null;
    try {
        Thread.sleep(2000);
    } catch (InterruptedException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Binding();
}
@Override
public void onServiceConnected(ComponentName name, IBinder service) {
    woyouService = IWoyouService.Stub.asInterface(service);
    setButtonEnable(true);
    try {
        serviceVersion = woyouService.getServiceVersion();
        info.setText("service version : " + serviceVersion + "\n");
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
private void Binding(){
    Intent intent=new Intent();
    intent.setPackage("woyou.aidlservice.jiui5");
    intent.setAction("woyou.aidlservice.jiui5.IWoyouService");
    bindService(intent, connService, Context.BIND_AUTO_CREATE);
}

```

1.1.3. AIDL 接口定义说明

通过 AIDL 方式与打印服务建立连接后，就可调用 IWoyouService 封装好的常用打印指令接口（仅 aidl 连接方式才可调用 IWoyouService 接口函数）。

1.1.3.1. 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);
```

1.1.3.2. 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() 获取打印机型接口（打印头尺寸）
3	String getPrinterVersion() 获取打印机固件版本号
4	int getPrintedLength(ICallback callback) 获取打印头打印长度
5	Build. MODEL （常量） 设备名称
6	int updatePrinterState() 获取打印机的最新状态
7	String getServiceVersion() 获取打印服务版本号

1. 获取打印机型接口（打印头类型）

函数：String **getPrinterModal ()**

返回值：T1mini- 58； T1mini- 80。

2. 获取打印机的最新状态

函数：String **updatePrinterState ()**

返回值：

- 1 → 打印机正常
- 2 → 打印机更新状态
- 3 → 获取状态异常
- 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

备注：目前V1设备暂不支持该接口。

1.1.3.3. 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
```

1.1.3.4. 黑标打印

编号	方法 / 指令
1	int getPrinterMode () 获取打印机模式
2	int getPrinterBBMDistance () 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；

备注：仅支持 T1、T2设备。

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。

1.1.3.5. 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) 设置打印字体 (暂不支持)
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. 设置打印字体 (暂不支持)

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

参数：

typeface → 字体名称，目前中支持字体"gh"，gh 是一种等宽中文字体。

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设置 (现有版本暂时不支持设置字体)。

示例：

```
woyouService.setFontName("gh", callback);
```

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 → 字体名称（**现有版本暂时不支持设置字体，默认**）。

fontsize → 字体大小，只对该方法有效。

callback → 结果回调。

备注：字体设置只对本次有效。

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);
```

1.1.3.6. 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);
```

1.1.3.7. 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：最大宽度 384 像素，超过部分无法打印并且回调 callback 异常函数。

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1--**printBitmap**();

1 → 阈值200的黑白化图片

2 → 灰度图片

callback → 结果回调。

备注：最大宽度 384 像素，超过部分无法打印并且回调 callback 异常函数。

示例：

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

1.1.3.8. 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: 每种编码的最多打印内容。

示例:

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

2. 打印二维条码

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

参数:

- data → 二维码内容。

modulesize → 二维码块大小, 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。

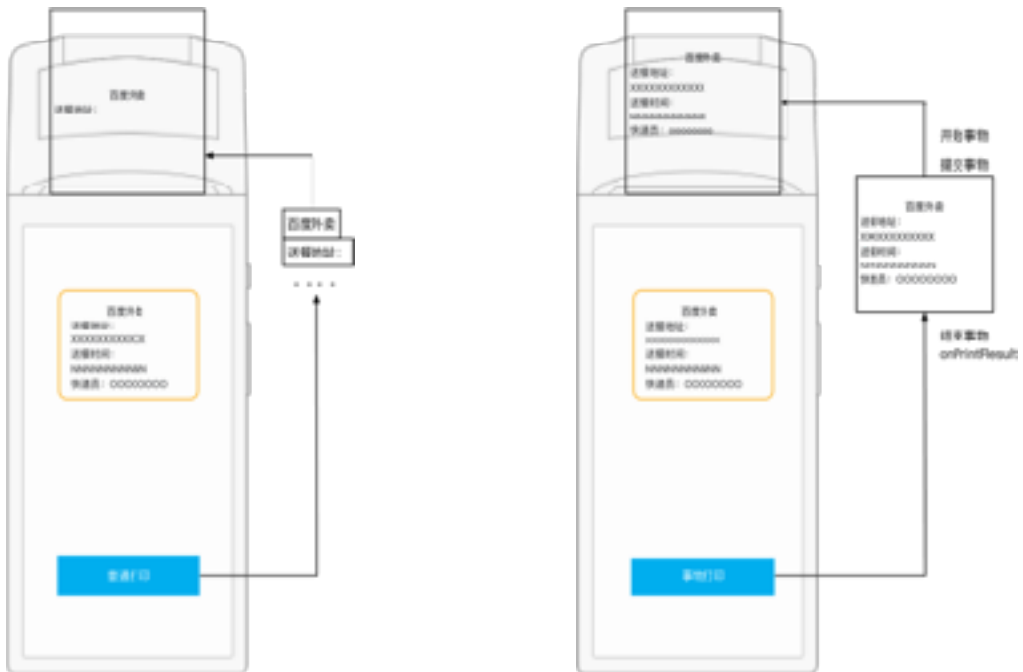
备注: 普通打印状态下在调用该方法后会直接输出打印, 更多参考 [二维码说明](#)。

示例:

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

1.1.3.9. 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1.1.3.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()、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（会有一些的耗时，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，推荐整张小票的使用事务打印），对应code如下：
 - a) 0 → 打印成功，msg为null；
 - b) 1 → 打印失败，msg为“woyou.aidlservice.jiuv5.ERROR_ACTION”。
5. 整个事务打印伪代码示例如下：

```

enterPrinterBuffer(true)
printText(/*something*/)
printBitmap(/*bitmap resource*/)
..... (1.1.3.中的其它打印相关方法)
commitPrinterBuffer()//exitPrinterBuffer(true)// commitPrinterBufferWithCallback(callback)/
exitPrinterBufferWithCallback(true, callback)//在 callback 的OnPrintResult 中回掉打印结果
    
```

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。

备注：使用 lib 的用户调用此接口开始事务打印，需打印服务 2.0.8 以上。

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：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，直到提交事务或退出提交事务，需打印服务 2.0.8 以上。

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：需打印服务 2.0.8 以上。

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：需在P1、V1s设备且打印服务 2.3.2 以上。

示例：

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

5. 提交事务打印

函数：void **commitPrinterBuffer**()

备注：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，需打印服务 2.0.8 以上。

示例：

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

6. 提交事务打印并回调结果

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

参数：

callback → 结果回调。

备注：需在P1、V1s设备且打印服务 2.3.2 以上。

示例：

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

1.1.3.10. 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) ;
```

1.1.3.11. 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：需设备支持。

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：需设备支持。

1.1.3.12. 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

1. 打开钱箱

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

参数：

callback → 结果回调。

备注：需设备支持。

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：需设备支持。

1.2. 方式二：虚拟蓝牙

1.2.1. 虚拟蓝牙简介

在V1的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

1.2.2. 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。

1.2.2.1. 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();
    }
}
```

1.2.2.2. 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 InnerPrinter!",
        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);
```

1.2.2.3. 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"/>
```

1.3. 方式三：HTML中的JS

1.3.1. JS简介

通过 JS 调用打印机本质上是通过 JS 桥在 HTML 中操作 android 原生的代码实现打印，通过蓝牙打印或者 AIDL 的方式实现打印。（sunmi 打印机本身不是网络打印机，网页应用无法直接与打印机通信，需要在 android 上有接受数据的应用）

1.3.2. HTML 使用

以下为具体的调用流程：

1. 在 HTML 中定义 android 中要使用的方法。

```
document.getElementsByTagName('a')[0].addEventListener('click',
function(){
    var a = "wellcome to sunmi"; javascript:lee.funAndroid(a);
    return false;
}, false);
```

2. 初始化 WebView.

```
WebView mWebView = (WebView) findViewById(R.id.wv_view);
// 设置编码
mWebView.getSettings().setDefaultTextEncodingName("utf-8");
// 支持 js
mWebView.getSettings().setJavaScriptEnabled(true); mWebView.setWebChromeClient(new
WebChromeClient());
```

3. 初始化打印服务

```
Intent intent = new Intent();
intent.setPackage("woyou.aidlservice.jiuiiv5");
intent.setAction("woyou.aidlservice.jiuiiv5.IWoyouService");
startService(intent);//Start printer service
bindService(intent, connService, Context.BIND_AUTO_CREATE);
```

4. 给 WebView 添加监听，在 onPageFinished 回调方法中调用

```
WebView.addJavascriptInterface(new JsObject(), 'lee');
```

在 JsObject 类中通过 @JavascriptInterface 定义在 HTML 中指定要操作的方法，在方法中通过调用 AIDL 打印方式打印一张小票。

```

//添加一个页面相应监听类 class
mWebView.setWebViewClient(new WebViewClientDemo());
WebViewClientDemo extends WebViewClient {
    @Override
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        // 当打开新链接时, 使用当前的 WebView, 不会使用系统其他浏览器
        view.loadUrl(url);
        return true;
    }
    @Override
    public void onPageFinished(WebView view, String url) {
        super.onPageFinished(view, url);
        /**
         * 注册 JavascriptInterface, 其中"lee"的名字随便取
         * 如果你用"lee", 那么在 html中只要用 lee.方法名()
         * 即可调用 MyJavascriptInterface 里的同名方法, 参数也要一致
         */
        mWebView.addJavascriptInterface(new JsObject(), "lee");
    }
}
class JsObject {
    @JavascriptInterface
    public void funAndroid(final String i) {
        Toast.makeText(getApplicationContext(), "calling the local method funAndroid by JS. " + i,
        Toast.LENGTH_SHORT).show();
        try {
            woyouService.printerSelfChecking(callback);//Using AIDL to print something.
        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }
}
}

```

5. 加载 HTML 文件, 在点击 HTML 中的按钮的时候就会打印一张小票了

```

// 载入包含 js 的 html
mWebView.loadData("", "text/html", null);
mWebView.loadUrl("file:///android_asset/test.html");//这里是您业务 html 所
所在的网页

```

2. 状态反馈

2.1. 打印机状态反馈

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"

2.2. 指令回调说明

2.2.1. ICallback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) 针对事务打印的反馈

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. 针对事务打印的反馈

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）。**

2.2.2. Callback对象示例

```

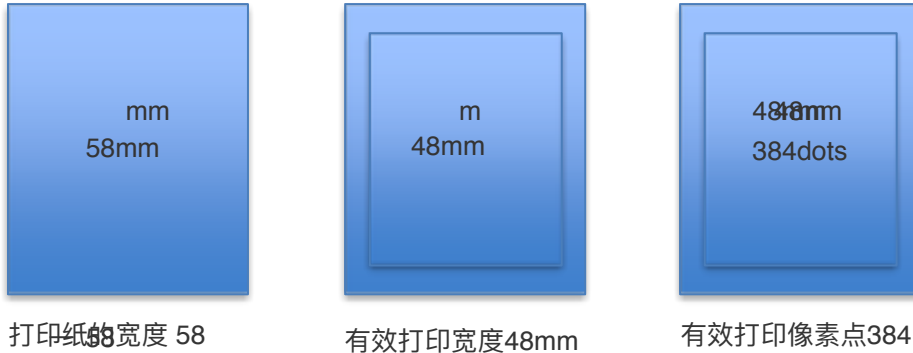
new ICallback.Stub(){
    @Override
    public void onRunResult(boolean b) throws RemoteException {
        // 指令执行结果
    }
    @Override
    public void onReturnString(String s) throws RemoteException {
        // 指令执行结果返回String
    }
    @Override
    public void onRaiseException(int i, String s) throws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

```

2.2.3. 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"

3. 打印服务参数介绍



打印纸的宽度 58

有效打印宽度48mm

有效打印像素点384

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

3.1. 打印机分辨率

打印机分辨率为 205DPI，计算公式如下

$$\text{DPI} = 384\text{dots} / 48\text{mm} = 8\text{dots} / 1\text{mm} = 205\text{dots/in} = 205$$

3.2. 查询有无打印机

T1 设备分为有打印机硬件和无打印机硬件两个版本，用户可以通过打印机查询接口从软件上判断。仅针对 T1 设备用户。

查询接口：

函数：Settings.Global.getInt(getContentResolver(), "sunmi_printer", 0);

参数：固定值。

返回值：

0 → 无打印机

1 → 有打印机

-1 → 正在查询中

3.3. 字体说明

默认字体 24，中文为 24*24 的矩阵，英文为 12*24 的矩阵。

3.4. 二维码说明

sunmi 打印机打印二维码，每个二维码块为 4 个像素点（小于 4 扫码解析不出）。最大支持 version19 (93*93) 的模式。

3.5. 图片说明

sunmi 打印机支持最大打印图片大小为 1M，最大支持宽度为 384 个像素点。如果用户需要打印超过 1M 或者超过宽度 384 像素的图片，需要手动压缩图片。


```

if( mBitmap == null ){
    / * * * * 1M 以内的图片 * * * * * /
    mBitmap = BitmapFactory.decodeResource(getResources(), R.raw.sunmi);
    / * * * * 超过 1M 的图片 * * * * * * * * * * /
    mBitmap1 = BitmapFactory.decodeResource(getResources(), R.raw.sunmi1);
}
/ * * * * * 压缩图片 * * * * * * * * * * /
double gh = (double)mBitmap.getWidth()/384;
if( mBitmap1 == null ){
    mBitmap2 = BitmapUtils.zoomBitmap(mBitmap1, 384, (int)(mBitmap.getHeight()/gh));
}
/ * * * * * 压缩完成 * * * * * * * * * * /
try {
    woyouService.setAlignment(1, callback);
    woyouService.printBitmap(mBitmap, callback);
//    woyouService.printBitmap(mBitmap2, callback);
    woyouService.lineWrap(3, null);
} catch (RemoteException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();//当图片过大, 也没有压缩, 打印服务会反馈异常
}
    
```

3.6. 条码说明

依赖版本 V1_printerservice2.1.13V1 (不同机型有略为不同)

编码	说明
code39	最长打印13个数字
code93	最长打印17个数字
code128	最长打印15个数字, Code128分三类: {A},{B},{C; A类: 包大写字母、数字、标点等, B类: 大小写字母, 数字; C类: 纯数字; 默认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{B13Bxc{C12”。
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)

3.7. 字符集设置

默认 character 设置为简体中文。

切换字符集命令如下：

```
close multibyte(single byte fit for europe area): 0x1C 0x2E
open multibyte(fit for east asia) : 0x1C 0x26
Set Single byte character set: 0x1B 0x74 [parameter]
```

Single byte character set parameter list:

```
[para] [character set] [area]
0 "CP437";
2 "CP850";
3 "CP860";
4 "CP863";
5 "CP865";
13 "CP857";
14 "CP737";
15 "CP928";
16 "Windows-1252";
17 "CP866";
18 "CP852";
19 "CP858";
21 "CP874";
33 "Windows-775";
34 "CP855";
36 "CP862";
37 "CP864";
254 "CP855";
Set multibyte character set: 1C 43 [parameter]
```

multibyte character set parameter list:

```
[para] [character set]
0x00 || 0x48 "GB18030";
//0x01 || 0x49 "BIG5";
0x02 || 0x50 "KSC5601";
(byte) 0xff "utf-8";
```

3.8. 黑标打印说明

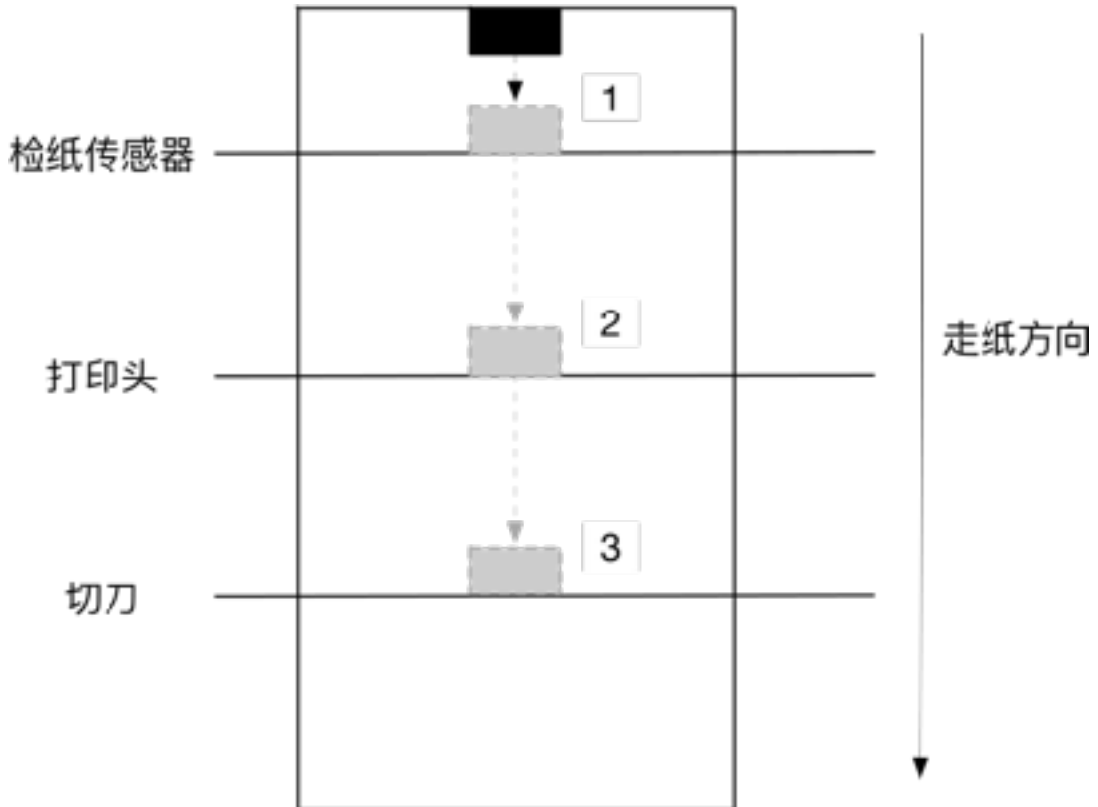
商米 T1 可使用符合规范的黑标打印纸进行打印。

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。

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。



文档更新说明

编号	更新日期	组件版本	更新内容	撰写人
1.0.0			原始版本	徐赟庭
1.1.170301			增加：打印图片的规格说明 增加：有无打印机硬件查询接口说明	徐赟庭
1.1.170315			增加：aidl切刀接口 增加：aidl 开钱柜接口 增加：aidl 获取切刀次数接口 增加：aidl 获取钱柜打开次数接口	徐赟庭
1.1.170322			增加：aidl带反馈的事务打印接口 增加：callback 事务打印结果反馈 修改：条码格式说明	徐赟庭
1.1.170329			增加：字符集修改说明	徐赟庭
1.1.170615			增加：事务打印说明	徐赟庭
1.1.170726			修改：AIDL调用的说明	徐赟庭
1.1.170802			增加：T1黑标打印说明	徐赟庭
1.1.170803			增加：T1黑标指令说明	徐赟庭
1.1.180523	2018/04/22		修改：全新文档格式，标准化整理	Darren
1.1.180601	2018/06/01		添加：打印图片扩展接口，ICallback实例说明	Darren
1.1.180612	2018/06/12		修改：加粗指令及部分描述	Darren
1.1.180629	2018/06/29		修改：ICallback接口方法错误及描述	Darren