PrinterSDK 使用说明书

> 版本：1.6.3
> 适用平台：Android 7.0+（API 24+）
> 包名：`com.hspos.printerSDK`
> 最后更新：2026-07-10
> English edition: PrinterSDK_User_Manual.md

目录

1. 产品简介
2. 系统要求与集成方式
3. SDK 架构说明
4. 快速开始
5. 权限配置
6. 打印机连接
7. 标准 POS 打印
8. 页模式打印
9. 标签模式打印
10. 状态查询与设备控制
11. 常量参考
12. 完整示例
13. 常见问题与排查
14. API 速查表
15. 函数方法说明（独立文档）
16. API 说明书（参考手册体例） · English
17. English User Manual

---

1. 产品简介

PrinterSDK 是一套面向 Android 平台的热敏打印机开发库，支持 ESC/POS 标准小票打印和 JPL 标签打印。通过统一的 `PrinterSDK` 入口类，开发者可以用同一套 API 完成设备发现、连接、打印与状态查询。

主要特性

特性	说明
多通道连接	USB、蓝牙、网络（TCP/IP）、串口
标准小票打印	文本、条码、二维码、PDF417、图片、切纸、钱箱、蜂鸣

特性	说明
页模式	坐标定位绘制文本、条码、图片、矩形
标签模式	标签纸专用布局与 JPL 指令
多编码支持	UTF-8、GBK、BIG5、Shift-JIS、EUC-KR
权限自动化	USB / 蓝牙权限由 SDK 自动申请（v1.5.0+）
异步网络连接	避免主线程 `NetworkOnMainThreadException`

对外核心文件

集成后，业务代码通常只需关注以下两个类：

```
import com.hspos.printerSDK.PrinterSDK;
import com.hspos.printerSDK.PrinterConstants;
import com.hspos.printerSDK.communication.PrinterConnection;
```

---

2. 系统要求与集成方式

2.1 环境要求

项目	要求
最低 Android 版本	API 24（Android 7.0）
目标 SDK	API 34
Java 版本	17
构建产物	AAR 库（`PrinterSDK.aar`）

2.2 集成 AAR 包

步骤 1： 将 `app/build/outputs/aar/PrinterSDK.aar`（或 `app-release.aar`）复制到宿主工程的 `libs/` 目录。

步骤 2： 在宿主 `build.gradle` 中添加依赖：

```
dependencies {
    implementation files('libs/PrinterSDK.aar')
}
```

步骤 3： 在宿主 `AndroidManifest.xml` 中声明所需权限（见 第 5 节）。

步骤 4： 构建 Release 版本：

```
./gradlew :app:assembleRelease
```

构建完成后，AAR 文件位于 `app/build/outputs/aar/` 目录。

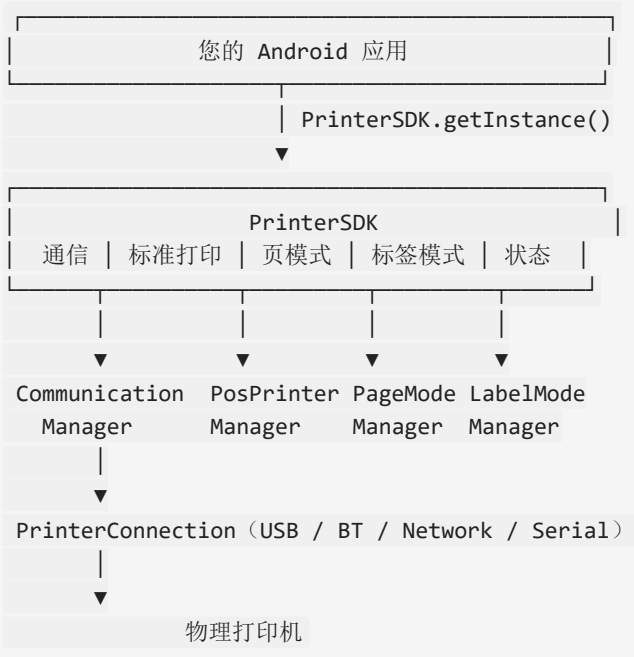
2.3 源码模块结构

```
com.hspos.printerSDK/
├── PrinterSDK.java              ← 对外统一入口（推荐使用）
├── PrinterConstants.java        ← 常量定义
├── CommunicationManager.java    ← 通信管理
├── PosPrinterManager.java       ← 标准 POS 打印实现
├── PageModeManager.java         ← 页模式实现
├── LabelModeManager.java        ← 标签模式实现
├── PrinterCommands.java         ← ESC/POS 指令生成
├── JPLCommands.java             ← JPL 标签指令生成
├── communication/
│   ├── PrinterConnection.java   ← 连接抽象接口
│   ├── usb/                     ← USB 连接
│   ├── bluetooth/               ← 蓝牙连接
│   ├── network/                 ← 网络连接
│   └── serial/                  ← 串口连接
└── utils/
    └── ImageUtils.java          ← 图片转打印数据
```

> **建议：** 业务层统一调用 `PrinterSDK`，不要直接调用内部 Manager 类，以便后续 SDK 升级时保持兼容。

---

3. SDK 架构说明



连接对象 `PrinterConnection` 是所有打印操作的第一参数，代表与某台打印机的通信通道。每次打印前需确保 `connection != null && connection.isConnected()`。

---

4. 快速开始

4.1 初始化

```
public class MainActivity extends AppCompatActivity {
    private PrinterSDK printerSDK;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        printerSDK = PrinterSDK.getInstance(this);
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        // 必须转发给 SDK，以便处理蓝牙权限回调
        printerSDK.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        printerSDK.release();
    }
}
```

4.2 最小打印流程

```
// 1. 连接打印机（以网络为例，推荐异步）
printerSDK.connectNetworkPrinterAsync("192.168.1.100", 9100,
    new CommunicationManager.NetworkConnectionCallback() {
        @Override
        public void onConnectionSuccess(PrinterConnection connection) {
            runOnUiThread(() -> doPrint(connection));
        }

        @Override
        public void onConnectionFailed(String errorMessage) {
            runOnUiThread(() -> Toast.makeText(context, errorMessage,
Toast.LENGTH_SHORT).show());
        }
    });

// 2. 执行打印（建议在子线程）
private void doPrint(PrinterConnection connection) {
    new Thread(() -> {
        printerSDK.initPrinter(connection);
        printerSDK.setAlignment(connection, PrinterConstants.ALIGNMENT_CENTER);
        printerSDK.printText(connection, "Hello PrinterSDK\n");
        printerSDK.cutPaper(connection, PrinterConstants.CUT_PARTIAL);
        printerSDK.disconnectPrinter(connection);
    }).start();
}
```

---

5. 权限配置

5.1 AndroidManifest 声明

根据实际使用的连接方式，在宿主 `AndroidManifest.xml` 中添加权限：

```
<!-- 蓝牙 -->
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
<uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

<!-- USB -->
<uses-permission android:name="android.permission.USB_PERMISSION" />
<uses-feature android:name="android.hardware.usb.host" android:required="false" />

<!-- 网络 -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<!-- 图片打印（从文件读取时需要） -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
```

5.2 各 Android 版本差异

Android 版本	蓝牙权限	其他
6.0–11 (API 23–30)	`BLUETOOTH`、`BLUETOOTH_ADMIN` + 位置权限	—
12+ (API 31+)	`BLUETOOTH_CONNECT`、`BLUETOOTH_SCAN` + 位置权限	—
13+ (API 33+)	同上	图片读取推荐 `READ_MEDIA_IMAGES`

5.3 SDK 自动权限处理（v1.5.0+）

连接方式	权限处理
USB	SDK 自动检查并申请 USB 权限
蓝牙	SDK 自动检查并申请蓝牙 + 位置权限
网络	无需运行时权限
串口	无需运行时权限（可能需要 root 设置设备权限）

Activity 中必须转发权限回调：

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
{
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    printerSDK.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

5.4 USB 设备过滤器（可选）

在 `res/xml/device_filter.xml` 中配置已知打印机的 Vendor ID / Product ID，以便 USB 插入时自动唤起应用：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <usb-device vendor-id="0x0483" product-id="0x5740" />
    <usb-device vendor-id="0x04b8" product-id="0x0202" />
</resources>
```

---

6. 打印机连接

所有连接方法返回 `PrinterConnection` 对象。连接失败时返回 `null`。

6.1 USB 连接

```
List<UsbDevice> devices = printerSDK.getUsbDevices();
if (devices != null && !devices.isEmpty()) {
    PrinterConnection conn = printerSDK.connectUsbPrinter(devices.get(0));
}
```

- SDK 会自动申请 USB 权限
- 也可手动检查：`hasUsbPermission(device)` / `requestUsbPermission(device)`

6.2 蓝牙连接

```
// 获取已配对设备
List<BluetoothDevice> paired = printerSDK.getBluetoothPairedDevices();

// 获取可用蓝牙设备（含已配对 + 扫描到的）
List<BluetoothDevice> devices = printerSDK.getBluetoothDevices();

// 获取打印机设备（过滤后的列表）
List<BluetoothDevice> printers = printerSDK.getBluetoothPrinters();

// 扫描新设备
printerSDK.startBluetoothDiscovery();
// 通过监听器接收发现结果
printerSDK.addBluetoothListener(new BluetoothPrinterManager.BluetoothDeviceListener() {
    @Override
    public void onDeviceFound(BluetoothDevice device) { /* 发现设备 */ }

    @Override
    public void onDiscoveryFinished() { /* 扫描结束 */ }

    @Override
    public void onBondStateChanged(BluetoothDevice device, int bondState) { /* 配对状态变化 */ }
});
// 不再需要监听时移除
printerSDK.removeBluetoothListener(listener);
printerSDK.stopBluetoothDiscovery();

// 连接
PrinterConnection conn = printerSDK.connectBluetoothPrinter(device);
```

> 蓝牙设备需先在系统设置中配对，或通过 SDK 扫描后配对。

6.3 网络连接

推荐：异步连接（可在主线程调用）

```
printerSDK.connectNetworkPrinterAsync("192.168.1.100", 9100,
    new CommunicationManager.NetworkConnectionCallback() {
        @Override
        public void onConnectionSuccess(PrinterConnection connection) {
            // ⚠ 回调在后台线程，更新 UI 需 runOnUiThread
            runOnUiThread(() -> { /* 连接成功 */ });
        }

        @Override
        public void onConnectionFailed(String errorMessage) {
            runOnUiThread(() -> { /* 显示错误 */ });
        }
    });
```

同步连接（不可在主线程调用）

```
// 必须在子线程中调用
PrinterConnection conn = printerSDK.connectNetworkPrinter("192.168.1.100", 9100);
```

参数	说明
IP 地址	打印机局域网 IP，如 `192.168.1.100`
端口	默认 `9100`（RAW 打印端口）
连接超时	5 秒（`NetworkPrinterConnection.TIMEOUT`）

6.4 串口连接

```
List<String> ports = printerSDK.getSerialDevices();
// 常见路径：/dev/ttyUSB0、/dev/ttyS0、/dev/ttyACM0
// 常见波特率：9600、19200、38400、57600、115200

PrinterConnection conn = printerSDK.connectSerialPrinter("/dev/ttyUSB0", 9600);
```

> 串口通常用于工业平板或 root 设备。普通手机一般不支持。

6.5 断开连接与通信辅助

```
// 断开连接
printerSDK.disconnectPrinter(connection);

// USB 权限（connectUsbPrinter 会自动申请，也可手动）
boolean ok = printerSDK.hasUsbPermission(usbDevice);
boolean granted = printerSDK.requestUsbPermission(usbDevice);

// 清除缓冲区 / 取消打印
printerSDK.clearPrinterBuffer(connection);
printerSDK.cancelPrint(connection);

// 高级：获取底层通信管理器
CommunicationManager comm = printerSDK.getCommunicationManager();
```

方法	说明
`disconnectPrinter(connection)`	断开打印机连接
`hasUsbPermission(device)`	检查 USB 权限
`requestUsbPermission(device)`	申请 USB 权限
`removeBluetoothListener(listener)`	移除蓝牙扫描监听器
`clearPrinterBuffer(connection)`	清除打印机缓冲区
`cancelPrint(connection)`	取消当前打印任务

方法	说明
`getCommunicationManager()`	获取底层 `CommunicationManager`（高级用法）
`onRequestPermissionsResult(...)`	转发运行时权限回调（见第 5 节）
`release()`	释放 SDK 全部资源

---

7. 标准 POS 打印

以下所有方法第一个参数均为 `PrinterConnection connection`，返回 `boolean` 表示操作是否成功。

7.1 初始化与重置

方法	说明
`initPrinter(connection)`	初始化打印机（ESC @）
`resetPrinter(connection)`	重置打印机
`sendSelfTest(connection)`	发送自检命令
`sendCommand(connection, byte[])`	发送原始指令

7.2 文本打印

方法	编码
`printText(connection, text)`	默认编码
`printTextInUTF8(connection, text)`	UTF-8
`printTextInGBK(connection, text)`	GBK（简体中文）
`printTextInBIG5(connection, text)`	BIG5（繁体中文）
`printTextInShiftJIS(connection, text)`	Shift-JIS（日文）
`printTextInEUCKR(connection, text)`	EUC-KR（韩文）

```
printerSDK.initPrinter(connection);
printerSDK.setAlignment(connection, PrinterConstants.ALIGNMENT_CENTER);
printerSDK.setChineseDoubleSize(connection, true);  // 汉字倍宽倍高 FS W
printerSDK.setBold(connection, true);
printerSDK.printTextInGBK(connection, "欢迎光临\n");
printerSDK.setBold(connection, false);
printerSDK.setChineseDoubleSize(connection, false);  // 恢复汉字正常尺寸
```

左右分列一行（80mm 小票常用）

方法	说明
`printTextLineLeftRight(connection, left, right)`	80mm 默认宽度 576 点、字符宽 12 点
`printTextLineLeftRight(connection, left, right, areaWidthDots, charWidthDots)`	自定义区域宽与字符宽

纸宽常量（`PrinterConstants`）：

常量	值	说明
`PAPER_WIDTH_58MM`	384	58mm 纸有效宽度（点）
`PAPER_WIDTH_80MM`	576	80mm 纸有效宽度（点）

```
// 80mm 商业小票：左商品名、右价格
printerSDK.initPrinter(connection);
printerSDK.setPrintAreaWidth(connection, PrinterConstants.PAPER_WIDTH_80MM);
printerSDK.setPrintAreaLeftMargin(connection, 0);
printerSDK.setAlignment(connection, PrinterConstants.ALIGNMENT_LEFT);

printerSDK.printTextLineLeftRight(connection, "Lorem ipsum", "$1.25");
printerSDK.printTextLineLeftRight(connection, "Sub Total", "$150.70");

// 合计行放大时，字符宽改为 24（GS ! 倍宽倍高后）
printerSDK.setBold(connection, true);
printerSDK.setTextScale(connection, 1, 1);
printerSDK.printTextLineLeftRight(connection, "TOTAL", "$155.99",
    PrinterConstants.PAPER_WIDTH_80MM, 24);
printerSDK.setTextScale(connection, 0, 0);
printerSDK.setBold(connection, false);
```

> 实现原理：`ESC $` 定位到行首打印左列，再 `ESC $` 定位到右侧打印右列并换行。倍宽倍高后请同步增大 `charWidthDots`（通常 12→24）。

7.2.1 字体倍宽倍高（三种指令，勿混淆）

打印机有 **三套不同的倍宽倍高指令**，适用场景不同：

方法	指令	码	适用
`setChineseDoubleSize()`	**FS W n**	`1C 57 n`	**汉字**倍宽倍高（开/关）
`setTextScale()`	GS ! n	`1D 21 n`	中外文字符尺寸（0–7 档）
`setPrintMode()`	ESC ! n	`1B 21 n`	打印模式倍宽/倍高

> 打印机会按 **最后收到的指令** 生效；混用时请注意顺序。

① 汉字倍宽倍高 — `setChineseDoubleSize()`（推荐打中文）

对应手册 **「选择/取消 汉字倍高倍宽」**，指令 **FS W n**：

```
// 开启汉字倍宽倍高 → 1C 57 01
printerSDK.setChineseDoubleSize(connection, true);
// 或
printerSDK.setChineseDoubleSize(connection, PrinterConstants.CHINESE_DOUBLE_SIZE_ON);

printerSDK.printTextInGBK(connection, "爱尚自身\n");

// 取消，恢复汉字正常尺寸 → 1C 57 00
printerSDK.setChineseDoubleSize(connection, false);
// 或传入 int（仅最低位有效：0/1）
printerSDK.setChineseDoubleSize(connection, PrinterConstants.CHINESE_DOUBLE_SIZE_NORMAL);
```

参数 `n` 仅 **最低位** 有效：`0`=取消，`1`=汉字倍宽倍高同时开启。

② 字符尺寸 — `setTextScale()`（GS ! n）

独立设置宽、高倍数（0=1倍，1=2倍 … 7=8倍），适用于中外文（HRI 字符除外）：

```
printerSDK.setTextScale(connection, 0, 0);  // 正常
printerSDK.setTextScale(connection, 1, 1);  // 2倍宽 + 2倍高
printerSDK.setTextScale(connection, 1, 0);  // 仅2倍宽
printerSDK.setTextScale(connection, 0, 1);  // 仅2倍高
```

③ 打印模式 — `setPrintMode()`（ESC ! n）

```
printerSDK.setPrintMode(connection, PrinterConstants.PRINT_MODE_DOUBLE_WIDTH);
printerSDK.setPrintMode(connection, PrinterConstants.PRINT_MODE_DOUBLE_HEIGHT);
printerSDK.setPrintMode(connection, PrinterConstants.PRINT_MODE_DOUBLE);
printerSDK.setPrintMode(connection, PrinterConstants.PRINT_MODE_NORMAL);
```

7.3 格式设置

方法	说明
`setAlignment(connection, alignment)`	对齐：左/中/右
`setChineseDoubleSize(connection, enable)`	**汉字倍宽倍高**（FS W n，`boolean`）
`setChineseDoubleSize(connection, n)`	**汉字倍宽倍高**（FS W n，`int`，0/1）
`setTextScale(connection, widthScale, heightScale)`	字符尺寸 GS ! n（0=正常，1=2倍 …）
`setBold(connection, bold)`	加粗
`setUnderline(connection, underline)`	下划线
`setReverse(connection, reverse)`	反白

方法	说明
`setLineSpacing(connection, spacing)`	行间距
`setCharSpacing(connection, spacing)`	字符间距
`setPrintMode(connection, mode)`	打印模式 ESC ! n（含倍宽/倍高）
`setTextUpsideDown(connection, upsideDown)`	上下倒置
`setTextRotate(connection, rotate)`	旋转 90°
`setTextLineHeight(connection, lineHeight)`	行高
`setPrintPosition(connection, position)`	打印起始位置
`setPrintAreaLeftMargin(connection, margin)`	左边距
`setPrintAreaWidth(connection, width)`	打印区域宽度
`setHorizontalAbsolutePrintPosition(connection, pos)`	水平绝对位置
`setHorizontalRelativePrintPosition(connection, offset)`	水平相对偏移
`setVerticalAbsolutePrintPosition(connection, pos)`	垂直绝对位置
`setVerticalRelativePrintPosition(connection, offset)`	垂直相对偏移
`setSingleByteMode(connection)`	单字节模式
`setCharacterSet(connection, charset)`	字符集
`setCharacterCodepage(connection, codepage)`	代码页
`setChineseMode(connection)`	中文模式
`setDoubleByteEncoding(connection, n)`	双字节编码
`setAsciiTextFontType(connection, fontType)`	ASCII 字体 A/B
`setAsciiTextCharRightSpacing(connection, spacing)`	ASCII 右间距
`setKanjiTextCharSpacing(connection, left, right)`	汉字间距

7.4 条码与二维码

如何选择条码 TYPE（重要）

`printBarcode()` 的第二个参数 `type` 必须传入 `PrinterConstants.BARCODE_TYPE_*` **常量**，不要手写 `0`、`1`、`2` 等数字，也不要与标签模式的条码类型编号（0–29）混用。

测试时不知道选哪个？ 优先使用 **Code 128**——兼容性最好，支持字母和数字，适合订单号、物流单号等大多数场景：

```
// 推荐：首次测试条码功能时使用此组合
printerSDK.printBarcode(connection,
    PrinterConstants.BARCODE_TYPE_CODE128,  // ← TYPE 选这个
    "1234567890",                            // 任意字母数字均可
    PrinterConstants.BARCODE_HRI_BELOW,
    80, 2);
```

按业务场景选择：

业务场景	选用常量	示例数据	数据要求
通用测试 / 订单号 / 物流单号	`BARCODE_TYPE_CODE128`	`"1234567890"`	字母、数字、符号均可
北美零售商品（UPC）	`BARCODE_TYPE_UPC_A`	`"012345678905"`	11–12 位数字
国际零售商品（EAN）	`BARCODE_TYPE_EAN13`	`"6901234567892"`	12–13 位数字
小包装商品	`BARCODE_TYPE_EAN8`	`"96385074"`	7–8 位数字
工业 / 仓储标签	`BARCODE_TYPE_CODE39`	`"ABC-1234"`	数字、大写字母及 `-. $/+%`
箱码（ITF）	`BARCODE_TYPE_ITF`	`"123456789012"`	必须为偶数位数字
图书馆 / 血库等	`BARCODE_TYPE_CODABAR`	`"A40156B"`	以 A–D 开头和结尾
北美零售压缩码	`BARCODE_TYPE_UPC_E`	`"01234565"`	需符合 UPC-E 编码规则
高密度编码	`BARCODE_TYPE_CODE93`	`"TEST93"`	字母数字

> **注意：** 标准小票（ESC/POS）与标签模式（JPL）的条码 `type` **不是同一套编号。**标准打印用 `BARCODE_TYPE_*`（值为 `0x41`–`0x49`）；标签打印用 `LABEL_BARCODE_TYPE_*`（值为 0–29），详见 第 9 节。

```
// 一维条码
printerSDK.printBarcode(connection,
    PrinterConstants.BARCODE_TYPE_CODE128,  // 条码类型
    "1234567890",                           // 数据
    PrinterConstants.BARCODE_HRI_BELOW,     // 文字位置
    80,                                     // 高度（点）
    2);                                     // 宽度

// 二维码（简易）
printerSDK.printQRCode(connection, "https://example.com", 3);

// 二维码（标准，可指定对齐和纠错级别）
printerSDK.printQRCodeStandard(connection,
    "https://example.com", 3,
    PrinterConstants.ALIGNMENT_CENTER,
    PrinterConstants.QR_ECC_LEVEL_M);

// 双二维码
printerSDK.printDoubleQRCode(connection, n, qr1Pos, qr1Data, qr1Ecc, qr1Ver,
    qr2Pos, qr2Data, qr2Ecc, qr2Ver);

// PDF417
printerSDK.printPDF417(connection, columns, rows, moduleWidth, rowHeight,
    errorCorrectionLevel, option, data);
```

条码格式预设置（可选，在 `printBarcode` 之前调用）：

方法	说明
`setBarcodeHRIPosition(connection, pos)`	HRI 文字位置（条码下方/上方等）
`setBarcodeHeight(connection, height)`	条码高度（点）
`setBarcodeWidth(connection, width)`	条码窄条宽度
`setBarcodeReadableTextFontType(connection, fontType)`	HRI 文字字体
`setBarcodeReadableTextPosition(connection, position)`	HRI 文字相对位置

```
printerSDK.setBarcodeHRIPosition(connection, PrinterConstants.BARCODE_HRI_BELOW);
printerSDK.setBarcodeHeight(connection, 80);
printerSDK.setBarcodeWidth(connection, 2);
printerSDK.printBarcode(connection, PrinterConstants.BARCODE_TYPE_CODE128, "1234567890",
    PrinterConstants.BARCODE_HRI_BELOW, 80, 2);
```

条码类型常量：

常量	类型	ESC/POS m 值	适用场景
`BARCODE_TYPE_UPC_A`	UPC-A	`0x41` (65)	北美 12 位商品码
`BARCODE_TYPE_UPC_E`	UPC-E	`0x42` (66)	北美 8 位压缩商品码
`BARCODE_TYPE_EAN13`	EAN-13	`0x43` (67)	国际 13 位商品码
`BARCODE_TYPE_EAN8`	EAN-8	`0x44` (68)	国际 8 位商品码

常量	类型	ESC/POS m 值	适用场景
`BARCODE_TYPE_CODE39`	Code 39	`0x45` (69)	工业标识、物流
`BARCODE_TYPE_ITF`	ITF	`0x46` (70)	箱码（偶数位数字）
`BARCODE_TYPE_CODABAR`	Codabar	`0x47` (71)	图书馆、血库
`BARCODE_TYPE_CODE93`	Code 93	`0x48` (72)	高密度编码
`BARCODE_TYPE_CODE128`	Code 128	`0x49` (73)	**通用首选**，测试推荐

HRI 文字位置：

常量	说明
`BARCODE_HRI_NONE`	不打印条码下方/上方文字
`BARCODE_HRI_BELOW`	在条码下方打印可读文字（常用）

7.5 图片打印

`printRasterImageFrom*` 会将图片转为黑白点阵后下发。自 **1.6.3** 起默认使用 **AUTO 压缩**（在 GS v 0 与 JBIG 间自动择优，传输更少、打印更快）。旧代码无需修改，行为向后兼容。

```
// 从文件（默认 AUTO 压缩）
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png");
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png", 384, 200);

// 从字节数组
printerSDK.printRasterImageFromData(connection, imageBytes);
printerSDK.printRasterImageFromData(connection, imageBytes, 384, 200);

// 从像素数组
int[] pixels = ...; // ARGB 格式
printerSDK.printRasterImageFromPixels(connection, pixels, width, height);
printerSDK.printRasterImageFromPixels(connection, pixels, width, height, 384, 200);

// 指定压缩模式（第 5 个参数 compressionMode，三种 API 均支持）
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png", 384, 200,
    PrinterConstants.RASTER_COMPRESSION_AUTO);   // 推荐，默认
printerSDK.printRasterImageFromData(connection, imageBytes, 384, 200,
    PrinterConstants.RASTER_COMPRESSION_JBIG);   // 强制 JBIG
printerSDK.printRasterImageFromPixels(connection, pixels, width, height, 384, 200,
    PrinterConstants.RASTER_COMPRESSION_GS_V0);  // 强制标准 GS v 0
```

压缩模式常量（`PrinterConstants.RASTER_COMPRESSION_*`）：

常量	值	指令	说明
`RASTER_COMPRESSION_GS_V0`	0	`GS v 0`（`1D 76 30`）	标准光栅，兼容性最好
`RASTER_COMPRESSION_LINE_RLE`	1	`US ( P`（`1F 28 50`）	逐行 RLE；**需固件支持**，多数机型不兼容
`RASTER_COMPRESSION_JBIG`	2	`US ( J`（`1F 28 4A`）	JBIG 压缩，体积更小、速度更快

常量	值	指令	说明
`RASTER_COMPRESSION_AUTO`	3	自动	**默认**；在 GS v 0 与 JBIG 间选传输量更小者

> - 图片会自动转为黑白点阵。宽度建议不超过打印机有效打印宽度（58mm 纸约 384 点，80mm 纸约 576 点）。
> - `targetHeight` 传 `0` 表示按宽度等比缩放。
> - 日常业务使用默认 `AUTO` 即可；若 JBIG 打印异常，可改为 `RASTER_COMPRESSION_GS_V0`。
> - `LINE_RLE` 仅建议在确认打印机支持 `US ( P` 指令时手动指定，**不要**用于 AUTO。

7.6 走纸与切纸

方法	说明
`feedLine(connection)`	换一行
`feedPaper(connection, lines)`	走纸 N 行
`feedPaper(connection)`	走纸（按点数）
`reverseFeed(connection, lines)`	反向走纸 N 行（ESC e）
`horizontalTab(connection)`	水平制表（HT）
`setTabPositions(connection, positions)`	设置制表位（ESC D）
`feedAndCutPaper(connection)`	走纸后切纸
`cutPaper(connection, cutType)`	切纸（全切/半切）
`printAndFeedDots(connection, n)`	打印并走纸 N 点
`printHorizontalLine(connection, start, end)`	打印水平线
`fixedHorizontalLine(connection)`	固定宽度水平线

```
printerSDK.cutPaper(connection, PrinterConstants.CUT_FULL);     // 全切
printerSDK.cutPaper(connection, PrinterConstants.CUT_PARTIAL);  // 半切

// 制表：先设置制表位（字符宽度），再发送 HT
printerSDK.setTabPositions(connection, new int[] { 8, 16, 24, 32 });
printerSDK.printText(connection, "列1");
printerSDK.horizontalTab(connection);
printerSDK.printText(connection, "列2");

// 反向走纸（需打印机支持）
printerSDK.reverseFeed(connection, 2);
```

7.7 外设控制

```
// 打开钱箱
printerSDK.openCashDrawer(connection, PrinterConstants.CASH_DRAWER_PIN2);
printerSDK.openCashDrawer(connection, PrinterConstants.CASH_DRAWER_PIN5);

// 蜂鸣
printerSDK.beep(connection);                    // 默认 1 次
printerSDK.beep(connection, 3, 2);              // 3 次，每次约 100ms
```

7.8 缓冲区控制

方法	说明
`clearBuffer(connection)`	清除缓冲区
`clearPrinterBuffer(connection)`	清除打印机缓冲区
`cancelPrint(connection)`	取消打印
`cancelPrintCmd(connection)`	取消打印命令

---

8. 页模式打印

页模式（Page Mode）允许在指定区域内按坐标绘制内容，适合制作复杂票据版式。

8.1 基本流程

```
printerSDK.initPrinter(connection);

// 1. 进入页模式（自动完成 GS P + ESC W + ESC T 初始化）
printerSDK.selectPageMode(connection);
// 或自定义打印区域（80mm 纸宽约 576 点）
printerSDK.selectPageModeEx(connection, 0, 0, 576, 800);

// 2. 可选：调整区域或方向
printerSDK.setPrintAreaInPageMode(connection, 0, 0, 576, 800);
printerSDK.setPrintAreaDirection(connection, PrinterConstants.PAGE_DIRECTION_LEFT_TO_RIGHT);

// 3. 绘制内容（坐标单位为点，原点在打印区域左上角）
printerSDK.drawTextGBKInPageMode(connection, 100, 100, "商品名称");
printerSDK.drawBarcodeInPageMode(connection, 100, 200, PrinterConstants.BARCODE_TYPE_CODE128,
"123456");
printerSDK.drawQRCodeInPageMode(connection, 100, 300, 3, PrinterConstants.QR_ECC_LEVEL_M,
"https://example.com");
printerSDK.drawImageFromFileInPageMode(connection, 100, 400, "/sdcard/logo.png", 100, 50);
printerSDK.drawRectangleBorderInPageMode(connection, 50, 50, 500, 100);   // 边框
printerSDK.drawFilledRectangleInPageMode(connection, 60, 60, 200, 80);     // 实心

// 4. 打印并退出
printerSDK.pagePrintPage(connection);   // ESC FF，将页缓冲内容打印出来
printerSDK.exitPageMode(connection);    // ESC S，返回标准模式
```

页模式排版辅助（80mm 小票常用）

方法	说明
`estimateTextWidthDots(text, charWidthDots)`	估算文本宽度（Font A 正常约 12 点/字符）
`drawTextCenteredInPageMode(connection, pageWidth, y, text)`	水平居中
`drawTextRightInPageMode(connection, pageWidth, y, text)`	右对齐
`drawTextLineLeftRightInPageMode(connection, pageWidth, y, left, right)`	一行左右分列

```
int w = PrinterConstants.PAPER_WIDTH_80MM;  // 576
int y = 40;
printerSDK.drawTextCenteredInPageMode(connection, w, y, "2847 Madison Avenue");
y += 36;
printerSDK.drawTextLineLeftRightInPageMode(connection, w, y, "Bank Card", "**** **** **** 1234");
```

NV 位图 / Logo（打印机内存，FS p）

打印机出厂或工具预存的 Logo 保存在 NV 存储器中，无需传图即可打印：

方法	说明
`printNvBitmap(connection, nvNumber, mode)`	标准模式打印 NV 位图
`drawNvBitmapInPageMode(connection, x, y, nvNumber, mode)`	页模式指定坐标
`drawNvBitmapCenteredInPageMode(connection, pageWidth, y, nvNumber, mode, imageWidthDots)`	页模式水平居中

`mode` 使用 `PrinterConstants.NV_BITMAP_MODE_*`（0 正常 / 1 倍宽 / 2 倍高 / 3 倍宽倍高）。

```
// 页模式顶部居中 Logo（nvNumber、imageWidthDots 按打印机手册调整）
printerSDK.drawNvBitmapCenteredInPageMode(connection, 576, 40, 1,
    PrinterConstants.NV_BITMAP_MODE_NORMAL, 256);
y += 100;  // 预留 Logo 高度
```

> **注意：** 页模式与标签模式（JPL）是不同指令体系，请勿混用。`selectPageMode()` / `selectPageModeEx()` 会自动设置移动单位、打印区域和默认方向；若只调用 `selectPageMode()` 而不设置区域，打印机无法正确定位内容。

8.2 页模式 API 列表

分类	方法
模式控制	`selectPageMode`、`exitPageMode`、`selectPageModeEx`（3 个重载）
区域设置	`setMovementUnits`、`pageSetPageArea`、`pageSetPageDrawDirection`、`setPrintAreaInPageMode`、`setPrintAreaDirection`
打印控制	`pagePrintPage`、`pageClearPage`
文本绘制	`drawTextInPageMode`、`drawTextUTF8InPageMode`、`drawTextGBKInPageMode`、`drawTextBIG5InPageMode`、`drawTextShiftJISInPageMode`、`drawTextEUCKRInPageMode`
页模式排版	`drawTextCenteredInPageMode`、`drawTextRightInPageMode`、`drawTextLineLeftRightInPageMode`、`estimateTextWidthDots`
NV 位图	`printNvBitmap`、`drawNvBitmapInPageMode`、`drawNvBitmapCenteredInPageMode`
图形绘制	`drawRectangleInPageMode`、`drawRectangleBorderInPageMode`、`drawFilledRectangleInPageMode`
条码/码	`drawBarcodeInPageMode`、`drawQRCodeInPageMode`
图片	`drawImageFromFileInPageMode`、`drawImageFromDataInPageMode`、`drawImageFromPixelsInPageMode`

---

9. 标签模式打印

标签模式适用于标签纸打印机，使用 JPL 指令集进行精确布局。

9.1 基本流程

```
// 1. 启用标签模式
printerSDK.enableLabelMode(connection);

// 2. 开始标签页（x, y, width, height, rotation）
printerSDK.labelPageBegin(connection, 0, 0, 576, 400, 0);

// 3. 绘制内容
printerSDK.printLabelTextGBK(connection, "商品标签", 50, 50, 24, 0);
printerSDK.printLabelBarcode(connection,
    PrinterConstants.LABEL_BARCODE_TYPE_CODE128, "1234567890", 50, 100, 80, 2);
printerSDK.printLabelQRCode(connection, "https://example.com", 50, 200, 3);
printerSDK.drawLabelImageFromFile(connection, 300, 50, "/sdcard/logo.png");
printerSDK.drawLabelLine(connection, 50, 300, 500, 300);
printerSDK.drawLabelRect(connection, 50, 320, 500, 380);

// 4. 打印
printerSDK.backPaperToPrintPosition(connection);
printerSDK.labelPagePrint(connection);       // 打印 1 份
printerSDK.labelPagePrint(connection, 3);    // 打印 3 份

// 5. 退出标签模式
printerSDK.disableLabelMode(connection);
```

9.2 标签模式 API 列表

分类	方法
模式控制	`enableLabelMode`、`disableLabelMode`、`labelPageBegin`、`backPaperToPrintPosition`、`labelPagePrint`（含份数重载）
文本	`printLabelText`、`printLabelTextUTF8`、`printLabelTextGBK`（均含完整参数重载）
条码	`printLabelBarcode`（2 个重载）、`setLabelBarcodeTextPosition`
二维码/PDF417	`printLabelQRCode`（2 个重载）、`printLabelPDF417Code`（2 个重载）
图片	`drawLabelImageFromFile`、`drawLabelImageFromData`、`drawLabelImageFromPixels`、`drawLabelImageAdvanced`
图形	`drawLabelLine`（2 个重载）、`drawLabelRect`（2 个重载）、`drawLabelBox`

---

10. 状态查询与设备控制

SDK 提供 **两种** 状态获取方式，适用场景不同：

方式	API	方向	SDK 是否解析	适用场景
便捷判断	`getPrinterStatus` / `isPrinterOnline` 等	App 主动查	部分（简化常量）	快速判断在线/缺纸/开盖
主动查询	`queryRTStatus` / `queryTransmitStatus`	App 主动查	否（返回原始字节）	按 ESC/POS 标准自行解析

> **说明：** 除「便捷判断」外，SDK **不对状态字节做业务解析**，位含义请按下文对照表在业务层自行判断。不同机型可能有差异，以实机为准。

10.1 SDK 便捷状态 API

```
int status = printerSDK.getPrinterStatus(connection);
boolean online    = printerSDK.isPrinterOnline(connection);
boolean hasPaper  = printerSDK.hasPaper(connection);
boolean coverOpen = printerSDK.isCoverOpen(connection);
```

内部通过 `queryRTStatus(connection, STATUS_TYPE_PRINTER, 5000)` 取 1 字节，再与下列**简化常量**比较：

常量	值	`getPrinterStatus` 含义	用于
`STATUS_OFFLINE`	`0x00`	查询失败或无响应	默认返回值
`STATUS_ONLINE`	`0x01`	与在线比较	`isPrinterOnline()`
`STATUS_PAPER_OUT`	`0x02`	缺纸位（按位与）	`hasPaper()` 取反
`STATUS_COVER_OPEN`	`0x04`	开盖位（按位与）	`isCoverOpen()`

> 上述常量为 SDK **封装用简化值**，与 ESC/POS 标准状态字节的位定义**不完全相同**。精确判断请用 10.2 的原始字节按位解析。

10.2 主动查询（DLE EOT / GS r）

API

```
// DLE EOT: 10 04 n
byte[] rt = printerSDK.queryRTStatus(connection, type, timeoutMs);

// GS r: 1D 72 n（部分机型更兼容，返回格式与 DLE EOT 同类）
byte[] tx = printerSDK.queryTransmitStatus(connection, type, timeoutMs);
```

查询类型 `type`（`PrinterConstants.STATUS_TYPE_*`）

常量	n 值	说明	典型返回
`STATUS_TYPE_PRINTER`	1	打印机状态	1 字节
`STATUS_TYPE_OFFLINE`	2	脱机/机械状态	1 字节
`STATUS_TYPE_ERROR`	3	错误状态	1 字节
`STATUS_TYPE_PAPER_SENSOR`	4	纸传感器状态	1 字节

`timeoutMs` 为等待响应毫秒数；无响应返回 `null`。

n=1 打印机状态字节（DLE EOT 1 / GS r 1）

Bit	0	1
0	（固定 0）	—
1	纸量正常	纸将尽（Near-end，视机型）
2	（固定 0）	—
3	在线	**脱机**
4	（固定 0）	—
5	正常	**恢复请求**（Recoverable error）
6	（固定 0）	—
7	（固定 0）	—

n=2 脱机状态字节

Bit	0	1
0	（固定 0）	—
1	关盖	**开盖**
2	未走纸	**走纸键走纸中**
3	正常	**纸尽停止**
4	正常	**错误**（Recoverable）
5	正常	**不可恢复错误**
6	（固定 0）	—
7	（固定 0）	—

n=3 错误状态字节

Bit	0	1
0	（固定 0）	—
1	正常	**切刀错误**

Bit	0	1
2	正常	**不可恢复错误**
3	正常	**可恢复错误**
4	正常	**打印头温度异常**
5	（固定 0）	—
6	（固定 0）	—
7	（固定 0）	—

n=4 纸传感器状态字节

Bit	含义（0 / 1）
0, 1	纸将尽传感器：00=纸充足，11=纸将尽（视机型）
2, 3	纸尽传感器：00=有纸，11=纸尽（视机型）
4–7	（固定 0）

10.3 状态相关 API 汇总

方法	说明
`getPrinterStatus(connection)`	便捷取状态 int
`isPrinterOnline(connection)`	是否在线
`hasPaper(connection)`	是否有纸
`isCoverOpen(connection)`	是否开盖
`queryRTStatus(connection, type, timeout)`	DLE EOT 主动查询
`queryTransmitStatus(connection, type, timeout)`	GS r 主动查询

---

11. 常量参考

> **说明：** `PrinterConstants` 中的 int 常量主要供**集成方**在业务代码中使用（如 `PrinterConstants.ALIGNMENT_CENTER`）。SDK 内部对多数参数做透传，因此 IDE 可能显示部分常量「no usages」，这是正常现象。旧版条码常量 `BARCODE_CODE128` 等已标记 `@Deprecated`，请改用 `BARCODE_TYPE_*` 系列。

11.1 对齐与切纸

```
PrinterConstants.ALIGNMENT_LEFT      // 0 - 左对齐
PrinterConstants.ALIGNMENT_CENTER    // 1 - 居中
PrinterConstants.ALIGNMENT_RIGHT     // 2 - 右对齐

PrinterConstants.CUT_FULL            // 0 - 全切
PrinterConstants.CUT_PARTIAL         // 1 - 半切
```

11.2 二维码

```
PrinterConstants.QRCODE_SIZE_SMALL   // 1
PrinterConstants.QRCODE_SIZE_NORMAL  // 2
PrinterConstants.QRCODE_SIZE_LARGE   // 3

PrinterConstants.QR_ECC_LEVEL_L      // 7%  纠错
PrinterConstants.QR_ECC_LEVEL_M      // 15% 纠错
PrinterConstants.QR_ECC_LEVEL_Q      // 优化纠错
PrinterConstants.QR_ECC_LEVEL_H      // 30% 纠错
```

11.3 页模式方向

```
PrinterConstants.PAGE_DIRECTION_LEFT_TO_RIGHT   // 0
PrinterConstants.PAGE_DIRECTION_BOTTOM_TO_TOP   // 1
PrinterConstants.PAGE_DIRECTION_RIGHT_TO_LEFT   // 2
PrinterConstants.PAGE_DIRECTION_TOP_TO_BOTTOM   // 3
```

11.4 打印模式组合

```
PrinterConstants.PRINT_MODE_NORMAL          // 正常
PrinterConstants.PRINT_MODE_DOUBLE_WIDTH    // 倍宽
PrinterConstants.PRINT_MODE_DOUBLE_HEIGHT   // 倍高
PrinterConstants.PRINT_MODE_DOUBLE          // 倍宽+倍高
PrinterConstants.PRINT_MODE_BOLD            // 加粗
PrinterConstants.PRINT_MODE_UNDERLINE       // 下划线
PrinterConstants.PRINT_MODE_REVERSE         // 反白
```

11.4.1 光栅图压缩模式

用于 `printRasterImageFrom*` 第 5 个参数 `compressionMode`：

```
PrinterConstants.RASTER_COMPRESSION_GS_V0    // 0 — GS v 0 标准光栅
PrinterConstants.RASTER_COMPRESSION_LINE_RLE   // 1 — US ( P 逐行 RLE（需固件支持）
PrinterConstants.RASTER_COMPRESSION_JBIG       // 2 — US ( J JBIG 压缩
PrinterConstants.RASTER_COMPRESSION_AUTO       // 3 — 自动（默认，GS v 0 ⇔ JBIG）
```

11.5 钱箱与行间距

```
PrinterConstants.CASH_DRAWER_PIN2          // 钱箱引脚 2
PrinterConstants.CASH_DRAWER_PIN5          // 钱箱引脚 5

PrinterConstants.LINE_SPACING_DEFAULT      // 默认行间距
PrinterConstants.LINE_SPACING_1_8          // 1/8 英寸
PrinterConstants.LINE_SPACING_1_6          // 1/6 英寸
```

11.6 状态相关常量

完整位定义见 第 10 节。

```
// SDK 便捷判断（简化值，非 ESC/POS 原位）
PrinterConstants.STATUS_OFFLINE            // 0x00
PrinterConstants.STATUS_ONLINE             // 0x01
PrinterConstants.STATUS_PAPER_OUT          // 0x02
PrinterConstants.STATUS_COVER_OPEN         // 0x04

// 主动查询 type（DLE EOT / GS r）
PrinterConstants.STATUS_TYPE_PRINTER       // 1
PrinterConstants.STATUS_TYPE_OFFLINE       // 2
PrinterConstants.STATUS_TYPE_ERROR         // 3
PrinterConstants.STATUS_TYPE_PAPER_SENSOR  // 4
```

11.7 条码类型（标准小票 ESC/POS）

> 测试条码功能时，不确定选哪个 TYPE 请直接用 `BARCODE_TYPE_CODE128`。完整选型说明见 7.4 节。

```
// 通用首选（测试推荐）
PrinterConstants.BARCODE_TYPE_CODE128   // 0x49 — 字母数字均可

// 商品零售
PrinterConstants.BARCODE_TYPE_EAN13     // 0x43 — 12~13 位数字
PrinterConstants.BARCODE_TYPE_EAN8      // 0x44 — 7~8 位数字
PrinterConstants.BARCODE_TYPE_UPC_A     // 0x41 — 11~12 位数字
PrinterConstants.BARCODE_TYPE_UPC_E     // 0x42 — UPC-E 压缩码

// 其他
PrinterConstants.BARCODE_TYPE_CODE39      // 0x45 — 工业标识
PrinterConstants.BARCODE_TYPE_ITF         // 0x46 — 偶数位数字
PrinterConstants.BARCODE_TYPE_CODABAR     // 0x47 — 以 A~D 起止
PrinterConstants.BARCODE_TYPE_CODE93      // 0x48 — 高密度

// HRI 可读文字位置
PrinterConstants.BARCODE_HRI_NONE         // 0 — 不显示
PrinterConstants.BARCODE_HRI_BELOW        // 2 — 条码下方（常用）
```

11.8 标签模式条码类型（JPL，与 ESC/POS 不同）

标签模式 `printLabelBarcode()` 的 `type` 使用 **0–29** 编号，与上方 `BARCODE_TYPE_*` 无关：

```
PrinterConstants.LABEL_BARCODE_TYPE_CODE128  // 8 — 标签模式 Code 128（常用）
```

---

12. 完整示例

12.1 小票打印（蓝牙）

```
public void printReceipt() {
    List<BluetoothDevice> devices = printerSDK.getBluetoothPairedDevices();
    if (devices == null || devices.isEmpty()) {
        Toast.makeText(this, "未找到已配对打印机", Toast.LENGTH_SHORT).show();
        return;
    }

    PrinterConnection conn = printerSDK.connectBluetoothPrinter(devices.get(0));
    if (conn == null || !conn.isConnected()) {
        Toast.makeText(this, "连接失败", Toast.LENGTH_SHORT).show();
        return;
    }

    new Thread(() -> {
        try {
            printerSDK.initPrinter(conn);

            // 标题
            printerSDK.setAlignment(conn, PrinterConstants.ALIGNMENT_CENTER);
            printerSDK.setChineseDoubleSize(conn, true);
            printerSDK.setBold(conn, true);
            printerSDK.printTextInGBK(conn, "=== 收银小票 ===\n\n");

            // 正文
            printerSDK.setBold(conn, false);
            printerSDK.setChineseDoubleSize(conn, false);
            printerSDK.setAlignment(conn, PrinterConstants.ALIGNMENT_LEFT);
            printerSDK.printTextInGBK(conn, "商品：苹果 x2\n");
            printerSDK.printTextInGBK(conn, "单价：5.00 元\n");
            printerSDK.printTextInGBK(conn, "合计：10.00 元\n\n");

            // 分隔线
            printerSDK.fixedHorizontalLine(conn);

            // 条码
            printerSDK.printBarcode(conn,
                PrinterConstants.BARCODE_TYPE_CODE128, "20260707001",
                PrinterConstants.BARCODE_HRI_BELOW, 80, 2);

            // 二维码
            printerSDK.printQRCode(conn, "https://example.com/order/001", 3);

            // 切纸
            printerSDK.feedLine(conn);
            printerSDK.cutPaper(conn, PrinterConstants.CUT_PARTIAL);

            runOnUiThread(() -> Toast.makeText(this, "打印完成", Toast.LENGTH_SHORT).show());
        } catch (Exception e) {
            runOnUiThread(() -> Toast.makeText(this, "打印失败: " + e.getMessage(),
Toast.LENGTH_SHORT).show());
        } finally {
```

```
            printerSDK.disconnectPrinter(conn);
        }
    }).start();
}
```

12.2 标签打印（网络）

```
public void printLabel() {
    printerSDK.connectNetworkPrinterAsync("192.168.1.100", 9100,
        new CommunicationManager.NetworkConnectionCallback() {
            @Override
            public void onConnectionSuccess(PrinterConnection conn) {
                new Thread(() -> {
                    printerSDK.enableLabelMode(conn);
                    printerSDK.labelPageBegin(conn, 0, 0, 576, 400, 0);

                    printerSDK.printLabelTextGBK(conn, "产品名称", 30, 30, 24, 0);
                    printerSDK.printLabelBarcode(conn,
                        PrinterConstants.LABEL_BARCODE_TYPE_CODE128,
                        "6901234567890", 30, 80, 80, 2);
                    printerSDK.printLabelQRCode(conn, "https://example.com/p/001", 30, 200, 3);
                    printerSDK.drawLabelRect(conn, 10, 10, 566, 390);

                    printerSDK.labelPagePrint(conn);
                    printerSDK.disableLabelMode(conn);
                    printerSDK.disconnectPrinter(conn);
                }).start();
            }

            @Override
            public void onConnectionFailed(String errorMessage) {
                runOnUiThread(() -> Toast.makeText(context, errorMessage,
Toast.LENGTH_SHORT).show());
            }
        });
}
```

---

13. 常见问题与排查

13.1 连接问题

现象	可能原因	解决方案
蓝牙连接失败 / bt socket closed	缺少蓝牙或位置权限	确认 Manifest 声明 + 转发 `onRequestPermissionsResult`
蓝牙扫描无结果	未授予位置权限	申请 `ACCESS_FINE_LOCATION`
USB 连接失败	未授权 USB 设备	调用 `connectUsbPrinter` 时 SDK 会自动申请；检查 USB 线材

现象	可能原因	解决方案
网络连接失败	IP 不可达或端口未开放	ping 打印机 IP，确认端口 9100 开放
NetworkOnMainThreadException	主线程同步连接	改用 `connectNetworkPrinterAsync`
串口连接失败	设备无串口或权限不足	确认设备路径存在，可能需要 root

13.2 打印问题

现象	可能原因	解决方案
中文乱码	编码不匹配	使用 `printTextInGBK` 等对应编码方法
图片不打印	文件路径无效或无读取权限	检查存储权限与文件路径
切纸无效	打印机不支持切刀	改用 `feedLine` 走纸代替
条码无法扫描	数据格式与类型不匹配	确认 `type` 使用 `BARCODE_TYPE_*` 常量；测试时优先用 `BARCODE_TYPE_CODE128` + `"1234567890"`；EAN/UPC 需符合位数要求

13.3 开发建议

1. **打印操作放子线程**：避免阻塞 UI，网络连接使用异步方法
2. **检查返回值**：所有打印方法返回 `boolean`，失败时应记录日志
3. **及时释放资源**：Activity 销毁时调用 `printerSDK.release()`
4. **使用常量**：不要硬编码对齐、条码类型等数值
5. **连接后先 init**：打印前调用 `initPrinter(connection)` 确保打印机就绪
6. **一次连接多次打印**：同一 `PrinterConnection` 可复用，无需每次重连

13.4 日志调试

SDK 使用 Android Log 输出，Tag 包括：

Tag	模块
`PrinterSDK`	主入口
`CommunicationManager`	通信管理
`NetworkPrinterConnection`	网络连接
`BluetoothPrinterManager`	蓝牙管理
`UsbPrinterManager`	USB 管理

在 Logcat 中过滤以上 Tag 可快速定位问题。

---

14. API 完整速查表

以下列出 `PrinterSDK.java` 中全部 **公开方法**（含重载）。

14.1 单例与生命周期

方法	返回值	说明
`getInstance(Context)`	`PrinterSDK`	获取单例
`release()`	`void`	释放 SDK 资源
`onRequestPermissionsResult(int, String[], int[])`	`void`	转发权限回调
`getCommunicationManager()`	`CommunicationManager`	获取底层通信管理器

14.2 连接与通信

方法	返回值	说明
`getUsbDevices()`	`List<UsbDevice>`	USB 设备列表
`connectUsbPrinter(UsbDevice)`	`PrinterConnection`	连接 USB
`hasUsbPermission(UsbDevice)`	`boolean`	检查 USB 权限
`requestUsbPermission(UsbDevice)`	`boolean`	申请 USB 权限
`getBluetoothPairedDevices()`	`List<BluetoothDevice>`	已配对蓝牙
`getBluetoothDevices()`	`List<BluetoothDevice>`	可用蓝牙设备
`getBluetoothPrinters()`	`List<BluetoothDevice>`	打印机类蓝牙
`connectBluetoothPrinter(BluetoothDevice)`	`PrinterConnection`	连接蓝牙
`startBluetoothDiscovery()`	`boolean`	开始蓝牙扫描
`stopBluetoothDiscovery()`	`void`	停止蓝牙扫描
`addBluetoothListener(Listener)`	`void`	添加蓝牙监听器
`removeBluetoothListener(Listener)`	`void`	移除蓝牙监听器
`connectNetworkPrinter(String, int)`	`PrinterConnection`	同步网络连接
`connectNetworkPrinterAsync(String, int, Callback)`	`void`	异步网络连接
`getSerialDevices()`	`List<String>`	串口设备路径列表
`connectSerialPrinter(String, int)`	`PrinterConnection`	连接串口
`disconnectPrinter(PrinterConnection)`	`void`	断开连接
`clearPrinterBuffer(PrinterConnection)`	`boolean`	清除打印机缓冲区
`cancelPrint(PrinterConnection)`	`boolean`	取消打印

方法	返回值	说明
`queryRTStatus(PrinterConnection, int, int)`	`byte[]`	查询实时状态（DLE EOT）
`queryTransmitStatus(PrinterConnection, int, int)`	`byte[]`	传送状态（GS r）

14.3 标准打印 — 初始化与文本

方法	说明
`initPrinter(connection)`	初始化打印机
`resetPrinter(connection)`	重置打印机
`sendSelfTest(connection)`	自检
`sendCommand(connection, byte[])`	发送原始指令
`printText(connection, text)`	默认编码文本
`printTextInUTF8(connection, text)`	UTF-8 文本
`printTextInGBK(connection, text)`	GBK 文本
`printTextInBIG5(connection, text)`	BIG5 文本
`printTextInShiftJIS(connection, text)`	Shift-JIS 文本
`printTextInEUCKR(connection, text)`	EUC-KR 文本

14.4 标准打印 — 格式设置

方法	说明
`setAlignment(connection, alignment)`	对齐
`setChineseDoubleSize(connection, enable)`	汉字倍宽倍高（boolean）
`setChineseDoubleSize(connection, n)`	汉字倍宽倍高（int，0/1）
`setTextScale(connection, w, h)`	字符尺寸 GS ! n
`setPrintMode(connection, mode)`	打印模式 ESC ! n
`setBold(connection, bold)`	加粗
`setUnderline(connection, underline)`	下划线
`setReverse(connection, reverse)`	反白
`setLineSpacing(connection, spacing)`	行间距
`setCharSpacing(connection, spacing)`	字符间距
`setSingleByteMode(connection)`	单字节模式
`setCharacterSet(connection, charset)`	字符集
`setCharacterCodepage(connection, codepage)`	代码页

方法	说明
`setChineseMode(connection)`	汉字模式 FS &
`setDoubleByteEncoding(connection, n)`	双字节编码
`setAsciiTextFontType(connection, fontType)`	ASCII 字体 A/B
`setTextUpsideDown(connection, upsideDown)`	上下倒置
`setTextRotate(connection, rotate)`	旋转 90°
`setTextLineHeight(connection, lineHeight)`	行高
`setAsciiTextCharRightSpacing(connection, spacing)`	ASCII 右间距
`setKanjiTextCharSpacing(connection, left, right)`	汉字左右间距
`setPrintPosition(connection, position)`	打印起始位置
`setPrintAreaLeftMargin(connection, margin)`	左边距
`setPrintAreaWidth(connection, width)`	打印区域宽度
`setHorizontalAbsolutePrintPosition(connection, pos)`	水平绝对位置
`setHorizontalRelativePrintPosition(connection, offset)`	水平相对偏移
`setVerticalAbsolutePrintPosition(connection, pos)`	垂直绝对位置
`setVerticalRelativePrintPosition(connection, offset)`	垂直相对偏移

14.5 标准打印 — 条码 / 二维码 / 图片

方法	说明
`printBarcode(connection, type, data, hri, height, width)`	一维条码
`setBarcodeHRIPosition(connection, pos)`	条码 HRI 位置
`setBarcodeHeight(connection, height)`	条码高度
`setBarcodeWidth(connection, width)`	条码窄条宽度
`setBarcodeReadableTextFontType(connection, fontType)`	条码 HRI 字体
`setBarcodeReadableTextPosition(connection, position)`	条码 HRI 文字位置
`printQRCode(connection, content, size)`	二维码（简易）
`printQRCodeStandard(connection, content, size, align, ecc)`	二维码（标准）
`printDoubleQRCode(connection, …)`	双二维码
`printPDF417(connection, …)`	PDF417
`printRasterImageFromFile(connection, path)`	图片（文件，AUTO 压缩）
`printRasterImageFromFile(connection, path, w, h)`	图片（文件+尺寸，AUTO）
`printRasterImageFromFile(connection, path, w, h, mode)`	图片（文件+尺寸+压缩模式）
`printRasterImageFromData(connection, bytes)`	图片（字节，AUTO）

方法	说明
`printRasterImageFromData(connection, bytes, w, h)`	图片（字节+尺寸，AUTO）
`printRasterImageFromData(connection, bytes, w, h, mode)`	图片（字节+尺寸+压缩模式）
`printRasterImageFromPixels(connection, pixels, w, h)`	图片（像素，AUTO）
`printRasterImageFromPixels(connection, pixels, w, h, tw, th)`	图片（像素+尺寸，AUTO）
`printRasterImageFromPixels(connection, pixels, w, h, tw, th, mode)`	图片（像素+尺寸+压缩模式）

14.6 标准打印 — 走纸 / 切纸 / 外设 / 缓冲

方法	说明
`feedLine(connection)`	换行
`feedPaper(connection, lines)`	走纸 N 行
`feedPaper(connection)`	走纸（点数）
`reverseFeed(connection, lines)`	反向走纸 N 行
`horizontalTab(connection)`	水平制表
`setTabPositions(connection, positions)`	设置制表位
`feedAndCutPaper(connection)`	走纸并切纸
`cutPaper(connection, cutType)`	切纸
`printAndFeedDots(connection, n)`	打印并走纸 N 点
`printHorizontalLine(connection, start, end)`	水平线
`fixedHorizontalLine(connection)`	固定宽度水平线
`openCashDrawer(connection, drawerIndex)`	开钱箱
`beep(connection)`	蜂鸣（默认）
`beep(connection, count, duration)`	蜂鸣（自定义）
`clearBuffer(connection)`	清除缓冲区
`cancelPrintCmd(connection)`	取消打印命令

14.7 页模式

方法	说明
`selectPageMode(connection)`	进入页模式
`exitPageMode(connection)`	退出页模式
`selectPageModeEx(connection, …)`	进入页模式（3 个重载）
`pagePrintPage(connection)`	打印页内容
`pageClearPage(connection)`	清除页缓冲

方法	说明
`setMovementUnits(connection, x, y)`	移动单位
`pageSetPageArea(connection, x, y, w, h)`	设置页区域
`pageSetPageDrawDirection(connection, dir)`	页绘制方向
`setPrintAreaInPageMode(connection, x, y, w, h)`	页模式打印区域
`setPrintAreaDirection(connection, dir)`	页模式区域方向
`drawRectangleInPageMode(connection, mode, x, y, w, h)`	绘制矩形
`drawRectangleBorderInPageMode(connection, …)`	矩形边框
`drawFilledRectangleInPageMode(connection, …)`	实心矩形
`drawTextInPageMode(connection, x, y, text)`	绘制文本
`drawTextUTF8InPageMode(connection, …)`	UTF-8 文本
`drawTextGBKInPageMode(connection, …)`	GBK 文本
`drawTextBIG5InPageMode(connection, …)`	BIG5 文本
`drawTextShiftJISInPageMode(connection, …)`	Shift-JIS 文本
`drawTextEUCKRInPageMode(connection, …)`	EUC-KR 文本
`drawBarcodeInPageMode(connection, …)`	绘制条码
`drawQRCodeInPageMode(connection, …)`	绘制二维码
`drawImageFromFileInPageMode(connection, …)`	绘制图片（文件）
`drawImageFromDataInPageMode(connection, …)`	绘制图片（数据）
`drawImageFromPixelsInPageMode(connection, …)`	绘制图片（像素）

14.8 标签模式

方法	说明
`enableLabelMode(connection)`	启用标签模式
`disableLabelMode(connection)`	禁用标签模式
`labelPageBegin(connection, x, y, w, h, rotation)`	开始标签页
`backPaperToPrintPosition(connection)`	回纸到打印位
`labelPagePrint(connection)`	打印标签
`labelPagePrint(connection, printNum)`	打印 N 份标签
`printLabelText(connection, …)`	标签文本（2 个重载）
`printLabelTextUTF8(connection, …)`	标签 UTF-8（2 个重载）
`printLabelTextGBK(connection, …)`	标签 GBK（2 个重载）
`printLabelBarcode(connection, …)`	标签条码（2 个重载）

方法	说明
`setLabelBarcodeTextPosition(connection, position)`	标签条码文字位置
`printLabelQRCode(connection, …)`	标签二维码（2 个重载）
`printLabelPDF417Code(connection, …)`	标签 PDF417（2 个重载）
`drawLabelImageFromFile(connection, …)`	标签图片（文件）
`drawLabelImageFromData(connection, …)`	标签图片（数据）
`drawLabelImageFromPixels(connection, …)`	标签图片（像素）
`drawLabelImageAdvanced(connection, …)`	标签图片（高级）
`drawLabelLine(connection, …)`	标签线段（2 个重载）
`drawLabelRect(connection, …)`	标签矩形（2 个重载）
`drawLabelBox(connection, …)`	标签实心框

14.9 状态查询

方法	返回值	说明
`getPrinterStatus(connection)`	`int`	获取状态字节
`isPrinterOnline(connection)`	`boolean`	是否在线
`hasPaper(connection)`	`boolean`	是否有纸
`isCoverOpen(connection)`	`boolean`	是否开盖
`queryRTStatus(connection, type, timeout)`	`byte[]`	DLE EOT 实时状态
`queryTransmitStatus(connection, type, timeout)`	`byte[]`	GS r 传送状态

14.10 接口

名称	说明
`PrinterSDK.NetworkConnectionCallback`	异步网络连接回调（继承 `CommunicationManager.NetworkConnectionCallback`）
`BluetoothPrinterManager.BluetoothDeviceListener`	蓝牙扫描/配对监听器

> 方法签名与参数详见 `PrinterSDK.java` 源码或 IDE 自动补全。

---

15. ESC/POS 指令覆盖说明（对照 Epson EPOS 标准）

本节对照 Epson ESC/POS 标准指令，说明 SDK 当前支持情况。图例：✅ **已实现并暴露** · ⚠️ **底层有实现但未接入/未暴露** · ❌ **未实现**

15.1 标准模式 — 已实现 ✅

EPOS 指令	十六进制	SDK 方法
ESC @ 初始化	`1B 40`	`initPrinter()` / `resetPrinter()`
ESC a 对齐	`1B 61`	`setAlignment()`
ESC ! 打印模式	`1B 21`	`setPrintMode()`
ESC E 加粗	`1B 45`	`setBold()`
ESC - 下划线	`1B 2D`	`setUnderline()`
ESC { 倒置	`1B 7B`	`setTextUpsideDown()`
ESC V 旋转90°	`1B 56`	`setTextRotate()`
ESC M 字体	`1B 4D`	`setAsciiTextFontType()`
ESC 3 行间距	`1B 33`	`setLineSpacing()` / `setTextLineHeight()`
ESC SP 字间距	`1B 20`	`setCharSpacing()` / `setAsciiTextCharRightSpacing()`
ESC R 国际字符集	`1B 52`	`setCharacterSet()`
ESC t 代码页	`1B 74`	`setCharacterCodepage()`
ESC $ 绝对水平位置	`1B 24`	`setPrintPosition()` / `setHorizontalAbsolutePrintPosition()`
ESC \ 相对水平位置	`1B 5C`	`setHorizontalRelativePrintPosition()`
GS L 左边距	`1D 4C`	`setPrintAreaLeftMargin()`
GS W 打印区宽度	`1D 57`	`setPrintAreaWidth()`
GS ! 字符尺寸	`1D 21`	`setTextScale()`
GS B 反白	`1D 42`	`setReverse()`
GS V 切纸	`1D 56`	`cutPaper()` / `feedAndCutPaper()`
ESC i / ESC m 切纸	`1B 69` / `1B 6D`	`cutPaper(CUT_FULL/PARTIAL)`
ESC d 走纸 n 行	`1B 64`	`feedPaper(lines)`
ESC J 走纸 n 点	`1B 4A`	`feedPaper()` / `printAndFeedDots()`
ESC e 反向走纸	`1B 65`	`reverseFeed(lines)`
HT 水平制表	`09`	`horizontalTab()`
ESC D 制表位	`1B 44`	`setTabPositions()`
GS r 传送状态	`1D 72`	`queryTransmitStatus()`
DLE EOT 实时状态	`10 04`	`queryRTStatus()`
LF 换行	`0A`	`feedLine()` / 文本末尾自动追加

EPOS 指令	十六进制	SDK 方法
ESC p 钱箱	`1B 70`	`openCashDrawer()`
ESC B 蜂鸣	`1B 42`	`beep()`
GS h/H/w/k 条码	`1D 68/48/77/6B`	`printBarcode()` + 条码格式设置
GS (k QR/PDF417	`1D 28 6B`	`printQRCode*()` / `printPDF417()` / `printDoubleQRCode()`
GS v 0 光栅图	`1D 76 30`	`printRasterImageFrom*()`（`RASTER_COMPRESSION_GS_V0`）
US (J JBIG 光栅	`1F 28 4A`	`printRasterImageFrom*()`（`RASTER_COMPRESSION_JBIG` / AUTO）
US (P 行 RLE	`1F 28 50`	`printRasterImageFrom*()`（`RASTER_COMPRESSION_LINE_RLE`，需固件支持）
CAN 取消	`18`	`clearBuffer()` / `cancelPrintCmd()`
GS (A 自检	`1D 28 41 02 00 00 02`	`sendSelfTest()`
原始指令	—	`sendCommand()`

15.2 汉字指令 — 部分实现

EPOS 指令	十六进制	状态	SDK 方法
FS & 汉字模式	`1C 26`	✅	`setChineseMode()`
FS . 取消汉字模式	`1C 2E`	✅	`setSingleByteMode()`
FS W 汉字倍宽倍高	`1C 57`	✅	`setChineseDoubleSize()`
ESC C 汉字间距	`1B 43`	✅	`setKanjiTextCharSpacing()`
ESC 39 双字节编码	`1B 39`	✅	`setDoubleByteEncoding()`
FS ! 汉字打印模式	`1C 21`	❌	—
FS - 汉字下划线	`1C 2D`	❌	—
FS S 汉字间距	`1C 53`	❌	—
FS 2 自定义汉字	`1C 32`	❌	—

15.3 页模式 — ✅ 已修复 ESC/POS 指令

此前页模式不可用的主要原因：

问题	原因	修复
文本/图片坐标无效	误用 `GS ( N`（字符颜色指令）定位	改为 `ESC $` + `GS $`
条码/二维码无效	使用了不存在的页模式专用格式	改为先定位，再发标准 `GS k` / QR 指令
矩形绘制无效	误用 `GS v 0`（光栅图指令）画矩形	改为 `GS ( Q` fn=49（边框）/ fn=48（填充）
进入页模式后无输出	未设置 `ESC W` 打印区域和 `ESC T` 方向	`selectPageMode()` 自动完成初始化

`PageModeManager` 统一调用 `PrinterCommands` 中的 ESC/POS 页模式指令：

功能	EPOS 指令	PrinterCommands	PageModeManager
进入页模式	ESC L `1B 4C`	✅	✅
退出页模式	ESC S `1B 53`	✅	✅
页区域	ESC W `1B 57`	✅	✅
页方向	ESC T `1B 54`	✅	✅
打印页	ESC FF `1B 0C`	✅	✅
清除页	CAN `18`	✅	✅
移动单位	GS P `1D 50`	✅	✅
页内文本	ESC $ + GS $ + 文本	✅	✅
页内矩形	GS (Q fn=49（边框）/ fn=48（填充）	✅	✅
页内条码/QR	ESC $ + GS $ + GS k / QR	✅	✅
页内光栅图	ESC $ + GS $ + GS v 0	✅	✅

> **说明：** 页模式与标签模式（JPL，`FS L` / `FS P`）是不同指令体系，请勿混用。

15.4 标签模式 — JPL 指令（非 ESC/POS 页模式）

标签模式使用 **JPL 指令集**（`FS L`、`FS P` 等），由 `LabelModeManager` / `JPLCommands` 实现，与 ESC/POS 页模式是不同体系。当前标签模式功能较完整。

15.5 常见但未实现的 EPOS 指令 ❌

以下指令在 Epson EPOS 标准中存在，**当前 SDK 尚未封装**（可用 `sendCommand()` 手动发送）：

分类	指令	说明	常用度
走纸	**ESC 2**	恢复默认 1/6 行距	低
字符	**ESC G**	双重击打（double-strike）	低
字符	**ESC % / ESC & / ESC ?**	用户自定义字符	低
字符	**ESC ***	位图模式（旧式，已被 GS v 0 取代）	低
布局	**CR** `0D`	回车	低
颜色	**ESC r**	双色打印	低（仅双色机）
状态	**GS I**	传送打印机 ID	低
传感器	**ESC c 3/4/5**	纸传感器/面板键	低

分类	指令	说明	常用度
图片	**GS * / GS /**	下载位图	低
图片	**FS p / FS q**	NV 位图（`printNvBitmap` / 页模式 `drawNvBitmapInPageMode`）	中
宏	**GS : / GS ^**	宏定义/执行	低
其他	**GS b**	平滑模式	低
其他	**ESC =**	选择外设	低

15.6 底层有但未暴露到 PrinterSDK 的指令 ⚠

位置	方法	说明
`PrinterCommands`	`queryPrintResult()`	查询打印结果
`PrinterCommands`	`printImage()`	ESC * 旧式位图
`PrinterCommands`	`printBarcodeRaw()`	原始条码字节
`PrinterCommands`	页模式全套	已通过 PageModeManager 对外暴露

15.7 覆盖度总结

模块	覆盖情况
标准小票打印（文本/格式/条码/QR/图片/切纸）	**高** — 核心 ESC/POS 已覆盖
汉字打印	**中高** — 缺 FS !、FS - 等辅助指令
页模式	**高** — 已接入 ESC/POS 页模式指令
标签模式	**高** — JPL 体系独立实现
状态查询	**中高** — 主动查询（DLE EOT / GS r）+ 便捷判断 API
高级/少见指令	**低** — 宏、NV 图、双色等未封装

附录：版本更新记录

版本	日期	主要变更
1.0.0	2025-08-28	初始版本，支持 USB/蓝牙/网络/串口
1.0.2	2025-08-29	修复网络连接缺少 `connect()` 的问题
1.2.0	2025-08-29	新增 `connectNetworkPrinterAsync` 异步连接
1.3.0	2025-08-29	USB/蓝牙权限自动申请

版本	日期	主要变更
1.5.0	2025-08-29	权限申请完全自动化，修复 bt socket closed 等问题
1.6.0	2026-07-07	页模式接入 ESC/POS；新增主动状态查询（DLE EOT / GS r）
1.6.1	2026-07-08	修复页模式坐标/条码/矩形指令；补充条码 TYPE 选型说明
1.6.2	2026-07-09	修复自检指令（改用 GS (A）；修复标签页打印（`1A 4F`，原 `FS P` 导致不出纸）；修复标签文本子命令 `1A 54` 的 fn；新增 `printTextLineLeftRight`、页模式排版辅助、`printNvBitmap` / `drawNvBitmapInPageMode`；纸宽常量

技术支持： 如有问题，请查看 Logcat 日志并参考本文 第 13 节。项目内另有 PrinterSDK_函数方法说明.md、`connectNetworkPrinterAsync使用指南.md` 可供参考。