PrinterSDK User Manual

> Version: 1.6.3
> Platform: Android 7.0+ (API 24+)
> Package: `com.hspos.printerSDK`
> Last updated: 2026-07-10
> Chinese edition: PrinterSDK_使用说明书.md
> API reference: PrinterSDK_API_Documentation.md

Table of Contents

1. Product Introduction
2. System Requirements and Integration
3. SDK Architecture
4. Quick Start
5. Permission Configuration
6. Printer Connection
7. Standard POS Printing
8. Page Mode Printing
9. Label Mode Printing
10. Status Query and Device Control
11. Constants Reference
12. Complete Examples
13. FAQ and Troubleshooting
14. API Quick Reference
15. ESC/POS Command Coverage
16. API Reference (handbook style) · 中文

---

1. Product Introduction

PrinterSDK is a thermal printer development library for Android. It supports ESC/POS receipt printing and JPL label printing. Through the unified `PrinterSDK` entry class, developers can discover devices, connect, print, and query status with one consistent API.

Key Features

Feature	Description
Multi-channel connection	USB, Bluetooth, network (TCP/IP), serial

Feature	Description
Standard receipt printing	Text, barcode, QR code, PDF417, image, cut, cash drawer, beep
Page mode	Coordinate-based drawing of text, barcodes, images, and rectangles
Label mode	Label-paper layouts and JPL commands
Multi-encoding support	UTF-8, GBK, BIG5, Shift-JIS, EUC-KR
Automated permissions	USB / Bluetooth permissions requested by the SDK automatically (v1.5.0+)
Async network connect	Avoids `NetworkOnMainThreadException` on the main thread

Core Public Types

After integration, application code typically only needs these types:

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

---

2. System Requirements and Integration

2.1 Environment Requirements

Item	Requirement
Minimum Android version	API 24 (Android 7.0)
Target SDK	API 34
Java version	17
Build artifact	AAR library (`PrinterSDK.aar`)

2.2 Integrating the AAR

Step 1: Copy `app/build/outputs/aar/PrinterSDK.aar` (or `app-release.aar`) into the host project's `libs/` directory.

Step 2: Add the dependency in the host `build.gradle`:

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

Step 3: Declare required permissions in the host `AndroidManifest.xml` (see Section 5).

Step 4: Build the Release package:

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

When the build finishes, the AAR is under `app/build/outputs/aar/`.

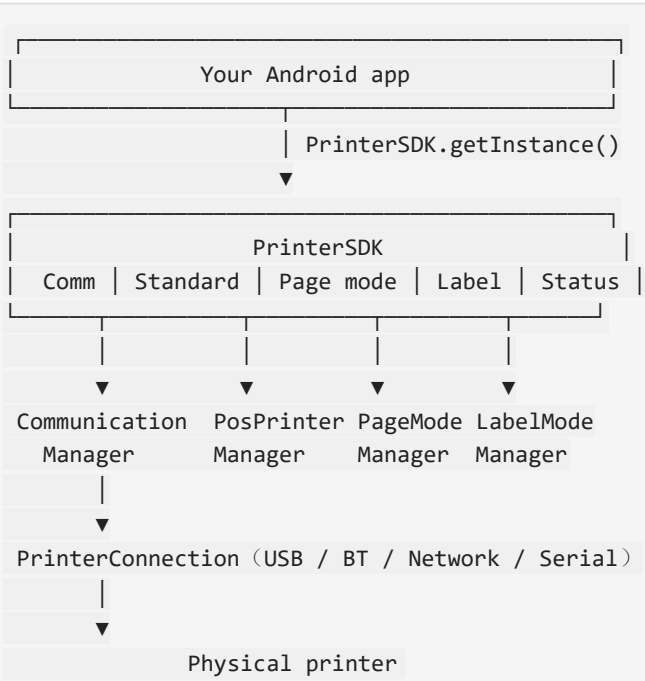
2.3 Source Module Structure

```
com.hspos.printerSDK/
├── PrinterSDK.java              ← Unified public entry (recommended)
├── PrinterConstants.java        ← Constant definitions
├── CommunicationManager.java    ← Communication management
├── PosPrinterManager.java       ← Standard POS print implementation
├── PageModeManager.java         ← Page mode implementation
├── LabelModeManager.java        ← Label mode implementation
├── PrinterCommands.java         ← ESC/POS command generation
├── JPLCommands.java             ← JPL label command generation
├── communication/
│   ├── PrinterConnection.java   ← Connection abstraction
│   ├── usb/                     ← USB connection
│   ├── bluetooth/               ← Bluetooth connection
│   ├── network/                 ← Network connection
│   └── serial/                  ← Serial connection
└── utils/
    └── ImageUtils.java          ← Image to print-data conversion
```

> **Recommendation:** Call `PrinterSDK` from the business layer only. Do not call internal Manager classes directly, so upgrades remain compatible.

---

3. SDK Architecture



Connection object `PrinterConnection` is the first parameter of every print operation. It represents the communication channel to a printer. Before each print job, ensure `connection != null && connection.isConnected()`.

---

4. Quick Start

4.1 Initialization

```
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);
        // Must forward to the SDK so Bluetooth permission callbacks are handled
        printerSDK.onRequestPermissionsResult(requestCode, permissions, grantResults);
    }

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

4.2 Minimal Print Flow

```
// 1. Connect the printer (network example; async recommended)
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. Print (prefer a background thread)
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. Permission Configuration

5.1 AndroidManifest Declarations

Depending on the connection types you use, add permissions to the host `AndroidManifest.xml`:

```
<!-- Bluetooth -->
<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" />

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

<!-- Image printing (required when reading from files) -->
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_MEDIA_IMAGES" />
```

5.2 Differences Across Android Versions

Android version	Bluetooth permissions	Other
6.0–11 (API 23–30)	`BLUETOOTH`, `BLUETOOTH_ADMIN` + location	—
12+ (API 31+)	`BLUETOOTH_CONNECT`, `BLUETOOTH_SCAN` + location	—
13+ (API 33+)	Same as above	Prefer `READ_MEDIA_IMAGES` for image reads

5.3 Automatic Permission Handling (v1.5.0+)

Connection	Permission handling
USB	SDK checks and requests USB permission automatically
Bluetooth	SDK checks and requests Bluetooth + location automatically
Network	No runtime permission required
Serial	No runtime permission required (device node permissions may need root)

You must forward the permission callback in the Activity:

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

5.4 USB Device Filter (Optional)

Configure known printer Vendor ID / Product ID values in `res/xml/device_filter.xml` so inserting USB can auto-launch the app:

```
<?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. Printer Connection

All connection methods return a `PrinterConnection` object, or `null` on failure.

6.1 USB Connection

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

- The SDK requests USB permission automatically
- You can also check manually: `hasUsbPermission(device)` / `requestUsbPermission(device)`

6.2 Bluetooth Connection

```
// Get paired devices
List<BluetoothDevice> paired = printerSDK.getBluetoothPairedDevices();

// Get available Bluetooth devices (paired + discovered)
List<BluetoothDevice> devices = printerSDK.getBluetoothDevices();

// Get printer devices (filtered list)
List<BluetoothDevice> printers = printerSDK.getBluetoothPrinters();

// Scan for new devices
printerSDK.startBluetoothDiscovery();
// Receive discovery results via listener
printerSDK.addBluetoothListener(new BluetoothPrinterManager.BluetoothDeviceListener() {
    @Override
    public void onDeviceFound(BluetoothDevice device) { /* Device found */ }

    @Override
    public void onDiscoveryFinished() { /* Discovery finished */ }

    @Override
    public void onBondStateChanged(BluetoothDevice device, int bondState) { /* Bond state
changed */ }
});
// Remove the listener when no longer needed
printerSDK.removeBluetoothListener(listener);
printerSDK.stopBluetoothDiscovery();

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

> Pair the Bluetooth device in system settings first, or discover and pair via the SDK.

6.3 Network Connection

Recommended: async connect (safe to call on the main thread)

```
printerSDK.connectNetworkPrinterAsync("192.168.1.100", 9100,
    new CommunicationManager.NetworkConnectionCallback() {
        @Override
        public void onConnectionSuccess(PrinterConnection connection) {
            // ⚠ Callback runs on a background thread; use runOnUiThread for UI updates
            runOnUiThread(() -> { /* Connected */ });
        }

        @Override
        public void onConnectionFailed(String errorMessage) {
            runOnUiThread(() -> { /* Show error */ });
        }
    });
```

Sync connect (must not be called on the main thread)

```
// Must be called on a background thread
PrinterConnection conn = printerSDK.connectNetworkPrinter("192.168.1.100", 9100);
```

Parameter	Description
IP address	Printer LAN IP, e.g. `192.168.1.100`
Port	Default `9100` (RAW print port)
Connect timeout	5 seconds (`NetworkPrinterConnection.TIMEOUT`)

6.4 Serial Connection

```
List<String> ports = printerSDK.getSerialDevices();
// Common paths: /dev/ttyUSB0, /dev/ttyS0, /dev/ttyACM0
// Common baud rates: 9600, 19200, 38400, 57600, 115200

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

> Serial is typically used on industrial tablets or rooted devices. Ordinary phones usually do not support it.

6.5 Disconnect and Communication Helpers

```
// Disconnect
printerSDK.disconnectPrinter(connection);

// USB permission (connectUsbPrinter requests automatically; can also be manual)
boolean ok = printerSDK.hasUsbPermission(usbDevice);
boolean granted = printerSDK.requestUsbPermission(usbDevice);

// Clear buffer / cancel print
printerSDK.clearPrinterBuffer(connection);
printerSDK.cancelPrint(connection);

// Advanced: obtain the underlying communication manager
CommunicationManager comm = printerSDK.getCommunicationManager();
```

Method	Description
`disconnectPrinter(connection)`	Disconnect the printer
`hasUsbPermission(device)`	Check USB permission
`requestUsbPermission(device)`	Request USB permission
`removeBluetoothListener(listener)`	Remove Bluetooth discovery listener

Method	Description
`clearPrinterBuffer(connection)`	Clear the printer buffer
`cancelPrint(connection)`	Cancel the current print job
`getCommunicationManager()`	Get underlying `CommunicationManager` (advanced)
`onRequestPermissionsResult(...)`	Forward runtime permission callbacks (see Section 5)
`release()`	Release all SDK resources

---

7. Standard POS Printing

For all methods below, the first parameter is `PrinterConnection connection`. The return value is `boolean` indicating success.

7.1 Initialize and Reset

Method	Description
`initPrinter(connection)`	Initialize printer (ESC @)
`resetPrinter(connection)`	Reset printer
`sendSelfTest(connection)`	Send self-test command
`sendCommand(connection, byte[])`	Send raw commands

7.2 Text Printing

Method	Encoding
`printText(connection, text)`	Default encoding
`printTextInUTF8(connection, text)`	UTF-8
`printTextInGBK(connection, text)`	GBK (Simplified Chinese)
`printTextInBIG5(connection, text)`	BIG5 (Traditional Chinese)
`printTextInShiftJIS(connection, text)`	Shift-JIS (Japanese)
`printTextInEUCKR(connection, text)`	EUC-KR (Korean)

```
printerSDK.initPrinter(connection);
printerSDK.setAlignment(connection, PrinterConstants.ALIGNMENT_CENTER);
printerSDK.setChineseDoubleSize(connection, true);  // Chinese double width/height FS W
printerSDK.setBold(connection, true);
printerSDK.printTextInGBK(connection, "欢迎光临\n");
printerSDK.setBold(connection, false);
printerSDK.setChineseDoubleSize(connection, false);  // Restore normal Chinese size
```

Left–Right Two-Column Line (common for 80 mm receipts)

Method	Description
`printTextLineLeftRight(connection, left, right)`	80 mm default width 576 dots, character width 12 dots
`printTextLineLeftRight(connection, left, right, areaWidthDots, charWidthDots)`	Custom area width and character width

Paper-width constants (`PrinterConstants`):

Constant	Value	Description
`PAPER_WIDTH_58MM`	384	58 mm paper printable width (dots)
`PAPER_WIDTH_80MM`	576	80 mm paper printable width (dots)

```
// 80 mm commerce receipt: product name on the left, price on the right
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");

// When the total line is enlarged, use character width 24 (after GS ! double width/height)
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);
```

> How it works: `ESC $` positions to the line start for the left column, then `ESC $` positions to the right for the right column and advances a line. After double width/height, increase `charWidthDots` accordingly (typically 12→24).

7.2.1 Font Double Width/Height (Three Commands — Do Not Confuse)

Printers expose **three different double-width/height command sets**, each for a different use case:

Method	Command	Bytes	Use for
`setChineseDoubleSize()`	**FS W n**	`1C 57 n`	**Chinese** double width/height (on/off)
`setTextScale()`	GS ! n	`1D 21 n`	Character size for Chinese and Latin (levels 0–7)
`setPrintMode()`	ESC ! n	`1B 21 n`	Print-mode double width/height

> The printer applies the **last command received**; watch ordering when mixing them.

① Chinese Double Width/Height — `setChineseDoubleSize()` (recommended for Chinese)

Corresponds to the handbook **"Select/cancel Chinese double height and double width"**, command **FS W n**:

```
// Enable Chinese double width/height → 1C 57 01
printerSDK.setChineseDoubleSize(connection, true);
// or
printerSDK.setChineseDoubleSize(connection, PrinterConstants.CHINESE_DOUBLE_SIZE_ON);

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

// Cancel and restore normal Chinese size → 1C 57 00
printerSDK.setChineseDoubleSize(connection, false);
// or pass int (only the least significant bit is used: 0/1)
printerSDK.setChineseDoubleSize(connection, PrinterConstants.CHINESE_DOUBLE_SIZE_NORMAL);
```

Parameter `n` uses only the **least significant bit**: `0` = cancel, `1` = enable Chinese double width and height together.

② Character Size — `setTextScale()` (GS ! n)

Set width and height multipliers independently (0 = 1×, 1 = 2× … 7 = 8×). Applies to Chinese and Latin (except HRI characters):

```
printerSDK.setTextScale(connection, 0, 0);  // Normal
printerSDK.setTextScale(connection, 1, 1);  // 2× width + 2× height
printerSDK.setTextScale(connection, 1, 0);  // 2× width only
printerSDK.setTextScale(connection, 0, 1);  // 2× height only
```

③ Print Mode — `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 Format Settings

Method	Description
`setAlignment(connection, alignment)`	Alignment: left / center / right
`setChineseDoubleSize(connection, enable)`	**Chinese double width/height** (FS W n, `boolean`)

Method	Description
`setChineseDoubleSize(connection, n)`	**Chinese double width/height** (FS W n, `int`, 0/1)
`setTextScale(connection, widthScale, heightScale)`	Character size GS ! n (0 = normal, 1 = 2× …)
`setBold(connection, bold)`	Bold
`setUnderline(connection, underline)`	Underline
`setReverse(connection, reverse)`	Reverse (white on black)
`setLineSpacing(connection, spacing)`	Line spacing
`setCharSpacing(connection, spacing)`	Character spacing
`setPrintMode(connection, mode)`	Print mode ESC ! n (includes double width/height)
`setTextUpsideDown(connection, upsideDown)`	Upside-down text
`setTextRotate(connection, rotate)`	Rotate 90°
`setTextLineHeight(connection, lineHeight)`	Line height
`setPrintPosition(connection, position)`	Print start position
`setPrintAreaLeftMargin(connection, margin)`	Left margin
`setPrintAreaWidth(connection, width)`	Print area width
`setHorizontalAbsolutePrintPosition(connection, pos)`	Absolute horizontal position
`setHorizontalRelativePrintPosition(connection, offset)`	Relative horizontal offset
`setVerticalAbsolutePrintPosition(connection, pos)`	Absolute vertical position
`setVerticalRelativePrintPosition(connection, offset)`	Relative vertical offset
`setSingleByteMode(connection)`	Single-byte mode
`setCharacterSet(connection, charset)`	Character set
`setCharacterCodepage(connection, codepage)`	Code page
`setChineseMode(connection)`	Chinese mode
`setDoubleByteEncoding(connection, n)`	Double-byte encoding
`setAsciiTextFontType(connection, fontType)`	ASCII font A/B
`setAsciiTextCharRightSpacing(connection, spacing)`	ASCII right spacing
`setKanjiTextCharSpacing(connection, left, right)`	Chinese character spacing

7.4 Barcodes and QR Codes

Choosing Barcode TYPE (Important)

The second parameter `type` of `printBarcode()` must be a `PrinterConstants.BARCODE_TYPE_*` **constant**. Do not hard-code `0`, `1`, `2`, and do not mix with label-mode barcode type numbers (0–29).

Not sure which to pick for testing? Prefer **Code 128** — best compatibility, supports letters and digits, suitable for order numbers, tracking numbers, and most cases:

```
// Recommended starting combination when testing barcodes
printerSDK.printBarcode(connection,
    PrinterConstants.BARCODE_TYPE_CODE128,  // ← use this TYPE
    "1234567890",                           // any alphanumeric is fine
    PrinterConstants.BARCODE_HRI_BELOW,
    80, 2);
```

Choose by business scenario:

Scenario	Constant	Sample data	Data rules
General test / order / tracking	`BARCODE_TYPE_CODE128`	`"1234567890"`	Letters, digits, symbols OK
North America retail (UPC)	`BARCODE_TYPE_UPC_A`	`"012345678905"`	11–12 digits
International retail (EAN)	`BARCODE_TYPE_EAN13`	`"6901234567892"`	12–13 digits
Small-package goods	`BARCODE_TYPE_EAN8`	`"96385074"`	7–8 digits
Industrial / warehouse labels	`BARCODE_TYPE_CODE39`	`"ABC-1234"`	Digits, uppercase, and `-. $/+%`
Case code (ITF)	`BARCODE_TYPE_ITF`	`"123456789012"`	Even number of digits only
Library / blood bank, etc.	`BARCODE_TYPE_CODABAR`	`"A40156B"`	Starts and ends with A–D
North America compressed retail	`BARCODE_TYPE_UPC_E`	`"01234565"`	Must follow UPC-E rules
High-density encoding	`BARCODE_TYPE_CODE93`	`"TEST93"`	Alphanumeric

> **Note:** Standard receipt (ESC/POS) and label mode (JPL) barcode `type` values are **not the same numbering**. Standard print uses `BARCODE_TYPE_*` (`0x41` – `0x49`); label print uses `LABEL_BARCODE_TYPE_*` (0–29). See Section 9.

```
// 1D barcode
printerSDK.printBarcode(connection,
    PrinterConstants.BARCODE_TYPE_CODE128,  // Barcode type
    "1234567890",                           // Data
    PrinterConstants.BARCODE_HRI_BELOW,     // HRI position
    80,                                     // Height (dots)
    2);                                     // Width

// QR code (simple)
printerSDK.printQRCode(connection, "https://example.com", 3);

// QR code (standard; alignment and ECC level)
printerSDK.printQRCodeStandard(connection,
    "https://example.com", 3,
    PrinterConstants.ALIGNMENT_CENTER,
    PrinterConstants.QR_ECC_LEVEL_M);

// Dual QR code
printerSDK.printDoubleQRCode(connection, n, qr1Pos, qr1Data, qr1Ecc, qr1Ver,
    qr2Pos, qr2Data, qr2Ecc, qr2Ver);

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

Optional barcode format presets (call before `printBarcode`):

Method	Description
`setBarcodeHRIPosition(connection, pos)`	HRI text position (below/above barcode, etc.)
`setBarcodeHeight(connection, height)`	Barcode height (dots)
`setBarcodeWidth(connection, width)`	Narrow bar width
`setBarcodeReadableTextFontType(connection, fontType)`	HRI text font
`setBarcodeReadableTextPosition(connection, position)`	HRI text relative position

```
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);
```

Barcode type constants:

Constant	Type	ESC/POS m	Typical use
`BARCODE_TYPE_UPC_A`	UPC-A	`0x41` (65)	North America 12-digit product code
`BARCODE_TYPE_UPC_E`	UPC-E	`0x42` (66)	North America 8-digit compressed product code
`BARCODE_TYPE_EAN13`	EAN-13	`0x43` (67)	International 13-digit product code

Constant	Type	ESC/POS m	Typical use
`BARCODE_TYPE_EAN8`	EAN-8	`0x44` (68)	International 8-digit product code
`BARCODE_TYPE_CODE39`	Code 39	`0x45` (69)	Industrial ID, logistics
`BARCODE_TYPE_ITF`	ITF	`0x46` (70)	Case codes (even digit count)
`BARCODE_TYPE_CODABAR`	Codabar	`0x47` (71)	Libraries, blood banks
`BARCODE_TYPE_CODE93`	Code 93	`0x48` (72)	High-density encoding
`BARCODE_TYPE_CODE128`	Code 128	`0x49` (73)	**General first choice**; recommended for testing

HRI text position:

Constant	Description
`BARCODE_HRI_NONE`	Do not print HRI text above/below
`BARCODE_HRI_BELOW`	Print readable text below the barcode (common)

7.5 Image Printing

`printRasterImageFrom*` converts the image to a black-and-white bitmap and sends it. From **1.6.3**, the default is **AUTO compression** (chooses the smaller transfer between GS v 0 and JBIG for faster printing). Existing code needs no changes; behavior remains backward compatible.

```
 // From file (default AUTO compression)
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png");
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png", 384, 200);

// From byte array
printerSDK.printRasterImageFromData(connection, imageBytes);
printerSDK.printRasterImageFromData(connection, imageBytes, 384, 200);

// From pixel array
int[] pixels = ...; // ARGB format
printerSDK.printRasterImageFromPixels(connection, pixels, width, height);
printerSDK.printRasterImageFromPixels(connection, pixels, width, height, 384, 200);

// Explicit compression mode (5th parameter compressionMode; supported by all three APIs)
printerSDK.printRasterImageFromFile(connection, "/sdcard/logo.png", 384, 200,
    PrinterConstants.RASTER_COMPRESSION_AUTO);   // Recommended, default
printerSDK.printRasterImageFromData(connection, imageBytes, 384, 200,
    PrinterConstants.RASTER_COMPRESSION_JBIG);   // Force JBIG
printerSDK.printRasterImageFromPixels(connection, pixels, width, height, 384, 200,
    PrinterConstants.RASTER_COMPRESSION_GS_V0);  // Force standard GS v 0
```

Compression mode constants (`PrinterConstants.RASTER_COMPRESSION_*`):

Constant	Value	Command	Description
`RASTER_COMPRESSION_GS_V0`	0	`GS v 0` (`1D 76 30`)	Standard raster; best compatibility
`RASTER_COMPRESSION_LINE_RLE`	1	`US ( P` (`1F 28 50`)	Per-line RLE; **requires firmware support**; most models incompatible
`RASTER_COMPRESSION_JBIG`	2	`US ( J` (`1F 28 4A`)	JBIG compression; smaller and faster
`RASTER_COMPRESSION_AUTO`	3	Auto	**Default**; picks the smaller transfer between GS v 0 and JBIG

> - Images are converted to black-and-white bitmaps automatically. Keep width within the printable width (about 384 dots for 58 mm paper, 576 dots for 80 mm).
> - Pass `0` for `targetHeight` to scale proportionally by width.
> - For day-to-day use, keep default `AUTO`. If JBIG fails on a device, switch to `RASTER_COMPRESSION_GS_V0`.
> - Use `LINE_RLE` only after confirming the printer supports `US ( P`. Do **not** use it via AUTO.

7.6 Paper Feed and Cut

Method	Description
`feedLine(connection)`	Advance one line
`feedPaper(connection, lines)`	Feed N lines
`feedPaper(connection)`	Feed (by dots)
`reverseFeed(connection, lines)`	Reverse feed N lines (ESC e)
`horizontalTab(connection)`	Horizontal tab (HT)
`setTabPositions(connection, positions)`	Set tab stops (ESC D)
`feedAndCutPaper(connection)`	Feed then cut
`cutPaper(connection, cutType)`	Cut (full / partial)
`printAndFeedDots(connection, n)`	Print and feed N dots
`printHorizontalLine(connection, start, end)`	Print a horizontal line
`fixedHorizontalLine(connection)`	Fixed-width horizontal line

```
printerSDK.cutPaper(connection, PrinterConstants.CUT_FULL);     // Full cut
printerSDK.cutPaper(connection, PrinterConstants.CUT_PARTIAL);  // Partial cut

// Tabs: set tab stops (character widths) first, then send HT
printerSDK.setTabPositions(connection, new int[] { 8, 16, 24, 32 });
printerSDK.printText(connection, "列1");
printerSDK.horizontalTab(connection);
printerSDK.printText(connection, "列2");

// Reverse feed (printer must support it)
printerSDK.reverseFeed(connection, 2);
```

7.7 Peripheral Control

```
// Open cash drawer
printerSDK.openCashDrawer(connection, PrinterConstants.CASH_DRAWER_PIN2);
printerSDK.openCashDrawer(connection, PrinterConstants.CASH_DRAWER_PIN5);

// Beep
printerSDK.beep(connection);                    // Default once
printerSDK.beep(connection, 3, 2);              // 3 times, ~100 ms each
```

7.8 Buffer Control

Method	Description
`clearBuffer(connection)`	Clear buffer
`clearPrinterBuffer(connection)`	Clear printer buffer
`cancelPrint(connection)`	Cancel print
`cancelPrintCmd(connection)`	Cancel-print command

---

8. Page Mode Printing

Page mode draws content by coordinates inside a defined area. It suits complex receipt layouts.

8.1 Basic Flow

```
printerSDK.initPrinter(connection);

// 1. Enter page mode (automatically completes GS P + ESC W + ESC T init)
printerSDK.selectPageMode(connection);
// Or customize the print area (~576 dots for 80 mm paper width)
printerSDK.selectPageModeEx(connection, 0, 0, 576, 800);

// 2. Optional: adjust area or direction
printerSDK.setPrintAreaInPageMode(connection, 0, 0, 576, 800);
printerSDK.setPrintAreaDirection(connection, PrinterConstants.PAGE_DIRECTION_LEFT_TO_RIGHT);

// 3. Draw content (coordinates in dots; origin at top-left of the print area)
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);   // Border
printerSDK.drawFilledRectangleInPageMode(connection, 60, 60, 200, 80);     // Filled

// 4. Print and exit
printerSDK.pagePrintPage(connection);   // ESC FF — print page buffer
printerSDK.exitPageMode(connection);    // ESC S — return to standard mode
```

Page-Mode Layout Helpers (common for 80 mm receipts)

Method	Description
`estimateTextWidthDots(text, charWidthDots)`	Estimate text width (Font A normal ≈ 12 dots/char)
`drawTextCenteredInPageMode(connection, pageWidth, y, text)`	Horizontal center
`drawTextRightInPageMode(connection, pageWidth, y, text)`	Right align
`drawTextLineLeftRightInPageMode(connection, pageWidth, y, left, right)`	One line left–right columns

```
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 Bitmap / Logo (printer memory, FS p)

Factory- or tool-stored logos live in NV memory and can print without transferring an image:

Method	Description
`printNvBitmap(connection, nvNumber, mode)`	Print NV bitmap in standard mode
`drawNvBitmapInPageMode(connection, x, y, nvNumber, mode)`	Page mode at coordinates
`drawNvBitmapCenteredInPageMode(connection, pageWidth, y, nvNumber, mode, imageWidthDots)`	Page mode horizontal center

`mode` uses `PrinterConstants.NV_BITMAP_MODE_*` (0 normal / 1 double width / 2 double height / 3 double width and height).

```
// Centered logo at top of page mode (adjust nvNumber and imageWidthDots per printer handbook)
printerSDK.drawNvBitmapCenteredInPageMode(connection, 576, 40, 1,
    PrinterConstants.NV_BITMAP_MODE_NORMAL, 256);
y += 100;  // Reserve space for logo height
```

> **Note:** Page mode and label mode (JPL) are different command systems — do not mix them. `selectPageMode()` / `selectPageModeEx()` set movement units, print area, and default direction automatically. If you call only `selectPageMode()` without setting an area, the printer cannot position content correctly.

8.2 Page Mode API List

Category	Methods
Mode control	`selectPageMode`, `exitPageMode`, `selectPageModeEx` (3 overloads)
Area settings	`setMovementUnits`, `pageSetPageArea`, `pageSetPageDrawDirection`, `setPrintAreaInPageMode`, `setPrintAreaDirection`
Print control	`pagePrintPage`, `pageClearPage`
Text drawing	`drawTextInPageMode`, `drawTextUTF8InPageMode`, `drawTextGBKInPageMode`, `drawTextBIG5InPageMode`, `drawTextShiftJISInPageMode`, `drawTextEUCKRInPageMode`
Page-mode layout	`drawTextCenteredInPageMode`, `drawTextRightInPageMode`, `drawTextLineLeftRightInPageMode`, `estimateTextWidthDots`
NV bitmap	`printNvBitmap`, `drawNvBitmapInPageMode`, `drawNvBitmapCenteredInPageMode`
Graphics	`drawRectangleInPageMode`, `drawRectangleBorderInPageMode`, `drawFilledRectangleInPageMode`
Barcode / codes	`drawBarcodeInPageMode`, `drawQRCodeInPageMode`
Images	`drawImageFromFileInPageMode`, `drawImageFromDataInPageMode`, `drawImageFromPixelsInPageMode`

---

9. Label Mode Printing

Label mode targets label-paper printers and uses the JPL command set for precise layout.

9.1 Basic Flow

```
// 1. Enable label mode
printerSDK.enableLabelMode(connection);

// 2. Begin label page (x, y, width, height, rotation)
printerSDK.labelPageBegin(connection, 0, 0, 576, 400, 0);

// 3. Draw content
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. Print
printerSDK.backPaperToPrintPosition(connection);
printerSDK.labelPagePrint(connection);       // Print 1 copy
printerSDK.labelPagePrint(connection, 3);    // Print 3 copies

// 5. Exit label mode
printerSDK.disableLabelMode(connection);
```

9.2 Label Mode API List

Category	Methods
Mode control	`enableLabelMode`, `disableLabelMode`, `labelPageBegin`, `backPaperToPrintPosition`, `labelPagePrint` (including copy-count overload)
Text	`printLabelText`, `printLabelTextUTF8`, `printLabelTextGBK` (all with full-parameter overloads)
Barcode	`printLabelBarcode` (2 overloads), `setLabelBarcodeTextPosition`
QR / PDF417	`printLabelQRCode` (2 overloads), `printLabelPDF417Code` (2 overloads)
Images	`drawLabelImageFromFile`, `drawLabelImageFromData`, `drawLabelImageFromPixels`, `drawLabelImageAdvanced`
Graphics	`drawLabelLine` (2 overloads), `drawLabelRect` (2 overloads), `drawLabelBox`

---

10. Status Query and Device Control

The SDK provides **two** status approaches for different scenarios:

Approach	API	Direction	Parsed by SDK?	Typical use
Convenience checks	`getPrinterStatus` / `isPrinterOnline`, etc.	App polls	Partially (simplified constants)	Quick online / paper / cover checks
Active query	`queryRTStatus` / `queryTransmitStatus`	App polls	No (raw bytes)	Parse per ESC/POS yourself

> **Note:** Except for the convenience helpers, the SDK **does not interpret status bytes for business meaning**. Use the bit tables below in your app layer. Behavior may vary by model; verify on real hardware.

10.1 SDK Convenience Status APIs

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

Internally these call `queryRTStatus(connection, STATUS_TYPE_PRINTER, 5000)`, take 1 byte, and compare against the **simplified constants** below:

Constant	Value	Meaning for `getPrinterStatus`	Used by
`STATUS_OFFLINE`	`0x00`	Query failed or no response	Default return
`STATUS_ONLINE`	`0x01`	Compared for online	`isPrinterOnline()`
`STATUS_PAPER_OUT`	`0x02`	Paper-out bit (AND)	Negated by `hasPaper()`
`STATUS_COVER_OPEN`	`0x04`	Cover-open bit (AND)	`isCoverOpen()`

> These are SDK **wrapper simplification values**, and are **not identical** to ESC/POS status bit definitions. For precise checks, parse raw bytes from 10.2.

10.2 Active Query (DLE EOT / GS r)

API

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

// GS r: 1D 72 n (more compatible on some models; return format similar to DLE EOT)
byte[] tx = printerSDK.queryTransmitStatus(connection, type, timeoutMs);
```

Query type `type` (`PrinterConstants.STATUS_TYPE_*`)

Constant	n	Description	Typical return
`STATUS_TYPE_PRINTER`	1	Printer status	1 byte
`STATUS_TYPE_OFFLINE`	2	Offline / mechanical status	1 byte
`STATUS_TYPE_ERROR`	3	Error status	1 byte
`STATUS_TYPE_PAPER_SENSOR`	4	Paper sensor status	1 byte

`timeoutMs` is the wait time in milliseconds; returns `null` if there is no response.

n=1 Printer status byte (DLE EOT 1 / GS r 1)

Bit	0	1
0	(Fixed 0)	—
1	Paper OK	Near-end (model-dependent)
2	(Fixed 0)	—
3	Online	**Offline**
4	(Fixed 0)	—
5	Normal	**Recover request** (recoverable error)
6	(Fixed 0)	—
7	(Fixed 0)	—

n=2 Offline status byte

Bit	0	1
0	(Fixed 0)	—
1	Cover closed	**Cover open**
2	Not feeding	**Feed key feeding**
3	Normal	**Stopped for paper end**
4	Normal	**Error** (recoverable)
5	Normal	**Unrecoverable error**
6	(Fixed 0)	—
7	(Fixed 0)	—

n=3 Error status byte

Bit	0	1
0	(Fixed 0)	—
1	Normal	**Cutter error**
2	Normal	**Unrecoverable error**
3	Normal	**Recoverable error**
4	Normal	**Print-head temperature fault**
5	(Fixed 0)	—
6	(Fixed 0)	—
7	(Fixed 0)	—

n=4 Paper sensor status byte

Bit	Meaning (0 / 1)
0, 1	Near-end sensor: 00 = paper OK, 11 = near end (model-dependent)
2, 3	Paper-end sensor: 00 = paper present, 11 = paper end (model-dependent)
4–7	(Fixed 0)

10.3 Status API Summary

Method	Description
`getPrinterStatus(connection)`	Convenience status `int`
`isPrinterOnline(connection)`	Whether online
`hasPaper(connection)`	Whether paper is present
`isCoverOpen(connection)`	Whether cover is open
`queryRTStatus(connection, type, timeout)`	Active DLE EOT query
`queryTransmitStatus(connection, type, timeout)`	Active GS r query

---

11. Constants Reference

> **Note:** Most `int` constants in `PrinterConstants` are for **integrators** in app code (e.g. `PrinterConstants.ALIGNMENT_CENTER`). The SDK mostly passes parameters through, so the IDE may show “no usages” for some constants — that is expected. Legacy barcode constants such as `BARCODE_CODE128` are `@Deprecated`; use the `BARCODE_TYPE_*` series instead.

11.1 Alignment and Cut

```
PrinterConstants.ALIGNMENT_LEFT      // 0 - Left
PrinterConstants.ALIGNMENT_CENTER    // 1 - Center
PrinterConstants.ALIGNMENT_RIGHT     // 2 - Right

PrinterConstants.CUT_FULL            // 0 - Full cut
PrinterConstants.CUT_PARTIAL         // 1 - Partial cut
```

11.2 QR Code

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

PrinterConstants.QR_ECC_LEVEL_L      // 7%  ECC
PrinterConstants.QR_ECC_LEVEL_M      // 15% ECC
PrinterConstants.QR_ECC_LEVEL_Q      // Higher ECC
PrinterConstants.QR_ECC_LEVEL_H      // 30% ECC
```

11.3 Page Mode Direction

```
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 Print Mode Combinations

```
PrinterConstants.PRINT_MODE_NORMAL          // Normal
PrinterConstants.PRINT_MODE_DOUBLE_WIDTH    // Double width
PrinterConstants.PRINT_MODE_DOUBLE_HEIGHT   // Double height
PrinterConstants.PRINT_MODE_DOUBLE          // Double width + height
PrinterConstants.PRINT_MODE_BOLD            // Bold
PrinterConstants.PRINT_MODE_UNDERLINE       // Underline
PrinterConstants.PRINT_MODE_REVERSE         // Reverse
```

11.4.1 Raster Image Compression Modes

Used as the 5th parameter `compressionMode` of `printRasterImageFrom*`:

```
PrinterConstants.RASTER_COMPRESSION_GS_V0    // 0 — GS v 0 standard raster
PrinterConstants.RASTER_COMPRESSION_LINE_RLE   // 1 — US ( P per-line RLE (needs firmware
support)
PrinterConstants.RASTER_COMPRESSION_JBIG       // 2 — US ( J JBIG compression
PrinterConstants.RASTER_COMPRESSION_AUTO       // 3 — Auto (default, GS v 0 ⇔ JBIG)
```

11.5 Cash Drawer and Line Spacing

```
PrinterConstants.CASH_DRAWER_PIN2          // Cash drawer pin 2
PrinterConstants.CASH_DRAWER_PIN5          // Cash drawer pin 5

PrinterConstants.LINE_SPACING_DEFAULT      // Default line spacing
PrinterConstants.LINE_SPACING_1_8          // 1/8 inch
PrinterConstants.LINE_SPACING_1_6          // 1/6 inch
```

11.6 Status-Related Constants

See full bit definitions in Section 10.

```
// SDK convenience checks (simplified values, not raw ESC/POS bits)
PrinterConstants.STATUS_OFFLINE            // 0x00
PrinterConstants.STATUS_ONLINE             // 0x01
PrinterConstants.STATUS_PAPER_OUT          // 0x02
PrinterConstants.STATUS_COVER_OPEN         // 0x04

// Active query 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 Barcode Types (Standard Receipt ESC/POS)

> When testing barcodes and unsure which TYPE to use, pick `BARCODE_TYPE_CODE128`. Full selection guidance is in Section 7.4.

```
// General first choice (recommended for testing)
PrinterConstants.BARCODE_TYPE_CODE128   // 0x49 — alphanumeric OK

// Retail product codes
PrinterConstants.BARCODE_TYPE_EAN13     // 0x43 — 12~13 digits
PrinterConstants.BARCODE_TYPE_EAN8      // 0x44 — 7~8 digits
PrinterConstants.BARCODE_TYPE_UPC_A     // 0x41 — 11~12 digits
PrinterConstants.BARCODE_TYPE_UPC_E     // 0x42 — UPC-E compressed

// Other
PrinterConstants.BARCODE_TYPE_CODE39      // 0x45 — industrial ID
PrinterConstants.BARCODE_TYPE_ITF         // 0x46 — even digit count
PrinterConstants.BARCODE_TYPE_CODABAR     // 0x47 — starts/ends with A~D
PrinterConstants.BARCODE_TYPE_CODE93      // 0x48 — high density

// HRI readable text position
PrinterConstants.BARCODE_HRI_NONE         // 0 — hidden
PrinterConstants.BARCODE_HRI_BELOW        // 2 — below barcode (common)
```

11.8 Label Mode Barcode Types (JPL — Different from ESC/POS)

The `type` for label-mode `printLabelBarcode()` uses numbers **0–29** and is unrelated to `BARCODE_TYPE_*` above:

```
PrinterConstants.LABEL_BARCODE_TYPE_CODE128  // 8 — Label-mode Code 128 (common)
```

---

12. Complete Examples

12.1 Receipt Print (Bluetooth)

```
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);

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

            // Body
            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");

            // Separator
            printerSDK.fixedHorizontalLine(conn);

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

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

            // Cut
            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 Label Print (Network)

```
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. FAQ and Troubleshooting

13.1 Connection Issues

Symptom	Likely cause	Solution
Bluetooth connect fails / bt socket closed	Missing Bluetooth or location permission	Confirm Manifest declarations + forward `onRequestPermissionsResult`
Bluetooth scan returns nothing	Location permission not granted	Request `ACCESS_FINE_LOCATION`
USB connect fails	USB device not authorized	`connectUsbPrinter` requests automatically; check the USB cable

Symptom	Likely cause	Solution
Network connect fails	IP unreachable or port closed	Ping the printer IP; confirm port 9100 is open
NetworkOnMainThreadException	Sync connect on main thread	Use `connectNetworkPrinterAsync`
Serial connect fails	No serial port or insufficient permission	Confirm the device path exists; root may be required

13.2 Printing Issues

Symptom	Likely cause	Solution
Garbled Chinese	Encoding mismatch	Use matching methods such as `printTextInGBK`
Image does not print	Invalid path or no read permission	Check storage permission and file path
Cut does nothing	Printer has no cutter	Use `feedLine` instead
Barcode will not scan	Data format mismatch with type	Ensure `type` uses `BARCODE_TYPE_*`; for tests prefer `BARCODE_TYPE_CODE128` + `"1234567890"`; EAN/UPC must meet digit rules

13.3 Development Tips

1. **Print on a background thread**: Avoid blocking the UI; use async network connect
2. **Check return values**: All print methods return `boolean`; log failures
3. **Release resources promptly**: Call `printerSDK.release()` when the Activity is destroyed
4. **Use constants**: Do not hard-code alignment, barcode type, and similar values
5. **Init after connect**: Call `initPrinter(connection)` before printing so the printer is ready
6. **Reuse one connection**: The same `PrinterConnection` can print multiple jobs without reconnecting each time

13.4 Log Debugging

The SDK uses Android Log. Tags include:

Tag	Module
`PrinterSDK`	Main entry
`CommunicationManager`	Communication management
`NetworkPrinterConnection`	Network connection
`BluetoothPrinterManager`	Bluetooth management
`UsbPrinterManager`	USB management

Filter Logcat by these tags to locate issues quickly.

---

14. API Quick Reference

The following lists all **public methods** in `PrinterSDK.java` (including overloads).

14.1 Singleton and Lifecycle

Method	Return	Description
`getInstance(Context)`	`PrinterSDK`	Get singleton
`release()`	`void`	Release SDK resources
`onRequestPermissionsResult(int, String[], int[])`	`void`	Forward permission callback
`getCommunicationManager()`	`CommunicationManager`	Get underlying communication manager

14.2 Connection and Communication

Method	Return	Description
`getUsbDevices()`	`List<UsbDevice>`	USB device list
`connectUsbPrinter(UsbDevice)`	`PrinterConnection`	Connect USB
`hasUsbPermission(UsbDevice)`	`boolean`	Check USB permission
`requestUsbPermission(UsbDevice)`	`boolean`	Request USB permission
`getBluetoothPairedDevices()`	`List<BluetoothDevice>`	Paired Bluetooth devices
`getBluetoothDevices()`	`List<BluetoothDevice>`	Available Bluetooth devices
`getBluetoothPrinters()`	`List<BluetoothDevice>`	Printer-class Bluetooth devices
`connectBluetoothPrinter(BluetoothDevice)`	`PrinterConnection`	Connect Bluetooth
`startBluetoothDiscovery()`	`boolean`	Start Bluetooth discovery
`stopBluetoothDiscovery()`	`void`	Stop Bluetooth discovery
`addBluetoothListener(Listener)`	`void`	Add Bluetooth listener
`removeBluetoothListener(Listener)`	`void`	Remove Bluetooth listener
`connectNetworkPrinter(String, int)`	`PrinterConnection`	Sync network connect
`connectNetworkPrinterAsync(String, int, Callback)`	`void`	Async network connect

Method	Return	Description
`getSerialDevices()`	`List<String>`	Serial device path list
`connectSerialPrinter(String, int)`	`PrinterConnection`	Connect serial
`disconnectPrinter(PrinterConnection)`	`void`	Disconnect
`clearPrinterBuffer(PrinterConnection)`	`boolean`	Clear printer buffer
`cancelPrint(PrinterConnection)`	`boolean`	Cancel print
`queryRTStatus(PrinterConnection, int, int)`	`byte[]`	Real-time status (DLE EOT)
`queryTransmitStatus(PrinterConnection, int, int)`	`byte[]`	Transmit status (GS r)

14.3 Standard Print — Init and Text

Method	Description
`initPrinter(connection)`	Initialize printer
`resetPrinter(connection)`	Reset printer
`sendSelfTest(connection)`	Self-test
`sendCommand(connection, byte[])`	Send raw commands
`printText(connection, text)`	Text with default encoding
`printTextInUTF8(connection, text)`	UTF-8 text
`printTextInGBK(connection, text)`	GBK text
`printTextInBIG5(connection, text)`	BIG5 text
`printTextInShiftJIS(connection, text)`	Shift-JIS text
`printTextInEUCKR(connection, text)`	EUC-KR text

14.4 Standard Print — Format Settings

Method	Description
`setAlignment(connection, alignment)`	Alignment
`setChineseDoubleSize(connection, enable)`	Chinese double width/height (`boolean`)
`setChineseDoubleSize(connection, n)`	Chinese double width/height (`int` , 0/1)
`setTextScale(connection, w, h)`	Character size GS ! n
`setPrintMode(connection, mode)`	Print mode ESC ! n
`setBold(connection, bold)`	Bold
`setUnderline(connection, underline)`	Underline

Method	Description
`setReverse(connection, reverse)`	Reverse
`setLineSpacing(connection, spacing)`	Line spacing
`setCharSpacing(connection, spacing)`	Character spacing
`setSingleByteMode(connection)`	Single-byte mode
`setCharacterSet(connection, charset)`	Character set
`setCharacterCodepage(connection, codepage)`	Code page
`setChineseMode(connection)`	Chinese mode FS &
`setDoubleByteEncoding(connection, n)`	Double-byte encoding
`setAsciiTextFontType(connection, fontType)`	ASCII font A/B
`setTextUpsideDown(connection, upsideDown)`	Upside-down text
`setTextRotate(connection, rotate)`	Rotate 90°
`setTextLineHeight(connection, lineHeight)`	Line height
`setAsciiTextCharRightSpacing(connection, spacing)`	ASCII right spacing
`setKanjiTextCharSpacing(connection, left, right)`	Chinese left/right spacing
`setPrintPosition(connection, position)`	Print start position
`setPrintAreaLeftMargin(connection, margin)`	Left margin
`setPrintAreaWidth(connection, width)`	Print area width
`setHorizontalAbsolutePrintPosition(connection, pos)`	Absolute horizontal position
`setHorizontalRelativePrintPosition(connection, offset)`	Relative horizontal offset
`setVerticalAbsolutePrintPosition(connection, pos)`	Absolute vertical position
`setVerticalRelativePrintPosition(connection, offset)`	Relative vertical offset

14.5 Standard Print — Barcode / QR / Image

Method	Description
`printBarcode(connection, type, data, hri, height, width)`	1D barcode
`setBarcodeHRIPosition(connection, pos)`	Barcode HRI position
`setBarcodeHeight(connection, height)`	Barcode height
`setBarcodeWidth(connection, width)`	Barcode narrow-bar width
`setBarcodeReadableTextFontType(connection, fontType)`	Barcode HRI font
`setBarcodeReadableTextPosition(connection, position)`	Barcode HRI text position
`printQRCode(connection, content, size)`	QR code (simple)

Method	Description
`printQRCodeStandard(connection, content, size, align, ecc)`	QR code (standard)
`printDoubleQRCode(connection, …)`	Dual QR code
`printPDF417(connection, …)`	PDF417
`printRasterImageFromFile(connection, path)`	Image (file, AUTO compression)
`printRasterImageFromFile(connection, path, w, h)`	Image (file + size, AUTO)
`printRasterImageFromFile(connection, path, w, h, mode)`	Image (file + size + compression mode)
`printRasterImageFromData(connection, bytes)`	Image (bytes, AUTO)
`printRasterImageFromData(connection, bytes, w, h)`	Image (bytes + size, AUTO)
`printRasterImageFromData(connection, bytes, w, h, mode)`	Image (bytes + size + compression mode)
`printRasterImageFromPixels(connection, pixels, w, h)`	Image (pixels, AUTO)
`printRasterImageFromPixels(connection, pixels, w, h, tw, th)`	Image (pixels + size, AUTO)
`printRasterImageFromPixels(connection, pixels, w, h, tw, th, mode)`	Image (pixels + size + compression mode)

14.6 Standard Print — Feed / Cut / Peripherals / Buffer

Method	Description
`feedLine(connection)`	Line feed
`feedPaper(connection, lines)`	Feed N lines
`feedPaper(connection)`	Feed (dots)
`reverseFeed(connection, lines)`	Reverse feed N lines
`horizontalTab(connection)`	Horizontal tab
`setTabPositions(connection, positions)`	Set tab stops
`feedAndCutPaper(connection)`	Feed and cut
`cutPaper(connection, cutType)`	Cut paper
`printAndFeedDots(connection, n)`	Print and feed N dots
`printHorizontalLine(connection, start, end)`	Horizontal line
`fixedHorizontalLine(connection)`	Fixed-width horizontal line
`openCashDrawer(connection, drawerIndex)`	Open cash drawer
`beep(connection)`	Beep (default)
`beep(connection, count, duration)`	Beep (custom)

Method	Description
`clearBuffer(connection)`	Clear buffer
`cancelPrintCmd(connection)`	Cancel-print command

14.7 Page Mode

Method	Description
`selectPageMode(connection)`	Enter page mode
`exitPageMode(connection)`	Exit page mode
`selectPageModeEx(connection, …)`	Enter page mode (3 overloads)
`pagePrintPage(connection)`	Print page content
`pageClearPage(connection)`	Clear page buffer
`setMovementUnits(connection, x, y)`	Movement units
`pageSetPageArea(connection, x, y, w, h)`	Set page area
`pageSetPageDrawDirection(connection, dir)`	Page draw direction
`setPrintAreaInPageMode(connection, x, y, w, h)`	Page-mode print area
`setPrintAreaDirection(connection, dir)`	Page-mode area direction
`drawRectangleInPageMode(connection, mode, x, y, w, h)`	Draw rectangle
`drawRectangleBorderInPageMode(connection, …)`	Rectangle border
`drawFilledRectangleInPageMode(connection, …)`	Filled rectangle
`drawTextInPageMode(connection, x, y, text)`	Draw text
`drawTextUTF8InPageMode(connection, …)`	UTF-8 text
`drawTextGBKInPageMode(connection, …)`	GBK text
`drawTextBIG5InPageMode(connection, …)`	BIG5 text
`drawTextShiftJISInPageMode(connection, …)`	Shift-JIS text
`drawTextEUCKRInPageMode(connection, …)`	EUC-KR text
`drawBarcodeInPageMode(connection, …)`	Draw barcode
`drawQRCodeInPageMode(connection, …)`	Draw QR code
`drawImageFromFileInPageMode(connection, …)`	Draw image (file)
`drawImageFromDataInPageMode(connection, …)`	Draw image (data)
`drawImageFromPixelsInPageMode(connection, …)`	Draw image (pixels)

14.8 Label Mode

Method	Description
`enableLabelMode(connection)`	Enable label mode
`disableLabelMode(connection)`	Disable label mode
`labelPageBegin(connection, x, y, w, h, rotation)`	Begin label page
`backPaperToPrintPosition(connection)`	Back paper to print position
`labelPagePrint(connection)`	Print label
`labelPagePrint(connection, printNum)`	Print N label copies
`printLabelText(connection, …)`	Label text (2 overloads)
`printLabelTextUTF8(connection, …)`	Label UTF-8 (2 overloads)
`printLabelTextGBK(connection, …)`	Label GBK (2 overloads)
`printLabelBarcode(connection, …)`	Label barcode (2 overloads)
`setLabelBarcodeTextPosition(connection, position)`	Label barcode text position
`printLabelQRCode(connection, …)`	Label QR code (2 overloads)
`printLabelPDF417Code(connection, …)`	Label PDF417 (2 overloads)
`drawLabelImageFromFile(connection, …)`	Label image (file)
`drawLabelImageFromData(connection, …)`	Label image (data)
`drawLabelImageFromPixels(connection, …)`	Label image (pixels)
`drawLabelImageAdvanced(connection, …)`	Label image (advanced)
`drawLabelLine(connection, …)`	Label line (2 overloads)
`drawLabelRect(connection, …)`	Label rectangle (2 overloads)
`drawLabelBox(connection, …)`	Label filled box

14.9 Status Query

Method	Return	Description
`getPrinterStatus(connection)`	`int`	Get status byte
`isPrinterOnline(connection)`	`boolean`	Whether online
`hasPaper(connection)`	`boolean`	Whether paper is present
`isCoverOpen(connection)`	`boolean`	Whether cover is open
`queryRTStatus(connection, type, timeout)`	`byte[]`	DLE EOT real-time status
`queryTransmitStatus(connection, type, timeout)`	`byte[]`	GS r transmit status

14.10 Interfaces

Name	Description
`PrinterSDK.NetworkConnectionCallback`	Async network connect callback (extends `CommunicationManager.NetworkConnectionCallback`)
`BluetoothPrinterManager.BluetoothDeviceListener`	Bluetooth discovery / pairing listener

> Method signatures and parameters are in `PrinterSDK.java` source or IDE autocomplete. Full API details: English API reference · Chinese API documentation.

---

15. ESC/POS Command Coverage (Compared with Epson EPOS)

This section maps Epson ESC/POS standard commands to current SDK support. Legend: ✅ **Implemented and exposed** · ⚠️ **Implemented underneath but not wired/exposed** · ❌ **Not implemented**

15.1 Standard Mode — Implemented ✅

EPOS command	Hex	SDK method
ESC @ Initialize	`1B 40`	`initPrinter()` / `resetPrinter()`
ESC a Align	`1B 61`	`setAlignment()`
ESC ! Print mode	`1B 21`	`setPrintMode()`
ESC E Bold	`1B 45`	`setBold()`
ESC - Underline	`1B 2D`	`setUnderline()`
ESC { Upside-down	`1B 7B`	`setTextUpsideDown()`
ESC V Rotate 90°	`1B 56`	`setTextRotate()`
ESC M Font	`1B 4D`	`setAsciiTextFontType()`
ESC 3 Line spacing	`1B 33`	`setLineSpacing()` / `setTextLineHeight()`
ESC SP Char spacing	`1B 20`	`setCharSpacing()` / `setAsciiTextCharRightSpacing()`
ESC R International charset	`1B 52`	`setCharacterSet()`
ESC t Code page	`1B 74`	`setCharacterCodepage()`
ESC $ Absolute horizontal	`1B 24`	`setPrintPosition()` / `setHorizontalAbsolutePrintPosition()`
ESC \ Relative horizontal	`1B 5C`	`setHorizontalRelativePrintPosition()`

EPOS command	Hex	SDK method
GS L Left margin	`1D 4C`	`setPrintAreaLeftMargin()`
GS W Print area width	`1D 57`	`setPrintAreaWidth()`
GS ! Character size	`1D 21`	`setTextScale()`
GS B Reverse	`1D 42`	`setReverse()`
GS V Cut	`1D 56`	`cutPaper()` / `feedAndCutPaper()`
ESC i / ESC m Cut	`1B 69` / `1B 6D`	`cutPaper(CUT_FULL/PARTIAL)`
ESC d Feed n lines	`1B 64`	`feedPaper(lines)`
ESC J Feed n dots	`1B 4A`	`feedPaper()` / `printAndFeedDots()`
ESC e Reverse feed	`1B 65`	`reverseFeed(lines)`
HT Horizontal tab	`09`	`horizontalTab()`
ESC D Tab stops	`1B 44`	`setTabPositions()`
GS r Transmit status	`1D 72`	`queryTransmitStatus()`
DLE EOT Real-time status	`10 04`	`queryRTStatus()`
LF Line feed	`0A`	`feedLine()` / auto-appended at text end
ESC p Cash drawer	`1B 70`	`openCashDrawer()`
ESC B Beep	`1B 42`	`beep()`
GS h/H/w/k Barcode	`1D 68/48/77/6B`	`printBarcode()` + barcode format setters
GS (k QR/PDF417	`1D 28 6B`	`printQRCode*()` / `printPDF417()` / `printDoubleQRCode()`
GS v 0 Raster image	`1D 76 30`	`printRasterImageFrom*()` (`RASTER_COMPRESSION_GS_V0`)
US (J JBIG raster	`1F 28 4A`	`printRasterImageFrom*()` (`RASTER_COMPRESSION_JBIG` / AUTO)
US (P Line RLE	`1F 28 50`	`printRasterImageFrom*()` (`RASTER_COMPRESSION_LINE_RLE` , needs firmware support)
CAN Cancel	`18`	`clearBuffer()` / `cancelPrintCmd()`
GS (A Self-test	`1D 28 41 02 00 00 02`	`sendSelfTest()`
Raw commands	—	`sendCommand()`

15.2 Chinese Character Commands — Partially Implemented

EPOS command	Hex	Status	SDK method
FS & Chinese mode	`1C 26`	✅	`setChineseMode()`
FS . Cancel Chinese mode	`1C 2E`	✅	`setSingleByteMode()`

EPOS command	Hex	Status	SDK method
FS W Chinese double W/H	`1C 57`	✅	`setChineseDoubleSize()`
ESC C Chinese spacing	`1B 43`	✅	`setKanjiTextCharSpacing()`
ESC 39 Double-byte encoding	`1B 39`	✅	`setDoubleByteEncoding()`
FS ! Chinese print mode	`1C 21`	❌	—
FS - Chinese underline	`1C 2D`	❌	—
FS S Chinese spacing	`1C 53`	❌	—
FS 2 User-defined Chinese	`1C 32`	❌	—

15.3 Page Mode — ✅ ESC/POS Commands Fixed

Main reasons page mode previously failed:

Issue	Cause	Fix
Text/image coordinates ignored	Misused `GS ( N` (character color) for positioning	Changed to `ESC $` + `GS $`
Barcode/QR ignored	Used a non-existent page-mode-specific format	Position first, then send standard `GS k` / QR
Rectangle ignored	Misused `GS v 0` (raster) to draw rectangles	Changed to `GS ( Q` fn=49 (border) / fn=48 (fill)
No output after entering page mode	Missing `ESC W` print area and `ESC T` direction	`selectPageMode()` initializes automatically

`PageModeManager` uses ESC/POS page-mode commands from `PrinterCommands`:

Function	EPOS command	PrinterCommands	PageModeManager
Enter page mode	ESC L `1B 4C`	✅	✅
Exit page mode	ESC S `1B 53`	✅	✅
Page area	ESC W `1B 57`	✅	✅
Page direction	ESC T `1B 54`	✅	✅
Print page	ESC FF `1B 0C`	✅	✅
Clear page	CAN `18`	✅	✅
Movement units	GS P `1D 50`	✅	✅
In-page text	ESC $ + GS $ + text	✅	✅
In-page rectangle	GS (Q fn=49 (border) / fn=48 (fill)	✅	✅
In-page barcode/QR	ESC $ + GS $ + GS k / QR	✅	✅
In-page raster	ESC $ + GS $ + GS v 0	✅	✅

> **Note:** Page mode and label mode (JPL, `FS L` / `FS P`) are different command systems — do not mix them.

15.4 Label Mode — JPL Commands (Not ESC/POS Page Mode)

Label mode uses the **JPL command set** (`FS L`, `FS P`, etc.), implemented by `LabelModeManager` / `JPLCommands`. It is a separate system from ESC/POS page mode. Current label-mode coverage is relatively complete.

15.5 Common Unimplemented EPOS Commands ❌

These exist in the Epson EPOS standard but are **not wrapped by the SDK yet** (you can send them manually with `sendCommand()`):

Category	Command	Description	Frequency
Feed	**ESC 2**	Restore default 1/6 line spacing	Low
Character	**ESC G**	Double-strike	Low
Character	**ESC %** / **ESC &** / **ESC ?**	User-defined characters	Low
Character	**ESC *	Bit-image mode (legacy; superseded by GS v 0)	Low
Layout	**CR** `0D`	Carriage return	Low
Color	**ESC r**	Two-color printing	Low (two-color printers only)
Status	**GS I**	Transmit printer ID	Low
Sensor	**ESC c 3/4/5**	Paper sensor / panel keys	Low
Image	**GS *** / **GS /**	Download bit image	Low
Image	**FS p** / **FS q**	NV bitmap (`printNvBitmap` / page-mode `drawNvBitmapInPageMode`)	Medium
Macro	**GS :** / **GS ^**	Macro define / execute	Low
Other	**GS b**	Smoothing mode	Low
Other	**ESC =**	Select peripheral	Low

15.6 Commands Present Underneath but Not Exposed on PrinterSDK ⚠️

Location	Method	Description
`PrinterCommands`	`queryPrintResult()`	Query print result
`PrinterCommands`	`printImage()`	ESC * legacy bit image
`PrinterCommands`	`printBarcodeRaw()`	Raw barcode bytes

Location	Method	Description
`PrinterCommands`	Full page-mode set	Already exposed via PageModeManager

15.7 Coverage Summary

Module	Coverage
Standard receipt (text/format/barcode/QR/image/cut)	**High** — core ESC/POS covered
Chinese printing	**Medium–high** — missing helpers such as FS !, FS -
Page mode	**High** — ESC/POS page-mode commands wired
Label mode	**High** — independent JPL implementation
Status query	**Medium–high** — active query (DLE EOT / GS r) + convenience APIs
Advanced / rare commands	**Low** — macros, NV images, two-color, etc. not wrapped

---

Appendix: Version History

Version	Date	Highlights
1.0.0	2025-08-28	Initial release: USB / Bluetooth / network / serial
1.0.2	2025-08-29	Fixed missing `connect()` on network connection
1.2.0	2025-08-29	Added `connectNetworkPrinterAsync`
1.3.0	2025-08-29	Automatic USB / Bluetooth permission requests
1.5.0	2025-08-29	Fully automated permission requests; fixed bt socket closed and related issues
1.6.0	2026-07-07	Page mode wired to ESC/POS; added active status query (DLE EOT / GS r)
1.6.1	2026-07-08	Fixed page-mode coordinate / barcode / rectangle commands; added barcode TYPE selection guidance
1.6.2	2026-07-09	Fixed self-test (switch to GS (A); fixed label page print (`1A 4F` ; former `FS P` caused no paper out); fixed label text subcommand `1A 54` fn; added `printTextLineLeftRight` , page-mode layout helpers, `printNvBitmap` / `drawNvBitmapInPageMode` ; paper-width constants

---

Technical support: If you run into issues, check Logcat and see Section 13. Related docs in this project: PrinterSDK API Documentation (English), Chinese API documentation, and

`connectNetworkPrinterAsync使用指南.md`.