HSPOS iOS PrinterSDK User Manual

Item	Description
Product	HSPOS iOS PrinterSDK
Platform	iOS 15+
Entry class	`PrinterSDK`（`PrinterSDK.shared`）
Constants	`PrinterConstants`
Connection	`PrinterConnection`
Companion doc	*PrinterSDK iOS API Documentation*
Chinese edition	`PrinterSDK_iOS_使用说明书.pdf`

---

1. Introduction

PrinterSDK is a thermal-printer library for iOS. Through a single facade you can discover devices, connect, print, and query status.

Key capabilities:

- Connections: Bluetooth BLE and network TCP (typical port 9100)
- Standard ESC/POS receipt printing (text, barcode, QR, raster image, cut, etc.)
- Page-mode coordinate drawing
- JPL label printing
- Status query and buffer control

Prefer these public types in app code: `PrinterSDK`, `PrinterConstants`, `PrinterConnection`.

Full method and constant details are in the *API Documentation*.

---

2. Requirements & Integration (3 steps)

2.1 Environment

Item	Requirement
Minimum iOS	15.0
Language	Swift 5.9+
Xcode	15+ (recommended)
Distribution	`HSPOS-PrinterSDK-iOS-*.zip` (contains `Package.swift`)

2.2 Three-step integration (recommended)

Step 1 — Add the SDK to your project

1. Unzip the official package `HSPOS-PrinterSDK-iOS-x.x.x.zip`
2. Open your app project in Xcode
3. Choose **File → Add Package Dependencies...**
4. Click **Add Local...** and select the **unzipped folder** (it must contain `Package.swift`)
5. Check product **PrinterSDK** → **Add Package**

Then in your Swift files:

```
import PrinterSDK
```

Step 2 — Configure permissions

Copy the keys from the package file `Info-Plist-Snippet.plist` into your host app `Info.plist` (or add them in Xcode's Info panel):

Key	Purpose
`NSBluetoothAlwaysUsageDescription`	BLE scan / connect
`NSBluetoothPeripheralUsageDescription`	Legacy Bluetooth usage string
`NSLocalNetworkUsageDescription`	TCP LAN printers

Example strings:

- Bluetooth: `Need Bluetooth to connect to a thermal printer`
- Local Network: `Need local network access to reach a network printer`

iOS shows system permission dialogs; users can also manage them in Settings.

Step 3 — Print

```
import PrinterSDK

let sdk = PrinterSDK.shared

sdk.connectNetworkPrinterAsync(ip: "192.168.1.100", port: 9100) { conn in
    DispatchQueue.global(qos: .userInitiated).async {
        _ = sdk.initPrinter(conn)
        _ = sdk.setAlignment(conn, align: PrinterConstants.ALIGNMENT_CENTER)
        _ = sdk.printText(conn, text: "Hello PrinterSDK\n")
        _ = sdk.feedAndCutPaper(conn)
        sdk.disconnectPrinter(conn)
    }
} onFailed: { message in
    print(message)
}
```

Package contents:

Path	Content
`README.md`	Three-step summary
`Examples/MinimalUsage.swift`	Minimal sample
`Docs/*.pdf`	User manual & API docs (CN/EN)
`Sources/PrinterSDK/`	SDK sources (built by SPM)

Use only: `PrinterSDK`, `PrinterConstants`, `PrinterConnection`.

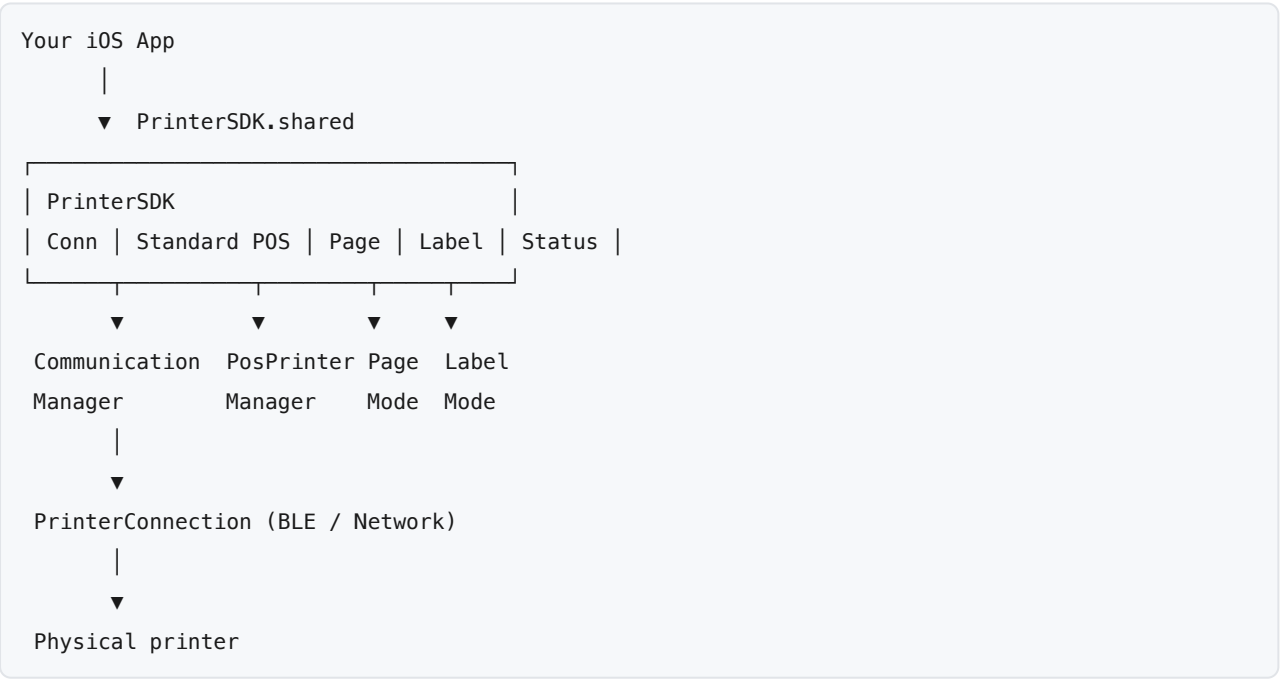
2.3 Package layout (for reference)

```
HSPOS-PrinterSDK-iOS-x.x.x/
├── Package.swift              ← select this folder for Add Local
├── Sources/PrinterSDK/        ← SDK sources
├── Docs/                      ← PDF docs
├── Examples/                  ← sample code
├── Info-Plist-Snippet.plist   ← permission keys
├── README.md
└── VERSION.txt
```

Do not call internal Manager types from app code, so upgrades stay compatible.

---

3. Architecture



Before every job: `connection != nil && connection.isConnected()`.

---

4. Quick Start

4.1 Initialization

```
let sdk = PrinterSDK.shared
// No extra Context required; call sdk.release() when done
```

4.2 Minimal print flow (network)

```
let sdk = PrinterSDK.shared

sdk.connectNetworkPrinterAsync(ip: "192.168.1.100", port: 9100) { conn in
    DispatchQueue.global(qos: .userInitiated).async {
        _ = sdk.initPrinter(conn)
        _ = sdk.setAlignment(conn, align: PrinterConstants.ALIGNMENT_CENTER)
        _ = sdk.printText(conn, text: "Hello PrinterSDK\n")
        _ = sdk.cutPaper(conn, cutType: PrinterConstants.CUT_PARTIAL)
        sdk.disconnectPrinter(conn)
    }
} onFailed: { message in
    print(message)
}
```

4.3 Minimal print flow (BLE)

```
final class MyPrinterHost: BluetoothDeviceListener {
    let sdk = PrinterSDK.shared

    func start() {
        sdk.addBluetoothListener(self)
        _ = sdk.startBluetoothDiscovery()
    }

    func onDeviceFound(_ peripheral: CBPeripheral, rssi: Int) {
        if let conn = sdk.connectBluetoothPrinter(peripheral) {
            _ = sdk.initPrinter(conn)
            _ = sdk.printText(conn, text: "BLE OK\n")
        }
    }

    func onDiscoveryFinished() {}
}
```

iOS shows a Bluetooth permission dialog. Power on the printer and ensure it supports HSPOS BLE (GATT Service `FF00`, write characteristic `FF02`).

---

5. Permissions (Info.plist)

See **§2.2 Step 2**. The package includes `Info-Plist-Snippet.plist` — copy it into the host app.

If you use Bonjour discovery, you may also declare `NSBonjourServices` (e.g. `_printer._tcp`).

---

6. Printer Connection

6.1 Supported transports

Transport	Notes
Bluetooth BLE	GATT Service `FF00`, write `FF02`
Network TCP	Typical RAW print port **9100**

6.2 BLE

```
sdk.addBluetoothListener(listener)
_ = sdk.startBluetoothDiscovery()
let devices = sdk.getBluetoothDevices()   // [CBPeripheral]
let conn = sdk.connectBluetoothPrinter(peripheral)
sdk.stopBluetoothDiscovery()
sdk.removeBluetoothListener(listener)
```

Notes:

- Device type is `CBPeripheral`
- Large payloads are chunked; the connection layer may coalesce small writes
- After complex jobs the connection flushes as needed

6.3 Network

```
// Recommended: async (safe on the main thread)
sdk.connectNetworkPrinterAsync(ip: "192.168.1.100", port: 9100,
    onSuccess: { conn in /* callback may not be on main */ },
    onFailed: { err in })

// Sync: do not call on the main thread
let conn = sdk.connectNetworkPrinter(ip: "192.168.1.100", port: 9100)
```

6.4 Disconnect & cleanup

```
sdk.disconnectPrinter(connection)
_ = sdk.clearPrinterBuffer(connection)
_ = sdk.cancelPrint(connection)
sdk.release()
```

---

7. Standard POS Printing

Common example:

```
_ = sdk.initPrinter(conn)
_ = sdk.setPrintAreaWidth(conn, width: PrinterConstants.PAPER_WIDTH_80MM)
_ = sdk.setAlignment(conn, align: PrinterConstants.ALIGNMENT_LEFT)
_ = sdk.printTextInGBK(conn, text: "Welcome\n")
_ = sdk.printTextLineLeftRight(conn, left: "Item A", right: "$12.00")
_ = sdk.printQRCode(conn, content: "https://hsprinter.com", size: 3)
_ = sdk.feedAndCutPaper(conn)
```

7.1 Text scaling (do not mix carelessly)

Method	Command	Use
`setChineseDoubleSize`	`FS W n`	CJK double width/height on/off
`setTextScale`	`GS ! n`	Width/height scale 0–7
`setPrintMode`	`ESC ! n`	Print-mode double width/height

The printer applies the last related command received.

7.2 Multi-encoding text

- `printText` (default)
- `printTextInUTF8` / `printTextInGBK` / `printTextInBIG5`
- `printTextInShiftJIS` / `printTextInEUCKR`

For Simplified Chinese receipts, prefer `printTextInGBK`.

7.3 Raster images

- `printRasterImageFromFile` / `FromData` / `FromPixels`
- Compression constants include RLE / JBIG / AUTO; **the current implementation always outputs GS v 0** (other modes fall back)

See the *API Documentation* for full parameters.

---

8. Page Mode

Recommended flow:

1. `selectPageMode` / `selectPageModeEx(...)`
2. `setPrintAreaInPageMode`, then draw text / lines / rectangles / barcodes / QR
3. `pagePrintPage`
4. `exitPageMode`

Some firmwares are unstable with rasters inside page mode. For complex layouts see Demo `PageModeExtendedPrinter` (text in page mode, raster after exit).

---

9. Label Mode (JPL)

```
guard sdk.enableLabelMode(conn) else { return }
defer { _ = sdk.disableLabelMode(conn) }

_ = sdk.labelPageBegin(conn, x: 0, y: 0, width: 400, height: 300, rotation: 0)
_ = sdk.printLabelTextGBK(conn, text: "Label", x: 20, y: 20, fontHeight: 24, fontType: 0)
_ = sdk.printLabelQRCode(conn, data: "ABC123", x: 20, y: 80, size: 4)
_ = sdk.labelPagePrint(conn)
```

Notes:

- Label commands target **label printers**; receipt printers may print nothing or return failure
- Invalid parameters return `false` and do not crash the app
- Label bitmaps default to source width (capped at 576 dots); use `targetWidth` to override

---

10. Status Query

```
let status = sdk.getPrinterStatus(conn)
let online = sdk.isPrinterOnline(conn)
let paper  = sdk.hasPaper(conn)
let cover  = sdk.isCoverOpen(conn)

_ = sdk.queryRTStatus(conn, type: PrinterConstants.STATUS_TYPE_PRINTER)
_ = sdk.queryTransmitStatus(conn, type: PrinterConstants.STATUS_TYPE_OFFLINE)
```

Status bits: `STATUS_ONLINE` / `STATUS_PAPER_OUT` / `STATUS_COVER_OPEN`.

BLE notification replies may take 1–2 seconds; do not set timeouts too short.

---

11. Common Constants

Constant	Value	Meaning
`ALIGNMENT_LEFT/CENTER/RIGHT`	0/1/2	Alignment
`CUT_FULL` / `CUT_PARTIAL`	0/1	Full / partial cut
`PAPER_WIDTH_58MM`	384	58 mm printable width (dots)
`PAPER_WIDTH_80MM` / `PAPER_WIDTH_72MM`	576	~72 mm effective width on 80 mm paper
`CHAR_COLUMNS_72MM`	48	~48 Font A columns

More constants: *API Documentation*, Constants chapter.

---

12. Demo

The in-repo Demo (`ContentView` + `Demo/Print/*`) covers:

- Connect: BLE / network
- Tests: 58/80 mm receipts, graphic receipts, raster, table, page mode, labels, status, etc.

Run print jobs on a background queue (`PrintJobRunner`) so the UI stays responsive.

---

13. FAQ

Q: Bluetooth will not connect?

A: Check system Bluetooth and app permission; confirm BLE GATT `FF00/FF02`; move closer and rescan.

Q: Network connect times out?

A: Ensure the phone/tablet and printer are on the same LAN, IP/port (often 9100) are correct, and Local Network permission is granted.

Q: Label APIs do nothing on a receipt printer?

A: JPL targets label printers. Confirm model and media; Demo shows an error on failure.

Q: Garbled CJK text?

A: Prefer `printTextInGBK`, or `setDoubleByteEncoding(..., GBK)` first. Page-mode encodings need firmware support.

Q: BLE print is slow or stalls?

A: Confirm the write characteristic and permissions; the SDK chunks large writes. Without notify, status query may fail while raw print still works.

---

14. Documents & Distribution

Item	Purpose
This *User Manual*	Integration, connection, typical flows
API Documentation	Signatures, parameters, returns, constants
`HSPOS-PrinterSDK-iOS-*.zip`	Customer delivery package (3-step integration)

Source: `DOC/PrinterSDK_iOS_User_Manual.md`.
