HSPOS iOS PrinterSDK 函数说明

项	说明
产品	HSPOS iOS PrinterSDK
平台	iOS 15+
入口类	`PrinterSDK`
常量类	`PrinterConstants`
连接协议	`PrinterConnection`
配套文档	《PrinterSDK iOS 使用说明书》
英文版	`PrinterSDK_iOS_API_Documentation.pdf`

---

1. 使用约定

- 绝大多数打印相关方法的第一个参数为 `PrinterConnection?`；连接无效时返回 `false` 或 `nil`。
- 返回 `Bool` 的方法：`true` 表示指令已成功写入连接（不等于打印机一定出纸成功）。
- 建议在业务层只调用 `PrinterSDK` 门面方法。
- 坐标、宽度等长度单位一般为**点（dots）**，约 8 dots/mm（203 DPI）。

2. 辅助类型

2.1 `PrinterConnection`

打印机连接抽象。常用方法：`connect()` / `disconnect()` / `isConnected()` / `write(_:)` / `read(timeout:)` / `clearInput()` / `flushWrites()`。

2.2 `BluetoothDeviceListener`

```
public protocol BluetoothDeviceListener: AnyObject {
    func onDeviceFound(_ peripheral: CBPeripheral, rssi: Int)
    func onDiscoveryFinished()
}
```

---

3. 入口类 `PrinterSDK` API

1. 生命周期

`shared`

全局单例入口。

```
public static let shared: PrinterSDK
```

`release`

释放 SDK 资源（断开连接等）。

```
public func release()
```

`getCommunicationManager`

获取底层通信管理器（高级用法）。

```
public func getCommunicationManager() -> CommunicationManager
```

返回： `CommunicationManager`

2. 连接 — 蓝牙 BLE

`getBluetoothDevices`

返回当前扫描到的 BLE 外设列表。

```
public func getBluetoothDevices() -> [CBPeripheral]
```

返回： `[CBPeripheral]`

`startBluetoothDiscovery`

开始 BLE 扫描。

```
public func startBluetoothDiscovery() -> Bool
```

返回： `Bool`

`stopBluetoothDiscovery`

停止 BLE 扫描。

```
public func stopBluetoothDiscovery()
```

`refreshBluetoothDevices`

刷新扫描（先停后启）。

```
public func refreshBluetoothDevices() -> Bool
```

返回： `Bool`

`connectBluetoothPrinter`

连接指定 BLE 打印机，成功返回连接对象。

```
public func connectBluetoothPrinter(_ peripheral: CBPeripheral) -> PrinterConnection?
```

参数	类型	说明
`peripheral`	`CBPeripheral`	BLE 外设

返回： `PrinterConnection?`

`addBluetoothListener`

注册蓝牙设备发现监听。

```
public func addBluetoothListener(_ listener: BluetoothDeviceListener)
```

参数	类型	说明
`listener`	`BluetoothDeviceListener`	蓝牙监听

`removeBluetoothListener`

移除蓝牙设备发现监听。

```
public func removeBluetoothListener(_ listener: BluetoothDeviceListener)
```

参数	类型	说明
`listener`	`BluetoothDeviceListener`	蓝牙监听

3. 连接 — 网口 TCP

`connectNetworkPrinter`

同步连接网口打印机（勿在主线程调用）。

```
public func connectNetworkPrinter(ip: String, port: Int) -> PrinterConnection?
```

参数	类型	说明
`ip`	`String`	打印机 IP
`port`	`Int`	端口，常见 9100

返回： `PrinterConnection?`

`connectNetworkPrinterAsync`

异步连接网口打印机（可在主线程调用）。

```
public func connectNetworkPrinterAsync( ip: String, port: Int, onSuccess: @escaping
(PrinterConnection) -> Void, onFailed: @escaping (String) -> Void )
```

参数	类型	说明
`ip`	`String`	打印机 IP
`port`	`Int`	端口，常见 9100
`onSuccess`	`@escaping (PrinterConnection) -> Void`	—
`onFailed`	`@escaping (String) -> Void`	—

返回： `Void, onFailed: @escaping (String) -> Void )`

`disconnectPrinter`

断开打印机连接。

```
public func disconnectPrinter(_ connection: PrinterConnection?)
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

4. 状态查询与控制

`queryRTStatus`

查询实时状态。

```
public func queryRTStatus(_ connection: PrinterConnection?, type: Int, timeout: Int = 3000) -> Data?
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`type`	`Int`	—
`timeout`	`Int`	超时（毫秒）；默认 `3000`

返回： `Data?`

`queryTransmitStatus`

查询传送状态。

```
public func queryTransmitStatus(_ connection: PrinterConnection?, type: Int, timeout: Int = 3000) -> Data?
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`type`	`Int`	—
`timeout`	`Int`	超时（毫秒）；默认 `3000`

返回： `Data?`

`clearPrinterBuffer`

清空打印机缓冲。

```
public func clearPrinterBuffer(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`cancelPrint`

取消当前打印。

```
public func cancelPrint(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`getPrinterStatus`

综合查询打印机状态位掩码。

```
public func getPrinterStatus(_ connection: PrinterConnection?) -> Int
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Int`

`isPrinterOnline`

是否在线。

```
public func isPrinterOnline(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`hasPaper`

是否有纸。

```
public func hasPaper(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`isCoverOpen`

盖板是否打开。

```
public func isCoverOpen(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`clearBuffer`

发送清缓冲指令。

```
public func clearBuffer(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`cancelPrintCmd`

发送取消打印指令。

```
public func cancelPrintCmd(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

5. POS — 初始化与文本

`initPrinter`

初始化打印机（ESC @）。

```
public func initPrinter(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`resetPrinter`

复位打印机。

```
public func resetPrinter(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`printText`

按默认编码打印文本。

```
public func printText(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本

返回： `Bool`

`printTextLineLeftRight`

左右两列对齐打印一行。

```
public func printTextLineLeftRight( _ connection: PrinterConnection?, left: String, right: String, areaWidthDots: Int = PrinterConstants.PAPER_WIDTH_80MM, charWidthDots: Int = 12 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`String`	坐标（点）
`right`	`String`	坐标（点）
`areaWidthDots`	`Int`	区域宽度（点）；默认 `PrinterConstants.PAPER_WIDTH_80MM`
`charWidthDots`	`Int`	默认 `12`

返回： `Bool`

```
public func printTextLineLeftRight(_ connection: PrinterConnection?, left: String, right: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`String`	坐标（点）
`right`	`String`	坐标（点）

返回： `Bool`

`printTextLineLeftRightGBK`

左右两列 GBK 行。

```
public func printTextLineLeftRightGBK( _ connection: PrinterConnection?, left: String, right: String, lineCharWidth: Int = 48 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`String`	坐标（点）
`right`	`String`	坐标（点）
`lineCharWidth`	`Int`	默认 `48`

返回： `Bool`

`printTextInUTF8`

以 UTF-8 打印文本。

```
public func printTextInUTF8(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本

返回： `Bool`

`printTextInGBK`

以 GBK 打印文本（简体中文推荐）。

```
public func printTextInGBK(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

`text`	`String`	文本

返回： `Bool`

`printTextInBIG5`

以 BIG5 打印文本。

```
public func printTextInBIG5(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本

返回： `Bool`

`printTextInShiftJIS`

以 Shift-JIS 打印文本。

```
public func printTextInShiftJIS(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本

返回： `Bool`

`printTextInEUCKR`

以 EUC-KR 打印文本。

```
public func printTextInEUCKR(_ connection: PrinterConnection?, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本

返回： `Bool`

`printTabLine`

按制表位打印多列一行。

```
public func printTabLine(_ connection: PrinterConnection?, columns: [String], encodingName:
String? = nil) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`columns`	`[String]`	列文本
`encodingName`	`String?`	默认 `nil`

返回： `Bool`

6. POS — 格式设置

`setAlignment`

设置对齐方式。

```
public func setAlignment(_ connection: PrinterConnection?, align: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`align`	`Int`	ALIGNMENT_*

返回： `Bool`

`setBold`

设置加粗。

```
public func setBold(_ connection: PrinterConnection?, bold: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`bold`	`Bool`	—

返回： `Bool`

`setUnderline`

设置下划线。

```
public func setUnderline(_ connection: PrinterConnection?, underline: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`underline`	`Bool`	—

返回： `Bool`

`setReverse`

设置反白。

```
public func setReverse(_ connection: PrinterConnection?, reverse: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`reverse`	`Bool`	—

返回： `Bool`

`setLineSpacing`

设置行间距。

```
public func setLineSpacing(_ connection: PrinterConnection?, spacing: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`spacing`	`Int`	—

返回： `Bool`

`setCharSpacing`

设置字符间距。

```
public func setCharSpacing(_ connection: PrinterConnection?, spacing: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`spacing`	`Int`	—

返回： `Bool`

`setPrintMode`

设置打印模式（ESC !）。

```
public func setPrintMode(_ connection: PrinterConnection?, mode: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`mode`	`Int`	—

返回： `Bool`

`setChineseMode`

进入中文模式。

```
public func setChineseMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`setDoubleByteEncoding`

设置双字节编码类型。

```
public func setDoubleByteEncoding(_ connection: PrinterConnection?, encoding: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`encoding`	`Int`	DOUBLE_BYTE_ENCODING_*

返回： `Bool`

`setTextScale`

设置字符放大倍率（GS !）。

```
public func setTextScale(_ connection: PrinterConnection?, widthScale: Int, heightScale: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`widthScale`	`Int`	—
`heightScale`	`Int`	—

返回： `Bool`

`setChineseDoubleSize`

中文倍宽倍高（FS W）。

```
public func setChineseDoubleSize(_ connection: PrinterConnection?, enable: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

`enable`	`Bool`	—

返回： `Bool`

```
public func setChineseDoubleSize(_ connection: PrinterConnection?, n: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`n`	`Int`	—

返回： `Bool`

`setCharacterSet`

设置国际字符集。

```
public func setCharacterSet(_ connection: PrinterConnection?, charset: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`charset`	`Int`	—

返回： `Bool`

`setCharacterCodepage`

设置代码页。

```
public func setCharacterCodepage(_ connection: PrinterConnection?, codepage: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`codepage`	`Int`	—

返回： `Bool`

`setSingleByteMode`

切换到单字节模式。

```
public func setSingleByteMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`setAsciiTextFontType`

设置 ASCII 字体类型。

```
public func setAsciiTextFontType(_ connection: PrinterConnection?, fontType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`fontType`	`Int`	—

返回： `Bool`

`setTextUpsideDown`

设置倒置打印。

```
public func setTextUpsideDown(_ connection: PrinterConnection?, upsideDown: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`upsideDown`	`Bool`	—

返回： `Bool`

`setTextRotate`

设置旋转打印。

```
public func setTextRotate(_ connection: PrinterConnection?, rotate: Bool) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`rotate`	`Bool`	旋转

返回： `Bool`

`setTextLineHeight`

设置行高。

```
public func setTextLineHeight(_ connection: PrinterConnection?, lineHeight: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`lineHeight`	`Int`	—

返回： `Bool`

`setAsciiTextCharRightSpacing`

设置 ASCII 右间距。

```
public func setAsciiTextCharRightSpacing(_ connection: PrinterConnection?, spacing: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`spacing`	`Int`	—

返回： `Bool`

`setKanjiTextCharSpacing`

设置汉字左右间距。

```
public func setKanjiTextCharSpacing(_ connection: PrinterConnection?, left: Int, right: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`Int`	坐标（点）
`right`	`Int`	坐标（点）

返回： `Bool`

`setPrintAreaWidth`

设置打印区域宽度（点）。

```
public func setPrintAreaWidth(_ connection: PrinterConnection?, width: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`width`	`Int`	尺寸（点）

返回： `Bool`

`setPrintAreaLeftMargin`

设置左边界。

```
public func setPrintAreaLeftMargin(_ connection: PrinterConnection?, leftMargin: Int) -> Bool
```

参数	类型	说明

`connection`	`PrinterConnection?`	打印机连接；无效时失败
`leftMargin`	`Int`	—

返回： `Bool`

`setPrintPosition`

设置打印绝对位置。

```
public func setPrintPosition(_ connection: PrinterConnection?, position: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`position`	`Int`	—

返回： `Bool`

`setHorizontalAbsolutePrintPosition`

水平绝对定位。

```
public func setHorizontalAbsolutePrintPosition(_ connection: PrinterConnection?, position: Int)
-> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`position`	`Int`	—

返回： `Bool`

`setHorizontalRelativePrintPosition`

水平相对定位。

```
public func setHorizontalRelativePrintPosition(_ connection: PrinterConnection?, offset: Int) ->
Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`offset`	`Int`	—

返回： `Bool`

`setVerticalAbsolutePrintPosition`

垂直绝对定位。

```
public func setVerticalAbsolutePrintPosition(_ connection: PrinterConnection?, position: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`position`	`Int`	—

返回： `Bool`

`setVerticalRelativePrintPosition`

垂直相对定位。

```
public func setVerticalRelativePrintPosition(_ connection: PrinterConnection?, offset: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`offset`	`Int`	—

返回： `Bool`

`horizontalTab`

水平制表。

```
public func horizontalTab(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`setTabPositions`

设置制表位。

```
public func setTabPositions(_ connection: PrinterConnection?, positions: [Int]) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`positions`	`[Int]`	制表位

返回： `Bool`

7. POS — 条码 / 二维码 / PDF417

`printBarcode`

打印一维条码。

```
public func printBarcode( _ connection: PrinterConnection?, type: Int, data: String, hri: Int = PrinterConstants.BARCODE_HRI_BELOW, height: Int = 80, width: Int = 2 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`type`	`Int`	—
`data`	`String`	数据内容
`hri`	`Int`	默认 `PrinterConstants.BARCODE_HRI_BELOW`
`height`	`Int`	尺寸（点）；默认 `80`
`width`	`Int`	尺寸（点）；默认 `2`

返回： `Bool`

`setBarcodeHeight`

设置条码高度。

```
public func setBarcodeHeight(_ connection: PrinterConnection?, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`height`	`Int`	尺寸（点）

返回： `Bool`

`setBarcodeWidth`

设置条码模块宽度。

```
public func setBarcodeWidth(_ connection: PrinterConnection?, width: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`width`	`Int`	尺寸（点）

返回： `Bool`

`setBarcodeHRIPosition`

设置条码 HRI 位置。

```
public func setBarcodeHRIPosition(_ connection: PrinterConnection?, pos: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pos`	`Int`	—

返回： `Bool`

`setBarcodeReadableTextFontType`

设置 HRI 字体。

```
public func setBarcodeReadableTextFontType(_ connection: PrinterConnection?, fontType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`fontType`	`Int`	—

返回： `Bool`

`setBarcodeReadableTextPosition`

设置可读文本位置。

```
public func setBarcodeReadableTextPosition(_ connection: PrinterConnection?, position: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`position`	`Int`	—

返回： `Bool`

`printQRCode`

打印二维码（简化接口）。

```
public func printQRCode(_ connection: PrinterConnection?, content: String, size: Int = 3) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`content`	`String`	内容
`size`	`Int`	默认 `3`

返回： `Bool`

`printQRCodeStandard`

打印二维码（完整参数）。

```
public func printQRCodeStandard( _ connection: PrinterConnection?, content: String, size: Int, align: Int, eccLevel: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`content`	`String`	内容
`size`	`Int`	—
`align`	`Int`	ALIGNMENT_*
`eccLevel`	`Int`	—

返回： `Bool`

`printDoubleQRCode`

打印双二维码。

```
public func printDoubleQRCode( _ connection: PrinterConnection?, n: Int, qr1Pos: Int, qr1Data: String, qr1Ecc: Int, qr1Ver: Int, qr2Pos: Int, qr2Data: String, qr2Ecc: Int, qr2Ver: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`n`	`Int`	—
`qr1Pos`	`Int`	—
`qr1Data`	`String`	—
`qr1Ecc`	`Int`	—
`qr1Ver`	`Int`	—
`qr2Pos`	`Int`	—
`qr2Data`	`String`	—
`qr2Ecc`	`Int`	—
`qr2Ver`	`Int`	—

返回： `Bool`

`printPDF417`

打印 PDF417。

```
public func printPDF417( _ connection: PrinterConnection?, columns: Int, rows: Int, moduleWidth: Int, rowHeight: Int, errorCorrectionLevel: Int, option: Int, data: String ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`columns`	`Int`	列文本
`rows`	`Int`	—
`moduleWidth`	`Int`	—
`rowHeight`	`Int`	—
`errorCorrectionLevel`	`Int`	—
`option`	`Int`	—
`data`	`String`	数据内容

返回： `Bool`

8. POS — 图像 / 切纸 / 外设

`printRasterImageFromFile`

从文件打印光栅图。

```
public func printRasterImageFromFile( _ connection: PrinterConnection?, filePath: String, targetWidth: Int = 384, targetHeight: Int = 0, compressionMode: Int = PrinterConstants.RASTER_COMPRESSION_GS_V0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`filePath`	`String`	文件路径
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `384`
`targetHeight`	`Int`	目标高度；0 常表示按比例; 默认 `0`
`compressionMode`	`Int`	当前实现回退为 GS v 0; 默认 `PrinterConstants.RASTER_COMPRESSION_GS_V0`

返回： `Bool`

`printRasterImageFromData`

从图片 Data 打印光栅图。

```
public func printRasterImageFromData( _ connection: PrinterConnection?, imageData: Data, targetWidth: Int = 384, targetHeight: Int = 0, compressionMode: Int = PrinterConstants.RASTER_COMPRESSION_GS_V0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`imageData`	`Data`	图片数据
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `384`
`targetHeight`	`Int`	目标高度；0 常表示按比例; 默认 `0`
`compressionMode`	`Int`	当前实现回退为 GS v 0; 默认 `PrinterConstants.RASTER_COMPRESSION_GS_V0`

返回： `Bool`

`printRasterImageFromPixels`

从像素数组打印光栅图。

```
public func printRasterImageFromPixels( _ connection: PrinterConnection?, pixels: [UInt32], width: Int, height: Int, targetWidth: Int = 384, targetHeight: Int = 0, compressionMode: Int = PrinterConstants.RASTER_COMPRESSION_GS_V0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pixels`	`[UInt32]`	ARGB 像素
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `384`
`targetHeight`	`Int`	目标高度；0 常表示按比例; 默认 `0`
`compressionMode`	`Int`	当前实现回退为 GS v 0; 默认 `PrinterConstants.RASTER_COMPRESSION_GS_V0`

返回： `Bool`

`printImage`

打印 UIImage。

```
public func printImage(_ connection: PrinterConnection?, image: UIImage) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`image`	`UIImage`	UIImage

返回： `Bool`

`printNvBitmap`

打印 NV 位图。

```
public func printNvBitmap(_ connection: PrinterConnection?, nvNumber: Int, mode: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`nvNumber`	`Int`	—
`mode`	`Int`	—

返回： `Bool`

`cutPaper`

切纸。

```
public func cutPaper(_ connection: PrinterConnection?, cutType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`cutType`	`Int`	CUT_FULL / CUT_PARTIAL

返回： `Bool`

`feedAndCutPaper`

走纸并切纸。

```
public func feedAndCutPaper(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`feedLine`

走纸一行。

```
public func feedLine(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`feedPaper`

走纸。

```
public func feedPaper(_ connection: PrinterConnection?, lines: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`lines`	`Int`	—

返回： `Bool`

```
public func feedPaper(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`reverseFeed`

反向走纸。

```
public func reverseFeed(_ connection: PrinterConnection?, lines: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`lines`	`Int`	—

返回： `Bool`

`printAndFeedDots`

打印并按点数走纸。

```
public func printAndFeedDots(_ connection: PrinterConnection?, dots: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`dots`	`Int`	—

返回： `Bool`

`printHorizontalLine`

打印水平分隔线。

```
public func printHorizontalLine(_ connection: PrinterConnection?, start: Int, end: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`start`	`Int`	—
`end`	`Int`	—

返回： `Bool`

`fixedHorizontalLine`

打印固定宽度横线。

```
public func fixedHorizontalLine(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`openCashDrawer`

打开钱箱。

```
public func openCashDrawer(_ connection: PrinterConnection?, drawerIndex: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`drawerIndex`	`Int`	—

返回： `Bool`

`beep`

蜂鸣。

```
public func beep(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

```
public func beep(_ connection: PrinterConnection?, count: Int, duration: Int) -> Bool
```

参数	类型	说明

`connection`	`PrinterConnection?`	打印机连接；无效时失败
`count`	`Int`	—
`duration`	`Int`	—

返回： `Bool`

`sendSelfTest`

发送自检页指令。

```
public func sendSelfTest(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`sendCommand`

发送原始指令字节。

```
public func sendCommand(_ connection: PrinterConnection?, command: Data) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`command`	`Data`	—

返回： `Bool`

9. 页模式

`selectPageMode`

进入页模式。

```
public func selectPageMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`selectPageModeEx`

扩展进入页模式并设置区域。

```
public func selectPageModeEx( _ connection: PrinterConnection?, nHorizontalMovementUnit: Int, nVerticalMovementUnit: Int, x: Int, y: Int, width: Int, height: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`nHorizontalMovementUnit`	`Int`	—
`nVerticalMovementUnit`	`Int`	—
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

```
public func selectPageModeEx(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

```
public func selectPageModeEx(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`exitPageMode`

退出页模式。

```
public func exitPageMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`pagePrintPage`

打印页模式缓冲内容。

```
public func pagePrintPage(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`pageClearPage`

清空页模式缓冲。

```
public func pageClearPage(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`setMovementUnits`

设置移动单位。

```
public func setMovementUnits(_ connection: PrinterConnection?, x: Int, y: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）

返回： `Bool`

`setPrintAreaInPageMode`

设置页模式打印区域。

```
public func setPrintAreaInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）

`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`setPrintAreaDirection`

设置页模式打印方向。

```
public func setPrintAreaDirection(_ connection: PrinterConnection?, direction: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`direction`	`Int`	—

返回： `Bool`

`pageSetPageArea`

设置页模式打印区域（别名）。

```
public func pageSetPageArea(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`pageSetPageDrawDirection`

设置页模式绘制方向（别名）。

```
public func pageSetPageDrawDirection(_ connection: PrinterConnection?, direction: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`direction`	`Int`	—

返回： `Bool`

`drawLineInPageMode`

页模式画线。

```
public func drawLineInPageMode(_ connection: PrinterConnection?, x1: Int, y1: Int, x2: Int, y2: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x1`	`Int`	—
`y1`	`Int`	—
`x2`	`Int`	—
`y2`	`Int`	—

返回： `Bool`

```
public func drawLineInPageMode(_ connection: PrinterConnection?, x1: Int, y1: Int, x2: Int, y2: Int, lineType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x1`	`Int`	—
`y1`	`Int`	—
`x2`	`Int`	—
`y2`	`Int`	—
`lineType`	`Int`	—

返回： `Bool`

`drawRectangleInPageMode`

页模式画矩形。

```
public func drawRectangleInPageMode(_ connection: PrinterConnection?, mode: Int, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`mode`	`Int`	—
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）

`height`	`Int`	尺寸（点）

返回： `Bool`

`drawRectangleBorderInPageMode`

页模式画矩形边框。

```
public func drawRectangleBorderInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`drawFilledRectangleInPageMode`

页模式画填充矩形。

```
public func drawFilledRectangleInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`drawTextInPageMode`

页模式绘制文本。

```
public func drawTextInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawTextCenteredInPageMode`

页模式居中文本。

```
public func drawTextCenteredInPageMode(_ connection: PrinterConnection?, pageWidth: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

```
public func drawTextCenteredInPageMode(_ connection: PrinterConnection?, pageWidth: Int, y: Int, text: String, charWidthDots: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`text`	`String`	文本
`charWidthDots`	`Int`	—

返回： `Bool`

`drawTextRightInPageMode`

页模式右对齐文本。

```
public func drawTextRightInPageMode(_ connection: PrinterConnection?, pageWidth: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—

`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

```
public func drawTextRightInPageMode(_ connection: PrinterConnection?, pageWidth: Int, y: Int, text: String, charWidthDots: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`text`	`String`	文本
`charWidthDots`	`Int`	—

返回： `Bool`

`drawTextLineLeftRightInPageMode`

页模式左右两列文本。

```
public func drawTextLineLeftRightInPageMode(_ connection: PrinterConnection?, pageWidth: Int, y: Int, left: String, right: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`left`	`String`	坐标（点）
`right`	`String`	坐标（点）

返回： `Bool`

```
public func drawTextLineLeftRightInPageMode( _ connection: PrinterConnection?, pageWidth: Int, y: Int, left: String, right: String, charWidthDots: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`left`	`String`	坐标（点）

`right`	`String`	坐标（点）
`charWidthDots`	`Int`	—

返回： `Bool`

`drawNvBitmapInPageMode`

页模式绘制 NV 位图。

```
public func drawNvBitmapInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, nvNumber: Int, mode: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`nvNumber`	`Int`	—
`mode`	`Int`	—

返回： `Bool`

`drawNvBitmapCenteredInPageMode`

页模式居中 NV 位图。

```
public func drawNvBitmapCenteredInPageMode( _ connection: PrinterConnection?, pageWidth: Int, y: Int, nvNumber: Int, mode: Int, imageWidthDots: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`pageWidth`	`Int`	—
`y`	`Int`	坐标（点）
`nvNumber`	`Int`	—
`mode`	`Int`	—
`imageWidthDots`	`Int`	—

返回： `Bool`

`drawTextUTF8InPageMode`

页模式 UTF-8 文本。

```
public func drawTextUTF8InPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawTextGBKInPageMode`

页模式 GBK 文本。

```
public func drawTextGBKInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawTextBIG5InPageMode`

页模式 BIG5 文本。

```
public func drawTextBIG5InPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawTextShiftJISInPageMode`

页模式 Shift-JIS 文本。

```
public func drawTextShiftJISInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawTextEUCKRInPageMode`

页模式 EUC-KR 文本。

```
public func drawTextEUCKRInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, text: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`text`	`String`	文本

返回： `Bool`

`drawImageFromFileInPageMode`

页模式从文件绘图。

```
public func drawImageFromFileInPageMode( _ connection: PrinterConnection?, x: Int, y: Int, filePath: String, width: Int, height: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`filePath`	`String`	文件路径
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`drawImageFromDataInPageMode`

页模式从 Data 绘图。

```
public func drawImageFromDataInPageMode( _ connection: PrinterConnection?, x: Int, y: Int, imageData: Data, width: Int, height: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`imageData`	`Data`	图片数据
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`drawImageFromPixelsInPageMode`

页模式从像素绘图。

```
public func drawImageFromPixelsInPageMode( _ connection: PrinterConnection?, x: Int, y: Int, pixels: [UInt32], imgWidth: Int, imgHeight: Int, width: Int, height: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`pixels`	`[UInt32]`	ARGB 像素
`imgWidth`	`Int`	—
`imgHeight`	`Int`	—
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）

返回： `Bool`

`drawBarcodeInPageMode`

页模式绘制条码。

```
public func drawBarcodeInPageMode( _ connection: PrinterConnection?, x: Int, y: Int, barcodeType: Int, data: String, height: Int = 55, width: Int = 4 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）

`y`	`Int`	坐标（点）
`barcodeType`	`Int`	—
`data`	`String`	数据内容
`height`	`Int`	尺寸（点）；默认 `55`
`width`	`Int`	尺寸（点）；默认 `4`

返回： `Bool`

```
public func drawBarcodeInPageMode( _ connection: PrinterConnection?, x: Int, y: Int, barcodeType: Int, data: String, hriPos: Int, height: Int, width: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`barcodeType`	`Int`	—
`data`	`String`	数据内容
`hriPos`	`Int`	—
`height`	`Int`	尺寸（点）
`width`	`Int`	尺寸（点）

返回： `Bool`

`drawBarcodeInPageModeWithCenteredHri`

页模式绘制居中 HRI 条码。

```
public func drawBarcodeInPageModeWithCenteredHri( _ connection: PrinterConnection?, x: Int, y: Int, barcodeType: Int, data: String, height: Int, width: Int, hriGapDots: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`barcodeType`	`Int`	—
`data`	`String`	数据内容
`height`	`Int`	尺寸（点）
`width`	`Int`	尺寸（点）
`hriGapDots`	`Int`	—

返回： `Bool`

`drawQRCodeInPageMode`

页模式绘制二维码。

```
public func drawQRCodeInPageMode(_ connection: PrinterConnection?, x: Int, y: Int, version: Int, eccLevel: Int, data: String) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`version`	`Int`	—
`eccLevel`	`Int`	—
`data`	`String`	数据内容

返回： `Bool`

10. 标签模式（JPL）

`enableLabelMode`

进入标签（JPL）模式。

```
public func enableLabelMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`disableLabelMode`

退出标签模式。

```
public func disableLabelMode(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`labelPageBegin`

开始标签页面。

```
public func labelPageBegin(_ connection: PrinterConnection?, x: Int, y: Int, width: Int, height: Int, rotation: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）
`rotation`	`Int`	旋转

返回： `Bool`

`backPaperToPrintPosition`

回纸到打印位置。

```
public func backPaperToPrintPosition(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

`labelPagePrint`

打印标签页。

```
public func labelPagePrint(_ connection: PrinterConnection?) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

返回： `Bool`

```
public func labelPagePrint(_ connection: PrinterConnection?, printNum: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`printNum`	`Int`	打印份数

返回： `Bool`

`printLabelText`

标签绘制文本。

```
public func printLabelText(_ connection: PrinterConnection?, text: String, x: Int, y: Int, fontSize: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`fontSize`	`Int`	—

返回： `Bool`

```
public func printLabelText(_ connection: PrinterConnection?, text: String, x: Int, y: Int, fontHeight: Int, fontType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`fontHeight`	`Int`	—
`fontType`	`Int`	—

返回： `Bool`

`printLabelTextUTF8`

标签绘制 UTF-8 文本。

```
public func printLabelTextUTF8(_ connection: PrinterConnection?, text: String, x: Int, y: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）

返回： `Bool`

```
public func printLabelTextUTF8(_ connection: PrinterConnection?, text: String, x: Int, y: Int, fontHeight: Int, fontType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`fontHeight`	`Int`	—
`fontType`	`Int`	—

返回： `Bool`

`printLabelTextGBK`

标签绘制 GBK 文本。

```
public func printLabelTextGBK(_ connection: PrinterConnection?, text: String, x: Int, y: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）

返回： `Bool`

```
public func printLabelTextGBK(_ connection: PrinterConnection?, text: String, x: Int, y: Int, fontHeight: Int, fontType: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`text`	`String`	文本
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`fontHeight`	`Int`	—
`fontType`	`Int`	—

返回： `Bool`

`printLabelBarcode`

标签绘制一维码。

```
public func printLabelBarcode(_ connection: PrinterConnection?, type: Int, data: String, x: Int, y: Int, height: Int, width: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`type`	`Int`	—
`data`	`String`	数据内容
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`height`	`Int`	尺寸（点）
`width`	`Int`	尺寸（点）

返回： `Bool`

```
public func printLabelBarcode( _ connection: PrinterConnection?, x: Int, y: Int, barcodeType: Int, barcodeHeight: Int, unitWidth: Int, rotate: Int, data: String ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`barcodeType`	`Int`	—
`barcodeHeight`	`Int`	—
`unitWidth`	`Int`	—
`rotate`	`Int`	旋转
`data`	`String`	数据内容

返回： `Bool`

`setLabelBarcodeTextPosition`

标签条码可读文本位置。

```
public func setLabelBarcodeTextPosition(_ connection: PrinterConnection?, position: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败

`position`	`Int`	—

返回： `Bool`

`printLabelQRCode`

标签绘制二维码。

```
public func printLabelQRCode(_ connection: PrinterConnection?, data: String, x: Int, y: Int, size: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`data`	`String`	数据内容
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`size`	`Int`	—

返回： `Bool`

```
public func printLabelQRCode( _ connection: PrinterConnection?, x: Int, y: Int, version: Int, eccLevel: Int, unitWidth: Int, rotate: Int, data: String ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`version`	`Int`	—
`eccLevel`	`Int`	—
`unitWidth`	`Int`	—
`rotate`	`Int`	旋转
`data`	`String`	数据内容

返回： `Bool`

`printLabelPDF417Code`

标签绘制 PDF417。

```
public func printLabelPDF417Code( _ connection: PrinterConnection?, x: Int, y: Int, colNum: Int, eccLevel: Int, lwRatio: Int, unitWidth: Int, rotate: Int, data: String ) -> Bool
```

参数	类型	说明

`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`colNum`	`Int`	—
`eccLevel`	`Int`	—
`lwRatio`	`Int`	—
`unitWidth`	`Int`	—
`rotate`	`Int`	旋转
`data`	`String`	数据内容

返回： `Bool`

```
public func printLabelPDF417Code(_ connection: PrinterConnection?, x: Int, y: Int, data: String)
-> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`data`	`String`	数据内容

返回： `Bool`

`drawLabelImageFromFile`

标签从文件绘制位图。

```
public func drawLabelImageFromFile( _ connection: PrinterConnection?, x: Int, y: Int, filePath:
String, targetWidth: Int = 0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`filePath`	`String`	文件路径
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `0`

返回： `Bool`

`drawLabelImageFromData`

标签从 Data 绘制位图。

```
public func drawLabelImageFromData( _ connection: PrinterConnection?, x: Int, y: Int, imageData: Data, targetWidth: Int = 0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`imageData`	`Data`	图片数据
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `0`

返回： `Bool`

`drawLabelImageFromPixels`

标签从像素绘制位图。

```
public func drawLabelImageFromPixels( _ connection: PrinterConnection?, x: Int, y: Int, pixels: [UInt32], width: Int, height: Int, targetWidth: Int = 0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`pixels`	`[UInt32]`	ARGB 像素
`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `0`

返回： `Bool`

`drawLabelImageAdvanced`

标签高级位图绘制。

```
public func drawLabelImageAdvanced( _ connection: PrinterConnection?, x: Int, y: Int, pixels: [UInt32], width: Int, height: Int, showType: Int, targetWidth: Int = 0 ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`x`	`Int`	坐标（点）
`y`	`Int`	坐标（点）
`pixels`	`[UInt32]`	ARGB 像素

`width`	`Int`	尺寸（点）
`height`	`Int`	尺寸（点）
`showType`	`Int`	—
`targetWidth`	`Int`	目标宽度（点）；标签侧 ≤0 表示原图宽（上限 576）；默认 `0`

返回： `Bool`

`drawLabelLine`

标签画线。

```
public func drawLabelLine(_ connection: PrinterConnection?, startX: Int, startY: Int, endX: Int, endY: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`startX`	`Int`	坐标（点）
`startY`	`Int`	坐标（点）
`endX`	`Int`	坐标（点）
`endY`	`Int`	坐标（点）

返回： `Bool`

```
public func drawLabelLine( _ connection: PrinterConnection?, startX: Int, startY: Int, endX: Int, endY: Int, width: Int, color: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`startX`	`Int`	坐标（点）
`startY`	`Int`	坐标（点）
`endX`	`Int`	坐标（点）
`endY`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`color`	`Int`	—

返回： `Bool`

`drawLabelRect`

标签画矩形。

```
public func drawLabelRect(_ connection: PrinterConnection?, left: Int, top: Int, right: Int, bottom: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`Int`	坐标（点）
`top`	`Int`	坐标（点）
`right`	`Int`	坐标（点）
`bottom`	`Int`	坐标（点）

返回： `Bool`

```
public func drawLabelRect( _ connection: PrinterConnection?, left: Int, top: Int, right: Int, bottom: Int, width: Int, color: Int ) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`Int`	坐标（点）
`top`	`Int`	坐标（点）
`right`	`Int`	坐标（点）
`bottom`	`Int`	坐标（点）
`width`	`Int`	尺寸（点）
`color`	`Int`	—

返回： `Bool`

`drawLabelBox`

标签画填充框。

```
public func drawLabelBox(_ connection: PrinterConnection?, left: Int, top: Int, right: Int, bottom: Int, color: Int) -> Bool
```

参数	类型	说明
`connection`	`PrinterConnection?`	打印机连接；无效时失败
`left`	`Int`	坐标（点）
`top`	`Int`	坐标（点）
`right`	`Int`	坐标（点）
`bottom`	`Int`	坐标（点）
`color`	`Int`	—

返回： `Bool`

11. 工具

`estimateCode128WidthDots`

估算 Code128 宽度（点）。

```
public func estimateCode128WidthDots(_ data: String, moduleWidth: Int) -> Int
```

参数	类型	说明
`data`	`String`	数据内容
`moduleWidth`	`Int`	—

返回： `Int`

`estimateTextWidthDots`

估算文本宽度（点）。

```
public func estimateTextWidthDots(_ text: String, charWidthDots: Int) -> Int
```

参数	类型	说明
`text`	`String`	文本
`charWidthDots`	`Int`	—

返回： `Int`

---

4. 常量 `PrinterConstants`

常量	值	说明
`ALIGNMENT_LEFT`	`0`	左对齐
`ALIGNMENT_CENTER`	`1`	居中
`ALIGNMENT_RIGHT`	`2`	右对齐
`CUT_FULL`	`0`	全切
`CUT_PARTIAL`	`1`	半切
`PAPER_WIDTH_58MM`	`384`	58mm 可打宽度
`PAPER_WIDTH_80MM`	`576`	80mm 有效宽（约72mm）
`PAPER_WIDTH_72MM`	`PAPER_WIDTH_80MM`	同 PAPER_WIDTH_80MM
`CHAR_COLUMNS_72MM`	`48`	72mm Font A 列数
`NV_BITMAP_MODE_NORMAL`	`0`	—
`NV_BITMAP_MODE_DOUBLE_WIDTH`	`1`	—
`NV_BITMAP_MODE_DOUBLE_HEIGHT`	`2`	—

NV_BITMAP_MODE_DOUBLE_BOTH	3	—
BARCODE_HRI_NONE	0	—
BARCODE_HRI_BELOW	2	—
BARCODE_TYPE_UPC_A	0x41	—
BARCODE_TYPE_UPC_E	0x42	—
BARCODE_TYPE_EAN13	0x43	—
BARCODE_TYPE_EAN8	0x44	—
BARCODE_TYPE_CODE39	0x45	—
BARCODE_TYPE_ITF	0x46	—
BARCODE_TYPE_CODABAR	0x47	—
BARCODE_TYPE_CODE93	0x48	—
BARCODE_TYPE_CODE128	0x49	—
BARCODE_CODE128	BARCODE_TYPE_CODE128	—
BARCODE_CODE39	BARCODE_TYPE_CODE39	—
BARCODE_EAN13	BARCODE_TYPE_EAN13	—
QRCODE_SIZE_SMALL	1	—
QRCODE_SIZE_NORMAL	2	—
QRCODE_SIZE_LARGE	3	—
QR_ECC_LEVEL_L	1	—
QR_ECC_LEVEL_M	2	—
QR_ECC_LEVEL_Q	3	—
QR_ECC_LEVEL_H	4	—
PAGE_DIRECTION_LEFT_TO_RIGHT	0	—
PAGE_DIRECTION_BOTTOM_TO_TOP	1	—
PAGE_DIRECTION_RIGHT_TO_LEFT	2	—
PAGE_DIRECTION_TOP_TO_BOTTOM	3	—
STATUS_OFFLINE	0x00	离线
STATUS_ONLINE	0x01	在线
STATUS_PAPER_OUT	0x02	缺纸
STATUS_COVER_OPEN	0x04	开盖
STATUS_TYPE_PRINTER	1	打印机状态
STATUS_TYPE_OFFLINE	2	离线状态
STATUS_TYPE_ERROR	3	错误状态
STATUS_TYPE_PAPER_SENSOR	4	纸传感器

`CASH_DRAWER_PIN2`	`0`	钱箱 pin2
`CASH_DRAWER_PIN5`	`1`	钱箱 pin5
`BEEP_DEFAULT_COUNT`	`1`	—
`BEEP_DEFAULT_DURATION`	`100`	—
`RASTER_COMPRESSION_GS_V0`	`0`	GS v 0（实际输出）
`RASTER_COMPRESSION_LINE_RLE`	`1`	RLE（回退 GS v 0）
`RASTER_COMPRESSION_JBIG`	`2`	JBIG（回退 GS v 0）
`RASTER_COMPRESSION_AUTO`	`3`	自动（回退 GS v 0）
`LABEL_FONT_HEIGHT_24`	`24`	—
`LABEL_BARCODE_TYPE_CODE128`	`8`	—
`LABEL_BARCODE_HEIGHT_DEFAULT`	`80`	—
`LABEL_BARCODE_UNIT_WIDTH_2`	`2`	—
`DOUBLE_BYTE_ENCODING_GBK`	`0`	GBK
`DOUBLE_BYTE_ENCODING_UTF8`	`1`	UTF-8
`DOUBLE_BYTE_ENCODING_SHIFT_JIS`	`2`	Shift-JIS
`DOUBLE_BYTE_ENCODING_BIG5`	`3`	BIG5
`DOUBLE_BYTE_ENCODING_EUC_KR`	`4`	EUC-KR
`ASCII_FONT_TYPE_A`	`0`	—
`ASCII_FONT_TYPE_B`	`1`	—
`LINE_SPACING_DEFAULT`	`0`	—
`LINE_SPACING_1_8`	`1`	—
`LINE_SPACING_1_6`	`2`	—
`PRINT_MODE_NORMAL`	`0`	—
`PRINT_MODE_DOUBLE_WIDTH`	`1`	—
`PRINT_MODE_DOUBLE_HEIGHT`	`2`	—
`PRINT_MODE_DOUBLE`	`3`	—
`PRINT_MODE_BOLD`	`8`	—
`PRINT_MODE_REVERSE`	`16`	—
`PRINT_MODE_UNDERLINE`	`128`	—
`CHINESE_DOUBLE_SIZE_NORMAL`	`0`	取消中文倍高倍宽
`CHINESE_DOUBLE_SIZE_ON`	`1`	开启中文倍高倍宽

---

5. 快速索引

- `addBluetoothListener`

- `backPaperToPrintPosition`
- `beep`
- `cancelPrint`
- `cancelPrintCmd`
- `clearBuffer`
- `clearPrinterBuffer`
- `connectBluetoothPrinter`
- `connectNetworkPrinter`
- `connectNetworkPrinterAsync`
- `cutPaper`
- `disableLabelMode`
- `disconnectPrinter`
- `drawBarcodeInPageMode`
- `drawBarcodeInPageModeWithCenteredHri`
- `drawFilledRectangleInPageMode`
- `drawImageFromDataInPageMode`
- `drawImageFromFileInPageMode`
- `drawImageFromPixelsInPageMode`
- `drawLabelBox`
- `drawLabelImageAdvanced`
- `drawLabelImageFromData`
- `drawLabelImageFromFile`
- `drawLabelImageFromPixels`
- `drawLabelLine`
- `drawLabelRect`
- `drawLineInPageMode`
- `drawNvBitmapCenteredInPageMode`
- `drawNvBitmapInPageMode`
- `drawQRCodeInPageMode`
- `drawRectangleBorderInPageMode`
- `drawRectangleInPageMode`
- `drawTextBIG5InPageMode`
- `drawTextCenteredInPageMode`
- `drawTextEUCKRInPageMode`
- `drawTextGBKInPageMode`
- `drawTextInPageMode`
- `drawTextLineLeftRightInPageMode`
- `drawTextRightInPageMode`
- `drawTextShiftJISInPageMode`
- `drawTextUTF8InPageMode`
- `enableLabelMode`
- `estimateCode128WidthDots`
- `estimateTextWidthDots`
- `exitPageMode`
- `feedAndCutPaper`
- `feedLine`
- `feedPaper`
- `fixedHorizontalLine`
- `getBluetoothDevices`
- `getCommunicationManager`

- `getPrinterStatus`
- `hasPaper`
- `horizontalTab`
- `initPrinter`
- `isCoverOpen`
- `isPrinterOnline`
- `labelPageBegin`
- `labelPagePrint`
- `openCashDrawer`
- `pageClearPage`
- `pagePrintPage`
- `pageSetPageArea`
- `pageSetPageDrawDirection`
- `printAndFeedDots`
- `printBarcode`
- `printDoubleQRCode`
- `printHorizontalLine`
- `printImage`
- `printLabelBarcode`
- `printLabelPDF417Code`
- `printLabelQRCode`
- `printLabelText`
- `printLabelTextGBK`
- `printLabelTextUTF8`
- `printNvBitmap`
- `printPDF417`
- `printQRCode`
- `printQRCodeStandard`
- `printRasterImageFromData`
- `printRasterImageFromFile`
- `printRasterImageFromPixels`
- `printTabLine`
- `printText`
- `printTextInBIG5`
- `printTextInEUCKR`
- `printTextInGBK`
- `printTextInShiftJIS`
- `printTextInUTF8`
- `printTextLineLeftRight`
- `printTextLineLeftRightGBK`
- `queryRTStatus`
- `queryTransmitStatus`
- `refreshBluetoothDevices`
- `release`
- `removeBluetoothListener`
- `resetPrinter`
- `reverseFeed`
- `selectPageMode`
- `selectPageModeEx`
- `sendCommand`

- `sendSelfTest`
- `setAlignment`
- `setAsciiTextCharRightSpacing`
- `setAsciiTextFontType`
- `setBarcodeHRIPosition`
- `setBarcodeHeight`
- `setBarcodeReadableTextFontType`
- `setBarcodeReadableTextPosition`
- `setBarcodeWidth`
- `setBold`
- `setCharSpacing`
- `setCharacterCodepage`
- `setCharacterSet`
- `setChineseDoubleSize`
- `setChineseMode`
- `setDoubleByteEncoding`
- `setHorizontalAbsolutePrintPosition`
- `setHorizontalRelativePrintPosition`
- `setKanjiTextCharSpacing`
- `setLabelBarcodeTextPosition`
- `setLineSpacing`
- `setMovementUnits`
- `setPrintAreaDirection`
- `setPrintAreaInPageMode`
- `setPrintAreaLeftMargin`
- `setPrintAreaWidth`
- `setPrintMode`
- `setPrintPosition`
- `setReverse`
- `setSingleByteMode`
- `setTabPositions`
- `setTextLineHeight`
- `setTextRotate`
- `setTextScale`
- `setTextUpsideDown`
- `setUnderline`
- `setVerticalAbsolutePrintPosition`
- `setVerticalRelativePrintPosition`
- `shared`
- `startBluetoothDiscovery`
- `stopBluetoothDiscovery`

完整重载以 `PrinterSDK.swift` 源码为准。