Hướng dẫn toàn diện về thư viện in Xprinter (printer-lib-3.2.0.aar)

Giới thiệu

Tài liệu này cung cấp một cái nhìn toàn diện và chi tiết về cách sử dụng thư viện `printer-lib-3.2.0.aar` của Xprinter. Dựa trên phân tích mã nguồn của thư viện và các tài liệu đi kèm, chúng tôi sẽ trình bày mọi hàm, thuộc tính, chức năng, và luồng sử dụng cần thiết để bạn có thể tích hợp và điều khiển máy in Xprinter một cách hiệu quả trong ứng dụng Android của mình. Tài liệu này được thiết kế để đi sâu vào chi tiết, giải thích rõ ràng mục đích của từng thành phần và cách chúng tương tác với nhau.

I. Cấu trúc tổng quan của thư viện

Thư viện `printer-lib-3.2.0.aar` được tổ chức xung quanh một số lớp và giao diện chính, cung cấp các chức năng từ quản lý kết nối đến gửi lệnh in cụ thể cho từng loại ngôn ngữ máy in (ESC/POS, CPCL, TSPL, ZPL).

Các thành phần cốt lõi bao gồm:

- **`POSConnect`** : Lớp tiện ích tĩnh để khởi tạo thư viện, tạo các đối tượng kết nối, và quản lý các thiết bị USB/Serial.
- **`IDeviceConnection`** : Giao diện định nghĩa các phương thức cơ bản để quản lý kết nối (kết nối, ngắt kết nối) và truyền dữ liệu (gửi/nhận byte).
- **`IConnectListener`** : Giao diện callback để lắng nghe các sự kiện liên quan đến trạng thái kết nối.
- **`IDataCallback`** : Giao diện callback để nhận dữ liệu phản hồi từ máy in.
- **`IPOSListener`** : Một giao diện lắng nghe trạng thái khác, có vẻ là phiên bản cũ hơn hoặc cho mục đích cụ thể.

- **Các lớp `Printer` (ví dụ: `POSPrinter`, `CPCLPrinter`, `TSPLPrinter`, `ZPLPrinter`)**: Các lớp này kế thừa từ một lớp cơ sở (`net.posprinter.a`) và cung cấp các phương thức tiện ích để tạo và gửi các lệnh in cụ thể cho từng ngôn ngữ máy in (ESC/POS, CPCL, TSPL, ZPL).
- **Các lớp `Const` (ví dụ: `POSConst`, `CPCLConst`, `TSPLConst`, `ZPLConst`)**: Chứa các hằng số định nghĩa các giá trị cho các tham số (ví dụ: loại font, chế độ in, mã lỗi).

II. Luồng sử dụng tổng quát (General Usage Flow)

Để sử dụng thư viện, bạn sẽ tuân theo một luồng chung như sau:

1. **Khởi tạo thư viện**: Gọi `POSConnect.init()` một lần duy nhất khi ứng dụng khởi động.
2. **Chọn loại kết nối**: Xác định loại kết nối bạn muốn sử dụng (Bluetooth, USB, Ethernet, Serial).
3. **Tạo đối tượng kết nối**: Sử dụng `POSConnect.createDevice()` để lấy một đối tượng `IDeviceConnection` cho loại kết nối đã chọn.
4. **Thiết lập `IConnectListener`**: Tạo một đối tượng `IConnectListener` để xử lý các sự kiện kết nối.
5. **Kết nối với máy in**: Gọi `connect()` trên đối tượng `IDeviceConnection`, truyền vào thông tin thiết bị và `IConnectListener`.
6. **Chọn lớp `Printer`**: Sau khi kết nối thành công, tạo một đối tượng từ lớp `Printer` tương ứng với ngôn ngữ máy in của bạn (ví dụ: `POSPrinter`, `CPCLPrinter`).
7. **Gửi lệnh in**: Sử dụng các phương thức của lớp `Printer` để tạo và gửi các lệnh in (ví dụ: `printString`, `printBitmap`, `cutPaper`).
8. **Đọc phản hồi (tùy chọn)**: Nếu cần, sử dụng `readData()` trên `IDeviceConnection` với một `IDataCallback` để nhận dữ liệu từ máy in.
9. **Ngắt kết nối**: Gọi `close()` trên đối tượng `IDeviceConnection` khi hoàn tất.
10. **Thoát dịch vụ (tùy chọn)**: Gọi `POSConnect.exit()` khi ứng dụng sắp đóng hoàn toàn.

III. Chi tiết API và luồng sử dụng theo từng thành phần

1. Lớp `POSConnect`

Đây là lớp tĩnh, đóng vai trò là cổng vào chính của thư viện. Nó quản lý các thiết lập toàn cục và cung cấp các phương thức để tạo các đối tượng kết nối.

Hàm/Thuộc tính	Chức năng	Tham số	Trả v
`static void init(Context appContext)`	**Khởi tạo SDK**. Phải được gọi **đầu tiên** và **duy nhất một lần** khi ứng dụng khởi động. Thiết lập `appCtx` và `backgroundThreadExecutor`.	`appContext`: Context của ứng dụng (ví dụ: `getApplicationContext()`).	`void`
`static IDeviceConnection createDevice(int deviceType)`	Tạo một đối tượng `IDeviceConnection` cho loại thiết bị được chỉ định. Đây là bước chuẩn bị để kết nối.	`deviceType`: Hằng số từ `POSConnect` (`DEVICE_TYPE_USB`, `DEVICE_TYPE_BLUETOOTH`, `DEVICE_TYPE_ETHERNET`, `DEVICE_TYPE_SERIAL`).	`IDev`
`static IDeviceConnection connectMac(String mac, IConnectListener listener)`	Kết nối với máy in mạng thông qua địa chỉ MAC. Hàm này sẽ tự động tìm kiếm thiết bị và kết nối.	`mac`: Địa chỉ MAC của máy in. `listener`: `IConnectListener` để nhận trạng thái kết nối.	`IDev`
`static void exit()`	Giải phóng tất cả tài nguyên toàn cục của SDK.	Không có	`void`
`static List<String> getUsbDevices(Context context)`	Lấy danh sách các đường dẫn của các thiết bị USB máy in đang được kết nối với thiết bị Android.	`context`: Context của ứng dụng hoặc Activity.	`List<` sách dẫn `/dev`
`static List<UsbDevice> getUsbDevice(Context`	Tương tự `getUsbDevices`, nhưng trả về danh sách	`context`: Context của ứng dụng hoặc Activity.	`List<` Danh

context)	các đối tượng UsbDevice.		tượn
static List<String> getSerialPort()	Lấy danh sách các cổng nối tiếp (serial port) có sẵn trên thiết bị Android.	Không có	List< sách tiếp (/dev

Luồng sử dụng POSConnect:

1. **Khởi động ứng dụng**: POSConnect.init(this) (trong Application.onCreate()).
2. **Người dùng chọn loại kết nối**: Ví dụ, người dùng muốn in qua Bluetooth.
3. **Tạo đối tượng kết nối**: val btConnection = POSConnect.createDevice(POSConnect.DEVICE_TYPE_BLUETOOTH).
4. **Tiến hành kết nối**: Sử dụng btConnection để gọi connect() (xem phần IDeviceConnection).
5. **Kết thúc ứng dụng**: POSConnect.exit() (tùy chọn, khi ứng dụng bị hủy).

2. Giao diện IDeviceConnection

Đây là giao diện trung tâm để tương tác với một kết nối máy in cụ thể. Bạn sẽ nhận được một đối tượng triển khai giao diện này từ POSConnect.createDevice() hoặc POSConnect.connectMac().

Hàm/Thuộc tính	Chức năng	Tham số	Trả về
void connect(String info, IConnectListener iConnectListener)	**(Quan trọng)** Bắt đầu quá trình kết nối bất đồng bộ tới máy in. Kết quả được trả về qua iConnectListener.	info: Thông tin kết nối (MAC, IP, USB path, Serial path+baudrate). iConnectListener: Đối tượng lắng nghe trạng thái kết nối.	void
boolean connectSync(String info, IConnectListener	Bắt đầu quá trình kết nối	Tương tự connect().	boolean: true nếu

`iConnectListener)`	đồng bộ. Hàm này sẽ chặn luồng hiện tại cho đến khi kết nối thành công/thất bại.		kết nối thành công, `false` nếu thất bại.
`void close()`	**(Quan trọng)** Đóng kết nối hiện tại với máy in và giải phóng tài nguyên.	Không có	`void`
`void closeSync()`	Đóng kết nối đồng bộ.	Không có	`void`
`void sendData(byte[] bArr)`	Gửi một mảng byte dữ liệu đến máy in.	`bArr` : Mảng byte chứa dữ liệu/lệnh in.	`void`
`void sendData(List<byte[]> list)`	Gửi một danh sách các mảng byte dữ liệu đến máy in.	`list` : Danh sách các mảng byte.	`void`
`int sendSync(byte[] bArr)`	Gửi dữ liệu đồng bộ.	`bArr` : Mảng byte chứa dữ liệu/lệnh in.	`int` : Số byte đã gửi, hoặc mã lỗi.
`void readData(int i, IDataCallback iDataCallback)`	Đọc dữ liệu từ máy in với một timeout.	`i` : Thời gian chờ (timeout) bằng mili giây. `iDataCallback` : Đối tượng callback để nhận dữ liệu.	`void`
`void readData(IDataCallback iDataCallback)`	Đọc dữ liệu từ máy in (không có timeout).	`iDataCallback` : Đối tượng	`void`

		callback để nhận dữ liệu.	
`byte[] readSync(int i)`	Đọc dữ liệu đồng bộ với timeout.	`i` : Thời gian chờ (timeout) bằng mili giây.	`byte[]` : Dữ liệu nhận được, hoặc `null` nếu timeout/lỗi.
`void startReadLoop(IDataCallback iDataCallback)`	Bắt đầu một vòng lặp liên tục để đọc dữ liệu từ máy in.	`iDataCallback` : Đối tượng callback để nhận dữ liệu.	`void`
`String getConnectInfo()`	Lấy thông tin kết nối hiện tại (ví dụ: địa chỉ MAC, IP, USB path).	Không có	`String`
`void setConnectInfo(String str)`	Đặt thông tin kết nối.	`str` : Chuỗi thông tin kết nối.	`void`
`int getConnectType()`	Lấy loại kết nối hiện tại.	Không có	`int` : Hằng số loại thiết bị từ `POSConnect` .
`boolean isConnect()`	**(Deprecated)** Kiểm tra xem có đang kết nối hay không.	Không có	`boolean`
`void setSendCallback(IStatusCallback iStatusCallback)`	Đặt callback để nhận trạng thái gửi dữ liệu.	`iStatusCallback` : Callback trạng thái.	`void`
`void isConnect(byte[] bArr, IStatusCallback iStatusCallback)`	Kiểm tra kết nối và gửi trạng thái qua callback.	`bArr` : Dữ liệu kiểm tra. `iStatusCallback` :	`void`

		Callback trạng thái.	

Luồng sử dụng `IDeviceConnection`:

1. **Tạo `IDeviceConnection`**: `val connection = POSConnect.createDevice(...)`.
2. **Tạo `IConnectListener`**: `val myListener = object : IConnectListener { ... }`.
3. **Kết nối**: `connection.connect(deviceInfo, myListener)`.
4. **Chờ `CONNECT_SUCCESS`**: Trong `myListener.onStatus()`, khi nhận được `CONNECT_SUCCESS`.
5. **Gửi dữ liệu**: `connection.sendData(byteArrayOf(...))`.
6. **Đọc dữ liệu (tùy chọn)**: `connection.readData(myReadCallback)`.
7. **Ngắt kết nối**: `connection.close()`.

3. Giao diện `IConnectListener`

Giao diện này được sử dụng để nhận các sự kiện về trạng thái kết nối. Bạn cần triển khai phương thức `onStatus()`.

Hàm/Thuộc tính	Chức năng	Tham số	Trả về	Luồng sử dụng	Ghi chú

`void onStatus(int code, String connectInfo, String message)`	Được gọi khi trạng thái kết nối thay đổi.	`code` : Mã trạng thái (xem bảng dưới). `connectInfo` : Thông tin kết nối. `message` : Mô tả trạng thái.	`void`	Triển khai giao diện này và truyền vào `IDeviceConnection.connect()` .	**Quan trọng** để theo dõi và phản ứng với trạng thái kết nối.

Các mã trạng thái trong `IConnectListener.onStatus()` :

Mã trạng thái (`POSConnect` hằng số)	Mô tả
`CONNECT_SUCCESS` (1)	Kết nối thành công.
`CONNECT_FAIL` (2)	Kết nối thất bại.
`SEND_FAIL` (3)	Gửi dữ liệu thất bại.
`CONNECT_INTERRUPT` (4)	Kết nối bị ngắt đột ngột.
`USB_ATTACHED` (5)	Thiết bị USB được gắn vào.
`USB_DETACHED` (6)	Thiết bị USB bị ngắt kết nối.
`BLUETOOTH_INTERRUPT` (7)	Kết nối Bluetooth bị ngắt.

Luồng sử dụng `IConnectListener` :

1. **Định nghĩa `IConnectListener`** : Tạo một đối tượng ẩn danh hoặc một lớp triển khai `IConnectListener` .
2. **Truyền vào `connect()`** : `connection.connect(deviceInfo, myConnectListener)` .
3. **Xử lý trạng thái**: Trong phương thức `onStatus()` , kiểm tra `code` để thực hiện các hành động tương ứng (ví dụ: hiển thị thông báo, kích hoạt nút in).

4. Giao diện `IDataCallback`

Giao diện này được sử dụng để nhận dữ liệu phản hồi từ máy in khi bạn gọi các hàm `readData()`.

Hàm/Thuộc tính	Chức năng	Tham số	Trả về	Luồng sử dụng	Ghi chú
`void receive(byte[] data)`	Được gọi khi dữ liệu được đọc từ máy in.	`data`: Mảng byte chứa dữ liệu nhận được.	`void`	Triển khai giao diện này và truyền vào `IDeviceConnection.readData()` hoặc `startReadLoop()`.	

Luồng sử dụng `IDataCallback`:

1. **Định nghĩa `IDataCallback`**: Tạo một đối tượng ẩn danh hoặc một lớp triển khai `IDataCallback`.
2. **Truyền vào `readData()`**: `connection.readData(timeout, myDataCallback)`.
3. **Xử lý dữ liệu**: Trong phương thức `receive()`, xử lý mảng byte nhận được (ví dụ: chuyển đổi thành chuỗi, phân tích trạng thái máy in).

5. Các lớp `Printer` (POS, CPCL, TSPL, ZPL)

Các lớp này cung cấp các phương thức tiện ích để tạo và gửi các lệnh in cụ thể cho từng ngôn ngữ máy in. Chúng kế thừa từ một lớp cơ sở (`net.posprinter.a`) có phương thức `sendData(byte[] data)` và `sendData(List<byte[]> datas)`.

Lưu ý quan trọng: Các hàm trong các lớp `Printer` này thường trả về chính đối tượng `Printer` đó (`return this;`), cho phép bạn gọi chuỗi các hàm (method chaining) để xây dựng lệnh in.

5.1. `POSPrinter` (ESC/POS)

Lớp này dành cho các máy in hỗ trợ lệnh ESC/POS, thường là máy in hóa đơn (receipt printer).

Hàm/Thuộc tính	Chức năng	Tham số	Trả về	L s d
`POSPrinter(IDeviceConnection connection)`	Constructor.	`connection` : Đối tượng `IDeviceConnection` đã kết nối.	`POSPrinter`	T đ t s k k n t c
`POSPrinter printString(String data)`	In một chuỗi văn bản.	`data` : Chuỗi cần in.	`POSPrinter`	G s k k n
`POSPrinter printTextSize(String data, int textSize)`	In văn bản với kích thước font cụ thể.	`data` : Chuỗi. `textSize` : Kích thước font (0-7).	`POSPrinter`	
`POSPrinter printTextAttribute(String data, int attribute)`	In văn bản với thuộc tính (đậm, gạch chân, v.v.).	`data` : Chuỗi. `attribute` : Hằng số thuộc tính từ `POSConst` .	`POSPrinter`	
`POSPrinter printTextAlignment(String data, int alignment)`	In văn bản với căn chỉnh (trái, giữa, phải).	`data` : Chuỗi. `alignment` : Hằng số căn chỉnh từ `POSConst` .	`POSPrinter`	

`POSPrinter printText(String data, int alignment, int attribute, int textSize)`	In văn bản với đầy đủ tùy chọn.	`data` : Chuỗi. `alignment` : Căn chỉnh. `attribute` : Thuộc tính. `textSize` : Kích thước.	`POSPrinter`	
`POSPrinter printBitmap(String bitmapPath, int alignment, int width, int model)`	In hình ảnh từ đường dẫn file.	`bitmapPath` : Đường dẫn file ảnh. `alignment` : Căn chỉnh. `width` : Chiều rộng. `model` : Chế độ in ảnh.	`POSPrinter`	
`POSPrinter printBitmap(Bitmap bmp, int alignment, int width, int model)`	In hình ảnh từ đối tượng `Bitmap` .	`bmp` : Đối tượng `Bitmap` . `alignment` : Căn chỉnh. `width` : Chiều rộng. `model` : Chế độ in ảnh.	`POSPrinter`	
`POSPrinter printBarCode(String data, int codeType, int width, int height, int alignment, int textPosition)`	In mã vạch 1D.	`data` : Dữ liệu mã vạch. `codeType` : Loại mã vạch. `width` : Độ rộng. `height` : Chiều cao. `alignment` : Căn chỉnh. `textPosition` : Vị trí văn bản.	`POSPrinter`	
`POSPrinter printQRCode(String data, int moduleSize, int ecLevel, int alignment)`	In mã QR.	`data` : Dữ liệu QR. `moduleSize` : Kích thước module. `ecLevel` : Mức độ sửa lỗi. `alignment` : Căn chỉnh.	`POSPrinter`	

`POSPrinter feedLine(int lineCount)`	Cuộn giấy lên `lineCount` dòng.	`lineCount` : Số dòng.	`POSPrinter`	
`POSPrinter feedDot(int dotCount)`	Cuộn giấy lên `dotCount` chấm.	`dotCount` : Số chấm.	`POSPrinter`	
`POSPrinter cutPaper(int model)`	Cắt giấy.	`model` : Chế độ cắt (0: full cut, 1: partial cut).	`POSPrinter`	
`POSPrinter openCashBox(int pinNum, int onTime, int offTime)`	Mở két tiền.	`pinNum` : Chân kích hoạt. `onTime` : Thời gian mở. `offTime` : Thời gian đóng.	`POSPrinter`	
`void printerCheck(int type, int timeout, IDataCallback callback)`	Kiểm tra trạng thái máy in.	`type` : Loại kiểm tra. `timeout` : Thời gian chờ. `callback` : `IDataCallback` để nhận trạng thái.	`void`	
`void printerStatus(IStatusCallback callback)`	Lấy trạng thái máy in.	`callback` : `IStatusCallback` để nhận trạng thái.	`void`	
`static void searchNetDevice(UdpCallback callback)`	Tìm kiếm máy in mạng qua UDP.	`callback` : `UdpCallback` để nhận thông tin thiết bị tìm thấy.	`void`	

`static void udpNetConfig(byte[] macAddress, byte[] ipAddress, byte[] mask, byte[] gateway, boolean dhcp)`	Cấu hình mạng cho máy in qua UDP.	Các thông số cấu hình mạng.	`void`	
`POSPrinter initializePrinter()`	Khởi tạo máy in (reset về trạng thái mặc định).	Không có	`POSPrinter`	
`POSPrinter setLineSpacing(int space)`	Đặt khoảng cách dòng.	`space` : Khoảng cách.	`POSPrinter`	
`POSPrinter setTurnUpsideDownMode(boolean on)`	Bật/tắt chế độ in ngược.	`on` : `true` để bật, `false` để tắt.	`POSPrinter`	
`POSPrinter selectCodePage(int page)`	Chọn bảng mã ký tự.	`page` : Mã bảng mã.	`POSPrinter`	
`POSPrinter selectCharacterFont(int font)`	Chọn font ký tự.	`font` : Mã font.	`POSPrinter`	
`POSPrinter setCharRightSpace(byte space)`	Đặt khoảng cách ký tự bên phải.	`space` : Khoảng cách.	`POSPrinter`	
`POSPrinter setTextStyle(int attribute, int textSize)`	Đặt kiểu văn bản (đậm, gạch chân, kích thước).	`attribute` : Thuộc tính. `textSize` : Kích thước.	`POSPrinter`	
`POSPrinter setAlignment(int alignment)`	Đặt căn chỉnh mặc định.	`alignment` : Căn chỉnh.	`POSPrinter`	
`POSPrinter downloadNVImage(List<Bitmap> bitmaps, int imageWidth)`	Tải hình ảnh vào bộ nhớ NV của máy in.	`bitmaps` : Danh sách `Bitmap` . `imageWidth` : Chiều rộng ảnh.	`POSPrinter`	

`POSPrinter clearNVImage()`	Xóa tất cả hình ảnh trong bộ nhớ NV.	Không có	`POSPrinter`	
`POSPrinter printNVImage(int index, int model)`	In hình ảnh từ bộ nhớ NV.	`index` : Chỉ số ảnh. `model` : Chế độ in.	`POSPrinter`	

Luồng sử dụng `POSPrinter` (Ví dụ: In hóa đơn đơn giản):

1. **Kết nối**: Đảm bảo `IDeviceConnection` đã kết nối thành công.
2. **Tạo `POSPrinter`** : `val printer = POSPrinter(connectedDevice)` .
3. **In văn bản**: `printer.printString("Cửa hàng ABC\n").printTextSize("Hóa đơn bán hàng\n", POSConst.FNT_SIZE_BIG).feedLine(2)` .
4. **In hình ảnh (logo)**: `printer.printBitmap(logoBitmap, POSConst.ALIGN_CENTER, 300)` .
5. **In mã vạch/QR**: `printer.printQRCode("https://example.com/invoice123", 8, POSConst.ALIGN_CENTER)` .
6. **Cắt giấy**: `printer.cutPaper()` .
7. **Đóng kết nối**: `connectedDevice.close()` .

5.2. `CPCLPrinter`

Lớp này dành cho các máy in hỗ trợ lệnh CPCL, thường là máy in nhãn (label printer).

Hàm/Thuộc tính	Chức năng	Tham số	Trả về
`CPCLPrinter(IDeviceConnection connection)`	Constructor.	`connection` : Đối tượng `IDeviceConnection` đã kết nối.	`CPCLPrinte`

`CPCLPrinter initializePrinter(int offset, int height, int qty)`	Khởi tạo máy in CPCL với các thông số trang.	`offset` : Offset. `height` : Chiều cao nhãn. `qty` : Số lượng bản sao.	`CPCLPrinte`
`void addPrint()`	Thêm lệnh in (thực hiện in nhãn đã định nghĩa).	Không có	`void`
`CPCLPrinter addForm()`	Thêm lệnh FORM.	Không có	`CPCLPrinte`
`CPCLPrinter addText(int x, int y, String font, String content)`	Thêm văn bản vào nhãn.	`x` , `y` : Tọa độ. `font` : Tên font (ví dụ: `CPCLConst.FNT_0`). `content` : Nội dung văn bản.	`CPCLPrinte`
`CPCLPrinter setMag(int w, int h)`	Đặt độ phóng đại cho văn bản/barcode.	`w` , `h` : Độ phóng đại chiều rộng/cao (1-16).	`CPCLPrinte`
`CPCLPrinter addBarcode(int x, int y, String type, int width, int ratio, int height, String data)`	Thêm mã vạch 1D.	`x` , `y` : Tọa độ. `type` : Loại mã vạch. `width` , `ratio` , `height` : Kích thước. `data` : Dữ liệu.	`CPCLPrinte`

`CPCLPrinter addQRCode(int x, int y, int codeModel, int cellWidth, String data)`	Thêm mã QR.	`x` , `y` : Tọa độ. `codeModel` : Chế độ mã QR. `cellWidth` : Độ rộng ô. `data` : Dữ liệu.	`CPCLPrinte`
`CPCLPrinter addBox(int x, int y, int width, int height, int thickness)`	Thêm hình hộp.	`x` , `y` : Tọa độ góc trên trái. `width` , `height` : Kích thước. `thickness` : Độ dày đường viền.	`CPCLPrinte`
`CPCLPrinter addLine(int x, int y, int xend, int yend, int thickness)`	Thêm đường thẳng.	`x` , `y` , `xend` , `yend` : Tọa độ điểm đầu/cuối. `thickness` : Độ dày.	`CPCLPrinte`
`CPCLPrinter addEGraphics(int x, int y, int width, Bitmap bmp, AlgorithmType algorithmType)`	Thêm hình ảnh (Expanded Graphics).	`x` , `y` : Tọa độ. `width` : Chiều rộng. `bmp` : `Bitmap` . `algorithmType` : Thuật toán xử lý ảnh.	`CPCLPrinte`
`CPCLPrinter addCGraphics(int x, int y, int width, Bitmap bmp, AlgorithmType algorithmType)`	Thêm hình ảnh (Compressed Graphics).	Tương tự `addEGraphics` .	`CPCLPrinte`
`CPCLPrinter addAlign(String align, int end)`	Đặt căn chỉnh cho các đối tượng tiếp theo.	`align` : Căn chỉnh (ví dụ: `CPCLConst.ALIGN_CENTER`). `end` : Vị trí kết thúc căn chỉnh.	`CPCLPrinte`
`CPCLPrinter addSpeed(int level)`	Đặt tốc độ in.	`level` : Mức tốc độ (0-5).	`CPCLPrinte`
`CPCLPrinter addBeep(int length)`	Kích hoạt tiếng bíp.	`length` : Thời gian bíp.	`CPCLPrinte`
`void printerStatus(IStatusCallback callback)`	Lấy trạng thái máy in.	`callback` : `IStatusCallback` để nhận trạng thái.	`void`
`void getBtMac(int timeout, IStrCallback callback)`	Lấy địa chỉ MAC	`timeout` : Thời gian chờ. `callback` : `IStrCallback` để	`void`

	Bluetooth của máy in.	nhận chuỗi MAC.	

Luồng sử dụng `CPCLPrinter` (Ví dụ: In nhãn sản phẩm):

1. **Kết nối**: Đảm bảo `IDeviceConnection` đã kết nối thành công.
2. **Tạo `CPCLPrinter`**: `val printer = CPCLPrinter(connectedDevice)`.
3. **Khởi tạo nhãn**: `printer.initializePrinter(200, 400, 1)` (offset 200, cao 400, 1 bản).
4. **Thêm FORM**: `printer.addForm()`.
5. **Thêm văn bản**: `printer.addText(10, 10, CPCLConst.FNT_0, "Sản phẩm ABC").addText(10, 50, CPCLConst.FNT_1, "Giá: 123.000 VND")`.
6. **Thêm mã QR**: `printer.addQRCode(10, 100, 2, 6, "https://example.com/product/abc")`.
7. **Thêm hình ảnh**: `printer.addEGraphics(200, 10, 100, productLogoBitmap)`.
8. **Thực hiện in**: `printer.addPrint()`.
9. **Đóng kết nối**: `connectedDevice.close()`.

5.3. `TSPLPrinter`

Lớp này dành cho các máy in hỗ trợ lệnh TSPL, cũng thường là máy in nhãn.

Hàm/Thuộc tính	Chức năng	Tham số	Trả về	Luồ sử dụ
`TSPLPrinter(IDeviceConnection connection)`	Constructor.	`connection`: Đối tượng `IDeviceConnection` đã kết nối.	`TSPLPrinter`	Tạo đối tượ sau khi kết nối thà côn

`TSPLPrinter sizeInch(double width, double height)`	Đặt kích thước nhãn theo inch.	`width` , `height` : Kích thước nhãn.	`TSPLPrinter`	Gọi đầu tiên khi bắt đầu nhã
`TSPLPrinter sizeMm(double width, double height)`	Đặt kích thước nhãn theo mm.	`width` , `height` : Kích thước nhãn.	`TSPLPrinter`	
`TSPLPrinter gapInch(double m, double n)`	Đặt khoảng cách giữa các nhãn theo inch.	`m` , `n` : Khoảng cách.	`TSPLPrinter`	
`TSPLPrinter speed(double speed)`	Đặt tốc độ in.	`speed` : Tốc độ.	`TSPLPrinter`	
`TSPLPrinter density(int density)`	Đặt mật độ in (độ đậm).	`density` : Mật độ.	`TSPLPrinter`	
`TSPLPrinter direction(int direction, boolean isMirror)`	Đặt hướng in và chế độ gương.	`direction` : Hướng. `isMirror` : `true` nếu in gương.	`TSPLPrinter`	
`TSPLPrinter cls()`	Xóa bộ đệm hình ảnh nhãn.	Không có	`TSPLPrinter`	Gọi trư khi bắt đầu định ngh nội dur nhã mớ

`TSPLPrinter box(int x, int y, int width, int height, int thickness)`	Vẽ hình hộp.	`x` , `y` : Tọa độ. `width` , `height` : Kích thước. `thickness` : Độ dày.	`TSPLPrinter`	
`TSPLPrinter bar(int x, int y, int width, int height)`	Vẽ thanh (bar).	`x` , `y` : Tọa độ. `width` , `height` : Kích thước.	`TSPLPrinter`	
`TSPLPrinter barcode(int x, int y, String codeType, int height, int readable, int rotation, int narrow, int wide, String content)`	Thêm mã vạch 1D.	Các thông số mã vạch.	`TSPLPrinter`	
`TSPLPrinter text(int x, int y, String font, int rotation, int xRatio, int yRatio, String content)`	Thêm văn bản.	`x` , `y` : Tọa độ. `font` : Tên font. `rotation` : Xoay. `xRatio` , `yRatio` : Tỷ lệ phóng đại. `content` : Nội dung.	`TSPLPrinter`	
`TSPLPrinter qrcode(int x, int y, String ecLevel, int cellWidth, String mode, int rotation, String data)`	Thêm mã QR.	Các thông số mã QR.	`TSPLPrinter`	
`void print(int count)`	Thực hiện in nhãn đã định nghĩa.	`count` : Số lượng bản sao.	`void`	Gọi sau khi định ngh xon nội dur nhâ

`TSPLPrinter feed(int length)`	Cuộn giấy lên một khoảng.	`length` : Khoảng cách.	`TSPLPrinter`	
`TSPLPrinter backFeed(int length)`	Cuộn giấy lùi một khoảng.	`length` : Khoảng cách.	`TSPLPrinter`	
`TSPLPrinter formFeed()`	Cuộn giấy đến đầu nhãn tiếp theo.	Không có	`TSPLPrinter`	
`TSPLPrinter home()`	Di chuyển đầu in về vị trí home.	Không có	`TSPLPrinter`	
`TSPLPrinter codePage(String page)`	Đặt bảng mã ký tự.	`page` : Tên bảng mã.	`TSPLPrinter`	
`TSPLPrinter sound(int level, int interval)`	Kích hoạt tiếng bíp.	`level` : Mức độ. `interval` : Khoảng thời gian.	`TSPLPrinter`	
`TSPLPrinter bitmap(int x, int y, int mode, int width, Bitmap bmp, AlgorithmType algorithmType)`	Thêm hình ảnh bitmap.	`x` , `y` : Tọa độ. `mode` : Chế độ. `width` : Chiều rộng. `bmp` : `Bitmap` . `algorithmType` : Thuật toán xử lý ảnh.	`TSPLPrinter`	
`TSPLPrinter cut()`	Cắt giấy.	Không có	`TSPLPrinter`	
`TSPLPrinter setPeel(boolean isOpen)`	Bật/tắt chế độ bóc nhãn.	`isOpen` : `true` để bật, `false` để tắt.	`TSPLPrinter`	
`TSPLPrinter setTear(boolean isOpen)`	Bật/tắt chế độ xé nhãn.	`isOpen` : `true` để bật, `false` để tắt.	`TSPLPrinter`	

`void printerStatus(IStatusCallback callback)`	Lấy trạng thái máy in.	`callback` : `IStatusCallback` để nhận trạng thái.	`void`	

Luồng sử dụng `TSPLPrinter` (Ví dụ: In nhãn vận chuyển):

1. **Kết nối**: Đảm bảo `IDeviceConnection` đã kết nối thành công.
2. **Tạo `TSPLPrinter`** : `val printer = TSPLPrinter(connectedDevice)` .
3. **Đặt kích thước nhãn**: `printer.sizeMm(100.0, 150.0)` (100mm x 150mm).
4. **Xóa bộ đệm**: `printer.cls()` .
5. **Thêm văn bản**: `printer.text(10, 10, "2", 1, 1, "Người gửi: Công ty XYZ")` .
6. **Thêm mã vạch**: `printer.barcode(10, 50, "128", 80, 1, 0, "1234567890")` .
7. **Thêm mã QR**: `printer.qrcode(10, 150, TSPLConst.EC_LEVEL_L, 4, TSPLConst.QRCODE_MODE_AUTO, 0, "Đơn hàng #ABCDEF")` .
8. **Thêm hình ảnh**: `printer.bitmap(200, 10, 0, 100, companyLogoBitmap)` .
9. **Thực hiện in**: `printer.print(1)` (in 1 bản).
10. **Cắt giấy**: `printer.cut()` .
11. **Đóng kết nối**: `connectedDevice.close()` .

5.4. `ZPLPrinter`

Lớp này dành cho các máy in hỗ trợ lệnh ZPL, thường là máy in nhãn cao cấp của Zebra.

Hàm/Thuộc tính	Chức năng	Tham số	Trả về	L s c
`ZPLPrinter(IDeviceConnection connection)`	Constructor.	`connection` : Đối tượng `IDeviceConnection` đã kết nối.	`ZPLPrinter`	T c t

`ZPLPrinter addStart()`	Bắt đầu một định dạng nhãn ZPL (`^XA`).	Không có	`ZPLPrinter`	
`ZPLPrinter addEnd()`	Kết thúc một định dạng nhãn ZPL (`^XZ`).	Không có	`ZPLPrinter`	
`ZPLPrinter addText(int x, int y, char fontName, String rotation, int sizeW, int sizeH, String content)`	Thêm văn bản.	`x` , `y` : Tọa độ. `fontName` : Tên font. `rotation` : Xoay. `sizeW` , `sizeH` : Kích thước. `content` : Nội dung.	`ZPLPrinter`	
`ZPLPrinter addTextBlock(int x, int y, char fontName, String rotation, int sizeW, int sizeH, int textblockWidth, int`	Thêm khối văn bản (text block).	Các thông số khối văn bản.	`ZPLPrinter`	

textblockHeight, String content)				
ZPLPrinter setCustomFont(String font, char alias, int codePage)	Đặt font tùy chỉnh.	font : Tên font. alias : Alias. codePage : Bảng mã.	ZPLPrinter	
ZPLPrinter setPrinterWidth(int width)	Đặt chiều rộng máy in.	width : Chiều rộng.	ZPLPrinter	
ZPLPrinter addReverse(int x, int y, int width, int height, int radius)	Vẽ hình chữ nhật đảo ngược màu.	x , y : Tọa độ. width , height : Kích thước. radius : Bán kính bo góc.	ZPLPrinter	
ZPLPrinter addBox(int x, int y, int width, int height, int thickness, int radius)	Vẽ hình hộp.	x , y : Tọa độ. width , height : Kích thước. thickness : Độ dày. radius : Bán kính bo góc.	ZPLPrinter	
ZPLPrinter addBarcode(int x, int y, String codeType, String ratio, byte textPosition, String data, int width, int height)	Thêm mã vạch 1D.	Các thông số mã vạch.	ZPLPrinter	
ZPLPrinter addQRCode(int x, int y, int size, String data)	Thêm mã QR.	x , y : Tọa độ. size : Kích thước. data : Dữ liệu.	ZPLPrinter	
ZPLPrinter downloadBitmap(int width, String bmpName, Bitmap bmp, AlgorithmType algorithmType)	Tải hình ảnh bitmap vào bộ nhớ máy in.	width : Chiều rộng. bmpName : Tên ảnh. bmp : Bitmap . algorithmType : Thuật toán xử lý ảnh.	ZPLPrinter	T ả t k
ZPLPrinter addBitmap(int x, int y, String bmpName, int mx, int my)	Thêm hình ảnh đã tải vào nhãn.	x , y : Tọa độ. bmpName : Tên ảnh. mx , my : Tỷ lệ phóng đại.	ZPLPrinter	

`ZPLPrinter printBitmap(int x, int y, Bitmap bmp, int width, AlgorithmType algorithmType)`	In hình ảnh bitmap trực tiếp.	`x`, `y`: Tọa độ. `bmp`: `Bitmap`. `width`: Chiều rộng. `algorithmType`: Thuật toán xử lý ảnh.	`ZPLPrinter`
`ZPLPrinter printBmpCompress(int x, int y, Bitmap bmp, int width, AlgorithmType algorithmType)`	In hình ảnh bitmap nén.	Tương tự `printBitmap`.	`ZPLPrinter`
`ZPLPrinter addPrintCount(int count)`	Đặt số lượng bản in.	`count`: Số lượng.	`ZPLPrinter`
`ZPLPrinter addGraphicDiagonalLine(int x, int y, char orientation, int width, int height, int thickness)`	Vẽ đường chéo.	Các thông số đường chéo.	`ZPLPrinter`
`ZPLPrinter addGraphicEllipse(int x, int y, int width, int height, int thickness)`	Vẽ hình elip.	Các thông số elip.	`ZPLPrinter`
`ZPLPrinter addGraphicCircle(int x, int y, int diameter, int thickness)`	Vẽ hình tròn.	`x`, `y`: Tọa độ tâm. `diameter`: Đường kính. `thickness`: Độ dày.	`ZPLPrinter`
`ZPLPrinter setPrintSpeed(int speed)`	Đặt tốc độ in.	`speed`: Tốc độ.	`ZPLPrinter`
`ZPLPrinter setLabelLength(int length)`	Đặt chiều dài nhãn.	`length`: Chiều dài.	`ZPLPrinter`
`ZPLPrinter setPrintOrientation(String orientation)`	Đặt hướng in.	`orientation`: Hướng (ví dụ: `ZPLConst.ROTATION_0`).	`ZPLPrinter`

`ZPLPrinter setPrintDensity(int density)`	Đặt mật độ in.	`density` : Mật độ (0-30).	`ZPLPrinter`	
`ZPLPrinter printPdf(int x, int y, String path, int pageWidth, AlgorithmType algorithmType)`	In file PDF.	`x` , `y` : Tọa độ. `path` : Đường dẫn file PDF. `pageWidth` : Chiều rộng trang. `algorithmType` : Thuật toán xử lý ảnh.	`ZPLPrinter`	
`void printerStatus(IStatusCallback callback)`	Lấy trạng thái máy in.	`callback` : `IStatusCallback` để nhận trạng thái.	`void`	

Luồng sử dụng `ZPLPrinter` (Ví dụ: In nhãn kho):

1. **Kết nối**: Đảm bảo `IDeviceConnection` đã kết nối thành công.
2. **Tạo `ZPLPrinter`** : `val printer = ZPLPrinter(connectedDevice)` .
3. **Bắt đầu định dạng**: `printer.addStart()` .
4. **Đặt chiều rộng máy in**: `printer.setPrinterWidth(600)` .
5. **Thêm văn bản**: `printer.addText(50, 50, 'E', 20, 20, "Mã sản phẩm: XYZ123")` .
6. **Thêm mã QR**: `printer.addQRCode(50, 100, 5, "http://warehouse.com/item/xyz123")` .
7. **Vẽ hộp**: `printer.addBox(10, 10, 580, 200, 2, 8)` .
8. **Kết thúc định dạng**: `printer.addEnd()` .
9. **Thực hiện in**: (Không có hàm `print()` riêng trong ZPLPrinter, bạn sẽ gửi chuỗi ZPL đã tạo thông qua `sendData` của `IDeviceConnection` hoặc các hàm `add...` đã tự động gửi dữ liệu).
10. **Đóng kết nối**: `connectedDevice.close()` .

IV. Các lớp hằng số (Const Classes)

Các lớp này chứa các hằng số được sử dụng làm tham số cho các hàm trong các lớp `Printer`.

- **`POSConst`**: Chứa các hằng số cho ESC/POS (ví dụ: `ALIGN_LEFT`, `ALIGN_CENTER`, `ALIGN_RIGHT`, `FNT_SIZE_NORMAL`, `FNT_SIZE_BIG`, `FNT_UNDERLINE`).
- **`CPCLConst`**: Chứa các hằng số cho CPCL (ví dụ: `FNT_0`, `FNT_1`, `ROTATION_0`, `ROTATION_90`, `ALIGN_LEFT`, `ALIGN_CENTER`, `ALIGN_RIGHT`).
- **`TSPLConst`**: Chứa các hằng số cho TSPL (ví dụ: `EC_LEVEL_L`, `QRCODE_MODE_AUTO`).
- **`ZPLConst`**: Chứa các hằng số cho ZPL (ví dụ: `ROTATION_0`, `ROTATION_90`, `BCS_CODE128`, `BCS_QR`).

Bạn nên tham khảo trực tiếp mã nguồn của các lớp này (trong thư mục `xprinter_aar_decompiled/sources/net/posprinter/`) để có danh sách đầy đủ và chính xác nhất các hằng số.

V. Các lớp tiện ích và nội bộ

Trong quá trình decompile, bạn có thể thấy các thư mục và lớp khác như `net.posprinter.a`, `net.posprinter.b`, `net.posprinter.c`, `net.posprinter.d`, `net.posprinter.esc`, `net.posprinter.model`, `net.posprinter.posprinterface`, `net.posprinter.serial`, `net.posprinter.utils`. Các lớp này thường là các lớp nội bộ, lớp tiện ích, hoặc các triển khai cụ thể của giao diện, không cần thiết phải tương tác trực tiếp từ ứng dụng của bạn. Chúng hỗ trợ các chức năng cốt lõi của thư viện (ví dụ: xử lý dữ liệu bitmap, giao tiếp serial, quản lý UDP).

VI. Các cân nhắc quan trọng

- **Quyền Android**: Đảm bảo `AndroidManifest.xml` của bạn khai báo đầy đủ các quyền cần thiết cho Bluetooth, USB, Internet, và vị trí (đối với quét Bluetooth). Đối với Android 6.0+ (API 23+), bạn cần yêu cầu các quyền nguy hiểm (`dangerous permissions`) trong thời gian chạy.

- **Xử lý lỗi**: Luôn kiểm tra mã trạng thái trả về từ `IConnectListener` và xử lý các trường hợp lỗi một cách thích hợp.
- **Luồng (Threading)**: Các thao tác kết nối, gửi/nhận dữ liệu, và xử lý hình ảnh có thể tốn thời gian. Luôn thực hiện chúng trên một luồng nền (background thread) để tránh làm treo ứng dụng (ANR - Application Not Responding).
- **Ngôn ngữ in**: Hiểu rõ máy in của bạn hỗ trợ ngôn ngữ in nào (ESC/POS, CPCL, TSPL, ZPL) để sử dụng đúng lớp `Printer` và các lệnh tương ứng. Việc gửi sai lệnh có thể dẫn đến kết quả in không mong muốn hoặc không in được.
- **Tài liệu gốc**: Mặc dù tài liệu này tổng hợp thông tin, các tài liệu PDF gốc (`Android CPCL Program Manual.pdf` , `Android POS Program Manual.pdf` , v.v.) vẫn là nguồn tham khảo chính xác nhất cho các lệnh in cụ thể và chi tiết kỹ thuật của từng ngôn ngữ máy in.

Kết luận

Thư viện `printer-lib-3.2.0.aar` cung cấp một bộ API mạnh mẽ để điều khiển máy in Xprinter. Bằng cách hiểu rõ cấu trúc lớp, chức năng của từng hàm, và tuân thủ luồng sử dụng được khuyến nghị, bạn có thể dễ dàng tích hợp khả năng in vào ứng dụng Android của mình. Việc phân tích mã nguồn đã giúp chúng ta có cái nhìn sâu sắc hơn về các khả năng của thư viện, đặc biệt là các lớp `Printer` chuyên biệt cho từng ngôn ngữ in.