iMin 内置打印机开发者文档

文档更新说明

版本号	更新日期	更新内容	撰写人
V1.0.0	2023/7/31	初始版本	谢华炎
V1.0.1	2024/4/18	对接 SDK 优化，添加通过监听广播的方式获取打印机状态	谢华炎

目录

简介

iMin 机器有内置了**热敏打印机**，允许 App 通过 sdk 直接打印热敏小票，具有打印机的产品有：

手持金融系列 —— M2-202、M2-203、M2 Pro、Swift 1 等
平板终端系列 —— M2 Max、D1、D1 Pro、Falcon 1 等
台式收银机设备 —— D4 等

iMin 产品的内置打印机一共有 2 种规格：

- 80mm 纸张宽度，带切刀，兼容 58mm， 如 Falcon 1 搭载了这种打印机
- 58mm 纸张宽度，不带切刀，如 D1、D1 Pro、M2 Max 搭载了这种打印机

App 开发者可以使用 3 种方式调用内置热敏打印机：

•**通过内置打印服务接口调用打印机** —— 此种方式适用于初次开发打印相关 app 且对 epson 指令没有了解的开发者，通过 iMin 打印服务提供的多个打印接口实现自己需要的打印效果；

• **通过内置虚拟蓝牙设备调用打印机** —— 此种方式适用于之前有过开发蓝牙、usb 打 印机经验或是开发者 app 已经实现蓝牙打印机打印的开发者，仅需稍微改动代码即可实现打印效果；

- **H5 Web页面通过 JS 桥调用打印机**—此种方式适用于接入H5页面的 app；

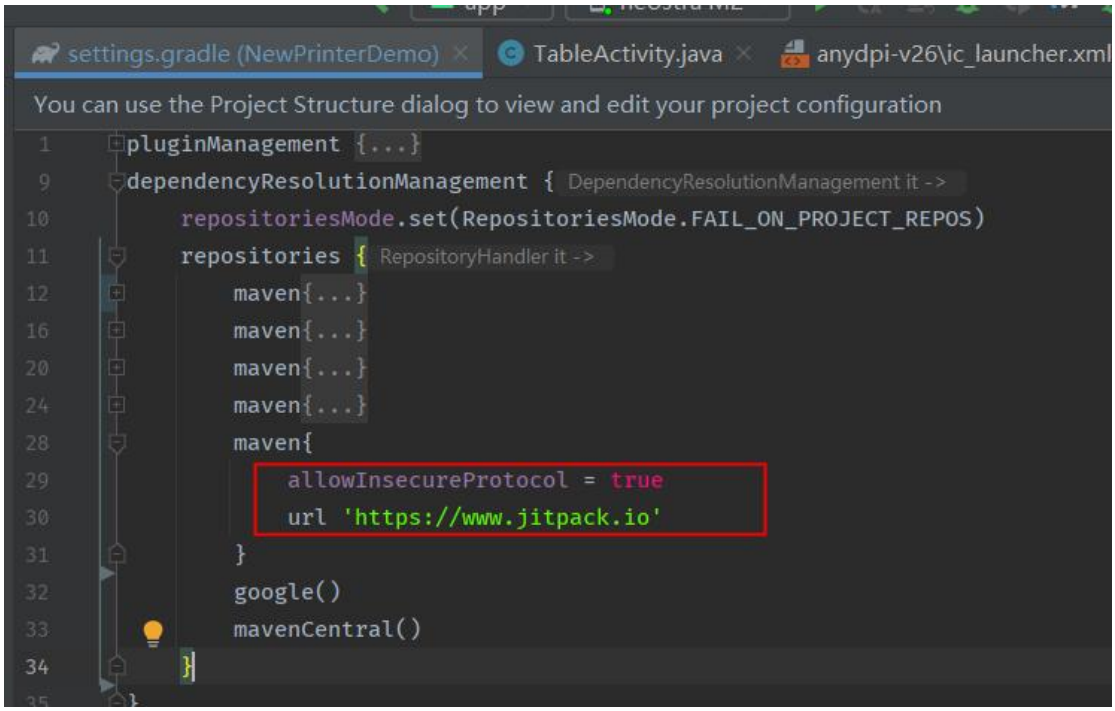
1.通过内置打印服务调用打印机接口

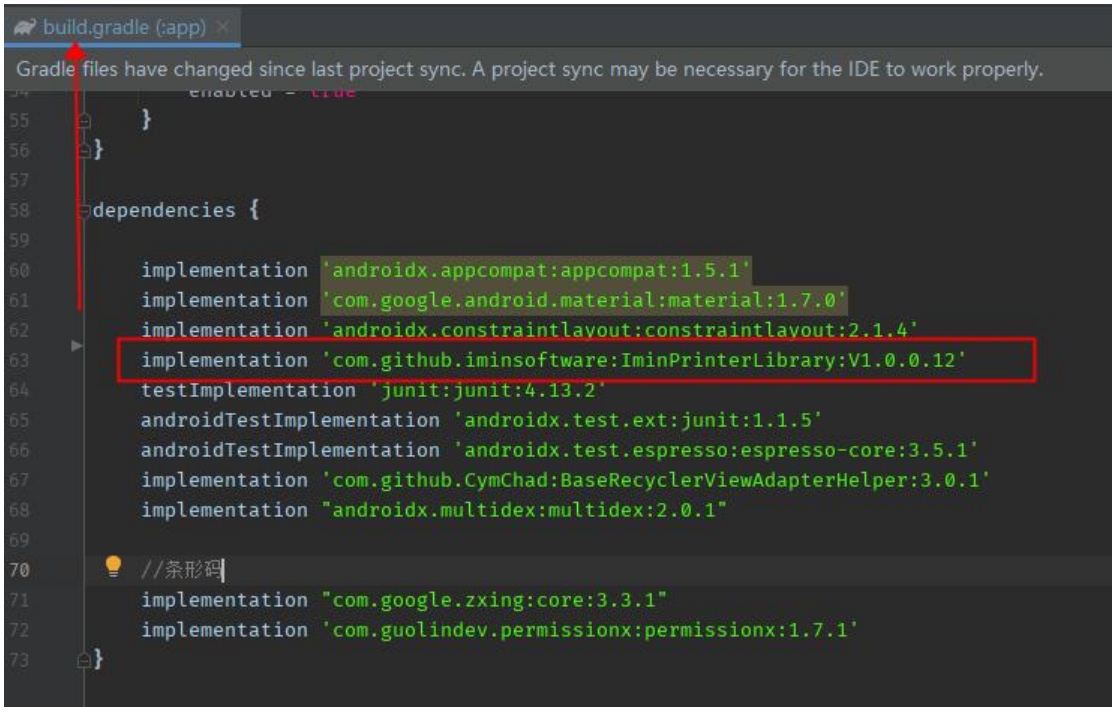
1. 通过内置打印服务接口调用打印机方式

1.1 通过 jitpack implementation 绑定远程库的方式（推荐）

com.github.iminsoftware:IminPrinterLibrary:V1.0.0.12' //已 github 上发布的最新版本为准

settings.gradle 文件配置示例：

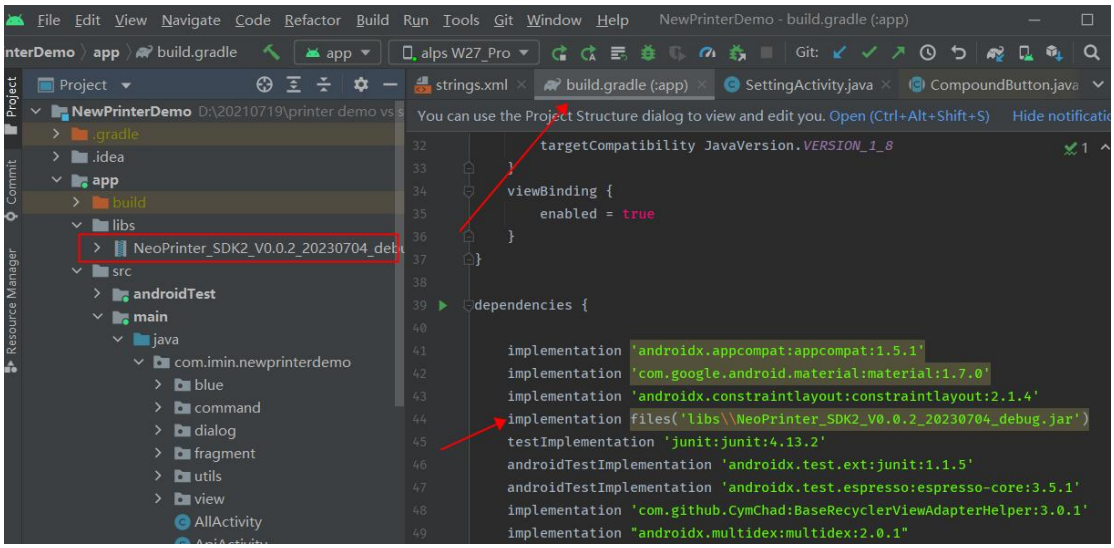




1.2 通过 jar 包集成方式（不推荐，更新成本大）

从 https://oss-sg.imin.sg/docs/en/Printer.html 开发者文档中下载 iMinPrinter_SDK2_V1.0.0.jar 包，把 jar 包放置 app-libs 目录下

如图所示：



1.3AndroidManifest.xml 文件配置(完成 1.1 或者 1.2 引入 SDK 之后需要配置)

通过包名和指定的 apk 进行交互（android 11 以上的版本）

```
<queries>
        <package android:name="com.neo.printer.sdk" />
        <intent>
            <action android:name="com.neo.printer.sdk.core.ApiAdapterManager.NeoPrinterService" />
        </intent>
    </queries>
```

示例：

注意：以上两种方式接入 SDK 后，并且在 androidMannifest.xml 文件配置完成之后进行 Sync Project clear Project build Project

2.绑定打印服务初始化工具类

2.1 初始化绑定打印服务

2.1.1 绑定服务

2.1.1.1 使用 sdk PrinterHelper.class 工具类进行初始化（默认推荐，后面的示例主要已使用 PrinterHelper 工具类交互为主）

函数：

```
PrinterHelper.getInstance().initPrinterService(this);
```

示例：

在 APK 的主页面（MainActivity） 实现类中的 finish()方法进行引用解绑服务

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    binding = ActivityMainBinding.inflate(LayoutInflater.from(this));
    setContentView(binding.getRoot());
    PrinterHelper.getInstance().initPrinterService(BaseApplication.mContext.getApplicationContext());

}
```

初始化配置代码示例（使用 SDK PrinterHelper 工具类代码示例）：

```
public class MainActivity extends BaseActivity {
    ActivityMainBinding binding;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(LayoutInflater.from(this));
        setContentView(binding.getRoot());

PrinterHelper.getInstance().initPrinterService(BaseApplication.mContext.getApplicationContext());

    }

    @Override
    public void finish() {
        super.finish();

PrinterHelper.getInstance().deInitPrinterService(BaseApplication.mContext.getApplicationContext());
        BluetoothUtil.closeBlueSocket();
    }
```

}

2.1.1.2 使用 NeoPrinterManager.class 工具类进行初始化

相关函数：NeoPrinterManager.getInstance().bindService(this,serviceConnectionCallback);

2.2 解绑服务释放资源

注意：释放资源方法用于客户应用 apk 完全退出之后调用。如果调用了该方法，重新开启 APK 的时候需要进行重新初始化

```
PrinterHelper.getInstance().deInitPrinterService(this);
或者  NeoPrinterManager.getInstance().unBindService(this,serviceConnectionCallback);
```

示例：

在 APK 的主页面（MainActivity） 实现类中的 finish()方法进行引用解绑服务

```
@Override
public void finish() {
    super.finish();
    PrinterHelper.getInstance().deInitPrinterService(BaseApplication.mContext.getApplicationContext());
    BluetoothUtil.closeBlueSocket();
}
```

或者

```
NeoPrinterManager.getInstance().unBindService(this,serviceConnectionCallback);
```

3.接口定义说明

通过 PrinterHelper.getInstance()对象,调用如下接口实现自己的打印功能

3.1 打印机初始化及设置

3.1.1 打印初始化

函数：int initPrinter(String packageName, IPrinterCallback callback);

参数：packageName ->当前 apk 的包名 ；

callback -> 初始化结果回调

注意：整个打印过程中，首次绑定服务之后进行初始化，初始化一次即可

如果已经使用 PrinterHelper 工具类则不需要调用该方法进行初始化

示例：

```
PrinterHelper.getInstance().initPrinter(getPackageName(), new INeoPrinterCallback() {
        @Override
        public void onRunResult(boolean b) throws RemoteException {
            Log.d("printerTest_PrintFragment", b ? "00000 绑定服务成功" : "uuuuu 绑定服务失败");
        }
```

```
            @Override
            public void onReturnString(String s) throws RemoteException {
            }
            @Override
            public void onRaiseException(int i, String s) throws RemoteException {
            }
            @Override
            public void onPrintResult(int i, String s) throws RemoteException {
            }
        });
```

备注：onRunResult(boolean b)初始化成功 / 失败 b=true 成功；b=false 失败

3.1.2 初始化，恢复打印参数

函数：void initPrinterParams(int fd); PrinterHelper 中对应的函数：void initPrinterParams();

参数：fd->每个应用分配的 id。

注意：因为引用了 PrinterHelper.getInstance()工具类进行调用打印方法，所以 fd 已经默认在接入的远程依赖库/jar 包中处理，后面的 API 调用中以 PrinterHelper.getInstance()工具类调用为准。客户可以借此参考

示例：

```
PrinterHelper.getInstance().initPrinterParams()
```

或者

```
//int fd:每个应用分配的 id
int fd = BaseApplication.getNeoPrinterService().initPrinter(getPackageName(), new INeoPrinterCallback()
{
                    @Override
                    public void onRunResult(boolean isSuccess) throws RemoteException {
                        Log.d(TAG,"initPrinter onRunResult isSuccess = " + isSuccess);
                    }

                    @Override
                    public void onReturnString(String result) throws RemoteException {
                        Log.d(TAG,"initPrinter onReturnString result = " + result);
                    }

                    @Override
                    public void onRaiseException(int code, String msg) throws RemoteException {
                        Log.d(TAG,"initPrinter onRaiseException code = " + code + " msg = " + msg);
                    }

                    @Override
                    public void onPrintResult(int code, String msg) throws RemoteException {
                        Log.d(TAG,"initPrinter onPrintResult code = " + code + " msg = " + msg);
                    }
                });
            } catch (RemoteException e) {
```

```
                e.printStackTrace();
            }
```

3.2 获取打印机相关信息

3.2.1 获取打印机固件版本号

函数：void getPrinterFirmwareVersion(IPrinterCallback callback);

参数：callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

```
PrinterHelper.getInstance().getPrinterFirmwareVersion(new INeoPrinterCallback() {
            @Override
            public void onRunResult(boolean isSuccess) throws RemoteException {
            }
            @Override
            public void onReturnString(String result) throws RemoteException {
getActivity().runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    
binding.textFirmwareVersion.setText(getString(R.string.printer_firmware_version,result));
                }
            });
            }
            @Override
            public void onRaiseException(int code, String msg) throws RemoteException {
            }
            @Override
            public void onPrintResult(int code, String msg) throws RemoteException {
            }
        });
```

3.2.2 获取打印服务版本号

函数：String getServiceVersion();

返回值说明：当前系统的打印服务版本号

示例：

```
PrinterHelper.getInstance().getServiceVersion();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getServiceVersion(int fd);
```

3.3 获取打印机状态

函数：int getPrinterStatus();

返回值说明：当前打印机状态

-1 ->未连接服务

3 -> 开盖

4 -> 打印头温度过高

7 -> 缺纸

0 -> 打印机正常

示例：

```
PrinterHelper.getInstance().getPrinterStatus();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterStatus(int fd);
```

3.4 打印机配置相关

3.4.1 获取 USB 连接的 pid vid

函数：String getUsbPrinterVidPid();

返回值说明：当前连接的 USBDevices 的 pid，vid

示例：

```
PrinterHelper.getInstance().getUsbPrinterVidPid();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getUsbPrinterVidPid(int fd);
```

3.4.2 获取连接的 USBDevices 的名称

函数： String getUsbDevicesName();

返回值说明：当前连接的 USBDevices 的名称

示例：

```
PrinterHelper.getInstance().getUsbDevicesName();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getUsbDevicesName(int fd);
```

3.4.3 设置打印浓度

函数：void setPrinterDensity(int density);

参数：density ->要设置的浓度值 取值（70,80,90,100,110,120,130,140,150）

示例：

```
PrinterHelper.getInstance().setPrinterDensity(100);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setPrinterDensity(int fd,100);
```

3.4.4 获取打印浓度

函数：int getPrinterDensity();

返回值说明：返回当前打印机设置的浓度

示例：

```
PrinterHelper.getInstance().getPrinterDensity();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterDensity(int fd);
```

3.4.5 获取打印长度

函数：void getPrinterPaperDistance(IPrinterCallback callback);

返回值说明：callback.onReturnString(String s) s 单位:cm

示例：

```
PrinterHelper.getInstance().getPrinterPaperDistance(new INeoPrinterCallback() {
            @Override
            public void onRunResult(boolean b) throws RemoteException {
            }
            @Override
            public void onReturnString(String s) throws RemoteException {
                Log.e(TAG,"getPrinterPaperDistance  ==》  "+s);
            }
            @Override
            public void onRaiseException(int i, String s) throws RemoteException {
            }
            @Override
            public void onPrintResult(int i, String s) throws RemoteException {
            }
        });
```

3.4.6 设定纸张规格

函数：void setPageFormat(int format);

参数：format ->纸张规格 取值（80/58）

示例：

```
PrinterHelper.getInstance().setPageFormat(80);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setPageFormat(int fd,80);
```

3.4.7 获取当前打印机设定的纸张规格

函数：int getPrinterPaperType();

返回参数：当前纸张规格 80/58

示例：

```
PrinterHelper.getInstance().getPrinterPaperType();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterPaperType(int fd);
```

3.4.8 获取切刀次数

函数：void getPrinterCutTimes(int fd, IPrinterCallback callback);

返回参数：callback.onReturnString(String s) s 单位:n

示例：

```
PrinterHelper.getInstance().getPrinterCutTimes(new INeoPrinterCallback() {
            @Override
            public void onRunResult(boolean b) throws RemoteException {
            }
            @Override
            public void onReturnString(String s) throws RemoteException {
                Log.e(TAG,"getPrinterCutTimes==》 "+s);
            }
            @Override
            public void onRaiseException(int i, String s) throws RemoteException {
            }
            @Override
            public void onPrintResult(int i, String s) throws RemoteException {
            }
        });
```

3.4.9 设置打印机模式

函数：void setPrinterMode(int fd, int mode);

参数：mode -> 1 正常模式 其他暂时不支持

示例：

```
PrinterHelper.getInstance().setPrinterMode(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setPrinterMode(int fd,1);
```

3.4.10 获取当前打印机模式

函数：int getPrinterMode(int fd);

返回值：当前设置的打印机模式 1：正常模式

示例：

```
PrinterHelper.getInstance().getPrinterMode();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterMode(int fd);
```

3.4.11 获取当前打印机配置信息

函数：void getConfigurationInfo(int fd, IPrinterCallback callback);

返回值：当前设置的打印机配置信息

示例：

```
PrinterHelper.getInstance().getConfigurationInfo(new INeoPrinterCallback() {
            @Override
            public void onRunResult(boolean b) throws RemoteException {
            }
            @Override
            public void onReturnString(final String s1) throws RemoteException {
                s = s1;
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (!Utils.isEmpty(s)){
                            Log.e(TAG,"返回数据 ==》 "+s);
                            if (s.contains(";")){
                                String[] strings = s.split(";");
                                Log.e(TAG,"返回数据 ==》 "+ Arrays.toString(strings));
                                if (strings != null && strings.length == 6){
                                    String s1 = strings[0];
                                    Log.e(TAG,"返回数据 ==》 "+ s1);
                                    if (s1.contains("=")){
                                        density                                                                    =
Integer.parseInt(s1.substring(s1.indexOf("=")+1,s1.length()));
                                    }
                                    Log.e(TAG,"返回数据 ==》 "+ density);
                                    String s2 = strings[1];
                                    if (s2.contains("=")){
                                        speed                                                                      =
Integer.parseInt(s2.substring(s2.indexOf("=")+1,s2.length()));
                                    }
                                    String s3 = strings[2];
                                    if (s3.contains("=")){
                                        format                                                                     =
Integer.parseInt(s3.substring(s3.indexOf("=")+1,s3.length()));
                                    }
                                    String s4 = strings[3];
                                    if (s4.contains("=")){
                                        codepage                                                                   =
Integer.parseInt(s4.substring(s4.indexOf("=")+1,s4.length()));
                                    }
                                    String s5 = strings[4];
                                    if (s5.contains("=")){
                                        alinment                                                                   =
Integer.parseInt(s5.substring(s5.indexOf("=")+1,s5.length()));
```

```
                                }
                                setContent();
                            }
                        }
                    }
                });
            }
            @Override
            public void onRaiseException(int i, String s) throws RemoteException {
            }
            @Override
            public void onPrintResult(int i, String s) throws RemoteException {
            }
        });
```

3.4.12 更新打印机信息

函数：void updatePrinterInfo(int fd);

参数：fd 初始化包名的返回当前应用的标记值；

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().updatePrinterInfo();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.updatePrinterInfo(int fd);
```

3.5 钱箱操作相关

3.5.1 开启钱箱

函数：void openDrawer(int fd);

参数：

示例：

```
PrinterHelper.getInstance().openDrawer();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.openDrawer(int fd);
```

3.5.2 获取当前钱箱的状态

函数： boolean getDrawerStatus(int fd);

返回值说明：true 钱箱打开；false 钱箱关闭

示例：

```
PrinterHelper.getInstance().getDrawerStatus();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getDrawerStatus(int fd);
```

3.5.3 获取钱箱打开的次数

函数：int getOpenDrawerTimes(int fd);

返回值说明：int times 返回 SDK 记录的钱箱打开的次数

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getOpenDrawerTimes();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getOpenDrawerTimes(int fd);
```

3.6 打印自检页

函数：void printerSelfChecking(int fd, IPrinterCallback callback);

参数：fd 初始化包名的返回当前应用的标记值；一般情况下不需要传输 callback

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

```
1.PrinterHelper.getInstance().printerSelfChecking(null);
2.PrinterHelper.getInstance().printerSelfChecking(new INeoPrinterCallback() {
                @Override
                public void onRunResult(boolean isSuccess) throws RemoteException {
                }
                @Override
                public void onReturnString(String result) throws RemoteException {
                }
                @Override
                public void onRaiseException(int code, String msg) throws RemoteException {
                }
                @Override
                public void onPrintResult(int code, String msg) throws RemoteException {
                }
            });
```

不使用 PrinterHelper 工具类：

```
3.iNeoPrinterService.printerSelfChecking(int fd,null);
4.iNeoPrinterService.printerSelfChecking(int fd,new INeoPrinterCallback() {
                @Override
                public void onRunResult(boolean isSuccess) throws RemoteException {
                }
                @Override
                public void onReturnString(String result) throws RemoteException {
                }
```

```
            @Override
            public void onRaiseException(int code, String msg) throws RemoteException {
            }
            @Override
            public void onPrintResult(int code, String msg) throws RemoteException {
            }
        });
```

3.7 ESC/POS 指令打印

函数：void sendRAWData(int fd, in byte[] bytes, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

bytes ->ESC/POS 指令

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().sendRAWData(bytes,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.sendRAWData(int fd,bytes,null);
```

3.8 打印走纸相关

3.8.1 走纸一行

函数：void printAndLineFeed(int fd);

参数：fd - >初始化包名的返回当前应用的标记值

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printAndLineFeed();
```

不使用：

```
iNeoPrinterService.printAndLineFeed(this.fd);
```

3.8.2 走纸若干行,自定义高度

函数：void printAndFeedPaper(int fd, int value);

参数：fd - >初始化包名的返回当前应用的标记值

value ->0<value<1016　如果 value 取值大于 1016 则选择 1016

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printAndFeedPaper(70);
```

不使用：

```
iNeoPrinterService.printAndFeedPaper(this.fd, value);
```

3.8.3 回退打印纸

函数：void printAndQuitPaper(int fd, int value);

参数：fd - >初始化包名的返回当前应用的标记值

value ->0<value<1016　如果 value 取值大于 1016 则选择 1016

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printAndQuitPaper(70);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printAndQuitPaper(this.fd, value);
```

3.9 切刀（切纸）相关

3.9.1 切纸（半切）

函数：void partialCut(int fd);

参数：fd - >初始化包名的返回当前应用的标记值

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().partialCut();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.partialCut(int fd);
```

3.9.2 切纸（全切）

函数：void fullCut(int fd);

参数：fd - >初始化包名的返回当前应用的标记值

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().fullCut();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.fullCut(int fd);
```

3.9.3 切刀复位

函数：void getPrinterKnifeReset(int fd, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

Callback ->

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getPrinterKnifeReset(null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterKnifeReset(int fd,null);
```

3.10 指令打印文本相关

3.10.1 设置字符倍数

函数：void setFontMultiple(int fd, int wide, int high);

参数：fd - >初始化包名的返回当前应用的标记值

wide ->倍宽　取值范围 0<= wide<=8

hight -> 倍高　取值范围 0<= wide<=8

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontMultiple(1,0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontMultiple(int fd,1,0);
```

3.10.2 设置字符是否加粗

函数：void setFontBold(int fd, boolean bold);

参数：fd - >初始化包名的返回当前应用的标记值

bold -> true 字符加粗　false 字符取消加粗

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontBold(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontBold(int fd,false);
```

3.10.3 设置字符反白

函数：void setFontAntiWhite(int fd, boolean antiWhite);

参数：fd - >初始化包名的返回当前应用的标记值

antiWhite -> true 字符反白 ， false 字符取消反白

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontAntiWhite(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontAntiWhite(int fd,false);
```

3.10.4 设置字符斜体

函数：void setFontItalic(int fd, boolean italic);

参数：fd - >初始化包名的返回当前应用的标记值

italic-> true 字符斜体 ， false 字符取消斜体

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontItalic(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontItalic(int fd,false);
```

3.10.5 设置字符下划线

函数：void setFontUnderline(int fd, int underline);

参数：fd - >初始化包名的返回当前应用的标记值

underline ->0 取消下划线

-> 1 轻下划线

->2 重下划线

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontUnderline(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontUnderline(int fd,0);
```

3.10.6 设置字符旋转

函数：void setFontRotate(int fd, int rotate);

参数：fd - >初始化包名的返回当前应用的标记值

rotate -> 0 接触旋转模式

->1 设置 90°顺时针旋转模式

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontRotate(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontRotate(int fd,0);
```

3.10.7 设置字体方向

函数：void setFontDirection(int fd, int direction);

参数：fd - >初始化包名的返回当前应用的标记值

direction ->0 左至右

->1 旋转 180 度

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontDirection(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontDirection(int fd,0);
```

3.10.8 设置字符间距

函数：void setFontLineSpacing(int fd, int space);

参数：fd - >初始化包名的返回当前应用的标记值

space ->设置字符间距位 anint 个锤子点距，（1mm = 8dots）

->范围（0<= anint <= 255）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontLineSpacing(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontLineSpacing(int fd,0);
```

3.10.9 设置中文字符间距

函数：void setFontChineseSpace(int fd, int chsLeftSpace, int chsRightSpace);

参数：fd - >初始化包名的返回当前应用的标记值

chsLeftSpace ->距离小票左边距 范围（0<chsLeftSpace<255）

chsRightSpace->距离小票右边距 范围（0<chsRightSpace<255）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontChineseSpace(0,0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontChineseSpace(int fd,0,0);
```

3.10.10 设置字符间距

函数：void setFontCharSpace(int fd, int space);

参数：fd - >初始化包名的返回当前应用的标记值

space ->字符之间的距离

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontCharSpace(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontCharSpace(int fd,0);
```

3.10.11 设置字符尺寸

函数：void setFontCharSize(int fd, int height,int width,int underLine, int asciitype);

参数：fd - >初始化包名的返回当前应用的标记值

height -> 倍高 0 无效 1 有效

width -> 倍宽 0 无效 1 有效

underLine -> 下划线 0 无效 1 有效

asciitype -> ASCII 字形 0 : 12*24 1: 1 9*17

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontCharSize(0,0,0,0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontCharSize(int fd,0,0,0,0);
```

3.10.12 设定汉字打印模式

函数：void setFontChineseMode(int fd, int mode);

参数：fd - >初始化包名的返回当前应用的标记值

mode -> 0 进入汉字模式 1 汉字 ASCII 字符混合模式（退出汉字模式）

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().setFontChineseMode(1);

不使用 PrinterHelper 工具类：

iNeoPrinterService.setFontChineseMode(int fd,1);

3.10.13 设置汉字尺寸

函数：void setFontChineseSize(int fd, int height, int width, int underLine, int chineseType);

参数：fd - >初始化包名的返回当前应用的标记值

height -> 倍高 0 无效 1 有效

width -> 倍宽 0 无效 1 有效

underLine -> 下划线 0 无效 1 有效

asciitype -> 汉字字形 0 : 24*24 1: 16*16

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().setFontChineseSize(0,0,0,0);

不使用 PrinterHelper 工具类：

iNeoPrinterService.setFontChineseSize(int fd,0,0,0,0);

3.10.14 文本字符打印

函数：void printText(int fd, String text, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

text ->待打印的文本

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().printText("text\n",null);

不使用 PrinterHelper 工具类：

iNeoPrinterService.printText(int fd,"text\n",null);

3.10.15 文字打印添加对齐方式

函数：void printTextWithAli(int fd, String text, int anInt, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

text ->待打印的文本

anInt ->0 左对齐 1 居中 2 右对齐

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().printTextWithAli("text\n",0,null);

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printTextWithAli(int fd,“text\n”,0,null);
```

3.10.16 文字打印添加字符编码

函数：void printTextWithEncode(int fd, String text, String code, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

text ->待打印的文本

code ->默认 GB18030

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printTextWithEncode(“text\n”,”GB18030 ”,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printTextWithEncode(int fd,“text\n”,”GB18030 ”,null);
```

3.11 设置打印全局对齐方式

函数：void setCodeAlignment(int fd, int alignmentMode);

参数：fd - >初始化包名的返回当前应用的标记值

alignmentMode -> 0 居左 1 居中 2 居右

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setCodeAlignment(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setCodeAlignment(int fd,0);
```

3.12 设置字符国家码

函数：void setFontCountryCode(int fd, int country);

参数：fd - >初始化包名的返回当前应用的标记值

country ->国家码

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontCountryCode(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontCountryCode(int fd,0);
```

3.13 获取支持的字符国家码

函数：List<String> getFontCountryCode(int fd);

返回值：字符国家码列表

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getFontCountryCode();
```

不使用 PrinterHelper 工具类：

iNeoPrinterService.getFontCountryCode(int fd,);

3.14 设置字符代码页

函数：void setFontCodepage(int fd, int codepage);

参数：fd - >初始化包名的返回当前应用的标记值

codepage ->设置的代码页

<item>C0 (PC473:Standard Europe)</item> <item>C1 (Katakana)</item>

<item>C2 (PC850:Multilingual)</item> <item>C3 (PC860:Portuguese)</item>

<item>C4 (PC863:Canadian-French)</item> <item>C5 (PC865:Nordic)</item>

<item>C6 (West Europe)</item> <item>C7 (Greek)</item>

<item>C8 (Hebrew)</item> <item>C9 (East Europe)</item>

<item>C10 (Iran)</item> <item>C16 (WPC1252)</item>

<item>C17 (PC866:Cyrillic#2)</item> <item>C18 (PC852:Latin2)</item>

<item>C19 (PC858)</item> <item>C20 (IranII)</item>

<item>C21 (Latvian)</item>

<item>C22 (Arabic)</item>

<item>C23 (PT151,1251k1)</item>

<item>C24 (PC747)</item>

<item>C25 (WPC1257)</item>

<item>C27 (Vietnam)</item>

<item>C28 (PC864)</item>

<item>C29 (PC1001)</item>

<item>C30 (Uygur)</item>

<item>P31 (Hevrew)</item>

<item>P32 (WPC1255)</item>

<item>P33 (PC720)</item>

<item>P34 (WPC1256)</item>

<item>C35 (WPC1257)</item>

<item>P254 (PC874:Thai)</item>

<item>C255 (Thai)</item>

<item>P50 (PC437)</item>

<item>P51 (Katakana)</item>

<item>P52 (PC437)</item>

<item>P53 (PC858)</item>

<item>P54 (PC852)</item>

<item>P55 (PC860)</item>

<item>P56 (PC861)</item>

<item>P57 (PC863)</item>

<item>P58 (PC865)</item>

<item>P59 (PC866)</item>

<item>P60 (PC855)</item>

<item>P61 (PC857)</item>

<item>P62 (PC862)</item>
<item>P63 (PC864)</item>
<item>P64 (PC737)</item>
<item>P65 (PC851)</item>
<item>P66 (PC869)</item>
<item>P67 (PC928)</item>
<item>P68 (PC772)</item>
<item>P69 (PC774)</item>
<item>P70 (PC874)</item>
<item>P71 (WPC1252)</item>
<item>P72 (WPC1250)</item>
<item>P73 (WPC1251)</item>
<item>P74 (PC3840)</item>
<item>P75 (PC3841)</item>
<item>P76 (PC3843)</item>
<item>P77 (PC3844)</item>
<item>P78 (PC3845)</item>
<item>P79 (PC3846)</item>
<item>P80 (PC3847)</item>
<item>P81 (PC3848)</item>
<item>P82 (PC1001)</item>
<item>P83 (PC2001)</item>
<item>P84 (PC3001)</item>
<item>P85 (PC3002)</item>
<item>P86 (PC3011)</item>
<item>P87 (PC3012)</item>
<item>P88 (PC3021)</item>
<item>P89 (PC3041)</item>

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setFontCodepage(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setFontCodepage(int fd,0);
```

3.15 获取字符代码页

函数：List<String> getFontCodepage(int fd);

返回值：字符代码页列表

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getFontCodepage(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getFontCodepage(int fd,0);
```

3.16 文本图片打印相关

3.16.1 设置打印的字体

函数：void setTextBitmapTypeface(int fd, String typeface);

参数：fd - >初始化包名的返回当前应用的标记值

typeface ->"Typeface.DEFAULT" 设置默认字体

"Typeface.MONOSPACE" 设置等宽字体

"Typeface.DEFAULT_BOLD" 设置粗体

"Typeface.SANS_SERIF" 设置无衬线字体

"Typeface.SERIF" 设置衬线字体

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapTypeface(“Typeface.DEFAULT”);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapTypeface(int fd,“Typeface.DEFAULT”);
```

3.16.2 设置文本图片打印字体大小

函数：void setTextBitmapSize(int fd, int size);

参数：fd - >初始化包名的返回当前应用的标记值

size ->设置文本图片字体大小 默认 28

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapSize(28);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapSize(int fd,28);
```

3.16.3 设置文本图片打印字体样式

函数：void setTextBitmapStyle(int fd, int style);

参数：fd - >初始化包名的返回当前应用的标记值

style->0=正常 1=粗体 2=斜体 3=加粗斜体

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapStyle(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapStyle(int fd,0);
```

3.16.4 设置文本字体删除线

函数：void setTextBitmapStrikeThru(int fd, boolean strikeThru);

参数：fd - >初始化包名的返回当前应用的标记值

strikeThru ->true= 设置删除线，false=取消删除线

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapStrikeThru(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapStrikeThru(int fd,false);
```

3.16.5 设置文本字体下划线

函数：void setTextBitmapUnderline(int fd, boolean haveUnderline);

参数：fd - >初始化包名的返回当前应用的标记值

haveUnderline ->true= 设置下划线，false=取消下划线

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapUnderline(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapUnderline(int fd,false);
```

3.16.6 设置文本字体的行高

函数：void setTextBitmapLineSpacing(int fd, float space);

参数：fd - >初始化包名的返回当前应用的标记值

space -> 1<= space <=255

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapLineSpacing(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapLineSpacing(int fd,1);
```

3.16.7 设置文本字体字间距

函数：void setTextBitmapLetterSpacing(int fd, float space);

参数：fd - >初始化包名的返回当前应用的标记值

space -> 1<= space <=255

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapLetterSpacing(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapLetterSpacing(int fd,1);
```

3.16.8 设置文本字体反白

函数：void setTextBitmapAntiWhite(int fd, boolean antiWhite);

参数：fd - >初始化包名的返回当前应用的标记值

antiWhite -> true= 设置反白 false = 取消反白

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setTextBitmapAntiWhite(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setTextBitmapAntiWhite(int fd,false);
```

3.16.9 文本图片打印

函数：void printTextBitmap(int fd, String text, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

text ->要打印的内容

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printTextBitmap(“text\n”,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printTextBitmap(int fd,“text\n”,null);
```

3.16.10 文本图片打印添加对齐方式

函数：void printTextBitmapWithAli(int fd, String text, int align, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

text ->要打印的内容

align-> 打印文本的对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printTextBitmapWithAli(“text\n”,0,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printTextBitmapWithAli(int fd,“text\n”,0,null);
```

3.17 图片打印

3.17.1 图片打印

函数：void printBitmap(int fd, in Bitmap bitmap, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

bitmap- >图片 bitmap 对象

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBitmap(bitmap,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printBitmap(int fd,bitmap,null);
```

3.17.2 图片打印添加对齐方式

函数：void printBitmapWithAlign(int fd, in Bitmap bitmap, int alignmentMode, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

bitmap- >图片 bitmap 对象

alignmentMode-> 打印文本的对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBitmapWithAlign(bitmap,0,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printBitmapWithAlign(int fd,bitmap,0,null);
```

3.17.3 打印多张图片

函数：void printMultiBitmap(int fd, in List<Bitmap> bitmaps, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

bitmaps- >要打印的多张图片列表

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printMultiBitmap(bitmaps,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printMultiBitmap(int fd,bitmaps,null);
```

3.17.4 打印多张图片添加对齐方式

函数：void printMultiBitmapWithAlign(int fd, in List<Bitmap> bitmaps, int alignmentMode, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

alignmentMode-> 打印文本的对齐方式 0=居左 1=居中 2=居右

bitmaps- >要打印的多张图片列表

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printMultiBitmap(bitmaps,0,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printMultiBitmap(int fd,bitmaps,0,null);
```

3.17.5 图片单色处理并打印

函数：void printBitmapColorChart(int fd, in Bitmap bitmap, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

alignmentMode-> 打印文本的对齐方式 0=居左 1=居中 2=居右

bitmaps- >要打印的多张图片列表

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBitmapColorChart(bitmaps,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printBitmapColorChart(int fd,bitmaps,null);
```

3.17.6 图片单色处理并打印添加对齐方式

函数： void printBitmapColorChartWithAlign(int fd, in Bitmap bitmap,int alignmentMode, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

alignmentMode-> 打印文本的对齐方式 0=居左 1=居中 2=居右

bitmap- >要打印的图片

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBitmapColorChartWithAlign(bitmaps,0,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printBitmapColorChartWithAlign(int fd,bitmaps,0,null);
```

3.18 打印表格

3.18.1 按照宽度权重比例的方式打印表格

函数：void printColumnsString(int fd, in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

colsTextArr- > 各列文本字符串数组

colsWidthArr- >各列宽度权重，宽度所占的比例

colsAlignArr- >各列文本的对齐方式（0 居左，1 居中 ，2 居右）

colsSizeArr- >各列文本的字体大小

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

备注：四个传入的数组的长度应当保持一致

示例：

```
PrinterHelper.getInstance().printColumnsString(in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printColumnsString(int fd, in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);
```

3.18.2 按照宽度数值的方式打印表格

函数：void printColumnsText(int fd, in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

colsTextArr- > 各列文本字符串数组

colsWidthArr- >各列宽度数值数组(以英文字符计算，每个中文字符占两个英文字符，每个宽度大于 0)

colsAlignArr- >各列文本的对齐方式（0 居左，1 居中 ，2 居右）

colsSizeArr- >各列文本的字体大小

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

备注：四个传入的数组的长度应当保持一致

示例：

```
PrinterHelper.getInstance().printColumnsText(in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printColumnsText(int fd, in String[] colsTextArr, in int[] colsWidthArr, in int[] colsAlignArr, in int[] colsSizeArr,IPrinterCallback callback);
```

3.19 一维码打印相关

3.19.1 设置一维码的宽度

函数：void setBarCodeWidth(int fd, int width);

参数：fd - >初始化包名的返回当前应用的标记值

width - >一维码宽度取值范围 2<= width <= 6

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setBarCodeWidth(2);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setBarCodeWidth(int fd,2);
```

3.19.2 设置一维码的高

函数：void setBarCodeHeight(int fd, int height);

参数：fd - >初始化包名的返回当前应用的标记值

height- > 一维码的高度，取值范围 24<= height<= 250

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setBarCodeHeight(162);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setBarCodeHeight(int fd,162);
```

3.19.3 设置一维码 HRI 字符的位置

函数：void setBarCodeContentPrintPos(int fd, int pos);

参数：fd - >初始化包名的返回当前应用的标记值

pos- > HRI 字符的位置，取值范围 0<= height<= 3 ，

0 不打印，1 一维码上方，2 一维码下方，3 上下均打印

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setBarCodeContentPrintPos(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setBarCodeContentPrintPos(int fd,0);
```

3.19.4 打印一维码

函数：void printBarCode(int fd, String data,int barCodeType, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data- > 一维码的内容

barCodeType- > 一维码类型

<item>0 UPC-A</item>

<item>1 UPC-E</item>

<item>2 JAN13（EAN13）</item>

<item>3 JAN8（EAN8）</item>

<item>4 CODE39</item>

<item>5 ITF</item>

<item>6 CODABAR</item>

<item>7 CODE93</item>

<item>8 CODE128</item>

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBarCode(“123456”,4);
```

不使用 PrinterHelper 工具类：

iNeoPrinterService.printBarCode(int fd,“123456”,4);

3.19.5 打印一维码添加对齐方式

函数：void printBarCodeWithAlign(int fd, String data,int barCodeType, int alignmentMode,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data- > 一维码的内容

barCodeType- > 一维码类型

<item>0 UPC-A</item>

<item>1 UPC-E</item>

<item>2 JAN13（EAN13）</item>

<item>3 JAN8（EAN8）</item>

<item>4 CODE39</item>

<item>5 ITF</item>

<item>6 CODABAR</item>

<item>7 CODE93</item>

<item>8 CODE128</item>

alignmentMode-> 一维码对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().printBarCodeWithAlign(“123456”,4,0);

不使用 PrinterHelper 工具类：

iNeoPrinterService.printBarCodeWithAlign(int fd,“123456”,4,0);

3.19.6 一维码打印传全参数

函数：void printBarCodeWithFull(int fd, String data, int barCodeType, int width, int height, int textposition, int alignmentMode, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data- > 一维码的内容

barCodeType- > 一维码类型

<item>0 UPC-A</item>

<item>1 UPC-E</item>

<item>2 JAN13（EAN13）</item>

<item>3 JAN8（EAN8）</item>

<item>4 CODE39</item>

<item>5 ITF</item>

<item>6 CODABAR</item>

<item>7 CODE93</item>

<item>8 CODE128</item>

width - >一维码宽度取值范围 2<= width <= 6

height- > 一维码的高度，取值范围 24<= height<= 250

textposition- > HRI 字符的位置，取值范围 0<= height<= 3

alignmentMode-> 一维码对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printBarCodeWithFull("123456", 4, 2, 162, 0, 0, null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.printBarCodeWithFull(int fd, "123456", 4, 2, 162, 0, 0, null);
```

备注：

条形码类型（0-6,73,8）	支持的条形码内容长度	支持的 ASCII 代码范围	
0 --> UPC-A	条形码内容长度 = 11,12	48 ≤range≤ 57	所有机型机型
1 --> UPC-E	条形码内容长度 = 11,12	48 ≤range≤ 57	所有机型机型
2 --> JAN13 / EAN13	条形码内容长度= 12,13	48 ≤range≤ 57	所有机型机型
3 --> JAN8 / EAN8	条形码内容长度 = 7 ,8	48 ≤range≤ 57	所有机型机型
4 --> CODE39	条形码内容长度 >=1	48≤range≤57,65≤range≤90, range= 32, 36, 37, 42, 43, 45, 46, 47	所有机型机型
5 --> ITF	条形码内容长度 >=2	48 ≤range≤ 57	所有机型机型
6 --> CODABAR	条形码内容长度 >=2	48≤range≤57, 65≤range≤68, 97≤range≤100, range = 36, 43, 45, 46, 47, 58	Z2 系列不支持打印，其它机型支持
73 -->CODE128，8-->CODE128	条形码内容长度 >=2	0≤range≤127	所有机型机型
7 -->CODE93	1≤n≤255	1≤n≤255	

3.20 二维码打印

3.20.1 设置二维码的大小

函数：void setQrCodeSize(int fd, int size);

参数：fd - >初始化包名的返回当前应用的标记值

size- >二维码的大小 取值范围 1<= size <= 11

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setQrCodeSize(2);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setQrCodeSize(int fd,2);
```

3.20.2 设置二维码的纠错级别

函数：void setQrCodeErrorCorrectionLev(int fd, int level);

参数：fd - >初始化包名的返回当前应用的标记值

level- >二维码的纠错级别 取值范围 0<= level<= 3

0 ->纠错级别 L(7%)

1 ->纠错级别 M(15%)

2 ->纠错级别 Q(25%)

3 ->纠错级别 H(30%)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setQrCodeErrorCorrectionLev(2);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setQrCodeErrorCorrectionLev(int fd,2);
```

3.20.3 打印二维码

函数：void printQrCode(int fd, String data, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data - >二维码的内容

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printQrCode(“123456”,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setQrCodeErrorCorrectionLev(int fd,“123456”,null);
```

3.20.4 打印二维码添加对齐方式

函数：void printQrCodeWithAlign(int fd, String data, int alignments, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data- > 一维码的内容

alignments -> 一维码对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().printQrCodeWithAlign(“123456”,0,null);

不使用 PrinterHelper 工具类：

iNeoPrinterService.printQrCodeWithAlign(int fd,“123456”,null);

3.20.5 二维码打印传全参数

函数：void printQRCodeWithFull(int fd, String data, int size, int errorlevel, int alignments, IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

data- > 一维码的内容

size- >二维码的大小 取值范围 1<= size <= 11

level- >二维码的纠错级别 取值范围 0<= level<= 3

alignments -> 一维码对齐方式 0=居左 1=居中 2=居右

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().printQRCodeWithFull(“123456”,1,2,0,null);

不使用 PrinterHelper 工具类：

iNeoPrinterService.printQRCodeWithFull(int fd,“123456”,1,2,0,null);

3.21 设置左边距

函数：void setLeftMargin(int fd, int valve);

参数：fd - >初始化包名的返回当前应用的标记值

valve- > 设置左边距的值 取值范围：0<=valve <=255

备注：该属性设置是全局的，设置了一定的值需要自行还原

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().setLeftMargin(20);

不使用 PrinterHelper 工具类：

iNeoPrinterService.setLeftMargin(int fd,20);

3.22 打印双二维码

3.22.1 设置双二维码的大小

函数：void setDoubleQRSize(int fd, int size);

参数：fd - >初始化包名的返回当前应用的标记值

size - >双二维码的大小 取值范围 1<= size <= 8

示例：

使用 PrinterHelper 工具类：

PrinterHelper.getInstance().setDoubleQRSize(1);

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQRSize(int fd,1);
```

3.22.2 设置双二维码(QR1)左边边距

函数：void setDoubleQR1MarginLeft(int fd, int qr1Left);

参数：fd - >初始化包名的返回当前应用的标记值

qr1Left - >双二维码 QR1 左边距 取值范围（0<= qr1Left <=255）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR1MarginLeft(10);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR1MarginLeft(int fd,10);
```

3.22.3 设置双二维码(QR2)左边边距

函数：void setDoubleQR2MarginLeft(int fd, int qr2Left);

参数：fd - >初始化包名的返回当前应用的标记值

qr2Left - >双二维码 QR1 左边距 取值范围（0<= qr2Left<=255）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR2MarginLeft(200);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR2MarginLeft(int fd,200);
```

3.22.4 设置双二维码(QR1)误差

函数：void setDoubleQR1Level(int fd, int qr1Level);

参数：fd - >初始化包名的返回当前应用的标记值

qr1Level - >双二维码 QR1 误差 取值范围（0<= qr1Level<=3）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR1Level(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR1Level(int fd,1);
```

3.22.5 设置双二维码(QR2)误差

函数：void setDoubleQR2Level(int fd, int qr2Level);

参数：fd - >初始化包名的返回当前应用的标记值

qr2Level - >双二维码 QR1 误差 取值范围（0<= qr2Level<=3）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR2Level(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR2Level(int fd,1);
```

3.22.6 设置双二维码(QR1)版本

函数：void setDoubleQR1Version(int fd, int qr1Version);

参数：fd - >初始化包名的返回当前应用的标记值

qr1Version - >双二维码 QR1 误差 取值范围（0<= qr1Version<=3）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR1Version(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR1Version(int fd,1);
```

3.22.7 设置双二维码(QR2)版本

函数：void setDoubleQR2Version(int fd, int qr2Version);

参数：fd - >初始化包名的返回当前应用的标记值

qr2Version - >双二维码 QR1 误差 取值范围（0<= qr2Version<=3）

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDoubleQR2Version(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR2Version(int fd,1);
```

3.22.8 打印双二维码

函数：void printDoubleQR(int fd, String qr1Data,String qr2Data,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

qr1Data - >QR1 的内容

qr2Data - >QR2 的内容

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().printDoubleQR(“123456&147”,”fsdfsdfsdfsd144411444&&&&”,null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDoubleQR2Version(int fd, “123456&147” ,”fsdfsdfsd11444&&&&”,null);
```

3.23 事务打印相关

事务打印模式适用于需要控制打印内容并得到打印结果反馈(是否打印出小票)的需求，此模式相当于建立 一个事务队列缓冲区，当开发者进入事务打印模式，将开启一个事务队列，可以向其中增加打印方法，此时 打印机不会立刻打印内容，当提交事务后，打印机才会依次执行队列中的任务，执行结束将获得此次事务的 结果反馈。

事务打印注意事项：

1. 当进入缓冲（事务）打印后，提交打印成功将返回成功结果，但遇到打印机异常如缺纸、过热等，将 会丢掉本次提交事务中所有指令任务，同时反馈异常，即当一单任务执行前或执行中打印机异常，则 此单不会打出；
2. 当指令打印和缓冲（事务）打印交替使用时，如果打印机异常，不会清除指令打印的内容！

3. 进入事务打印模式后，但不会立即打印输出，会将输出 内容缓存到缓存区，当调用exitPrinterBuffer()或commitPrinterBuffer()等方法才会进行打印输出。

4. 事务打印结果回调在 IPrinterCallback方法中的 onPrintResult(int code, String msg)方法（会有一定的耗 时，要等物理打印出纸，不推荐单行频繁使用事务打印，将会影响打印速度，推荐整张小票使用事务 打印），

对应返回 code 如下：

a) 0 ！ 打印成功，msg 为“Transaction print successful!”；

b) 1 ！ 打印失败，msg 为“Transaction print failed!”；

3.23.1 进入事务模式

函数：void enterPrinterBuffer(int fd, boolean clean);

参数：fd - >初始化包名的返回当前应用的标记值

clean - > 是否清除事务队列的数据

true ->清除事务队列的未打印的数据，

false ->不清除事务队列未打印的数据，下次提交事务一起打印出来

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().enterPrinterBuffer(false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.enterPrinterBuffer(int fd,false);
```

3.23.2 提交事务

函数：void commitPrinterBuffer(int fd);

参数：fd - >初始化包名的返回当前应用的标记值

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().commitPrinterBuffer();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.commitPrinterBuffer(int fd);
```

3.23.2 提交事务回调结果

函数：void commitPrinterBufferWithCallback(int fd,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().commitPrinterBufferWithCallback(callback);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.commitPrinterBufferWithCallback(int fd,callback);
PrinterHelper.getInstance().commitPrinterBuffer(new INeoPrinterCallback() {
```

```
            @Override
            public void onRunResult(boolean isSuccess) throws RemoteException {
                Log.d(TAG," printTextWithAli  onRunResult ====>    "+isSuccess);
            }
            @Override
            public void onReturnString(String result) throws RemoteException {
                Log.d(TAG,"  onReturnString ====>    "+result);
            }
            @Override
            public void onRaiseException(int code, String msg) throws RemoteException {

            }
            @Override
            public void onPrintResult(int code, String msg) throws RemoteException {
                Log.d("NeoPrinterSDK_事务打印回调","  事务打印结果 ===== 》onPrintResult ====>    "+"
事务打印的 code=> "+code+"  , 事务打印的描述==》 "+msg);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        binding.tvResult.setText("事务打印的 code=> "+code+"  , 事务打印的描述==》
"+msg);
                    }
                });
            }
        })
```

3.23.3 退出事务打印

函数：void exitPrinterBuffer(int fd, boolean commit);

参数：fd - >初始化包名的返回当前应用的标记值

commit - > 是否清除事务队列的数据 true 清除事务队列的缓存数据，false 不清楚队列

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().exitPrinterBuffer(true);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.exitPrinterBuffer(int fd,true);
```

3.23.4 退出事务打印回调结果

函数：void exitPrinterBufferWithCallback(int fd, boolean commit,IPrinterCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

commit - > 是否清除事务队列的数据 true 清除事务队列的缓存数据，false 不清楚队列

callback ->onReturnString(String result) :返回接口执行的结果(字符串数据)

->onRaiseException(int code, String msg):接口执行失败时发生异常情况的具体原因

->onPrintResult(int code, String msg):返回打印机结果 code=0 成功 1 失败

->onReturnString(String result):返回接口执行的结果(字符串数据)

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().exitPrinterBufferWithCallback(true,callback);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.exitPrinterBufferWithCallback(int fd,true,callback);
PrinterHelper.getInstance().exitPrinterBufferWithCallback(true,new INeoPrinterCallback() {
            @Override
            public void onRunResult(boolean isSuccess) throws RemoteException {
                Log.d(TAG," printTextWithAli  onRunResult ====>    "+isSuccess);
            }
            @Override
            public void onReturnString(String result) throws RemoteException {
                Log.d(TAG,"  onReturnString ====>    "+result);
            }
            @Override
            public void onRaiseException(int code, String msg) throws RemoteException {

            }
            @Override
            public void onPrintResult(int code, String msg) throws RemoteException {
                Log.d("NeoPrinterSDK_事务打印回调","  事务打印结果 ===== 》onPrintResult ====>    "+"
事务打印的 code=> "+code+"  ，事务打印的描述==》 "+msg);
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        binding.tvResult.setText("事务打印的 code=> "+code+"  ，事务打印的描述==》
"+msg);
                    }
                });
            }
        })
```

整个事务打印示例：

```
 enterPrinterBuffer(true) //进入事务模式，此后所有命令不会立刻输出
   printText(/*something*/)
   printBitmap(/*bitmap resource*/)
   // …… 其它打印相关方法——打印一些内容
   commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)//提交一次事务，此时打印机将开始打印，
当打印成功或失败将在 callback 中返回
   ……等待上一次事务的返回
   printText(/*something*/)
   printBitmap(/*bitmap resource*/)
   //…… 其它打印相关方法——可以选择等待或不等待上一次事务的返回继续打印内容
   commitPrinterBuffer()/commitPrinterBufferWithCallback(callback)//继续提交下一次事务，此时打印机将继续
打印
```

exitPrinterBuffer(true)/exitPrinterBufferWithCallback(true, callback)//退出事物模式时调用，如果在上一次提交后又输入新的数据则会继续打印否则不打印

3.24 打印 Debug log 配置

3.24.1 设置 Debug log 的等级

函数：void setDebugLogLevel(int fd,int level);

参数：fd - >初始化包名的返回当前应用的标记值

level - >log 等级

1 表示打印普通调试 Log，

2 表示打印关键调试 Log，

3 表示打印打印机原始数据 Log

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDebugLogLevel(1);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDebugLogLevel(int fd,1);
```

3.24.2 设置 Debug log 的存储大小

函数：void setDebugLogSize(int fd,int size);

参数：fd - >初始化包名的返回当前应用的标记值

size - >存储数据的大小 默认 200M

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDebugLogSize(200*1024*1024);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDebugLogSize(int fd,200*1024*1024);
```

3.24.3 设置 Debug log 的各个模块的开关

函数：void setDebugLogModule(int fd,String module, boolean isOpen);

参数：fd - >初始化包名的返回当前应用的标记值

isOpen - >是否开启 debug true 开启 debug 模式，false 关闭 debug 模式

module ->

应用 API 接口与数据适配层"ApiAdapterManager" ->

DebugLogConfig.setLogApiAdapterManager(isOpen);

应用用户行为管控模块 "AppBehaviorManager" -> DebugLogConfig.setLogAppBehaviorManager(isOpen);

数据 管理和配置适配层 "DataAndConfigAdapter" -> DebugLogConfig.setLogDataAndConfigAdapter(isOpen);

数据库管理模块 "DatabaseManager" -> DebugLogConfig.setLogDatabaseManager(isOpen);

异常管控模块 "ExceptionManager" -> DebugLogConfig.setLogExceptionManager(isOpen);

硬件接口适配层 "PrinterHwAdapter" -> DebugLogConfig.setLogPrinterHwAdapter(isOpen);

状态管理模块 "PrinterStateCfgManager" ->
DebugLogConfig.setLogPrinterStateCfgManager(isOpen);

打印业务模块 "PrinterTaskManager" ->

DebugLogConfig.setLogPrinterTaskManager(isOpen);

升级模块 "UpgradeManager" -> DebugLogConfig.setLogUpgradeManager(isOpen);

蓝牙服务模块 "VirtualBluetoothService" ->
DebugLogConfig.setLogVirtualBluetoothService(isOpen);

打印升级模块 "PrinterUpdateManager" -> DebugLogConfig.setLogPrinterUpdateManager(isOpen);

JS 打印模块"JSPrinter" -> DebugLogConfig.setLogJSPrinter(isOpen);

打印状态服务 "PrintStatusManager" -> DebugLogConfig.setLogPrintStatusManager(isOpen);

Web 打印模块 "WebPrinterService" -> DebugLogConfig.setLogWebPrinterService(isOpen);

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setDebugLogModule("PrinterHwAdapter",false);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setDebugLogModule(int fd,"PrinterHwAdapter",false);
```

3.24.3 获取 Debug log 配置

函数：String getDebugLogState(int fd);

返回值：

```
"LogLevel:" + DebugLogConfig.getLogLevel() + "\tLogSize:" + DebugLogConfig.getLogSize() +
                "\tLogApiAdapterManager:" + DebugLogConfig.isLogApiAdapterManager() +
                "\tLogAppBehaviorManager:" + DebugLogConfig.isLogAppBehaviorManager() +
                "\tLogDataAndConfigAdapter:" + DebugLogConfig.isLogDataAndConfigAdapter() +
                "\tLogDatabaseManager:" + DebugLogConfig.isLogDatabaseManager() +
                "\tLogExceptionManager:" + DebugLogConfig.isLogExceptionManager() +
                "\tLogPrinterHwAdapter:" + DebugLogConfig.isLogPrinterHwAdapter() +
                "\tLogPrinterStateCfgManager:" + DebugLogConfig.isLogPrinterStateCfgManager() +
                "\tLogPrinterTaskManager:" + DebugLogConfig.isLogPrinterTaskManager() +
                "\tLogUpgradeManager:" + DebugLogConfig.isLogUpgradeManager()+
                "\tLogVirtualBluetoothService:" + DebugLogConfig.isLogVirtualBluetoothService()+
                "\tLogPrinterUpdateManager:" + DebugLogConfig.isLogPrinterUpdateManager()+
                "\tLogJSPrinter:" + DebugLogConfig.isLogJSPrinter()+
                "\tLogPrintStatusManager:" + DebugLogConfig.isLogPrintStatusManager()+
                "\tLogWebPrinterService:" + DebugLogConfig.isLogPrintStatusManager()
```

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getDebugLogState();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getDebugLogState(int fd);
```

3.25 打印机升级相关

3.25.1 设置打印机升级 bin 文件的路径

函数：void setPrinterUpdatePath(int fd, String path);

参数：fd - >初始化包名的返回当前应用的标记值

path - >待升级的打印机 bin 文件的路径

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setPrinterUpdatePath(path);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setPrinterUpdatePath(int fd,path);
```

3.25.2 获取打印机升级 bin 文件的路径

函数：String getPrinterUpdatePath(int fd);

返回值说明：path - >待升级的打印机 bin 文件的路径

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getPrinterUpdatePath();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterUpdatePath(int fd);
```

3.25.3 开启打印机固件升级

函数：void startPrinterUpdate(int fd);

参数：fd - >初始化包名的返回当前应用的标记值

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().startPrinterUpdate();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.startPrinterUpdate(int fd);
```

3.25.4 获取打印机固件升级的进度状态

函数：void getPrinterUpdateStatus(int fd, IPrinterUpdateCallback callback);

参数：fd - >初始化包名的返回当前应用的标记值

callback ->

printerUpdateStatus(int status, int arg1, String message) :

->status:当前执行的类型

->arg1:升级状态

->message:返回接口执行的结果(字符串数据)

error(String s) -> 升级失败

示例：

使用 PrinterHelper 工具类：

```
 PrinterHelper.getInstance().getPrinterUpdateStatus(new INeoPrinterUpdateCallback() {
                @Override
                public  void  printerUpdateStatus(int  status,  int  arg1,  String  message)  throws
RemoteException {
                    Log.e("printdemoTAG", "PrinterUpdateManager printerUpdateStatus 88888 ===> " + status
+ " ,message.obj==> " + message + " ,message.arg1==>" + arg1);

                    getActivity().runOnUiThread(new Runnable() {
                        @Override
```

```
                    public void run() {
                        ShowMessage("printerUpdateStatus ===> " + status + "    ,message.what==> " + arg1
+ "  ,message.obj==>  " + message);

                        switch (status) {
                            case 110://USB 口连接失败
                                break;
                            case 111://开始更新
//                          showUpdate();
                                break;
                            case 112:
                                setProgressBarMax(arg1);
                                break;
                            case 113:
                                setProgressBar(arg1);
                                break;
                            case 114://更新成功
                                handler.postDelayed(new Runnable() {
                                    @Override
                                    public void run() {
                                        binding.btGetPath.setEnabled(true);
                                        binding.btUpdate.setEnabled(true);
                                        binding.btUpdate.setAlpha(1f);
                                        binding.btGetPath.setAlpha(1f);
                                    }
                                }, 15000);
                                break;
                            case 115://更新完成

                                break;
                        }
                    }
                });
            }
            @Override
            public void error(String s) throws RemoteException {

            }
        });
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterUpdateStatus(int fd,new INeoPrinterUpdateCallback() {
            @Override
            public  void  printerUpdateStatus(int  status,  int  arg1,  String  message)  throws
RemoteException {
```

```
                Log.e("printdemoTAG", "PrinterUpdateManager printerUpdateStatus 88888 ===> " + status
+ " ,message.obj==> " + message + " ,message.arg1==>" + arg1);

                getActivity().runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        ShowMessage("printerUpdateStatus ===> " + status + "   ,message.what==> " + arg1
+ "  ,message.obj==>  " + message);
                        switch (status) {
                            case 110://USB 口连接失败
                                break;
                            case 111://开始更新
                                break;
                            case 112:
                                setProgressBarMax(arg1);
                                break;
                            case 113:
                                setProgressBar(arg1);
                                break;
                            case 114://更新成功
                                handler.postDelayed(new Runnable() {
                                    @Override
                                    public void run() {
                                        binding.btGetPath.setEnabled(true);
                                        binding.btUpdate.setEnabled(true);
                                        binding.btUpdate.setAlpha(1f);
                                        binding.btGetPath.setAlpha(1f);
                                    }
                                }, 15000);
                                break;
                            case 115://更新完成

                                break;
                        }
                    }
                });
            }
            @Override
            public void error(String s) throws RemoteException {

            }
        });
```

3.25.5 取消升级

函数：void setIsUpdatePrinter(int fd, int update);

参数：fd - >初始化包名的返回当前应用的标记值

update ->0 清除升级队列的数据

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setIsUpdatePrinter(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setIsUpdatePrinter(int fd,0);
```

3.25.6 获取打印机升级的状态

函数：int getPrinterIsUpdateStatus(int fd, IPrinterCallback callback);

返回值说明：0 ->打印机正常， 1 ->打印机手动升级中

2 -> 打印机自动升级中

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getPrinterIsUpdateStatus(null);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getPrinterIsUpdateStatus(int fd,null);
```

3.26 内置打印机重连配置

3.26.1 获取当前内置 USB 打印机连接的状态

函数：int getIsReconnectUsb(int fd);

返回值：0 -> 配置打印机重连模式

1 -> 配置打印机普通连接模式

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getIsReconnectUsb();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getIsReconnectUsb(int fd);
```

3.26.2 设置当前内置 USB 打印机连接的状态

函数：void setIsReconnectUsb(int fd,int isConnect);

参数：fd - >初始化包名的返回当前应用的标记值

isConnect ： 0 -> 配置打印机重连模式

1 -> 配置打印机普通连接模式

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setIsReconnectUsb(0);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setIsReconnectUsb(int fd,0);
```

3.27 内外置打印机切换

3.27.1 获取当前连接打印机的类型

函数：boolean getConnectInternalPrinter(int fd);

返回值说明：true ->内置打印机

false ->外置打印机机

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().getConnectInternalPrinter();
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.getConnectInternalPrinter(int fd);
```

3.27.2 内置外置打印机连接切换

函数：void setConnectInternalPrinter(int fd,boolean connect);

参数：fd - >初始化包名的返回当前应用的标记值

Connect ： true ->内置打印机

false ->外置打印机机 默认选择外置打印机列表的首个

备注：当 connect = false 的时候，必须设置内置打印机取消重连机制才能切换至外置打印机

即先执行 PrinterHelper.getInstance().setIsReconnectUsb(1);

或者 iNeoPrinterService.setIsReconnectUsb(int fd,1);

示例：

使用 PrinterHelper 工具类：

```
PrinterHelper.getInstance().setConnectInternalPrinter(true);
```

不使用 PrinterHelper 工具类：

```
iNeoPrinterService.setConnectInternalPrinter(int fd,true);
```

整个流程执行示例：

```
PrinterHelper.getInstance().setIsReconnectUsb(1);
PrinterHelper.getInstance().setConnectInternalPrinter(true);或者
PrinterHelper.getInstance().setConnectInternalPrinter(false);
-------
iNeoPrinterService.setIsReconnectUsb(int fd,1);
iNeoPrinterService.setConnectInternalPrinter(int fd,true);或者
iNeoPrinterService.setConnectInternalPrinter(int fd,false);
```

2 通过内置虚拟蓝牙调用打印机

1.1. 虚拟蓝牙简介

在蓝牙设备列表中可以看到一个已经配对，且永远存在的蓝牙设备“BluetoothPrinter”，这是由操作系统虚拟 出来的打印机设备，实际并不存在。虚拟蓝牙支持 iMin《esc/pos》指令。 其中有部分特殊的指令属于 iMin自定义指令，如：

功能	指令
开钱箱指令	byte［5］: 0x10 0x14 0x00 0x00 0x00
切刀指令 全部切割	byte［2］: 0x1B 0x69
切刀指令 切割（左边留一点不切）	byte［2］: 0x1d 0x56

1.2. 虚拟蓝牙使用

1.2.1. 与该蓝牙设备建立连接。

2.2.2. 将指令和文本内容拼接转码为 Bytes。

2.2.3. 发送给 InnerPrinter。

2.2.4. 底层打印服务驱动打印设备完成打印。

注：BluetoothUtil 一个蓝牙工具类，用于连接虚拟蓝牙设备 BluetoothPrinter。

2.2.4.1. 工具类 BluetoothUtil，标准的蓝牙连接工具类

示例代码：

```
public class BluetoothUtil {

    /**
     * 蓝牙是否打开
     */
    public static boolean isBluetoothOn() {
        BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
        if (mBluetoothAdapter != null)
            // 蓝牙已打开
            if (mBluetoothAdapter.isEnabled())
                return true;

        return false;
    }

    /**
     * 从已配对设配中，删选出某一特定类型的设备展示
     * @param deviceClass
     * @return
     */
    public static BluetoothDevice getDevice() {
     BluetoothDevice innerprinter_device = null;
     Set<BluetoothDevice> devices = BluetoothAdapter.getDefaultAdapter().getBondedDevices();
     for (BluetoothDevice device : devices) {
        if (device.getAddress().equals(Innerprinter_Address)) {
            innerprinter_device = device;
            break;
         }
      }
    return innerprinter_device;
   }

    /**
     * 弹出系统对话框，请求打开蓝牙
     */
    public static void openBluetooth(Activity activity) {
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
        activity.startActivityForResult(enableBtIntent, 666);
    }

    public static BluetoothSocket connectDevice(BluetoothDevice device) {
        //内置打印机蓝牙 00001101-0000-1000-8000-00805F9B34FB
```

```
            BluetoothSocket socket = null;
            try {
                socket = device.createRfcommSocketToServiceRecord(
                        UUID.fromString("00001101-0000-1000-8000-00805f9b34fb"));//第三方蓝牙打印机
00000000-0000-1000-8000-00805f9b34fb 00001101-0000-1000-8000-00805f9b34fb
                socket.connect();
                Log.i("imin_print_cmd_1111", "=================888888888888888=蓝牙连接成功 11111 ");
            } catch (IOException e) {
                Log.i("imin_print_cmd_1111",       "=================888888888888888=================
e.getMessage() "+e.getMessage());
                try {
                    socket.close();
                } catch (IOException closeException) {
                    return null;
                }
                return null;
            }
            return socket;
        }

        private static OutputStreamWriter mWriter = null;
        private static OutputStream mOutputStream = null;


        public static void openOutputStream(OutputStream outputStream, String encoding) throws IOException
{
            mWriter = new OutputStreamWriter(outputStream, encoding);
            mOutputStream = outputStream;
        }

        public static void sendData(byte[] bytes, BluetoothSocket socket) throws IOException {
            if(socket != new){
                OutputStream out = socket.getOutputStream();
            out.write(bytes, 0, bytes.length);
            out.close();
            }
        }

}
```

2.2.4.2. 蓝牙连接打印服务事例

1.判断系统蓝牙功能是否开启

```
if (BluetoothUtil.isBluetoothOn()) {
        //
  } else {
      BluetoothUtil.openBluetooth(BluetoothActivity.this);
  }
```

2. 获取 iMin 内置蓝牙设备

```
BluetoothDevice device = BluetoothUtil.getDevice(btAdapter);
if (device == null) {
    Toast.makeText(getBaseContext(),"Please Make Sure Bluetooth have InnterPrinter!",
      Toast.LENGTH_LONG).show();
return;
}
```

3. 待打印的小票数据

```
byte[] b = null;
```

4. 使用内置蓝牙打印小票数据

```
BluetoothSocket socket = null; socket = BluetoothUtil.getSocket(device);
BluetoothUtil.sendData(data, socket);
```

5. 使用蓝牙获取打印机状态方法

5.1 在应用 onCreate()中注册广播监听

```
          try {
           IntentFilter intentFilter = new IntentFilter();
           intentFilter.addAction(ACTION_PRINTER_STATUS_CHANGE);
           registerReceiver(mReceiver, intentFilter);
       } catch (Exception e) {

       }
```

5.2、在广播中处理状态值回调

```
 private static final String ACTION_PRINTER_STATUS = "status";
    ///状态值对应
    public static final int PRINTER_NORMAL = 0;//正常
    public static final int PRINTER_UNCAP = 3;////开盖
    public static final int PRINTER_LOWER_POWER = 4;//低压
    public static final int PRINTER_OVER_HEAT = 5;//过热
    public static final int PRINTER_CUTTING_ERROR = 6;//卡切刀(D4Pro)
    public static final int PRINTER_PAPER_OUT = 7;//缺纸
    public static final int PRINTER_OTHER_ERROR = 99;//其他错误
```

```
    private TextView tv_status;
    private BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            int status = intent.getIntExtra(ACTION_PRINTER_STATUS, -1);
            String PrinterStatus = "";
            switch (status) {
                case PRINTER_NORMAL:
                    PrinterStatus = "正常";
                    break;
                case PRINTER_UNCAP:
                    PrinterStatus = "开盖";
                    break;
                case PRINTER_OVER_HEAT:
                    PrinterStatus = "过热";
                    break;
                case PRINTER_LOWER_POWER:
                    PrinterStatus = "低压";
                    break;
                case PRINTER_CUTTING_ERROR:
                    PrinterStatus = "卡切刀";
                    break;
                case PRINTER_PAPER_OUT:
                    PrinterStatus = "缺纸";
                    break;
                case PRINTER_OTHER_ERROR:
                    PrinterStatus = "其他错误";
                    break;
            }
        }
    };
```

5.3、在应用 onDestroy()中注册广播监听

```
            try {
            unregisterReceiver(mReceiver);
        } catch (Exception e) {
        }
```

2.2.5 注意事项

需要在 App 的项目中添加蓝牙权限声明才能使用蓝牙设备：

```
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

```
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

3. H5 Web页面通过 JS 桥调用打印机

1. H5 Web页面调用打印插件示例

```
$('#btn').click(function() {

    // 第一种方式: 原生打印
    var mywindow = window.open('', 'PRINT', 'height=400,width=600');
    mywindow.document.write('<html><head><title>' + document.title + '</title>');
    mywindow.document.write('</head><body >');
    mywindow.document.write(document.getElementById('PrintContent').innerHTML);
```

```
    mywindow.document.write('</head><body >');
    mywindow.print();

  })
```

2. 调用 jquery 插件打印

一、 简单方式引入

1.引入打印 sdk 脚本

```
<script type="text/javascript" src="./imin-printer.min.js"></script>
```

2. 初始化

```
var IminPrintInstance = new IminPrinter();
```

3. 调用相关方法

```
IminPrintInstance.connect().then(async (isConnect) => {

 if (isConnect) {
    // 初始化打印机
     IminPrintInstance.initPrinter()
     // 获取打印机状态
     await IminPrintInstance.getPrinterStatus()
   }
})
```

二、vue2 中使用 vue-cli 脚手架引入

1. 在 vue.config.js 中配置 imin-printer 的包

```
 const { defineConfig } = require("@vue/cli-service");
const path = require('path');
console.log(path.join(__dirname,'src/assets/imin-printer.esm.browser.min.js'))
module.exports = defineConfig({
  transpileDependencies: true,
  configureWebpack: {
    resolve: {
      alias: {
        'imin-printer': path.join(__dirname, 'src/assets/imin-printer.esm.browser.min.js')
```

```
    }
  },
  plugins: [
  ]
 },
});
```

2. 在 main.js 中导入包

```
import Vue from 'vue'
import App from './App.vue'
import IminPrinter from 'imin-printer';

Vue.config.productionTip = false

Vue.use(IminPrinter)

new Vue({
 printer: new IminPrinter(),
 render: h => h(App),
}).$mount('#app')
```

3. 在 App.vue 或组件中使用

```
export default {
 name: 'App',
 data() {
  return {
   isConnect: false
  }
 },
 created() {
  this.init();
 },
 methods: {
  async init() {
   this.isConnect = await this.$printer.connect()
  },
  async getPrinterStatus() {
    return await this.$printer.getPrinterStatus()
   },
  handleClick() {
   if (!this.isConnect) return false
    console.log(this.getPrinterStatus());
  }
```

```
  },
  beforeDestroy() {
    this.isConnect = false
  }
}
```

三、 在 vue3 使用 vite 脚手架引入

1. 在 vite.config.js 中配置 imin-printer 的包

```
import { fileURLToPath, URL } from 'node:url'

import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
// https://vitejs.dev/config/
export default defineConfig({
  plugins: [
    vue(),
  ],
  resolve: {
    alias: {
      '@': fileURLToPath(new URL('./src', import.meta.url)),
      'imin-printer':      fileURLToPath(new      URL('./src/assets/imin-printer.esm.browser.min.js',
import.meta.url))
    }
  }
})
```

2. 在 main.js 中导入包

```
import IminPrinter from 'imin-printer';
import { createApp } from 'vue'
import App from './App.vue'
const app =createApp(App)
app.config.globalProperties.$printer = new IminPrinter('10.0.21.53');
app.mount('#app')
```

3. 在 App.vue 或组件中使用

```
 import { getCurrentInstance, ref } from 'vue'
const { proxy } = getCurrentInstance()
const isConnect = ref(false)
 const init = async () => {
    isConnect.value = await proxy.$printer.connect()
 }
 init();
 const getPrinterStatus = async() => {
  return await this.$printer.getPrinterStatus()
 }
 const handleClick = () => {
  if (!isConnect.value) return false
  console.log(getPrinterStatus());
 }
```

四. Api 说明

1. 初始化打印机（仅支持 SPI/USB 打印）

函数：initPrinter()

2. 获取打印机状态

函数： getPrinterStatus(IminPrintInstance.PrintConnectType, callback)

打印机状态说明：

-1 ->未连接服务

3 -> 开盖

4 -> 打印头温度过高

7 -> 缺纸

0 -> 打印机正常

示例：

```
IminPrintInstance.getPrinterStatus(IminPrintInstance.PrintConnectType.SPI, function (status) {2
console.log('printer status:' + status.value);3})
```

3. 走纸 1 行

函数：printAndLineFeed()

示例：

```
IminPrintInstance.printAndLineFeed();
```

4. 走纸自定义走纸距离

函数： printAndFeedPaper(value)

参数：value 走纸的距离范围（0<value<255）

示例：

```
IminPrintInstance.printAndFeedPaper(100);
```

5. 切纸

函数：partialCut()

示例：

```
IminPrintInstance.partialCut();
```

6. 设置对齐方式

函数：setAlignment(alignment)

参数：

alignment ->

0 = 左

1 = 中

2 =右

默认= 0

示例：

```
IminPrintInstance.setAlignment(1);
```

7. 设置打印字体大小

函数：setTextSize(size)

参数：size 默认 28

示例：

```
IminPrintInstance.setTextSize(26);
```

8. 设置打印字体

函数：setTextTypeface(typeface)

参数：typeface ->0 默认字体 1 等宽字体 2 粗体 3 无衬线字体 4 衬线字体

示例：

```
IminPrintInstance.setTextTypeface(0)
```

9. 设置字体样式

函数：setTextStyle(style)

参数：style ->0 标准正常 1 粗体 2 斜体 3 粗斜体

示例：

```
IminPrintInstance.setTextStyle(1);
```

10. 设置打印文本的行间距

函数：setTextLineSpacing(space)

参数：space 行间距 范围 0<space<255 默认 1.0f

示例：`IminPrintInstance.setTextLineSpacing(1.0f);`

11. 设置打印纸宽(可忽略调用)

函数：setTextWidth(width)

参数：width -> 576 80mm 纸宽 默认

->384 58mm 纸宽

示例：

```
IminPrintInstance.setTextWidth(576);
```

12．打印文本

函数：printText(text)

参数：text ->需要打印的文本内容

示例：

```
IminPrintInstance.printText('test print content');
```

13．打印文本添加类型

函数：printText(text, type)

参数：text -> 当打印的内容少于一行或多行时，需要在内容末尾添加换行符“n”以立即打印，否则会缓存在缓冲区中。

type -> 可以忽略不传输

注意：要更改打印文本的样式（如对齐方式、字体大小、粗体等），请在调用 printText 方法之前进行设置

示例：

```
IminPrintInstance.printText('test print content',0);
```

14：打印一行表格

函数：printColumnsText(colTextArr, colWidthArr, colAlign, width, size)

参数：colTextArr->列文本字符串数组

colWidthArr->每列宽度的数组，以英文字符计算，每个中文字符占用两个英文字符，每个宽度大于 0。

colAlign->对齐方式：0 向左，1 居中，2 向右

size->每列字符串数组的字体大小

width->打印一行的总宽度（80 毫米打印纸=576，50mm 的打印纸=384）

示例：

```
IminPrintInstance.printColumnsText(["1","iMin","iMin"],[1,2,1],[1,0,2],[26,26,26],576);
```

15.设置一维码的宽度

函数：setBarCodeWidth(int width)

参数：width->条形码宽度级别 2<=width<=6 如果未将默认条形码宽度级别设置为 3

示例：

```
IminPrintInstance.setBarCodeWidth(4);
```

16．设置一维码的高

函数：setBarCodeHeight(height)

参数：height->条形码高度 24<=高度<=250 同上，每 8 点为 1mm

示例：

```
IminPrintInstance.setBarCodeHeight(100);
```

17.一维码选择 HRI 字符的打印位置

函数：setBarCodeContentPrintPos(position)

参数：position->位置 HRI 字符打印位置

0->不打印

1->条形码上方

2->条形码下方

3->条形码打印在上面和下面

示例：

```
IminPrintInstance.setBarCodeContentPrintPos(2);
```

18 打印条形码

函数：printBarCode(barCodeType, barCodeContent) throws UnsupportedEncodingException

参数：barCodeType->条形码类型 0<=条形码类型<=6 和条形码类型=8

barCodeContent->打印的条形码字符内容，示例

示例：

```
IminPrintInstance.printBarCode(8,"0123456789");//Code128 示例
```

条形码类型（0-6,73,8）	支持的条形码内容长度	支持的 ASCII 代码范围	
0 --> UPC-A	条形码内容长度 = 11,12	48 ≤range≤ 57	所有机型机型
1 --> UPC-E	条形码内容长度 = 11,12	48 ≤range≤ 57	所有机型机型
2 --> JAN13 / EAN13	条形码内容长度= 12,13	48 ≤range≤ 57	所有机型机型
3 --> JAN8 / EAN8	条形码内容长度 = 7 ,8	48 ≤range≤ 57	所有机型机型
4 --> CODE39	条形码内容长度 >=1	48≤range≤57,65≤range≤90, range= 32, 36, 37, 42, 43, 45, 46, 47	所有机型机型
5 --> ITF	条形码内容长度 >=2	48 ≤range≤ 57	所有机型机型
6 --> CODABAR	条形码内容长度 >=2	48≤range≤57, 65≤range≤68, 97≤range≤100, range = 36, 43, 45, 46, 47, 58	Z2 系列不支持打印，其它机型支持
73 -->CODE128，8-->CODE128	条形码内容长度 >=2	0≤range≤127	所有机型机型
7 -->CODE93	1≤n≤255	1≤n≤255	

19．打印条形码并设置对齐方式

函数：printBarCode(barCodeType, barCodeContent, alignmentMode) throws UnsupportedEncodingException

参数：barCodeType->条形码类型 0<=条形码类型<=6 和条形码类型=73

barCodeContent->打印的条形码字符内容，如果是 code128 打印，则需要在前面添加｛A、｛B 或｛C，可以看到以下示例

alignmentMode->0=左/1=中/2=右

示例：

```
IminPrintInstance.printBarCode(73 ,"{B0123456789", 1);
```

20. 设置二维码的大小

函数：setQrCodeSize(level)

参数：level->二维码块大小，单位：点，1<=级别<=11

示例：

```
IminPrintInstance.setQrCodeSize(2);
```

21. 设置二维码的错误级别

函数：setQrCodeErrorCorrectionLev(level)

参数：level- >二维码的纠错级别 取值范围 0<= level<= 3

0 ->纠错级别 L(7%)

1 ->纠错级别 M(15%)

2 ->纠错级别 Q(25%)

3 ->纠错级别 H(30%)

示例：

```
IminPrintInstance.setQrCodeErrorCorrectionLev(51);
```

22. 设置左边距

函数：setLeftMargin(marginValue)

参数：marginValue->左间距值 0<marginValue<255

示例：

```
IminPrintInstance.setLeftMargin(100);
```

23. 打印二维码

函数：printQrCode(qrStr)

参数：qrStr->二维码内容

示例：

```
IminPrintInstance.printQrCode("https://www.imin.sg");
```

24. 打印二维码添加对齐方式

函数：printQrCode(qrStr, alignmentMode)

参数：qrStr->二维码内容。

alignmentMode->0=左/1=中/2=右。

示例：

```
IminPrintInstance.printQrCode("https://www.imin.sg", 1);
```

25. 设置纸张模式（不建议使用，使用会导致打印机复位）

函数：setPageFormat(style)

参数：style -> 0 80mm

1 58mm

示例：

```
IminPrintInstance.setPageFormat(1);
```

26. 打印图片

函数：printSingleBitmap(imgResources)

参数：imgResources->图片（base64 或 url）

示例：

```
1.IminPrintInstance.printSingleBitmap("...");
2.IminPrintInstance.printSingleBitmap('https://t7.baidu.com/it/u=1517419723,1472324058&fm=193&f=GIF')
```

27.开启钱箱

函数：openCashBox()

示例：

```
IminPrintInstance.openCashBox();
```

28.设置双二维码的大小

函数：setDoubleQRSize(size)

参数：size -> 1<= size <= 8

示例：

```
IminPrintInstance.setDoubleQRSize(1)
```

29. 设置双二维码 QR1 的纠错级别

函数：setDoubleQR1Level(level)

参数：level -> 1<= size <= 3

示例：

```
IminPrintInstance.setDoubleQR1Level(1)
```

30. 设置双二维码 QR2 的纠错级别

函数：setDoubleQR2Level(level)

参数：level -> 1<= size <= 3

示例：

```
IminPrintInstance.setDoubleQR2Level(1)
```

31. 设置双二维码 QR1 的左边距

函数：setDoubleQR1MarginLeft(marginValue)

参数：level -> 0<level<255

示例：

```
IminPrintInstance.setDoubleQR1Level(16)
```

32. 设置双二维码 QR2 的左边距

函数：setDoubleQR2MarginLeft(marginValue)

参数：level -> 0<level<255

示例：

```
IminPrintInstance.setDoubleQR2Level(200)
```

33. 设置双二维码 QR1 的版本

函数：setDoubleQR1Version(version)

参数：version-> 0<=version<=40

示例：

```
IminPrintInstance.setDoubleQR1Version(40)
```

34. 设置双二维码 QR2 的版本

函数：setDoubleQR2Version(version)

参数：version-> 0<=version<=40

示例：

```
IminPrintInstance.setDoubleQR2Version(40)
```

35. 打印双二维码

函数：printDoubleQR(colTextArr)

参数：colTextArr->列文本字符串数组

示例：

```
IminPrintInstance.printDoubleQR(["www.iMin.sg", "www.google.com"]);
```