

ESC SDK v1.0.0接口文档

文档修改记录

一.使用说明

1.1工程配置说明

1.2混淆配置

1.3初始化

二.样例演示

2.1连接打印机流程

2.1.1蓝牙搜索打印机

2.1.2连接打印机

三.接口说明

PrinterHelper引入

3.1.2蓝牙连接

3.1.3WIFI连接

3.1.4USB连接

3.1.5串口连接

3.1.6连接状态

3.1.7断开连接

3.2.1获取打印机标记

3.2.2设置SDK字符编码

3.2.3将打印数据发送给打印机

3.2.4添加数据给打印机

3.2.5从打印机中读取数据

3.2.6添加文本

3.2.7添加表格类型的文本数据

3.2.8添加对齐方式

3.2.9添加字体样式

3.2.10字体放大

3.2.11添加图片

3.2.12添加条码

3.2.13添加二维码

3.2.14添加PDF417

3.2.15添加页模式

3.2.16添加清除页模式缓存数据

3.2.17添加页模式打印区域

3.2.18添加页模式打印方向

3.2.19添加页模式绝对位置

3.2.20添加页模式打印

3.2.21添加定位

3.2.22添加走纸

3.2.23添加行间距

3.2.24添加打印机编码

3.2.25初始化打印机

3.2.25添加打印机浓度

3.2.26添加打印速度

3.2.27添加走纸切刀

3.2.28添加打开钱箱

3.2.29添加开启蜂鸣器

3.2.30添加左边距

3.2.31获取打印机状态

3.2.32获取打印机序列号

3.2.33获取打印机电量

3.2.34设置打印结束开关

3.2.35获取打印结束

4.1.1编码表

4.1.2条码类型

4.1.3PDF417纠错等级

文档修改记录

序号	版本号	修改内容	修改者	修改日期
01	v1.0.0	文档建立	邱贤	2024-11-15

一.使用说明

1.1工程配置说明

- 1.本SDK接口需运行在Android系统上
- 2.把ESC_SDK_Vx.x.x.jar和xxxx.so放到工程libs目录， build.gradle增加引入jar

▼ build.gradle中配置

Java

1 implementation fileTree(include: ['*.jar','*.aar'], dir: 'libs')

3.蓝牙权限配置

▼ 在AndroidManifest.xml中配置

XML

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

3 <uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
4 <uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
5 <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE"/>
6 </>

▼ 在代码中动态配置定位权限

Kotlin

1 disposable = RxPermissions(this).request(
2 Manifest.permission.BLUETOOTH,
3 Manifest.permission.ACCESS_FINE_LOCATION,
4 Manifest.permission.ACCESS_COARSE_LOCATION
5)
6 .subscribe {
7 if (it) {
8 }}
9 }

1.2混淆配置

1.需在proguard-rules.pro中配置一下代码防止SDK被混淆

▼ 不混淆下面的类

Kotlin |

```
1 -keep class com.prt.esc.PrinterHelper{*;}
2 -keep class com.prt.esc.printer.Printer{*};
```

1.3初始化

1.在Application中调用下面代码

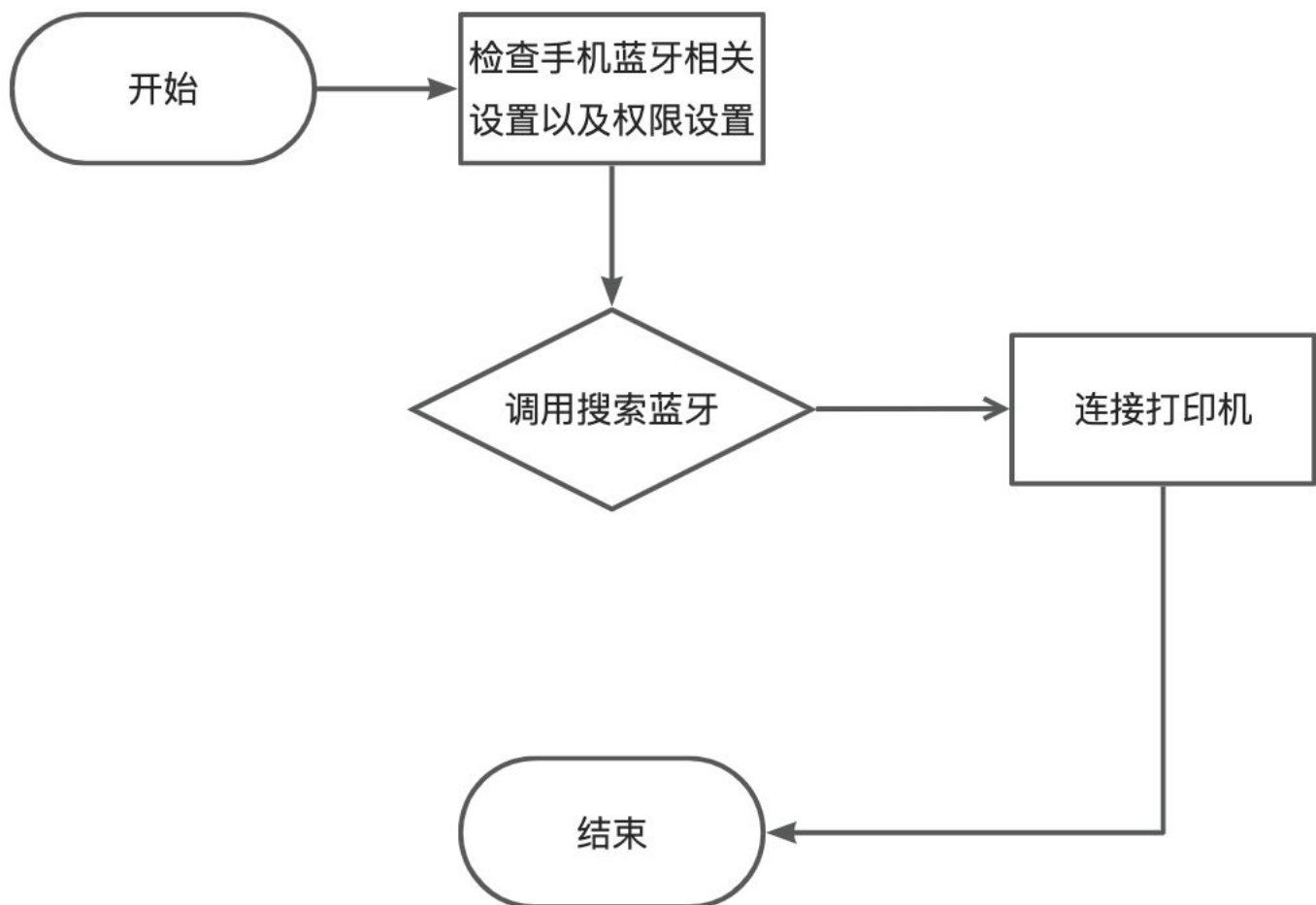
▼ 初始化

Kotlin |

```
1 PrinterHelper.init(this);
```

二.样例演示

2.1连接打印机流程



2.1.1 蓝牙搜索打印机

注册蓝牙广播

Java

```
1 private void registerBroadcast() {
2     IntentFilter intent = new IntentFilter();
3     intent.addAction(BluetoothDevice.ACTION_FOUND);
4     intent.addAction(BluetoothDevice.ACTION_BOND_STATE_CHANGED);
5     intent.addAction(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
6     context.registerReceiver(mReceiver, intent);
7 }
8
```

搜索蓝牙

Java

```
1 RxPermissions rxPermissions = new RxPermissions((AppCompatActivity) context);
2 rxPermissions.request(Manifest.permission.BLUETOOTH_ADMIN,
3     Manifest.permission.BLUETOOTH,
4     Manifest.permission.ACCESS_FINE_LOCATION)
5 .subscribe(aBoolean -> {
6     if (aBoolean) {
7         if (null == mBluetoothAdapter) {
8             mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
9         }
10        if (!mBluetoothAdapter.isEnabled()) {
11            mBluetoothAdapter.enable();
12        }
13        if (mBluetoothAdapter.isDiscovering()) {
14            mBluetoothAdapter.cancelDiscovery();
15        }
16        mBluetoothAdapter.startDiscovery();//开启搜索
17    } else {
18        Utility.show(context, "no bluetooth permission");
19    }
20 });
```

▼ 接受广播

Java

```
1 private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
2     @SuppressWarnings("MissingPermission")
3     @Override
4     public void onReceive(Context context, Intent intent) {
5         String action = intent.getAction();
6         if (BluetoothDevice.ACTION_FOUND.equals(action)) {
7             BluetoothDevice device =
8                 intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
9             if (device.getBluetoothClass().getMajorDeviceClass() == 1536)
10            {
11                //代表是经典蓝牙协议的打印机
12            }
13            break;
14        }
15    };
```

2.1.2连接打印机

▼ 蓝牙连接

Kotlin

```
1 Printer printer = PrinterHelper.connectBT(btMac);
```

三.接口说明

PrinterHelper引入

在APP启动时初始化PrinterHelper

▼ 初始化代码

Java

```
1 public class App extends Application {
2     @Override
3     public void onCreate() {
4         super.onCreate();
5         PrinterHelper.init(this);
6     }
7 }
```

3.1.2 蓝牙连接

- 注意权限配置

▼ 蓝牙权限配置 Java

```
1  蓝牙连接需要配置权限，在AndroidManifest.xml文件中添加
2  <uses-permission android:name="android.permission.BLUETOOTH" />
3  <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
4  <uses-permission android:name="android.permission.BLUETOOTH_CONNECT" />
5  <uses-permission android:name="android.permission.BLUETOOTH_SCAN" />
6  <uses-permission android:name="android.permission.BLUETOOTH_ADVERTISE" />
7
8  然后在代码中动态获取权限
9  RxPermissions rxPermissions = new RxPermissions(this);
10 rxPermissions.request(Manifest.permission.BLUETOOTH_ADMIN,
11                      Manifest.permission.BLUETOOTH,
12                      Manifest.permission.BLUETOOTH_CONNECT,
13                      Manifest.permission.BLUETOOTH_SCAN,
14                      Manifest.permission.ACCESS_FINE_LOCATION)
15  .subscribe(aBoolean -> {
16      if (aBoolean) {
17          //权限获取成功
18      }
19  });
```

- 描述
通过蓝牙mac地址连接打印机

▼ 蓝牙连接 Java

```
1  public static Printer connectBT(String mac)
```

- 参数

参数	描述
mac	蓝牙地址（大写）

- 例子

- ▼ 蓝牙连接示例

Java |

```
1 Printer printer = PrinterHelper.connectBT(btMac);
2 if(printer == null){
3     //连接失败
4 }else{
5     //连接成功，所有打印相关的接口都在Printer类里面
6 }
```

3.1.3WIFI连接

- 注意配置权限

- ▼ wifi权限配置

Java |

```
1 wifi连接需要配置权限，在AndroidManifest.xml文件中添加
2 <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
3 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
4 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
5 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
6 <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"
7     />
8 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
9 在代码中需要获取动态权限
10 RxPermissions rxPermissions = new RxPermissions(this);
11 rxPermissions.request(Manifest.permission.ACCESS_COARSE_LOCATION,
12     Manifest.permission.ACCESS_FINE_LOCATION)
13 .subscribe(aBoolean -> {
14     if (aBoolean) {
15         //获取权限成功
16     }
17 });
```

- 描述

通过打印机的IP地址连接

- ▼ wifi连接接口

Java |

```
1 public static Printer connectWifi(String address)
```

- 参数

参数	描述
address	打印机所在的IP地址，（例：192.168.1.100）

- 例子

▼ wifi连接示例

Java

```
1 Printer printer = PrinterHelper.connectWifi(ip);
2 if(printer == null){
3     //连接失败
4 }else{
5     //连接成功，所有打印相关的接口都在Printer类里面
6 }
```

3.1.4USB连接

- 注意权限配置

▼ USB权限配置

Java

```
1 wifi连接需要配置权限，在AndroidManifest.xml文件中添加
2 <uses-feature android:name="android.hardware.usb.host" />
3 <uses-permission android:name="android.permission.USB_PERMISSION" />
4 <uses-permission android:name="android.permission.ACCESS_USB_DEVICE" />
5 <meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
```

- 描述

通过usbdevice连接打印机

▼ USB连接接口

Java

```
1 public static Printer connectUSB(UsbDevice usbdevice)
```

- 参数

参数	描述
usbdevice	android 系统提供的usb设备类

- 例子

▼ USB连接示例

Java |

```

1  Printer printer = PrinterHelper.connectUSB(device);
2  if(printer == null){
3      //连接失败
4  }else{
5      //连接成功，所有打印相关的接口都在Printer类里面
6  }

```

▼ 获取USB设备

Java |

```

1  //device:可以参考Demo获取，也可以参考下面的代码
2  private void connectUSB() { //遍历系统中所有的USB设备
3      mUsbManager = (UsbManager) thisCon.getSystemService(Context.USB_SERVICE);
4      HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();
5      Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();
6
7      boolean HavePrinter = false;
8      while (deviceIterator.hasNext()) {
9          device = deviceIterator.next();
10         int count = device.getInterfaceCount();
11         for (int i = 0; i < count; i++) {
12             UsbInterface intf = device.getInterface(i);
13             if (intf.getInterfaceClass() == 7) { //代表是打印机类型
14                 HavePrinter = true;
15                 if (mPermissionIntent != null) {
16                     //申请USB权限
17                     mUsbManager.requestPermission(device, mPermissionIntent);
18                 }
19             }
20         }
21     }
22 }

```

▼ 注册USB系统权限广播

Java

```
1 Intent intent = new Intent(ACTION_USB_PERMISSION);
2 intent.setPackage(thisCon.getPackageName());
3 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
4     mPermissionIntent = PendingIntent.getBroadcast(thisCon, 0, intent, FLAG_MUTABLE);
5 } else {
6     mPermissionIntent = PendingIntent.getBroadcast(thisCon, 0, intent, 0);
7 }
8 IntentFilter filter = new IntentFilter(ACTION_USB_PERMISSION);
9 filter.addAction(UsbManager.ACTION_USB_DEVICE_DETACHED);
10 filter.addAction(BluetoothDevice.ACTION_ACL_DISCONNECTED);
11 filter.addAction(BluetoothAdapter.ACTION_STATE_CHANGED);
12 if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
13     thisCon.registerReceiver(mUsbReceiver, filter, RECEIVER_EXPORTED);
14 } else {
15     thisCon.registerReceiver(mUsbReceiver, filter);
16 }
```

▼ 接收广播

Java

```
1 private BroadcastReceiver mUsbReceiver = new BroadcastReceiver() {
2     public void onReceive(Context context, Intent intent) {
3         String action = intent.getAction();
4         if (ACTION_USB_PERMISSION.equals(action)) {
5             synchronized (this) {
6                 UsbDevice device =
7                     (UsbDevice) intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
8                 boolean result =
9                     intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)
10                 if (result) {
11                     Printer printer = PrinterHelper.connectUSB(device);
12                     if (printer == null) {
13                         //连接失败
14                         return;
15                     } else {
16                         //连接成功
17                     }
18                 } else {
19                     return;
20                 }
21             }
22         }
23     }
24 };
```

3.1.5串口连接

- 描述
通过连接节点和波特率串口连接打印机

▼ 串口连接接口 Java |

```
1 public static Printer connectSerial(String node, int baudRate)
```

- 参数

参数	描述
node	连接节点（例： /dev/ttyS1）
baudRate	波特率（例： 9600）

- 例子

▼ 串口连接示例 Java |

```
1 Printer printer = PrinterHelper.connectSerial(port, Integer.valueOf(baudRate));
2 if(printer == null){
3     //连接失败
4 }else{
5     //连接成功，所有打印相关的接口都在Printer类里面
6 }
```

3.1.6连接状态

- 描述
判断打印机是否已连接

▼ 连接状态接口 Java |

```
1 public boolean isConnect()
2 //该接口不能获取实时的状态，蓝牙可以通过系统蓝牙广播来获取，参考Demo
```

- 例子

▼ 连接状态示例

Java |

```
1 printer.isConnect()
2 //true: 已连接, false: 未连接
3 //printer是通过连接接口返回的
```

3.1.7断开连接

- 描述

断开已连接的设备

▼ 断开连接接口

Java |

```
1 public boolean closeOperator()
```

- 例子

▼ 断开接口示例

Java |

```
1 printer.closeOperator()
2 //printer是通过连接接口返回的
```

3.2.1获取打印机标记

- 描述

获取已连接打印机的一些标识，蓝牙：返回的是蓝牙地址，WIFI：返回的是IP地址，USB：返回的是设备节点，串口：返回的是串口节点

▼ 获取打印机标记接口

Java |

```
1 public String getPrinterTAG()
```

- 参数

无

- 例子

▼ 获取打印机标记示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.getPrinterTAG()
```

3.2.2设置SDK字符编码

- 描述
设置SDK所需的字符编码，例如中文设置GB2312

▼ 设置SDK字符编码接口 Java |

```
1 public void setLanguageEncode(String languageEncode)
```

- 参数

参数	描述
languageEncode	编码（例：GBK）详情可以查看4.1.1

- 例子

▼ 设置SDK字符编码示例 Java |

```
1 //printer是通过连接接口返回的
2 printer.setLanguageEncode(sLEncode);
```

3.2.3将打印数据发送给打印机

- 描述
将储存在Printer类中的数据发送给打印机

▼ 发送打印数据接口 Java |

```
1 public boolean print()
```

- 参数
无
- 例子

▼ 发送打印数据示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addText("text",1, false, false, false, false, 1, 1);
3 boolean result = printer.print();
4 //将printer中储存的数据发送给打印
5 //false: 发送失败, true: 发送成功
```

3.2.4添加数据给打印机

- 描述

将数据添加进Printer类中，通过print接口发送给打印机

▼ 添加数据接口

Java |

```
1 public void addData(byte[] data)//添加byte数组类型的数据给打印机
2 public void addData(String data)//添加字符串数据给打印机
```

- 参数

参数	描述
data	需要发送给打印机的数据

- 例子

▼ 添加数据示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addData(new byte[]{0x0d,0x0a});
3 printer.addData("test");
4 printer.print();
```

3.2.5从打印机中读取数据

- 描述

读取打印机返回给APP的数据,如果在超时间内没有读取到数据，将会返回长度为0的byte数组

▼ 读取数据接口

Kotlin

```
1 public byte[] readData(long timeout)
```

• 参数

参数	描述
timeout	超时时间（毫秒）

• 例子

▼ 读取数据示例

Java

```
1 //printer是通过连接接口返回的
2 byte[] data = printer.readData(2000)
3 //data:从打印机中读取到的数据
```

3.2.6 添加文本

• 描述

向Printer类添加文本数据，该接口可以设置对齐方式，字体样式，字体缩放，通过print接口发送给打印机

▼ 添加文本接口

Java

```
1 public void addText(String data, int alignment, boolean isFontSmall,
2                     boolean isBold, boolean isUnderline,
3                     boolean isTurnWhite, int widthMultiplier,
4                     int heightMultiplier)
```

• 参数

参数	描述
data	文本数据
alignment	对齐方式（0：左对齐，1：居中，2：右对齐）
isFontSmall	是否使用小字体（小字体：9X17，大字体：12X24）

isBold	是否加粗
isUnderline	是否使用下划线
isTurnWhite	是否反白
widthMultiplier	字体横向放大倍数（0-8）最大倍数跟随打印机
heightMultiplier	字体纵向放大倍数（0-8）最大倍数跟随打印机

- 例子

▼ 添加文本示例 Java

```
1 //printer是通过连接接口返回的，将文本数据添加进Printer里
2 printer.addText("text",1, false, false, false, false, 1, 1);
3 boolean result = printer.print();//发送数据
```

3.2.7添加表格类型的文本数据

- 描述

该接口也是添加文本的一种方式，该样式更像表格。

▼ 添加表格文本接口 Java

```
1 public void addTable(Table table)
2
3 public class Table { //表格类
4     //column: 标题栏数据（例：品名;数量;单价;金额）数据都用";"做分隔
5     //regularExpression: 分隔符（例：;）
6     //columnWidth: 每一列的字节长度，每个中文是2个字节长度，每个数字和字母是1个字节长度，
7     //例（new int[]{6, 6, 6, 6}）6表示可以放得下3个中文或者6个字母
8     public Table(String column, String regularExpression, int[] columnWidth)
9     //设置每列对应的对齐方式（0：左对齐，1：居中，2：右对齐）
10    public void setColumnAlign(int align)
11    //添加没一行的数据（例：保鲜袋;10.00;1;10.00）
12    public void addRow(String row)
13 }
```

- 参数

参数	描述
table	表格类对象

- 例子

添加表格文本示例

Java

```
1 //printer是通过连接接口返回的
2 String column = "品名;数量;单价;金额";
3 Table table = new Table(column, ";", new int[]{6, 6, 6, 6});
4 table.setColumnAlign(1);
5 table.addRow("保鲜袋" + ";10.00;1;10.00");
6 table.addRow("铁丝挂钩" + ";5.00;2;10.00");
7 table.addRow("雨伞" + ";5.00;3;15.00");
8 printer.addTable(table);
9 boolean result = printer.print();//将数据发送给打印机
```

3.2.8添加对齐方式

- 描述

该接口用于设置整体的对齐方式，但是对于一些接口本身有带设置对齐方式的无效（例如：`addText`,`addImage`,`addBarCode`,`addQRCode`等）

添加对齐方式接口

Kotlin

```
1 public void addAlignment(int alignment)
```

- 参数

参数	描述
alignment	对齐方式（0：左对齐，1：居中，2：右对齐）

- 例子

添加对齐方式示例

Java

```
1 //printer是通过连接接口返回的
2 printer.addAlignment(1)
3 printer.writeData("test");//将文本test居中打印
4 printer.print()
```

3.2.9添加字体样式

- 描述
该接口用于给文本数据添加字体样式，但是对于addText接口无效，addText自身已有设置字体样式的参数

▼ 添加字体样式接口 Java

```
1 public void addFontStyle(boolean isFontSmall, boolean isBold,
2                          boolean isUnderline)
```

- 参数

参数	描述
isFontSmall	是否使用小字体 （小字体： 9X17, 大字体： 12X24）
isBold	是否加粗
isUnderline	是否使用下划线

- 例子

▼ 添加字体样式示例 Java

```
1 //printer是通过连接接口返回的
2 printer.addFontStyle(true, true, true);
3 printer.addData("test");
4 printer.print();
```

3.2.10字体放大

- 描述
该接口用于给文本数据进行字体的放大，但是对于addText接口无效，addText自身已有设置字体放大的参数

▼ 字体放大接口 Java

```
1 public void addFontMultiplier(int widthMultiplier, int heightMultiplier)
```

- 参数

参数	描述
widthMultiplier	字体横向放大倍数（0–8）最大倍数跟随打印机
heightMultiplier	字体纵向放大倍数（0–8）最大倍数跟随打印机

- 例子

▼ 字体放大示例	Java
<pre> 1 //printer是通过连接接口返回的 2 printer.addFontMultiplier(1, 1); 3 printer.addData("test"); 4 printer.print(); </pre>	

3.2.11添加图片

- 描述

该接口可以添加图片打印，可以设置对齐方式，图片算法，需要打印的宽高，是否对数据压缩
 200dpi的打印机 8点 = 1毫米
 300dpi的打印机 11.8点 = 1毫米

▼ 添加图片接口	Java
<pre> 1 public void addImage(int alignment, int type, int imageWidth, 2 int imageHeight, boolean isCompress, Bitmap bitmap) </pre>	

- 参数

参数	描述
alignment	对齐方式（0：左对齐，1：居中，2：右对齐）
type	图片算法（0：二值，1：抖动）
imageWidth	需要打印的宽度（单位：点）
imageHeight	需要打印的高度（单位：点）

isCompress	是否压缩（部分打印机支持）
bitmap	图片

- 例子

▼ 打印图片示例 Java

```
1 //printer是通过连接接口返回的
2 printer.addImage(0, type, bitmapPrint.getWidth(), bitmapPrint.getHeight(),
  isLzo, bitmapPrint);
3 boolean result = printer.print();
```

3.2.12 添加条码

- 描述

该接口用于打印条码，可以打印 UPC-A，UPC-E，JAN13/EAN13，JAN8/EAN8，CODE39，ITF，CODABAR(NW-7)，CODE93，CODE128类型的条码

▼ 添加条码接口 Java

```
1 public void addBarCode(int bcType, String bcData, int width, int height,
2                        int hriPosition, int alignment)
```

- 参数

参数	描述
bcType	条码类型（详情查看表4.1.2）
bcData	条码内容
width	条码宽度（1-6等级）
height	条码高度（1-255等级）
hriPosition	文本位置 (0: 不打印, 1: 条码上方, 2: 条码下方, 3: 条码上方及下方)
alignment	对齐方式（0: 左对齐, 1: 居中, 2: 右对齐）

- 例子

打印条码示例

Java

```
1 //printer是通过连接接口返回的
2 printer.addBarCode(65, "12345678901", 2, 50, 2, 1);
3 boolean result = printer.print();
```

3.2.13 添加二维码

- 描述

该接口用于打印二维码

添加二维码接口

Java

```
1 public void addQRCode(String qrData, int sizeOfModule, int errorLevel,
2                        int alignment)
```

- 参数

参数	描述
qrData	二维码内容
sizeOfModule	尺寸（1–16等级）
errorLevel	纠错等级（48–51，对应 L，M，Q，R）
alignment	对齐方式（0：左对齐，1：居中，2：右对齐）

- 例子

打印二维码示例

Java

```
1 //printer是通过连接接口返回的
2 printer.addQRCode("Test print QRCode", 6, 48, 1);
3 boolean result = printer.print();
```

3.2.14 添加PDF417

- 描述

该接口用于打印PDF417码

▼ 添加PDF417接口

Java

```
1 public void addPDF417(String data, int dataColumns, int dataRows,  
2                       int moduleWidth,int rowHeight, int errorMode,  
3                       int errorLevel, int options)
```

• 参数

参数	描述
data	数据内容
dataColumns	设置列数（0–30） 0：自动处理，根据数据来确定列数
dataRows	设置行数（0，3–90） 0：自动处理，根据数据来确定行数
moduleWidth	设置模块宽度（2–8）
rowHeight	设置模块高度（2–8）
errorMode	纠错模式（48：等级模式，49：比率模式）
errorLevel	纠错等级（详情查看4.1.3）
options	模式（0：标准模式，1：压缩模式）

• 例子

▼ 打印PDF417示例

Java

```
1 //printer是通过连接接口返回的  
2 printer.addPDF417("123456",0, 0, 3, 3, 49, 1, 0);  
3 boolean result = printer.print();
```

3.2.15添加页模式

• 描述

该接口是让打印机进入页模式



Java

```
1 public void addSelectPageMode()
```

- 参数

无

- 例子

▼ 页模式打印示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.16 添加清除页模式缓存数据

- 描述

该接口可以清除上一次的页模式缓存的数据，最好是在为创建页模式打印区域前使用

▼ 清除页模式缓存数据接口

Java |

```
1 public void addClearPageModePrintAreaData()
```

- 例子

▼ 页模式打印示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.17添加页模式打印区域

- 描述
该接口用于创建页模式的画布，可以设置起始位置和宽高

▼ 页模式打印区域接口 Java

```
1 public void addPageModePrintArea(int horizontal, int vertical, int width,
2                                 int height)
```

- 参数

参数	描述
horizontal	起始点横坐标（单位：点）
vertical	起始点纵坐标（单位：点）
width	区域宽度（单位：点）
height	区域高度（单位：点）

- 例子

▼ 页模式打印示例 Java

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.18添加页模式打印方向

- 描述
该接口用于设置页模式里面的打印方向

▼ 页模式打印方向接口

Java |

```
1 public void addPageModePrintDirection(int direction)
```

• 参数

参数	描述
direction	方向 (0: 从左到右, 1: 从下到上, 2: 从右到左, 3: 从上到下)

• 例子

▼ 页模式打印示例

Kotlin |

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.19 添加页模式绝对位置

• 描述

该接口用于页模式下设置打印点的绝对位置，设置完该接口后，再跟随一个打印内容，那么该内容就可以在这个位置上打印。

▼ 添加页模式绝对位置接口

Java |

```
1 public void addPageModeAbsolutePosition(int xPosPosition, int yPosPosition)
```

• 参数

参数	描述
xPosition	打印横坐标

yPosition

打印纵坐标

- 例子

- ▼ 页模式打印示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.20添加页模式打印

- 描述

该接口用于让打印机知道打印页模式下储存的数据

- ▼ 添加页模式打印接口

Java |

```
1 public void addPrintPageModeData()
```

- 参数

无

- 例子

- ▼ 页模式打印示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addSelectPageMode();//进入页模式
3 printer.addClearPageModePrintAreaData();//清除页面模式缓存数据
4 printer.addPageModePrintArea(0,0,200,200);//设置打印区域
5 printer.addPageModePrintDirection(0);//设置打印方向
6 printer.addPageModeAbsolutePosition(0,0);//设置打印起始坐标
7 printer.addQRCode("Test print QRCode",6, 48, 1);//打印二维码
8 printer.addPrintPageModeData();//设置打印
9 printer.print();//发送给打印机
```

3.2.21添加定位

- 描述
该接口可以在打印标签或者黑标时用于定位功能

▼ 添加定位接口 Java |

```
1 public void addPositioning()
```

- 参数
无

- 例子

▼ 定位示例 Java |

```
1 //printer是通过连接接口返回的
2 printer.addPositioning();
3 printer.print();
```

3.2.22添加走纸

- 描述
该接口可以让打印机按你设置的行数走纸

▼ 添加走纸接口 Java |

```
1 public void addFeedLine(int lines)
```

- 参数

参数	描述
lines	走纸行数

- 例子

▼ 走纸示例

Java

```
1 //printer是通过连接接口返回的
2 printer.addFeedLine(100);
3 printer.print();
```

3.2.23 添加行间距

- 描述

该接口用于设置行间距，单位是行

▼ 添加行间距接口

Java

```
1 public void addLineSpace(int lineSpace)
```

- 参数

参数	描述
lineSpace	行间距（单位：行）

- 例子

▼ 设置行间距示例

Java

```
1 //printer是通过连接接口返回的
2 printer.addFeedLine(2);
3 printer.print();
```

3.2.24 添加打印机编码

- 描述

该接口可以用于设置打印机编码

▼ 添加打印机编码接口

Java

```
1 public void addPrinterCharacter(int character)
```

- 参数

参数	描述
character	打印机编码代号（详情查看4.1.1）

- 例子

▼ 设置打印机编码示例	Java
<pre>1 //printer是通过连接接口返回的 2 printer.addPrinterCharacter(1); 3 printer.print();</pre>	

3.2.25初始化打印机

- 描述

该接口可以初始化打印机，清除字体样式，对齐方式等等。

▼ 初始化打印机接口	Java
<pre>1 public void addInitialize()</pre>	

- 参数

无

- 例子

▼ 初始化打印机示例	Kotlin
<pre>1 //printer是通过连接接口返回的 2 printer.addInitialize(); 3 printer.print();</pre>	

3.2.25添加打印机浓度

- 描述

用于添加打印机浓度，但是不同的机型会有不同的浓度范围，详情咨询客服

▼ 添加打印机浓度接口

Java |

```
1 public void addPrintDensity(int density)
```

• 参数

参数	描述
density	打印浓度，范围根据机型而定

• 例子

▼ 设置浓度示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addPrintDensity(0);
3 printer.print();
```

3.2.26 添加打印速度

• 描述

用于添加打印速度，但是不同的机型会有不同的速度范围，详情咨询客服

▼ 添加打印速度接口

Java |

```
1 public void addPrintSpeed(int speed)
```

• 参数

参数	描述
speed	打印速度，范围根据机型而定

• 例子

▼ 设置打印速度示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addPrintSpeed(0);
3 printer.print();
```

3.2.27 添加走纸切刀

• 描述

该接口可以用于走纸后切刀

▼ 添加走纸切刀接口

Java |

```
1 public void addCutterPaperFeeding(int cutMode, int distance)
```

• 参数

参数	描述
cutMode	切刀模式（0：半切，1：全切）
distance	走纸距离（0–255 单位点）

• 例子

▼ 设置走纸切刀示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addCutterPaperFeeding(0, 100);
3 printer.print();
```

3.2.28 添加打开钱箱

• 描述

该接口用于打开跟打印机连接的钱箱

▼ 添加打开钱箱接口

Java |

```
1 public void addOpenCashDrawer(int openMode)
```

- 参数

参数	描述
openMode	0: 打开1号钱箱, 1: 打开2号钱箱, 2: 打开全部钱箱

- 例子

▼ 打开钱箱示例	Java
<pre>1 //printer是通过连接接口返回的 2 printer.addOpenCashDrawer(0); 3 printer.print();</pre>	

3.2.29添加开启蜂鸣器

- 描述

该接口用于让蜂鸣器鸣叫

▼ 添加开启蜂鸣器接口	Java
<pre>1 public void addBeepBuzzer(int times, int t1, int t2)</pre>	

- 参数

参数	描述
times	蜂鸣器响的次数
t1	响的时间 (单位 100ms)
t2	停止时间 (单位 100ms)

- 例子

▼ 开启蜂鸣器示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addBeepBuzzer(1, 2, 2);
3 printer.print();
```

3.2.30 添加左边距

- 描述

该接口可以设置打印内容的左边距

▼ 添加左边距接口

Java |

```
1 public void addLeftMargin(int margin)
```

- 参数

参数	描述
margin	左边距（单位：点）

- 例子

▼ 设置左边距示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.addLeftMargin(100);
3 printer.addText("text",1, false, false, false, false, 1, 1);
4 printer.print();
```

3.2.31 获取打印机状态

- 描述

该接口可以获取打印机的开合盖和纸张状态

▼ 获取打印机状态接口

Java |

```
1 public int getPrinterStatus(int model)
```

- 参数

参数	描述
model	2：获取开合盖状态，4：获取纸张状态

- 返回

参数	描述
-1	发送失败
其他	打印机状态

- 例子

▼ 获取打印机状态示例

Java |

```
1 //printer是通过连接接口返回的
2 int status = printer.getPrinterStatus(2);
3 if ((status & 4) == 4) {
4     //开盖
5 } else {
6     //合盖
7 }
8 status = printer.getPrinterStatus(4);
9 if ((status & 96) == 96) {
10     //无纸
11 } else {
12     //有纸
13 }
14 if ((status & 12) == 12) {
15     //纸将尽
16 } else {
17     //纸充足
18 }
```

3.2.32 获取打印机序列号

- 描述

该接口可以用于获取打印机的序列号

▼ 获取打印机序列号接口

Java |

```
1 public String getPrinterSN()
```

- 参数
无
- 例子

▼ 获取打印机序列号示例

Java |

```
1 //printer是通过连接接口返回的
2 String strSN = printer.getPrinterSN();
```

3.2.33 获取打印机电量

- 描述
该接口获取打印机电量百分比，该接口只适用于部分带电池的打印机

▼ 获取打印机电量接口

Java |

```
1 public int getPrinterQuantity()
```

- 参数
无
- 返回

参数	描述
-1	发送失败（仅有部分打印机支持）
其他	电量

- 例子

▼ 获取打印机电量示例

Java |

```
1 //printer是通过连接接口返回的
2 int quantity = printer.getPrinterQuantity();
```

3.2.34 设置打印结束开关

- 描述

该接口用于打印完成时是否需要回传app的开关设置，仅部分打印机支持

▼ 设置打印结束开关接口

Java |

```
1 public boolean setPrintEndSwitch(boolean isOpen)
```

- 参数

参数	描述
isOpen	true:开启，false:关闭

- 例子

▼ 开启打印完成时回传数据示例

Java |

```
1 //printer是通过连接接口返回的
2 printer.setPrintEndSwitch(true);
3 printer.addText("Test print", 1, false, false, false, false, 0, 0);
4 printer.addQRCode("Test print QRCode", 6, 48, 1);
5 printer.addCutterPaperFeeding(0, 100);
6 boolean result = printer.print();
7 int printResult = printer.getPrintResult(3000);
```

3.2.35 获取打印结束

- 描述

该接口用于接收打印完成时，打印机回传给app的响应，首先需要接口setPrintEndSwitch设置成true，该接口仅有部分打印机支持

▼ 接收打印完成时接口 Java

```
1 public int getPrintResult(long outTime)
```

• 参数

参数	描述
outTime	获取超时时间

• 返回

参数	描述
-1	发送失败
-2	获取超时（仅有部分打印机支持）
0	打印完成

• 例子

▼ 开启打印完成时回传数据示例 Java

```
1 //printer是通过连接接口返回的
2 printer.setPrintEndSwitch(true);
3 printer.addText("Test print", 1, false, false, false, false, 0, 0);
4 printer.addQRCode("Test print QRCode",6, 48, 1);
5 printer.addCutterPaperFeeding(0, 100);
6 boolean result = printer.print();
7 int printResult = printer.getPrintResult(3000);
```

4.1.1编码表

名称	打印机编码代号	SDK编码
Default	0	gb2312
Chinese Simplified	0	gb2312

Chinese Traditional	0	big5
PC437(USA)	0	iso8859-1
KataKana	1	Shift_JIS
PC850(Multilingual)	2	iso8859-3
PC860(Portuguese)	3	iso8859-6
PC863(Canadian- French)	4	iso8859-1
PC865(Nordic)	5	iso8859-1
PC857(Turkish)	13	IBM857
PC737(Greek)	14	iso8859-7
ISO8859-7(Greek)	15	iso8859-7
WCP1252	16	iso8859-1
PC866(Cyrillic #2)	17	iso8859-5
PC852(Latin 2)	18	iso8859-2
PC858(Euro)	19	iso8859-15
KU42	20	ISO8859-11
TIS11(Thai)	21	ISO8859-11
TIS18(Thai)	26	ISO8859-11
PC720	32	iso8859-6
WPC775	33	iso8859-1
PC855(Cyrillic)	34	iso8859-5
PC862(Hebrew)	36	iso8859-8

PC864(Arabic)	37	iso8859-6
ISO8859-2(Latin2)	39	iso8859-2
ISO8859-15(Latin9)	40	iso8859-15
WPC1250	45	iso8859-2
WPC1251(Cyrillic)	46	iso8859-5
WPC1253	47	iso8859-7
WPC1254	48	iso8859-3
WPC1255	49	iso8859-8
WPC1256	50	Windows-1256
WPC1257	51	iso8859-1
WPC1258	52	bg2312
MIK(Cyrillic/Bulgarian)	54	iso8859-15
CP755	55	iso8859-5
Iran	56	iso8859-6
Iran II	57	iso8859-6
Latvian	58	iso8859-4
ISO-8859-1(West Europe)	59	iso8859-1
ISO-8859-3(Latin 3)	60	iso8859-3
ISO-8859-4(Baltic)	61	iso8859-4
ISO-8859-5(Cyrillic)	62	iso8859-5
ISO-8859-6(Arabic)	63	iso8859-6

ISO-8859-8(Hebrew)	64	iso8859-8
ISO-8859-9(Turkish)	65	iso8859-9
PC856	66	iso8859-8
ABICOIM	67	iso8859-15

4.1.2条码类型

bcType	名称	长度	数据范围
65	UPC-A	11-12	数字
66	UPC-E	11-12	数字（第一个数字必须是0）
67	JAN13/EAN13	12-13	数字
68	JAN8/EAN8	7-8	数字
69	CODE39	1-255	数字，字母，部分符号
70	ITF	2-254	数字
71	CODABAR(NW-7)	2-255	数字，部分字母，部分符号（起始和结束必须是A,B,C,D）
72	CODE93	1-255	数字，字母，符号
73	CODE128	2-255	数字，字母，符号（先选择字符集,{A: ASCII 字符 00H 到 5FH {B: ASCII 字符 20H 到 7FH {C: 00-99的100个数字）

4.1.3PDF417纠错等级

值	模式	等级	纠错码字数量
n = 48	等级模式	0	2
n = 49	等级模式	1	4
n = 50	等级模式	2	8

n = 51	等级模式	3	16
n = 52	等级模式	4	32
n = 53	等级模式	5	64
n = 54	等级模式	6	128
n = 55	等级模式	7	256
n = 56	等级模式	8	512
数据码字 $\times n \times 0.1 = (0-3)$	比率模式	1	4
数据码字 $\times n \times 0.1 = (4-10)$	比率模式	2	8
数据码字 $\times n \times 0.1 = (11-20)$	比率模式	3	16
数据码字 $\times n \times 0.1 = (21-45)$	比率模式	4	32
数据码字 $\times n \times 0.1 = (46-100)$	比率模式	5	64
数据码字 $\times n \times 0.1 = (101-200)$	比率模式	6	128
数据码字 $\times n \times 0.1 = (201-400)$	比率模式	7	256
数据码字 $\times n \times 0.1 = (400以上)$	比率模式	8	512