ZOLON

商用设备开发手册

Shenzhen Zolon Technology Co., Ltd.

1 文档介绍

本文档主要介绍兆珑商用设备系列软件开发流程，包括打印功能，扫码功能，钱箱功能，刷卡工具功能等需要二次开发的功能，适用机型如下表格所示。由于各商用设备的功能不同，所支持的模块可能存在差异，请以实际配置为准。

名称	机型号	OS版本	名称	机型号	OS版本
Zolon Station	L1400	A13	Zolon Note	M9200/M9200N	A12
Zolon Tower	L1450	A13	Zolon PayTablet	M8100	A14
Zolon Litchi	L160X	A13	Zolon Eye Touch	T3300	Nuxiums
Zolon Litchi pro	L161X	A14	Zolon Eye	T3320	RunthOS
Zolon display 16	K2160	A11	Zolon Lens	T3350	RunthOS
Zolon display 22	K2220	A11	Zolon printer	T3180	/
Zolon PayTablet	M8	A11	Bump Bar	PB20	/

注意事项:

【1】若适配的商用设备机型不在表格中，请咨询兆珑相关开发人员。

【2】其他设备的适配请参考[支付设备开发手册]。

2 设备调试

2.1 开启调试模式

商用设备可通过组合设置切换到调试模式，具体步骤如下:

1. 启用开发者模式
   “设置”->“关于设备”->连续点击“版本号”或“Build号”。
2. 开启“USB调试”或“无线调试”
   “设置”->“系统”->“开发者选项”->开启“USB调试”或“无线调试”。

2.2 连接调试工具

调试工具可通过实际情况来选择调试方式，具体方式如下：

- **使用USB调试**
  1.使用USB转type-C线材连接到调试设备对应的type-c口上即可。
- **使用无线调试**
  1.商用设备和调试设备连接到同一个局域网
  2.记录IP地址：“设置”->“系统”->“开发者选项”->“无线调试”->“IP地址和端口”
  3.适配adb环境变量，通过电脑中的命令提示符或者powershell窗口来进行连接（adb connect xx.xx.xx.xx:端口号）

> *注意：以上步骤操作完成后，仍无法连接到设备，请打开：设置-连接设备-USB，修改USB使用方式为文件模式或无数据模式*

2.3 安装调试应用

安装调试应用有很多种方式可以实现，常用方法如下：

- **使用U盘安装**
  1.调试应用为打包后的apk形式，可以借用type-c转USB接口的转接器，插入U盘进行安装。调试应用路径：文件-U盘目录-调试应用
- **使用USB安装**
  1.调试应用为源代码的形式，需要执行完连接调试工具的步骤后，将运行设备改为调试的商用设备，点击运行即可。

注意事项：
【1】连续点击”版本号”或“Build号”，会有弹窗提示，直到弹窗显示为:”您已处于开发者模式，无需进行此操作”,开发者模式才会被打开。
【2】切换为正常使用模式，关闭开发者选项总开关即可，开发者模式下重启是不会

重置为正常使用模式，请注意个人数据暴露风险。

【3】打开USB调试开关时会提示否是运行设备使用，需要点击允许。

【4】调试工具上没显示连接的设备，可重新插拔USB线或者重启调试工具。

【5】若调试步骤无法成功操作，请联系兆珑相关技术人员进行咨询。

3 打印功能开发

3.1 简介

商用设备的打印机支持热敏小票以及热敏标签打印，具有如下特点：

- **接口种类多**
  调用方便，可以快速构建复杂样式的打印单据。
- **开发难度小**
  兆珑自定义接口已涵盖客户的各种排版需求。
- **打印速度快**
  最高可达到200mm/s（具体数据请查看各机型参数手册）。
- **兼容性好**
  支持标准的ESC/POS指令。

3.2 适配方式

Zolon系列内置了热敏打印机，可通过以下三套接口实现打印操作，开发该功能前请确保已完成设备调试步骤。

注意事项：

【1】linkup和printerkit接口相斥，建议直接使用printerkit接口,在LinkUp的环境中也可直接使用printerkit。

【2】打印接口有多套开发文档，实际请根据商用设备支持类型进行开发

【3】网址登陆账号请联系兆珑相关技术人员提供

【4】使用过 `posapilite` 这套接口进行开发，可选择posapilite接口

3.2.1 PrinterKit方案（推荐）

为了让客户使用打印机有更好的体验，我们提供了一套名为 `printerkit` 的内置打印机接口，支持打印机ESC/POS控制指令调用。这套接口包含了更丰富更全面的打印设置，还能操作加入了LinkUp方案的外置打印机。具体请访问PrinterKit文件链接自行下载zip包。内附有接口详细说明文件，以下列举部分接口，具体请查看文件说明。打印接口支持的机型请查看以下列表。

PrinterKit接口适用机型如所示:
L1400,L1450,M9200,A3700,A800,A8700,A920MAX,IM30,E800
E600M,E700M,E700A,SK300,Sunmi V2sPLUS,Google Pixel 3
Samsung Galaxy S10 LiteA3700,E700

3.2.1.1 初始化打印设备

- **getInstance:创建PrinterKitManager的实例**

```
public static PrinterKitManager getInstance()
```

参数	返回值	抛出	备注
无	PrinterKitManager对象	无	

- **connect:建立与打印服务的连接**

```
public void connect(Context context,IPrinterKitConnectCallback callback)
```

参数	返回值	抛出	备注
context	无	无	
callback	无	无	连接的状态回调，见IPrinterKitConnectCallback

注：连接后会回调IPrinterKitConnectCallback.onConnect方法

- **connect:建立与打印服务的连接（集成了NeptuneLiteApi时请调用此接口连接打印服务）**

```
public void connect(Context context,IDAL idal,
```

```
                    IPrinterKitConnectCallback callback)
```

参数	返回值	抛出	备注
context	无	无	上下文
idal	无	无	Neptune对外接口对象
callback	无	无	连接的状态回调，见IPrinterKitConnectCallback

注：连接后会回调IPrinterKitConnectCallback.onConnect方法

- **getPrinterList:获取打印机列表**

```
public java.util.List<java.lang.String> getPrinterList()
```

参数	返回值	抛出	备注
无	打印机deviceID列表	PrinterKitException	

- **getComboPrintAPI:获取打印服务代理对象，通过这个代理对象调用打印相关接口**

```
public IComboPrint getComboPrintAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	打印API接口	PrinterKitException	

- **getComboConfigAPI:获取配置服务代理对象，通过这个代理对象调用网络配置，打印模式切换等相关接口**

```
public IComboConfig getComboConfigAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	打印API接口	PrinterKitException	

- **initConfig:初始化打印配置，需在打印前调用**

```
void initConfig(BaseConfig baseConfig)
```

参数	返回值	抛出	备注
baseConfig	无	PrinterKitException	

`baseConfig` 参数如下:

可用方法	方法说明	参数说明	默认值
textSize	设置文本字符大小	6~96像素	24
width	设置行内可打印区域宽度	当超出可设置范围时，仅设置最大值	打印机纸张宽度，内置:384像素， 外置:576像素
mode	设置打印模式	目前只支持普通热敏纸打印	0
align	设置行内对齐方式	见Align	居左
leftIntent	设置行左边距	左边距大小不能超过纸张宽度，左边距设置可能会导致打印内容换行	0

- **connectPrinter:连接打印机**

```
public void connectPrinter(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	无	PrinterKitException	打印机SN

- **registerShareMode:注册共享模式**

```
public void registerShareMode(java.lang.String deviceId,int mode,
```

```
                                IPrintResultCallback callback)
```

参数	返回值	抛出	备注
deviceID	无	无	打印机deviceID
mode	无	无	打印机连接模式 0： 普通模式 1：共享模式
callback	无	无	共享模式下打印结果回调，见IPrintResultCallback

注：目前仅支持IP、WIFI的共享模式（多台上位机共享一台打印机，一台上位机打印完成后，另外一台打印机才能继续打印）； BT、USB仍为独占模式

- **sendESCCmd:发送esc指令**

```
public void sendESCCmd(java.lang.String deviceID,byte[] cmd)
```

参数	返回值	抛出	备注
deviceID	无	PrinterKitException	打印机deviceID
cmd	无	同上	esc指令

- **discoverPrinter:发现外置打印机IDiscoveryCallback#onResult(List)**

```
public void discoverPrinter(EDiscoveryMode mode,IDiscoveryCallback callbac
k)
```

参数	返回值	抛出	备注
mode	无	PrinterKitException	打印机连接方式
callback	无	同上	发现设备回调，见IDiscoveryCallback

- **disconnect:断开与打印服务的连接**

```
public void disconnect()
```

参数	返回值	抛出	备注
无	无	无	

注：断开连接后会执行IPrinterKitConnectCallback.onDisconnect方法

- **disconnectPrinter:通过deviceID断开打印机的连接**

```
public void disconnectPrinter(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	无	PrinterKitException	打印机SN

3.2.1.2 设置打印机属性

- **printText:设置文本格式与文本内容**

```
void printText(java.lang.String text,TextConfig textconfig)
```

参数	返回值	抛出	备注
`text`	无	PrinterKitException	要打印的文本内容
`textconfig`	无	同上	

`textConfig` 参数如下：

可用方法	方法说明	参数说明	默认值
textSize	设置文本字符大小	6~96像素	24
align	设置行内对齐方式	见Align	居左
leftIndent	设置行左边距	左边距大小不能超过纸张宽度，左边距设置可能会导致打印内容换行	0
noWrap	设置不换行	无	换行

- **printImage:设置图片格式与位图内容**

```
void printImage(android.graphics.Bitmap bitmap,ImageConfig imageconfig)
```

参数	返回值	抛出	备注
`bitmap`	无	PrinterKitException	预打印的Bitmap对象
`imageconfig`	无	同上	

`imageconfig` 参数如下：

可用方法	方法说明	参数说明	默认值
imgWidth	图片缩放宽度	1. 若设置imgWidth小于0或不设置，当原图宽度小于可打印区域宽度，按原图打印；当原图大于可打印区域宽度，按可打印区域宽度对原图进行等比例缩放后打印；2. 若设置imgWidth为0，不打印图片；3. 若设置imgWidth大于0且小于可打印区域宽度，按imgWidth对原图进行等比例缩放后打印；4. 若设置imgWidth大于等于可打印区域宽度，按可打印区域宽度对原图进行等比例缩放后打印；	-1（表示不设置）
imgHeight	图片缩放高度	1. 若设置imgHeight为0，不打印图片；2. 若设置imgHeight为其他值，根据imgWidth的设置，进行等比例调整	-1（根据imgWidth的设置，进行等比例调整）
align	设置行内对齐方式	见Align	居左

leftIntent	设置行左边距	左边距大小不能超过纸张宽度，左边距设置可能会导致打印内容换行	0
noWrap	设置不换行	无	换行

- **printBarCode:设置条形码格式与打印内容**

```
void printBarCode(java.lang.String text,BarCodeConfig barcodeconfig)
```

参数	返回值	抛出	备注
`text`	无	PrinterKitException	预打印的条形码内容
`barcodeconfig`	无	同上	

`barcodeconfig` 参数如下：

可用方法	方法说明	参数说明	默认值
imgWidth	生成指定宽度的条形码	用于生成条形码的宽度，如果大于纸宽，会被截掉	250
imgHeight	生成指定高度的条形码	用于生成条形码的高度，如果大于纸高，会被截掉	100
codeFormat	条形码格式	见BarcodeDataFormat	CODE_128
align	设置条形码行内对齐方式	见Align	居左

leftInte nt	设置行左边距	左边距大小不能超过纸张宽度，左边距设置可能会导致打印内容换行	0
noWra p	设置不换行	无	换行

- **printQRCode:设置二维码格式与二维码内容**

```
void printQRCode(java.lang.String text,QRCodeConfig qrcodeconfig)
```

参数	返回值	抛出	备注
`text`	无	PrinterKitException	预打印的二维码内容
`qrcodeconfig`	无	同上	

注意事项:

【1】二维码是正方形的，当传入的指定宽高不相等时，会取其中较小的值作为宽高；当传null时，使用默认配置打印二维码

`qrcodeconfig` 参数如下:

可用方法	方法说明	参数说明	默认值
imgWidth	生成指定宽度的二维码	如果imgWidth小于imgHeight，则以imgWidth作为二维码的宽高大小，如果最后宽度大于纸宽，会被截掉	150
imgHeight	生成指定高度的二维码	如果imgHeight小于imgWidth，则以imgHeight作为二维码的宽高大小，如果最后宽度大于纸宽，会被截掉	150
align	设置二维码行内对齐方式	见Align	居左
leftIntent	设置行左边距	左边距大小不能超过纸张宽度，左边距设置可能会导致打印内容换行	0
no			

Wrap	设置不换行	无	换行

3.2.1.3 启动打印机

- **printBitmap:打印Bitmap图片**

```
public void printBitmap(java.lang.String deviceID,Bitmap bitmap)
```

参数	返回值	抛出	备注
deviceID	无	PrinterKitException	打印机deviceID
bitmap	无	同上	bitmap图片

- **startPrint:开启打印**

```
void startPrint()
```

参数	返回值	抛出	备注
无	无	PrinterKitException	

3.2.1.4 打印机状态获取

- **getStatus:获取打印机状态**

```
public EStatus getStatus(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	EStatus	PrinterKitException	打印机deviceID

返回值参数如下:

参数	说明	备注
EStatus.ONLINE	printer is ready	打印机准备就绪
EStatus.ERR_OFFLINE	printer offline	打印机离线
EStatus.ERR_OUT_OF_PAPER	out of paper error	缺纸异常

EStatus.ERR_COVER_OPEN	cover open error	盖子打开异常
EStatus.ERR_OVER_HEATING	printer over heating	打印机过热
EStatus.ERR_VOLTAGE_TOO_LOW	pvoltage is too low	电压过低
EStatus.ERR_NO_EXIST	printer not exist	打印机不存在
EStatus.ERR_UNEXPECTED	unexpected error	意外错误

3.2.2 POSAPILite方案

开发者若是使用过 `posapilite` 的接口进行开发，商用产品也是支持这套接口，请访问posapilite文件链接自行下载，内附尾缀为chm文件有该打印功能接口的详细介绍。以下列举的方法为部分常用接口展示。

3.2.2.1 初始化打印设备

- **getInstance:创建PrintManager的实例**

```
public static PrintManager getInstance();
```

参数	返回值	抛出	备注
无	PrintManager对象	PrintException	

- **getInstance:创建PrintManager带参数的实例**

```
public static PrintManager getInstance(android.content.Context ctx);
```

参数	返回值	抛出	备注
`ctx`	PrintManager对象	PrintException	

- **finalize:关闭打印机并释放PrintManager**

```
public void finalize();
```

参数	返回值	抛出	备注

无	无	PrintException	覆盖：finalize在java.lang.Object中

- **prnInit:打印机初始化**

```
public void prnInit();
```

参数	返回值	抛出	备注
无	无	PrintException	

3.2.2.2 设置打印机属性

- **prnFontSet:设置打印的字体**

```
public void prnFontSet(byte Ascii,byte Cfont);
```

`Ascii` 码的字体设置。

取值	说明	取值	说明
0	8x16 字体 (基本)	5	12x24 字体 (横向放大)
1	12x24 字体 (基本)	6	8x16 字体 (整体放大)
2	8x16 字体 (纵向放大)	7	12x24 字体 (整体放大)
3	12x24 字体 (纵向放大)	其他值	不改变初始值
4	8x16 字体 (横向放大)		

`Cfont` 扩展码的字体设置。

取值	说明	取值	说明
0	16x16 font (基本)	5	24x24 font (横向放大)
1	24x24 font (基本)	6	16x16 font (整体放大)
2	16x16 font (纵向放大)	7	24x24 font (整体放大)
3	24x24 font (纵向放大)	其他值	不改变初始值(任意一个参数设置错误则均不作变化)

4	16x16 font (横向放大)		

- **prnSpaceSet:设置打印字间距、行间距**

```
public void prnSpaceSet(byte charSpace,byte lineSpace);
```

参数	返回值	抛出	备注
`charSpace`	无	无	字间距 [像素点]
`lineSpace`	无	无	行间距 [像素点]

- **prnBitmap:打印位图传两参数的方法**

```
public void prnBitmap(Bitmap bitmap,int grayThreshold);
```

参数	返回值	抛出	备注
`bitmap`	无	PrintException java.io.IOException	打印位图的大小限制：宽度最大为384个像素点，高度没有限制。如果位图的宽度超过打印机的限制，将会裁剪图片的右边。
`grayThreshold`	无	同上	灰度阈值。

- **prnBitmap:打印位图传一参数的方法**

```
public void prnBitmap(Bitmap bitmap);
```

参数	返回值	抛出	备注
`bitma`	无	PrintException java.io.IOExce	打印位图的大小限制：宽度最大为384个像素点，高度没有限制。如果位图的宽度超过打印机的限制，将会裁剪图片

p		ption	的右边。

- prnLeftIndent:设置字符打印左边界

```
public void prnLeftIndent(short Indent);
```

参数	返回值	抛出	备注
Indent	无	无	左边界空白像素点，范围：0~300。

- prnSetGray:设置打印灰度等级

```
public void prnSetGray(int Level);
```

Level 数值设置列表

数值	说明
1	缺省灰度
3	双层热敏打印
4	双层热敏打印，比3的黑度更高
50~500	黑度按照缺省黑度百分比进行设置，如50表示把黑度设置为默认值的50%，500则表示把黑度设置为500%
其他值	保留并且更改无效（注：E500不支持300~500）

- prnSetFontName:设置打印机字体名称

```
public void prnSetFontName(java.lang.String name);
```

参数	返回值	抛出	备注
name	无	PrintException	字体名称。

- prnSelectFont:选择打印机字体

```
public void prnSelectFont(ST_FONT singleCodeFontAttr,ST_FONT multiCodeFontAttr);
```

参数	返回值	抛出	备注
`singleCodeFontAttr`	无	PrintException	单内码字体（比如英文、俄文等），可以为NULL。
`multiCodeFontAttr`	无	同上	多内码字体（比如中文、日文等），可以为NULL

- **prnAttrSet:设置反显打印**

```
public void prnAttrSet(int reverse);
```

参数	返回值	抛出	备注
`reverse`	无	无	1表示反显打印，0表示正常。

- **prnDoubleWidth:设置打印字体宽度，可在基本字体上实现双倍宽度的打印**

```
public void prnDoubleWidth(int AscDoubleWidth,int LocalDoubleWidth);
```

参数	返回值	抛出	备注
`AscDoubleWidth`	无	无	单内码字体，1表示双倍宽度，0表示正常宽度。
`LocalDoubleWidth`	无	无	多内码字体，1表示双倍宽度，0表示正常宽度。

- **prnDoubleHeight:设置打印字体高度，可在基本字体上实现双倍高度的打印**

```
public void prnDoubleHeight(int AscDoubleHeight,int LocalDoubleHeight);
```

参数	返回值	抛出	备注
			单内码字体，1表示双倍高度，0表示正常高

AscDoubleHeight	无	无	度。
LocalDoubleHeight	无	无	多内码字体，1表示双倍高度，0表示正常高度。

- **prnPresetCutPaper:设置切纸模式，打印完成后立刻切纸**

```
public void prnPresetCutPaper(int mode);
```

参数	返回值	抛出	备注
mode	无	PrintException	请在执行prnStart()前调用该方法。

mode 数值设置列表:

取值	说明
-1	表示不切纸
0	表示全部切纸
1	表示部分切纸

- **prnSetEffect:设置打印效果**

```
public void prnSetEffect(int mode);
```

参数	返回值	抛出	备注
mode	无	PrintException	取值为0时, 表示默认打印效果；取值为1时, 表示定制打印效果（降低打印速度45mm/s，增加加热时间，优化打印晕染）。

- **prnPaperDetect:标签间隙纸学习**

```
public void prnPaperDetect();
```

参数	返回值	抛出	备注

参数	返回值	抛出	备注
无	无	PrintException	

- **prnFeedPaper:Feed一张标签**

```
public void prnFeedPaper();
```

参数	返回值	抛出	备注
无	无	PrintException	该函数使用前必须调用一次prnPaperDetect进行标签学习。

- **prnSetPrintMode:设置打印模式**

```
public void prnSetPrintMode(int mode);
```

参数	返回值	抛出	备注
`mode`	无	PrintException	取值为0时, 表示普通热敏纸（连续纸）打印模式。取值为1时, 表示标签热敏纸（间隙/黑标纸）打印模式。

- **enumFontr:枚举当前 POS 支持的字体**

```
public ST_FONT[] enumFont(int maxFontNum);
```

参数	返回值	抛出	备注
`maxFontNum`	存放字体的缓冲	PrintException	Fonts 缓冲可以存放字体的最大个数不超过64

- **prnStep:让打印机正向或反向走纸pixel个像素点**

```
public void prnStep(short pixel);
```

参数	返回值	抛出	备注

`pixel`	无	无	像素点个数

3.2.2.3 启动打印机

- **prnStart:启动打印机，并将缓冲区里的数据打印出来**

```
public void prnStart();
```

参数	返回值	抛出	备注
无	无	PrintException	1. 每行的点数 = 字符的高度+行距 2. 总的点数 =每行的点数 * 行数 3. 总的点数要小于30000，默认字符高度是24。

- **prnStr:打印字符串**

```
public void prnStr(java.lang.String str,java.lang.String charset() ;
```

参数	返回值	抛出	备注
`str`	无	PrintException	要打印的字符串。
`charset`	无	同上	字符串的编码方式。

- **prnCutPaper:进行切纸**

```
public int prnCutPaper(int mode);
```

参数	返回值	抛出	备注
`mode`	返回0表示执行成功，其他值表示失败。	PrintException	1.取值为0时, 表示全部切纸。 2.取值为1时, 表示部分切纸。

3.2.2.4 打印机状态获取

- **prnStatus:查询当前打印状态**

```
public byte prnStatus();
```

返回值状态查询表

数值	说明	数值	说明
0x00	打印成功	0x09	打印机电压过低
0x01	打印机忙	0xf0	打印未完成
0x02	打印机缺纸	0xfa	切刀异常(支持E500,E800)
0x03	打印数据包格式错误	0xfb	开盖错误(支持E500,E800)
0x04	打印机故障	0xfc	打印机未装字库
0x08	打印机过热	0xfe	数据包过长

- **prnGetDotLine:获取当前已打印的点阵行**

```
public int prnGetDotLine();
```

参数	返回值	抛出	备注
无	当前的点阵行数。	无	

- **prnGetTemperature:获取打印机温度**

```
public int prnGetTemperature();
```

参数	返回值	抛出	备注
无	打印机温度。	无	

- **prnGetFontDot:获取指定字符的打印点阵**

```
public byte[] prnGetFontDot(int fontSize, java.lang.String str)
```

参数	返回值	抛出	备注

`fontSize`	如下表格	PrintException java.io.UnsupportedEncodingException	0表示小字体（8x16 或 16x16）1表示大字体（12x24 或 24x24）
`str`	同上	同上	指定的字符。

返回值为点阵以及点阵占用空间，参数如下:

点阵	占用空间
8x16	16个字节
16x16	32个字节
12x24	48个字节
24x24	72个字节

- **prnGetEffect:获取当前打印效果**

```
public int prnGetEffect();
```

参数	返回值	抛出	备注
无	当前打印效果。	PrintException	当mode = 0时, 表示默认打印效果；当mode = 1时, 表示定制打印效果（降低打印速度45mm/s，增加加热时间，优化打印晕染）。

- **prnGetCutMode:获取商用设备支持的切纸模式**

```
public int prnGetCutMode();
```

参数	返回值	抛出	备注
无	如下表格	PrintException	

返回值为切纸模式，参数如下:

取值	说明

0	支持全切
1	支持半切
2	支持半切和全切
-1	不支持切刀

3.2.3 LinkUp方案

开发者若是使用过 `linkup` 的接口进行开发，商用产品也是支持这套接口，请访问linkup文件链接自行下载，内附尾缀为chm文件有该打印功能接口的详细介绍。以下列举的方法为部分常用接口展示，打印接口支持的机型请查看以下列表。

LinkUp接口适用机型如所示:
Built-in Printer (L1450/M9200/A800)
External Printer (T3180)
Third Party Printer

1.LinkUpSdk

- **init:初始化**

```
public void init(InitListener initListener)
```

参数	返回值	抛出	备注
initListener	无		

2.DeviceHelper

- **getSelfDeviceInfo:获取设备信息**

```
public LinkDevice getSelfDeviceInfo()
```

参数	返回值	抛出	备注
无	无	LinkException	

- **queryOfflineDeviceList:获取设备离线列表**

```
public List<LinkDevice> queryOfflineDeviceList()
```

参数	返回值	抛出	备注
无	返回设备离线列表	LinkException	

- **queryOnlineDeviceList:获取设备在线列表**

```
public List<LinkDevice> queryOnlineDeviceList()
```

参数	返回值	抛出	备注
无	返回设备在线列表	LinkException	

3.FileHelper

- **removeFile:删除文件**

```
public void removeFile(String str, String str2)
```

参数	返回值	抛出	备注
`str`	无	LinkException	从指定的设备中删除有绝对路径的文件

- **transferFile:传输文件**

```
public void transferFile(String str, String str2, String str3, boolean z,
IFileProcess iFileProcess)
```

参数	返回值	抛出	备注
`str` `str2` `str3` `z` `iFileProcess`	无	LinkException	从调用该方法的设备中传输到另一台具有ID的设备

- **cancelTransferFile:取消传输文件**

```
public void cancelTransferFile(String str, String str2, String str3, boole
an z, IFileProcess iFileProcess)
```

参数	返回值	抛出	备注
`str` `str2` `str3` `z` `iFileProcess`	无	LinkException	取消transferFile的执行

4.MiscHelper

- **getAllAppInfo:获取指定设备上的应用程序列表**

```
public List<AppInfo> getAllAppInfo(String str, String str2)
```

参数	返回值	抛出	备注
`str` `str2`	获取应用程序列表	LinkException	

- **installFile:在指定设备上安装应用**

```
public void installFile(String str, String str2, String str3, String str4)
```

参数	返回值	抛出	备注
`str` `str2` `str3` `str4`	无	LinkException	

- **reboot:重启指定设备**

```
public void reboot(String str, String str2)
```

参数	返回值	抛出	备注
`str` `str2`	无	LinkException	

- **shutdown:关机指定设备**

```
public void shutdown(String str, String str2)
```

参数	返回值	抛出	备注
str str2	无	LinkException	

5.PrinterHelper

- **queryPrinterInfoList:获取打印机设备列表**

```
public List<LinkDevice> queryPrinterInfoList()
```

参数	返回值	抛出	备注
无	打印机设备列表	LinkException	

- **printImage:打印图像**

```
public void printImage(String str, String str2, PrintParam printParam, ResultListener resultListener)
```

参数	返回值	抛出	备注
str str2 printParam resultListener	无	LinkException	

- **sendRawCommand:发送原始打印指令**

```
public byte[] sendRawCommand(String str, String str2, CommandChannelRequestContent commandChannelRequestContent)
```

参数	返回值	抛出	备注
str str2 commandChannelRequestContent		LinkException	

6.ScannerHelper

- queryScannerInfolist:获取扫码设备列表
- getParam:获取扫码参数
- setParam:设置扫码参数
- startScan:开始扫码
- stopScan:停止扫码

7.MediaHelper

- startCastScreen:开始投屏屏幕
- stopCastScreen:停止投票屏幕
- startCastVideo:开始播放投屏文件
- stopCastVideo:停止播放投屏文件
- setCustomPlayViewListener:设置自定义播放图片接口

3.2.4 其他打印接口方案

3.2.4.1 虚拟蓝牙打印服务

通过标准的ESC/POS指令调用商用设备自带的打印机，从而实现打印功能，方法如下：

1. 连接内置打印机。

通过操作“设置-连接设备-添加蓝牙设备”跳转到蓝牙搜索界面后选择“内置蓝牙打印机”，界面如下：

2. 建立蓝牙打印通道。

```
//获取所有已配对的设备
public static List<BluetoothDevice> getPairedDevices() {
   List deviceList = new ArrayList<>();
   Set<BluetoothDevice> pairedDevices = BluetoothAdapter.getDefaultAdapter
().getBondedDevices();
   if (pairedDevices.size() > 0) {
      for (BluetoothDevice device : pairedDevices) {
         deviceList.add(device);
      }
   }
   return deviceList;
}

//根据传入的device，建立蓝牙连接
public static BluetoothSocket connectDevice(BluetoothDevice device) {
   BluetoothSocket socket = null;
   try {
```

```
            socket = device.createRfcommSocketToServiceRecord(
                    UUID.fromString("00001101-0000-1000-8000-00805F9B34FB"));
            socket.connect();
        } catch (IOException e) {
            try {
                socket.close();
            } catch (IOException closeException) {
                return null;
            }
            return null;
        }
        return socket;
    }

    //根据connectDevice返回的BluetoothSocket获取输出流管道outputStream
        try {
        OutputStream outputStream = bluetoothSocket.getOutputStream();
    } catch (IOException e) {
        e.printStackTrace();
    }
```

3. 将构建好的打印单据信息传给打印机。

```
//outputStream是根据connectDevice返回的BluetoothSocket获取到的输出流管道
public void print(byte[] bs) throws IOException {
    outputStream.write(bs);
}
```

打印信息的构建可参考 打印机ESCPOS控制指令.pdf。

3.2.4.2 Android打印服务

商用设备也支持Android源生系统打印接口，若选择调用源生接口进行开发，可自行查找相关资料。

4 扫码功能开发

4.1 简介

兆珑提供了适配其商用设备的专业扫码SDK，相对于其它开源扫码方案，其具有以下优势：

【1】 高识别率
具备条码污损容差能力强、解码率高的特点，经过大量、多种模拟真实场景的测试，扫码识别率可接近100%。

【2】快速扫码
解码速度快，平均解码速度可达50毫秒。

【3】安全稳定
使用专业的解码算法，稳定性高，安全性强。

【4】 简单灵活
与商用设备完美适配，提供简单且功能强大的SDK接口，只需少量接口即可添加扫码功能。

【5】丰富码制
支持以下二十余种码制，其中Code128的扫码精度可达到5mil及以上。

条码支持表

类型	格式
一维	Code11、Code39、Code93、Code128、EAN-8、Codabar、EAN-13、HK25、Interleaved 2 of 5、GS1_DATABAR、MSI、Postal、Straight 25、Tri-Optic、Telepen、MATRIX 25、UCC/EAN 128、UPC-A/E、NW-7。
二维	QR、PDF417、Data Matrix、Codablock F、MicroPDF、MaxiCode、Aztec Code、Han Xin Code。

4.2 适配方式

目前扫码接口分扫码枪接口和摄像头扫码接口，以下只展示扫码接口的部分接口定义，具体开发说明请访问相关扫码的API文件。

4.2.1 扫码枪扫码

兆珑自定义的扫码接口，该接口具备丰富的设置参数，建议使用这一套sdk进行开发，具体请访问ScannerKit文件链接自行下载zip包。内附有接口详细说明文件，以下列举部分接口，具体请查看文件说明。

> *注意：兆珑自研的扫描枪和目前市场上大部分的扫码设备都是支持即插即用，也可以需要需求使用以下扫码枪接口进行适配*

适用机型:	适用设备
L1400,L1450,L160X	T3300,T3320,T3350

注意事项:

【1】扫码接口有多套开发文档，实际请根据商用设备支持类型进行开发

【2】网址登陆账号请联系相关技术人员提供

- **getInstance:获取扫码sdk的实例**

```
public static ScannerKitManager getInstance()
```

参数	返回值	抛出	备注
无	无	返回sdk的实例	

- **connect:建立与扫码服务的连接**

```
public void connect(android.content.Context context,IScannerKitConnectCallback callBack)
```

参数	返回值	抛出	备注

context	无	ScannerKitException	上下文
callBack	无	同上	连接的状态回调，见IScannerKitConnectCallback

注：连接后会回调IScannerKitConnectCallback.onConnect方法

- **disconnect:断开与扫码服务的连接**

```
public void disconnect()
```

参数	返回值	抛出	备注
context	无	ScannerKitException	上下文
callBack	无	同上	连接的状态回调，见IScannerKitConnectCallback

注：连接后会回调IScannerKitConnectCallback.onDisconnect方法

- **getScannerList:获取连接的扫码设备的deviceID列表**

```
public java.util.List<java.lang.String> getScannerList()
```

参数	返回值	抛出	备注
无	扫码设备deviceID列表	ScannerKitException	

- **getScannerAPI:获取扫码相关接口**

```
public IScanner getScannerAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	扫码API接口	ScannerKitException	连接的扫码设备的deviceID

- **getConfigAPI:获取扫码配置相关接口**

```
public IScannerConfig getConfigAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	扫码配置类接口	ScannerKitException	连接的扫码设备的deviceID

- **getScannerFileAPI:获取扫码文件类相关接口**

```
public IScannerFile getScannerFileAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	扫码文件类接口	ScannerKitException	连接的扫码设备的deviceID

- **getSystemAPI:获取扫码设备类相关接口**

```
public IScannerSystem getSystemAPI(java.lang.String deviceID)
```

参数	返回值	抛出	备注
deviceID	扫码设备类接口	ScannerKitException	连接的扫码设备的deviceID

4.2.2 摄像头扫码

以下介绍的是轻量级posapilite的摄像头接口，若之前开发使用过posapilite，可继续使用该接口进行适配，具体请访问PosApiLite-Scanner文件链接自行下载zip包。内附有接口详细说明文件，以下列举部分接口，具体请查看文件说明。

注意事项:

【1】posapilite的摄像头接口目前只支持M9200，其他机型请使用常规源生的摄像头接口即可。

【2】使用该扫码接口必须先添加以下三种权限。

所需权限
android.permission.READ_PHONE_STATE

android.permission.INTERNET
android.permission.ACCESS_WIFI_STATE

- **getInstance:创建ScannerManagerLite实例**

```
public static IScannerManager getInstance();
```

参数	返回值	抛出	备注
无	ScannerManagerLite对象。如果为null，表示不支持。	无	

- **getInstance:创建ScannerManagerLite实例(带参数)**

```
public static IScannerManager getInstance(int type);
```

参数	返回值	抛出	备注
type	ScannerManagerLite对象。如果为null，表示不支持。	无	type - 解码库类型：0:zmxf 1:kudw 只有当对应解码库激活时才生效，其他值表示默认

- **init:初始化ScannerManagerLite**

```
public boolean init (android.content.Context ctx,int width,int height);
```

参数	返回值	抛出	备注
`ctx`	初始化的结果。	无	上下文。
`width`	同上	无	图像宽度。
`height`	同上	无	图像高度。

> *注意:目前只支持1280 * 720，640 * 480， 480 * 640*

- **unInit:释放ScannerManagerLite资源**

```
public void unInit();
```

参数	返回值	抛出	备注
无	无	无	

- startDecode:开始扫描解码

```
public ScanResult startDecode(byte[] data);
```

参数	返回值	抛出	备注
`data`	扫描结果	无	需要解码的原始数据。

- startDecodeRaw:开始扫描解码，返回未经编码转换的原始数据

```
public ScanResultRaw startDecodeRaw(byte[] data);
```

参数	返回值	抛出	备注
`data`	未经编码转换的原始数据。	无	需要解码的原始数据。

- startDecodeRawWithTimeout:开始扫描解码，返回未经编码转换的原始数据（仅支持智码讯付）

```
public ScanResultRaw startDecodeRawWithTimeout(byte[] data, long timeOut);
```

参数	返回值	抛出	备注
`data`	未经编码转换的原始数据。	无	需要解码的原始数据。
`timeOut`	无	无	超时时长（单位ms，>=500ms）。

- enableCodec:启用指定的扫描类型

```
public void enableCodec(int codec);
```

`codec` 条码类型

取值	条码类型	取值	条码类型	取值	条码类型
1	CODEC_UPC	9	CODEC_PDF417	17	CODEC_TRIOPTIC
2	CODEC_C39	10	CODEC_MICROPDF	18	CODEC_STRAIGHT25
3	CODEC_C128	11	CODEC_MAXICODE	19	CODEC_TELEPEN
4	CODEC_I25	12	CODEC_QRCODE	20	CODEC_C11
5	CODEC_C93	13	CODEC_DATAMATRIX	21	CODEC_NEC25
6	CODEC_GS1_DATABAR	14	CODEC_AZTEC	22	CODEC_CODABAR
7	CODEC_MSI	15	CODEC_HAXIN	23	CODEC_HK25
8	CODEC_CODEBLOCK_F	16	CODEC_MATRIX	24	CODEC_POSTAL

个别商用设备支持的条码类型如下：

取值	条码类型	取值	条码类型
1	CODEC_UPC	5	CODEC_C93
2	CODEC_C39	6	CODEC_GS1_DATABAR
3	CODEC_C128	12	CODEC_QRCODE
4	CODEC_I25	22	CODEC_CODABAR

- **disableCodec:禁用指定的扫描类型**

```
public void disableCodec(int codec);
```

`codec` 条码类型。

取值	条码类型	取值	条码类型	取值	条码类型
1	CODEC_UPC	9	CODEC_PDF417	17	CODEC_TRIOPTIC

2	CODEC_C39	10	CODEC_MICROPDF	18	CODEC_STRAIGHT25
3	CODEC_C128	11	CODEC_MAXICODE	19	CODEC_TELEPEN
4	CODEC_I25	12	CODEC_QRCODE	20	CODEC_C11
5	CODEC_C93	13	CODEC_DATAMATRIX	21	CODEC_NEC25
6	CODEC_GS1_DATABAR	14	CODEC_AZTEC	22	CODEC_CODABAR
7	CODEC_MSI	15	CODEC_HAXIN	23	CODEC_HK25
8	CODEC_CODEBLOCK_F	16	CODEC_MATRIX	24	CODEC_POSTAL

个别商用设备支持的条码类型如下：

取值	条码类型	取值	条码类型
1	CODEC_UPC	5	CODEC_C93
2	CODEC_C39	6	CODEC_GS1_DATABAR
3	CODEC_C128	12	CODEC_QRCODE
4	CODEC_I25	22	CODEC_CODABAR

- **ScannerManagerLite接口实例**

```
    private int IMAGE_WIDTH = 1280;//640;
    private int IMAGE_HEIGHT = 720;//480;
    private byte[] preivewBuf;
    private SurfaceView mSurfaceView;
    private SurfaceHolder mSurfaceHolder;
    private Camera mCamera;
    private TextView tvBarcode;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        preivewBuf = new byte[10*IMAGE_WIDTH*IMAGE_HEIGHT];
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_main);
        ScannerManagerLite.getInstance().init(this, IMAGE_WIDTH,IMAGE_HEI
GHT);//调用初始化接口
        for(int i = 1; i < 24; i++) {
        ScannerManagerLite.getInstance().enableCodec(i);//使能扫码类型
        }
        isStartDecode = false;
        initView();//初始化界面及摄像头等
    }
    private class DecoderTask extends AsyncTask<byte[], String, String> {
        private DecoderTask() {}
        @Override
        protected String doInBackground(byte[]... params) {
            if (mCamera == null) {
                Log.d(TAG, "doInBackground: camera is null...");
                return null;
            }
            byte[] ImageBuf = params[0];
            String barcodeData = null;
            isStartDecode = true;
            runDecodeOK = false;
            ScanResult decRes = ScannerManagerLite.getInstance().startDec
ode(ImageBuf);//开始使用ImageBuf的数据解码
            if (decRes != null) {
                totalCount = 0;
                Log.d(TAG, "data:"+ decRes.getContent());
            } else {
            }
            runDecodeOK = true;
            if (decRes != null) {
                barcodeData = decRes.getContent();
                String format = decRes.getFormat();
                String str = "CodeType: " + format + " \nData: " + barcod
eData;
                publishProgress(str, format, barcodeData);
```

```
            }
            if (mCamera != null) {
                Log.d("AAA", "addCallbackBuffer：doInBackground");
                mCamera.addCallbackBuffer(params[0]);
            }
            isDecoding = false;
            return null;
        }
        @Override
        protected void onProgressUpdate(String... values) {
            Date date = new Date();
            count++;
            String text = sdf.format(date) + "    Count=" + count + "\n";
            tvBarcode.setText(text + values[0]); //显示扫码结果

            try {
                BaseSystemManager.getInstance().beep();//调用百富接口，在扫码动作完成后发beep音提示用户
            } catch (BaseSystemException e) {
                e.printStackTrace();
            }
            isDecoding = false;
            frameCount = 0;
        }
    }

    private void initView() {
        mSurfaceView = (SurfaceView) findViewById(R.id.preview_view);
        mSurfaceHolder = mSurfaceView.getHolder();
        mSurfaceHolder.setType(3);
        mSurfaceHolder.addCallback(this);
        tvBarcode = (TextView) findViewById(R.id.tv_barcode);
        cameraId = 0; //使用后置
//        cameraId = 1; //使用前置
        mCamera = Camera.open(cameraId);
        mCamera.setDisplayOrientation(90);
        Camera.Parameters parameters = mCamera.getParameters();
```

```
        //设置预览界面大小
        parameters.setPreviewSize(IMAGE_WIDTH, IMAGE_HEIGHT);
        parameters.setPictureSize(IMAGE_WIDTH, IMAGE_HEIGHT);
        //设置缩放
        int zoomValue = parameters.getZoom();
        parameters.setZoom(zoomValue+ 15);
//        parameters.setFocusMode(Parameters.FOCUS_MODE_CONTINUOUS_PICTURE
); //设置自动定焦
        mCamera.setParameters(parameters);
    }
```

5 钱箱功能开发

部分商用设备可通过调用以下接口连接外置钱箱。

5.1 简介

根据客户具体的使用场景，兆珑提供了外置钱箱的解决方案，以下是具体的钱箱开发接口

5.2 适配方式

5.2.1 钱箱接口

具体请访问PosApiLite-CashBox文件链接自行下载zip包。内附有接口详细说明文件，以下列举部分接口，具体请查看文件说明。

- **cashBoxPopup:弹出钱箱**

```
public void cashBoxPopup();
```

参数	返回值	抛出	备注
无	无	BaseSystemException	

- **cashBoxStatus:检查钱箱状态**

```
public int cashBoxStatus();
```

参数	返回值	抛出	备注
无	如下表格	BaseSystemException	

返回值参数如下:

取值	说明
1	打开
0	关闭，或无法监测到钱箱状态
<0	出现错误

6 外置刷卡功能开发

部分商用设备可外接刷卡小工具，以下为具体的开发函数方法

6.1 简介

为了方便客户在使用兆珑的商用产品下支付更加便捷，我们推出了L1450+D135以及M8+R20组合，大幅度满足了客户对于支付环境的安全性和效率性，客户可根据实际需求进行功能开发

6.2 适配方式

目前刷卡小工具有D135和R20这两种类型，均需搭配相关商用设备进行使用，以下仅提供基础的控制接口，具体的交易功能接口请咨询兆珑相关的技术人员。

6.2.1 刷卡接口

具体请访问paymentdevice文件链接自行下载zip包。内附有接口详细说明文件，以下列举部分接口，具体请查看文件说明。

> *注意：paymentdevice的接口*

- **getInstance:创建PaymentDeviceManager实例**

```
public static PaymentDeviceManager getInstance()
```

参数	返回值	抛出	备注
无	PaymentDeviceManager对象	PaymentDeviceException	

- **controlPower:给支付模块上/下电**

```
public void controlPower(boolean on)
```

参数	返回值	抛出	备注
`on`	无	PaymentDeviceException	on - true:上电 false:下电

- **controlPower:给支付模块上/下电（带参数）**

```
public void controlPower(java.lang.String exdev,boolean on)
```

参数	返回值	抛出	备注
`exdev`	无	PaymentDeviceException	设备名称
`on`	无	同上	true:上电，false：下电

- **wakeup:唤醒支付模块**

```
public void wakeup()
```

参数	返回值	抛出	备注
无	无	PaymentDeviceException	

- **wakeup:唤醒支付模块（带参数）**

```
public void wakeup(java.lang.String exdev)
```

参数	返回值	抛出	备注
`exdev`	无	PaymentDeviceException	设备名称

- **getStatus:查询支付模块状态**

```
public int getStatus()
```

参数	返回值	抛出	备注
无	如下表格	PaymentDeviceException	

返回值数值如下:

参数	状态
1	mpos未上电
2	mpos上电但未就绪
3	mpos就绪
4	mpos休眠

- **getStatus:扩展接口，获取特定支付设备的状态**

```
public int getStatus(java.lang.String exdev)
```

参数	返回值	抛出	备注
`exdev`	无	PaymentDeviceException	设备名

返回值数值如下:

参数	状态
1	mpos未上电
2	mpos上电但未就绪
3	mpos就绪
4	mpos休眠

- **getExDeviceInfos:获取当前的扩展名称列表**

```
public java.lang.String[] getExDeviceInfos()
```

参数	返回值	抛出	备注
无	设备名称数组	PaymentDeviceException	

- **mutiExDevSupport:当前是否支持多外部设备**

```
public boolean mutiExDevSupport()
```

参数	返回值	抛出	备注
无	无	PaymentDeviceException	true:支持，false：不支持

> *注意：外接USB设备R15未上电或者移除时，返回false*

- **finalize:关闭 RPC并且释放 manager**

```
public void finalize()
```

参数	返回值	抛出	备注
无	无	PaymentDeviceException	finalize 在类中 java.lang.Object

7 屏幕显示

商用设备支持以下方式来实现不同场景下客户需求的屏幕显示。

7.1 全屏显示

商用设备使用场景为全屏模式，可以使用以下方法来处理，示例如下。
开发应用可以在Activity中添加如下代码实现全屏化：

```
    @Override
```

```
    protected void onResume() {
        super.onResume();
        hideSystemUI();
    }

   private void hideSystemUI() {
        getWindow().getDecorView().setSystemUiVisibility(
                View.SYSTEM_UI_FLAG_LAYOUT_STABLE
                        | View.SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION
                        | View.SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN
                        | View.SYSTEM_UI_FLAG_HIDE_NAVIGATION /*隐藏导航栏
*/
                        | View.SYSTEM_UI_FLAG_FULLSCREEN /*隐藏状态栏*/
                        | View.SYSTEM_UI_FLAG_IMMERSIVE_STICKY);
    }
```

7.2 双屏显示

商用设备使用场景为双屏模式，提高了与客户交互的便捷性，以下主要介绍三种情景的调用方法。

7.2.1 情景展示

1.应用在副屏中打开activity

副屏打开的activity需要继承于 presentation 类

```
public class SecondaryActivity extends Presentation {
    public SecondaryActivity(Context outerContext, Display display) {
        super(outerContext, display);
        //绑定activity的布局，其他执行代码
    }
}
```

主屏中来启动副屏的activity

```
SecondaryActivity myPresentation = new SecondaryActivity(MainActivity.this
, displays[1]);
myPresentation.show();
```

2.将应用中的某个activity声明为副屏的luancher

```
<activity
 android:name=".secondScreenMainActivity"
 android:exported="true">
 <intent-filter>
 <action android:name="android.intent.action.MAIN"/>
 <category android:name="android.intent.category.SECONDARY_HOME"/>
 <category android:name= "android.intent.category.DEFAULT"/>
 </intent-filter>
</activity>
```

7.3 拓展

1.chrome浏览器在双屏中显示不同的网页

> *注意：以下是以E800的Andoroid12的方案来解决的*

操作步骤：

【1】通过chrome浏览器的设置来开打新窗口，新窗口会在副屏显示

【2】通过移至其他窗口来选择主屏要显示的窗口

2.解决副屏显示比例错误的 Bug

副屏有可能会出现尺寸比例或者字体大小异常显示的问题，解决问题我们可以通过ADB设置参数修改或者联系兆珑技术人员获取新的OS进行优化

操作步骤：

【1】启动 ADB，输入 adb shell getprop persist.sys.rotation.einit，查看返回值是否为1。

【2】若没有返回值或者返回值不为 1，执行adb shell setprop persist.sys.rotation.einit val 命令来修改设备显示方向

【3】val 可以为1，2，3，4（1为旋转值 90 度，2为旋转值 180 度，3为旋转值 270 度，0为旋转值 0 度），修改旋转值后重新启动设备即可获得正确的显示比

8 NFC功能调用

兆珑的部分商用设备都内置有NFC硬件模块，以下是屏内置nfc硬件模块的机型列表。

适用机型
M8100，K20，K21，L161X

8.1 适配方式

根据以下适配步骤即可制作出简易的nfc测试应用，如开发过程中出现异常，请联系兆珑开发人员。

- **添加应用NFC权限**
  在AndroidManifest.xml（以下都简称为Manifest）中添加以下权限说明

```
<uses-permission android:name="android.permission.NFC" />
<uses-feature android:name="android.hardware.nfc" android:required="true"
/>
```

- **添加NFC的支持列表**
  新建一个nfc_tech_filter.xml，把目前支持的nfc类型添加进文件中

```
    <tech-list>
        <tech>android.nfc.tech.NfcA</tech>
        <tech>android.nfc.tech.NfcB</tech>
        <tech>android.nfc.tech.IsoDep</tech>
    </tech-list>
```

- **添加NFC过滤器**

  在Manifest的activity中添加NFC的过滤器以及nfc的支持列表，当NFC组件被调

用时才启动

```
<!-- 添加 NFC intent 过滤器 -->
<intent-filter>
    <action android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" /></intent-filter>
<intent-filter>
    <action android:name="android.nfc.action.TECH_DISCOVERED" /></intent-f
ilter>
<intent-filter>
    <action android:name="android.nfc.action.TAG_DISCOVERED" />
</intent-filter>
<meta-data
     android:name="android.nfc.action.TECH_DISCOVERED"
     android:resource="@xml/nfc_tech_filter" />
```

- **初始化NFC功能**
  在主程序中先获取nfc的实例，在依次判断设备是否支持nfc，是否打开nfc功能

```
//获取nfc的实例
nfcAdapter = NfcAdapter.getDefaultAdapter(this);
//判断硬件是否支持nfc功能
if (nfcAdapter == null)
//判断系统是否打开nfc功能
if (!nfcAdapter.isEnabled())
```

- **创建处理nfc功能**
  PendingIntent：nfc检测功能需要特殊的Intent,保证nfc识别时能预操作处理
  onNewIntent：activity运行过程中，nfc功能会在onNewtent中处理

> *注意：*
> *1.tag只是nfc卡的UID，并不是实际卡号，输出tag需要把字节转化为十六进制输出*

- **onResume和onPause保持nfc的前台调度**

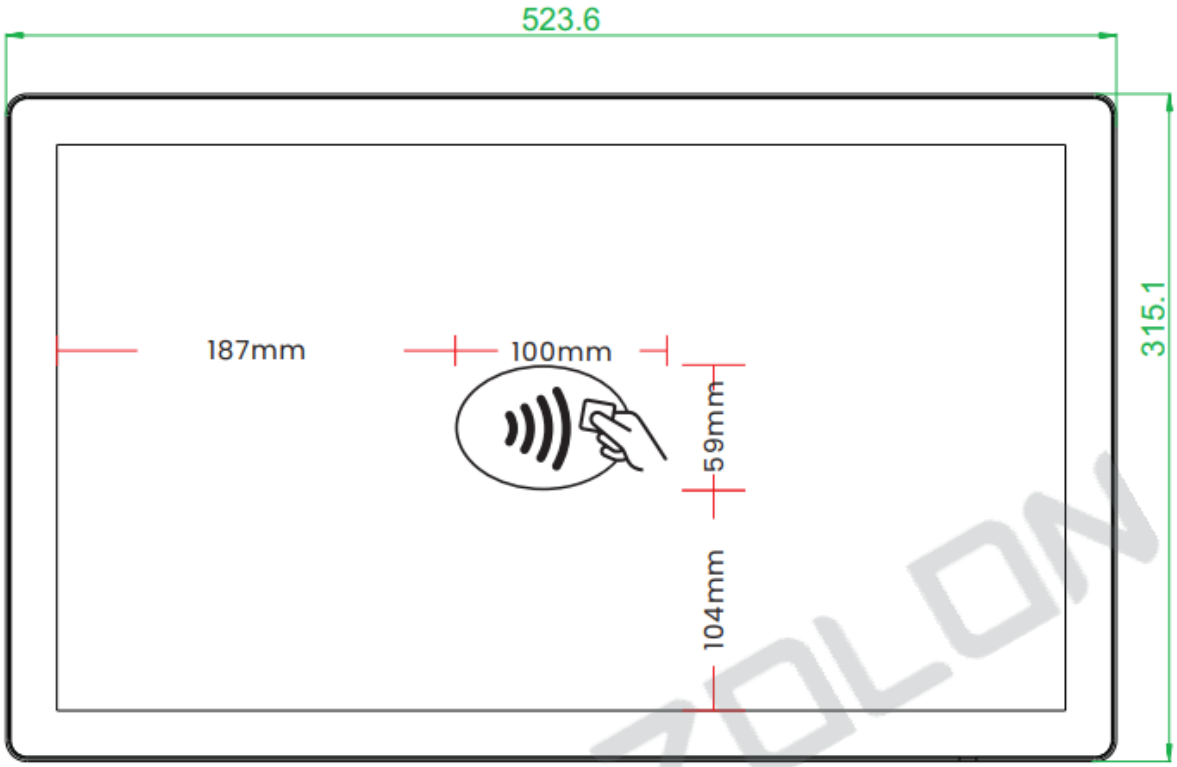
```
//onResume
nfcAdapter.enableForegroundDispatch(this, mPendingIntent, mFilters, mTechLists);
//onPause
nfcAdapter.disableForegroundDispatch(this);
```

编写好以上步骤就可制作出nfc功能的测试工具，目前仅作为屏内置nfc模块的测试场景

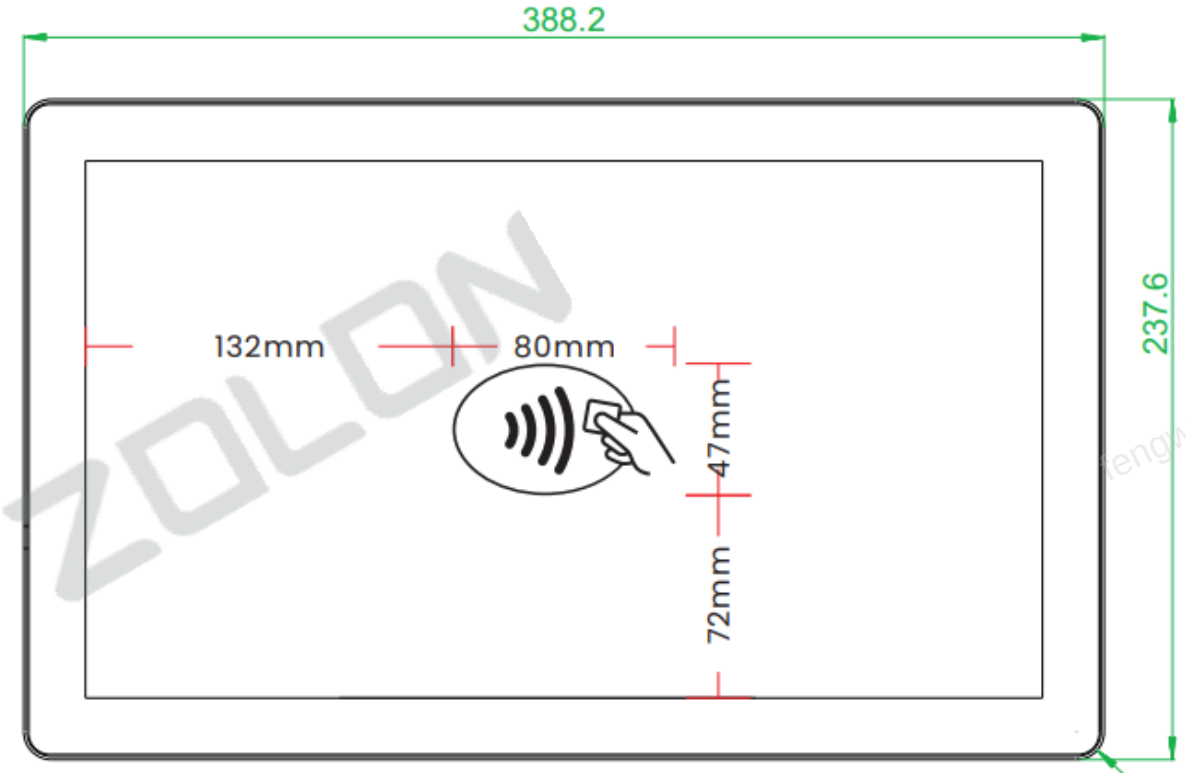
8.2 屏内nfc方位图

以下展示是兆珑部分屏内nfc硬件模块位置的商用设备图，可根据实际机型环境进行间距的设置。

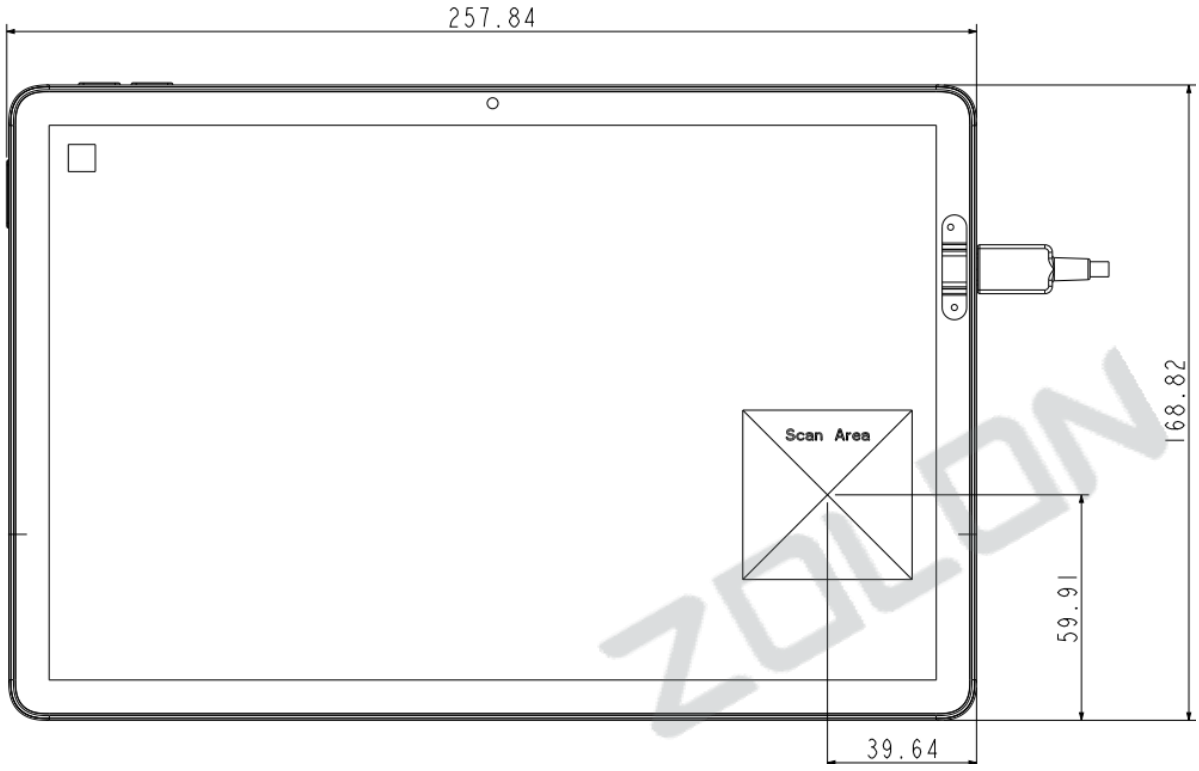
- **K2220nfc硬件模块屏内方位图**



- **K2160nfc硬件模块屏内方位图**



- **M8100nfc硬件模块屏内方位图**



9 串口