

Gprinter Android SDK使用手册

sdk大致分为两部分，打印机与客显，具体可点击下面查看具体部分

- [打印机部分](#)
- [客显部分](#)

打印机部分的**sdk**使用

前言

佳博提供Gprinter Android SDK给用户开发和使用佳博品牌的打印机。
用户通过使用Gprinter Android SDK能更快速、更高效的开发AndroidAPP。

下载GprinterSDK

最新版的GprinterSDKV2.x.x可在佳博官网（<http://www.gainscha.cn/cn/download.aspx?current=down>）下载。

新建工程，向工程的**libs**文件中拷贝库文件

- gprinter-x.x.x.jar;
- jcc-bate-0.7.3.jar;
- ksoap2-android-assembly-2.5.2-jar-with-dependencies.jar;
- xUtils-2.6.14.jar;
- armeabi目录;
- armeabi-v7a目录;
- x86目录。

注册服务和权限

注册服务在**AndroidManifest.xml**文件中添加服务，gprinter-x.x.x.jar中提供了打印服务,不添加服务可能造成无法连接，请确认配置好该文件。

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sample"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="14"
        android:targetSdkVersion="22" />

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.hardware.usb.accessory" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <uses-permission android:name="android.permission.GET_TASKS" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-feature android:name="android.hardware.usb.host" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <service
            android:name="com.gprinter.service.GpPrintService"
            android:enabled="true"
            android:exported="true"
            android:label="GpPrintService" >
            <intent-filter>
                <action android:name="com.gprinter.aidl.GpPrintService" />
            </intent-filter>
        </service>

        <service android:name="com.gprinter.service.AllService" >
        </service>
    </application>

</manifest>

```

添加aidl文件

新建包，包名为com.gprinter.aidl,向包中添加文件，文件名为GpService.aidl内容为：

```
package com.gprinter.aidl;
interface GpService{
    int openPort(int PrinterId,int PortType,String DeviceName,int PortNumber);
    void closePort(int PrinterId);
    int getPrinterConnectStatus(int PrinterId);
    int printeTestPage(int PrinterId);
    void queryPrinterStatus(int PrinterId,int Timesout,int requestCode);
    int getPrinterCommandType(int PrinterId);
    int sendEscCommand(int PrinterId, String b64);
    int sendLabelCommand(int PrinterId, String b64);
    void isUserExperience(boolean userExperience);
    String getClientID();
    int setServerIP(String ip, int port);
}
```

启动并绑定PrinterPrintService服务

```
private PrinterServiceConnection conn = null;

class PrinterServiceConnection implements ServiceConnection {
    @Override
    public void onServiceDisconnected(ComponentName name) {
        Log.i("ServiceConnection", "onServiceDisconnected() called");
        mGpService = null;
    }

    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        mGpService = GpService.Stub.asInterface(service);
    }
}

@Override
public void onCreate(Bundle savedInstanceState) {
    conn = new PrinterServiceConnection();
    Intent intent = new Intent(this, GpPrintService.class);
    bindService(intent, conn, Context.BIND_AUTO_CREATE); // bindService
}
```

使用打印服务的AIDL接口

- int openPort(int PrinterId,int PortType,String DeviceName,int PortNumber);

(具体查看工程GpSample中的PrinterConnectDialog.java文件)

打开客户端与打印机通讯端口，该接口会通过广播返回PrinterId的打印机的连接状态。广播的action为**GpCom.ACTION_CONNECT_STATUS**，EXTRA分别为**GpPrintService.CONNECT_STATUS**与**GpPrintService.PRINTER_ID**，对应得到连接的状态值与打印机的索引id

PrinterId为连接打印机的索引id

PortType为连接的类型，具体类型可以查看API文档中的PortParameters类说明

DeviceName为连接设备的地址，一般为蓝牙地址，usb设备名或者ip地址

PortNumber为使用网口或者wifi连接打印机下使用（默认端口号为**9100**），蓝牙与usb连接均可设置为0。

- void closePort(int PrinterId);

断开客户端与打印机通讯端口（断开的时候，如果需要再次连接的话，请确保打

印机的状态已经是**GpDevice.STATE_NONE**的状态)

PrinterId为连接打印机的索引id

- **int getPrinterConnectStatus(int PrinterId);**

获取打印机当前的连接状态

PrinterId为连接打印机的索引id

- **int printeTestPage(int PrinterId);**

索引为PrinterId的打印机打印测试页

PrinterId为连接打印机的索引id

- **void queryPrinterStatus(int PrinterId,int Timesout,int requestCode);**

(具体查看GpSample中的MainActivity.java文件中的广播声明)

查询打印机的实时状态，该查询将会发送一

个**action**为**GpCom.ACTION_DEVICE_REAL_STATUS**的广播，请在项目中去接收这个广播，该广播中带有的**EXTRA**有两个，分别

为**GpCom.EXTRA_PRINTER_REQUEST_CODE**（得到该接口调用时的requestCode）与**GpCom.EXTRA_PRINTER_REAL_STATUS**（打印机状态），两个值都为int类型。

PrinterId为打印机索引

requestCode为查询请求码

- **int getPrinterCommandType(int PrinterId);**

获取打印机的命令类型，是ESC命令还是Label命令

PrinterId为打印机索引

- **int sendEscCommand(int PrinterId, String b64);**

发送票据内容给索引为PrinterId的打印机

PrinterId为打印机索引

b64为字节数组进行base64编码后的数据

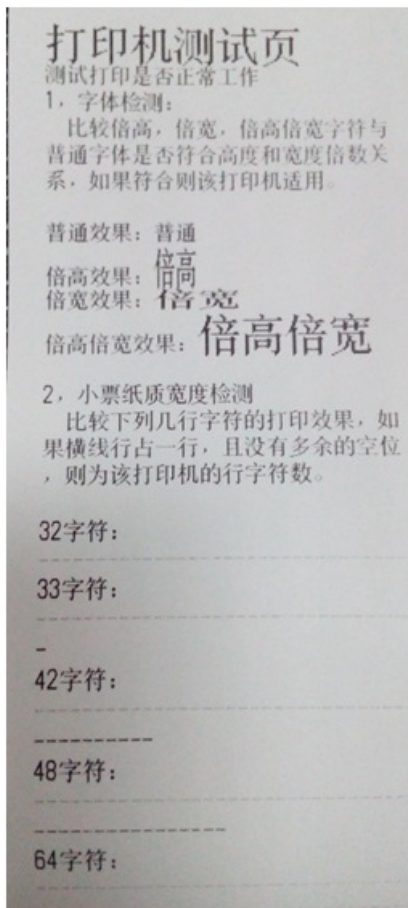
- **int sendLabelCommand(int PrinterId, String b64);**

发送标签内容给索引为PrinterId的打印机

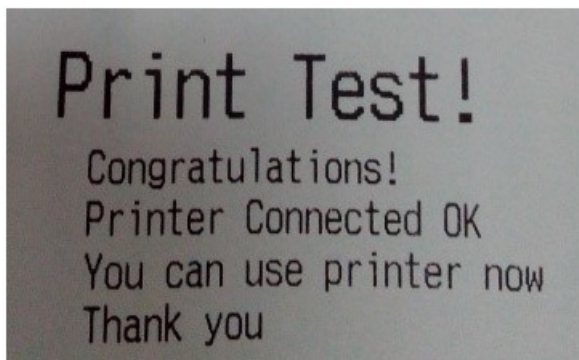
b64为字节数组进行base64编码后的数据

向打印机发送数据

- 票据打印机测试页内容如图



- 标签打印机测试页内容如图



使用打印命令组装内容

- 组装票据

```
EscCommand esc = new EscCommand();
esc.addPrintAndFeedLines((byte)3);
// 设置打印居中
esc.addSelectJustification(JUSTIFICATION.CENTER);

// 设置为倍高倍宽
esc.addSelectPrintModes(FONT.FONTA,
                        ENABLE.OFF,
                        ENABLE.ON,
                        ENABLE.ON, ENABLE.OFF);

// 打印文字
esc.addText("Sample\n");
esc.addPrintAndLineFeed();
```

```

/*打印文字*/
//取消倍高倍宽
esc.addSelectPrintModes(FONT.FONTA,
                        ENABLE.OFF,
                        ENABLE.OFF,
                        ENABLE.OFF,
                        ENABLE.OFF);

esc.addSelectJustification(JUSTIFICATION.LEFT); // 设置打印左对齐
esc.addText("Print text\n"); // 打印文字
esc.addText("Welcome to use printer!\n"); // 打印文字
esc.addPrintAndLineFeed();
/*打印图片*/
esc.addText("Print bitmap!\n"); // 打印文字
Bitmap b = BitmapFactory.decodeResource(getResources(),
R.drawable.printer);
esc.addRastBitImage(b,b.getWidth(),0); //打印图片
/*打印一维条码*/
esc.addText("Print code128\n"); // 打印文字

// 设置条码可识别字符位置在条码下方
esc.addSelectPrintingPositionForHRICharacters(HRI_POSITION.BELOW);
// 设置条码高度为60点
esc.addSetBarcodeHeight((byte)60);
// 设置条码单元宽度为1点
esc.addSetBarcodeWidth((byte)1);
// 打印Code128码
esc.addCODE128(esc.genCode("123456789"));
esc.addCODE128(esc.genCodeB("Gprinter"));
esc.addPrintAndLineFeed();
/*QRCode命令打印
此命令只在支持QRCode命令打印的机型才能使用。
在不支持二维码指令打印的机型上，则需要发送二维条码图片
*/
esc.addText("Print QRcode\n"); // 打印文字
esc.addSelectErrorCorrectionLevelForQRCode((byte)0x31); // 设置纠错等级
esc.addSelectSizeOfModuleForQRCode((byte)3); // 设置qrcode模块大小
esc.addStoreQRCodeData("www.printer.com.cn"); // 设置qrcode内容
esc.addPrintQRCode(); // 打印QRCode
esc.addPrintAndLineFeed();
/*打印文字*/
esc.addSelectJustification(JUSTIFICATION.CENTER); // 设置打印居中
esc.addText("Completed!\r\n"); // 打印结束
// 开钱箱
esc.addGeneratePlus(LabelCommand.FOOT.F5, (byte) 255, (byte) 255);
// 立即开钱箱
//esc.addGeneratePluseAtRealtime(LabelCommand.FOOT.F2, (byte) 8);
esc.addPrintAndFeedLines((byte)8);
Vector<Byte>datas = esc.getCommand(); // 发送数据
Byte[] Bytes = datas.toArray(new Byte[datas.size()]);
byte[] bytes = GpUtils.ByteTo_byte(Bytes);
String str = Base64.encodeToString(bytes, Base64.DEFAULT);
intrel;
try {
    rel = mService.sendEscCommand(mPrinterIndex, str);
    GpCom.ERROR_CODE r=GpCom.ERROR_CODE.values()[rel];
    if(r != GpCom.ERROR_CODE.SUCCESS){
        Toast.makeText(getApplicationContext(),GpCom.getErrorText(r),
        Toast.LENGTH_SHORT).show();
    }
} catch (RemoteException e) {
    e.printStackTrace();
}
}

```

效果选下图



- 组装标签

```
LabelCommand label = new LabelCommand();

// 设置标签尺寸, 按照实际尺寸设置
label.addSize(60, 60);

// 设置标签间隙, 按照实际尺寸设置, 如果为无间隙纸则设置为0
label.addGap(0);

// 设置打印方向
label.addDirection(DIRECTION.BACKWARD, MIRROR.NORMAL);

// 设置原点坐标
label.addReference(0, 0);

// 撕纸模式开启
label.addTear(ENABLE.ON);

// 清除打印缓冲区
label.addCls();

// 绘制简体中文
label.addText(20, 20,
    FONTTYPE.SIMPLIFIED_CHINESE,
    ROTATION.ROTATION_0,
    FONTMUL.MUL_1,
    FONTMUL.MUL_1,
    "Welcome to use Printer!");

// 绘制图片
Bitmap b = BitmapFactory
    .decodeResource(getResources(), R.drawable.printer);

label.addBitmap(20, 50,
    BITMAP_MODE.OVERWRITE,
    b.getWidth(),
    b);

label.addQRCode(250, 80, EEC.LEVEL_L, 5,
    ROTATION.ROTATION_0,
    "gprinter.com");
```

```
//绘制一维条码
label.add1DBarcode(20,250,
                    BARCODETYPE.CODE128,
                    100,
                    READABEL.EANBEL,
                    ROTATION.ROTATION_0,
                    "printer");

label.addPrint(1,1); // 打印标签
label.addSound(2, 100); //打印标签后蜂鸣器响
Vector<Byte>datas = label.getCommand(); //发送数据
Byte[] Bytes = datas.toArray(new Byte[datas.size()]);
byte[] bytes = ArrayUtils.toPrimitive(Bytes);
String str = Base64.encodeToString(bytes, Base64.DEFAULT);
intrel;
try {
    rel = mGpService.sendLabelCommand(mPrinterIndex, str);
    GpCom.ERROR_CODE r=GpCom.ERROR_CODE.values()[rel];
    if(r != GpCom.ERROR_CODE.SUCCESS){
        Toast.makeText(getApplicationContext(),
                        GpCom.getErrorText(r),
                        Toast.LENGTH_SHORT)
            .show();
    }
} catch (RemoteException e) {
    e.printStackTrace();
}
```

效果选下图



客显部分的sdk使用

具体的接口方法，请查看**API文档**中的**CustomerDisplay**类的说明

打开端口、关闭端口和获取端口状态

具体如下图所示



- 打开端口

```
/* 打开端口 */
private CustomerDisplay port;

private void openPort() {
    port = CustomerDisplay.getInstance(this);
    try {
        //打开端口
        port.openPort();
    } catch (IOException e) {
        e.printStackTrace();
    }
    //设置监听回调数据
    port.setReceivedListener(this);
    port.getBacklightTimeout();
}
```

- 关闭端口

```
// 关闭端口
private void closePort() {
    if (port != null) {
        port.closeSerialPort();
        toast("关闭端口");
    }
}
```

- 获取端口状态

```
// 获取端口状态
private void getPortStatus() {
    if (port != null) {
        toast("端口状态: " + (port.isPortOpen() ? "打开" : "关闭"));
    } else {
        toast("端口状态: 关闭");
    }
}
}
```

- 设置背光灯亮度

```
// 设置背光灯亮度
private void setBrightness() {
    Spinner spinner = (Spinner) findViewById(R.id.spinner_brightness);
    byte brightness=Byte.valueOf(spinner.getSelectedItem().toString());
    port.setBrightness(brightness);
}
}
```

- 设置客显对比度

```
private void setContrast() {
    Spinner spinner = (Spinner) findViewById(R.id.spinner_contrast);
    byte contrast=Byte.valueOf(spinner.getSelectedItem().toString());
    port.setContrast(contrast);
}
}
```

- 关闭背光灯

```
private void turnOffBacklight() {
    port.setBacklight(false);
}
}
```

- 复位客显

```
// 复位客显
private void reset() {
    port.reset();
}
}
```

- 清屏

```
// 清屏
private void clear() {
    port.clear();
}
}
```

- 显示位图

```
// 显示位图
private void displayBitmap() {
    Bitmap bitmap=BitmapFactory
        .decodeStream(getResources().openRawResource(R.raw.fl));
    port.displayBitmap(bitmap, 48);
}
}
```

- 打开背光灯

```
// 打开背光灯
private void turnOnBacklight() {
    port.setBacklight(true);
}
```

- 设置客显背光灯超时时间

```
// 设置客显背光灯超时时间
private void setDisplayTimeout() {
    String displayTimeoutStr = mEtDisplayTimeout.getText().toString();
    if (TextUtils.isEmpty(displayTimeoutStr)) {
        Toast.makeText(CustomerDiaplayActivity.this,
            R.string.str_input_display_timeout,
            Toast.LENGTH_SHORT).show();

        return;
    }
    mTimeoutMsg.setText(R.string.str_set_successful);
    int timeout = Integer.parseInt(displayTimeoutStr);
    port.setBacklightTimeout(timeout);
}
```

- 设置客显光标位置

```
// 设置客显光标位置
private void setCursorPosition() {
    String xStr = mEtX.getText().toString();
    String yStr = mEtY.getText().toString();
    if (TextUtils.isEmpty(xStr) || TextUtils.isEmpty(yStr)) {
        Toast.makeText(this,
            R.string.str_please_input_x_y,
            Toast.LENGTH_SHORT).show();

        return;
    }
    int x = Integer.parseInt(xStr);
    int y = Integer.parseInt(yStr);
    port.setCursorPosition(x, y);
}
```

- 当前光标位置输入文字，光标位置不变

```
private void setTextCurrentCursor() {
    String inputText = mInputText.getText().toString();
    port.setTextCurrentCursor(inputText);
}
```

- 当前光标位置输入文字，光标位置跟随输入的文字

```
private void setTextBebindCursor() {
    String inputText = mInputText.getText().toString();
    port.setTextBebindCursor(inputText);
}
```

获取客显信息,以及实现信息回调

```
/**
```

```

    * get 部分
    */

    private void getDisplayStatus() {
        port.getBacklight();
    }

    private void getDisplayTimeout() {
        port.getBacklightTimeout();
    }

    private void getCursorPosition() {
        port.getCursorPosition();
    }

    private void getDisplayRowAndColumn() {
        port.getDisplayRowAndColumn();
    }

    /**
     * 获取客显示屏的状态开启或关闭
     *
     * @param isOpen
     *      客显示屏背光灯开启或关闭
     */
    @Override
    public void onPortOpen(boolean isOpen) {
        if (isOpen) {
            toast("打开端口成功");
        } else {
            toast("打开端口失败");
        }
    }

    /**
     * 获取客显示屏背光灯开启或关闭
     *
     * @param isOn
     *      客显示屏背光灯开启或关闭
     */
    @Override
    public void onBacklightStatus(final boolean isOn) {
        Log.d("==onBacklightStatus==", String.valueOf(isOn));

        toast("==onBacklightStatus== 背光灯状态->" + String.valueOf(isOn));
    }

    /**
     * 获取客显示屏光标的位置
     *
     * @param x
     *      横坐标
     * @param y
     *      纵坐标
     */
    @Override
    public void onCursorPosition(final int x, final int y) {
        toast("==onCursorPosition==x = " + x + ",y = " + y);
        Log.d("==onCursorPosition==", "x坐标 = " + x + ",y坐标 = " + y);
    }

```

```
/**
 * 获取客显屏的行和列
 * @param row
 *      行
 * @param column
 *      列
 */
@Override
public void onDisplayRowAndColumn(final int row, final int column) {
    toast("行数 = " + row + ",列数 = " + column);
    Log.d("==onCursorPosition==", "row = " + row + ",column = " + column);
}

/**
 * 获取客显屏背光灯超时时间
 *
 * @param timeout
 *      单位: 秒
 */
@Override
public void onBacklightTimeout(final int timeout) {
    toast("超时时间 = " + timeout);
    Log.d("==onBacklightTimeout==", "timeout = " + timeout);
}

/**
 * 更新客显固件完成回调方法
 */
@Override
public void onUpdateSuccess() {
    //nothing to do
}

@Override
public void onUpdateFail(String error) {
    //nothing to do
}
```