基础包使用说明:

本套开发指南是针对 Eclipse 开发工具编写的。

一、包介绍:

1. ipaynow_base_v_XXX.jar (现在支付基础包，调起支付插件的节点)
2. libplugin_phone.so(现在支付动态库)

二、接入步骤:

(一) 配置环境:

1.jar 包导入:

将技术包中 jars\ipaynow_base_v_XXX.jar 拷贝到商户项目目录中 libs 目录下。

2.so 动态库导入:

将技术包中 so\libplugin_phone.so copy 到商户项目目录中 libs\<与目录同名的 cpu 架构名>目录下，可参考 demo 给出的目录结构。

3.在 AndroidManifest 文件中添加基础包所需权限，**除此之外请商户查阅各子包文档并添加支付渠道所需的权限信息**:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

4.在 AndroidManifest 文件中添加基础包的 Activity 组件,**当前的 Activity 只支持竖屏状态**，所以配置信息先不要更改:

```
<!-- 现在支付 -->
<activity
    android:name="com.ipaynow.plugin.presenter.PayMethodActivity"
    android:configChanges="keyboardHidden|navigation|orientation|screenS
ize"
    android:theme="@android:style/Theme.Dialog"
    android:exported="false"
    android:screenOrientation="portrait" >
</activity>
```

5.进行插件的初始化操作:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
// 初始化支付插件，传入 context 对象
        IpaynowPlugin.getInstance().init(MainActivity.this);
        setContentView(R.layout.activity_main);
    }
}
```

（二）调起支付请求接口:(该接口必须在主线程调用)

支付请求流程介绍:支付请求流程为商户生成订单信息并在服务器对现在支付规定的订单信息进行签名后，将订单信息、商户信息、签名等发送给现在支付插件进行支付请求的整个流程。(此文档只叙述该流程中客户端部分,服务端部分参看现在支付服务端文档)

第一步.创建支付接口请求参数：

方式一（客户端传统请求方式相对更安全）：

该方式为订单信息在服务端进行拼接以及签名并返回客户端的流程。(由于该部分主要的拼接与签名均在服务器端处理，所以请参看现在支付服务器端文档)

服务端应该返回符合现在支付接口规范的请求信息 requestMessage 供第二步调起聚合支付请求接口执行。

方式二（客户端简便请求方式）：

1) **创建待签名串对象:**

```
PreSignMessageUtil preSign=new PreSignMessageUtil();
```

2) **将属性进行赋值（具体属性请看附录表单）：**

```
preSign.appId="xxxxxxx";
PreSign.mhtOrderName="支付样例-手机版";
preSign.mhtOrderType="01";
preSign.mhtCurrencyType="156";
preSign.mhtOrderAmt="10";
preSign.mhtOrderDetail="关于支付的演示";
preSign.mhtOrderTimeOut="3600";
```

3) **生成待签名串：**

```
String preSignStr=preSign.generatePreSignMessage();
```

4) **生成签名串（考虑到信息安全，签名串要由商户自己的服务端生成，生成方法请查看支付服务器端文档）：**

商户服务器仅需做两次 MD5 加密并将"签名值"以及"签名方法"串发送给客户端

5) **将待签名串与服务器返回的签名串拼接成支付接口请求参数：**

```
preSignStr+"&"+ 服务器返回的签名串
```

待签名串 preSignStr 样例：

```
"appId=1408709961320306&consumerId=456123&consumerName=test&mhtCharset=UTF-8&mhtCurrencyType=156&mhtOrderAmt=10&mhtOrderDetail= 关于支付的演示 &mhtOrderName=支付样例-手机版 mhtOrderNo=20150729165543 &mhtOrderStartTime=20150729165540& mhtOrderTimeOut=3600&mhtOrderType=01&mhtReserved=test& notifyUrl=http://localhost:10802/&payChannelType=12"
```

服务器返回串样例:

```
"mhtSignature=588744b272f39e26c729fac78d7096c9&mhtSignType=MD5"
```

第二步.调起聚合支付请求接口（必须在主线程调用支付接口）：

```
//该接口需要传入符合现在支付接口规范的请求信息。
//支付接口请求参数格式 requestMessage = preSignStr+"&"+ 服务器返回的签名串
IpaynowPlugin.getInstance().pay(requestMessage);
```

（三）接收支付响应结果：

（1）在非 Activity 类中接收通知(推荐商户使用该方法)：

第一步：调用 setCallResultReceiver(ReceivePayResult);//设置接收通知的对象

第二步：在接收通知的类中实现 ReceivePayResult 接口中的方法：

```
@Override
public void onIpaynowTransResult(ResponseParams arg0) {
  String respCode = arg0.respCode;
  String errorCode = arg0.errorCode;
  String errorMsg = arg0.respMsg;
  StringBuilder temp = new StringBuilder();
  if (respCode.equals("00")) {
    temp.append("交易状态:成功");
  } else if (respCode.equals("02")) {
    temp.append("交易状态:取消");
  } else if (respCode.equals("01")) {
    temp.append("交易状态:失败").append("\n").append("错误码:").append(errorCode)
        .append("原因:" + errorMsg);
  } else if (respCode.equals("03")) {
    temp.append("交易状态:未知").append("\n").append("原因:" + errorMsg);
  } else {
    temp.append("respCode=").append(respCode).append("\n").append("respMsg=").append(errorMsg);
  }
  Toast.makeText(this, temp.toString(), Toast.LENGTH_LONG).show();
}
```

（2）在 Activity 中接收通知(使用老版本插件的商户使用此方法)：

第一步:调用 setCallResultActivity(Activity)//传入回调用的 Activity 对象。

第二步：实现 Activity 中的 onActivityResult(int requestCode,int resultCode,Intent data)。

当用户支付完毕、支付失败、中途取消支付均会通过该方法通知商户 APP。通知时 requestCode=0，resultCode=1 样例代码：

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (data == null) {
    return;
  }
  if (requestCode == 0 && resultCode == 1) {
    Bundle bundle = data.getExtras();
    String respCode = bundle.getString("respCode");
    String errorCode = bundle.getString("errorCode");
    String errorMsg = bundle.getString("respMsg");
    StringBuilder temp = new StringBuilder();
    if (respCode.equals("00")) {
      temp.append("交易状态:成功");
    } else if (respCode.equals("02")) {
      temp.append("交易状态:取消");
    } else if (respCode.equals("01")) {
      temp.append("交易状态:失败").append("\n").append("错误码:").append(errorCode)
          .append("原因:" + errorMsg);
    } else if (respCode.equals("03")) {
      temp.append("交易状态:未知").append("\n").append("原因:" + errorMsg);
    } else {
      temp.append("respCode=").append(respCode).append("\n").append("respMsg=").append(errorMsg);
    }
    Toast.makeText(this, temp.toString(), Toast.LENGTH_LONG).show();
  }
}
```

注：以上步骤可参考提供的实例 Demo。

（四）代码混淆：

打包应用如需代码混淆，需要保留与支付 SDK 相关的方法不被混淆，请加入以下配置，否则会出错：

```
-keep class com.alipay.android.app.IAlixPay{*;}
-keep class com.alipay.android.app.IAlixPay$Stub{*;}
-keep class com.alipay.android.app.IRemoteServiceCallback{*;}
-keep class com.alipay.android.app.IRemoteServiceCallback$Stub{*;}
-keep class com.alipay.sdk.app.PayTask{ public *;}
-keep class com.alipay.sdk.auth.AlipaySDK{ public *;}
-keep class com.alipay.sdk.auth.APAuthInfo{ public *;}
-keep class com.alipay.mobilesecuritysdk.*
-keep class com.ut.*
-keep class cn.gov.pbc.tsm.*{*;}
-keep class com.UCMobile.PayPlugin.*{*;}
-keep class com.unionpay.*{*;}
-dontwarn com.unionpay.**

-keep class com.ipaynow.plugin.api.IpaynowPlugin{
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.utils.PreSignMessageUtil{
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.utils.MerchantTools{
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.manager.route.dto.RequestParams{
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.manager.route.dto.ResponseParams{
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.manager.route.impl.ReceivePayResult{
    <fields>;
    <methods>;
}
-keep class com.alipay.android.app.IAlixPay {
```

```
    <fields>;
    <methods>;
}
-keep class com.ipaynow.plugin.utils.StringUtils{
    <fields>;
    <methods>;
}
-keep class com.alipay.android.app.IRemoteServiceCallback {
    <fields>;
    <methods>;
}
```

附录 A

调起插件接口信息规范:

字段名称	字段 Key	格式	必填	备注
商户应用唯一标识	appId	String(1,40)	Y	现在支付业务提供
商户订单号	mhtOrderNo	String(1,40)	Y	字母、数字
商户商品名称	mhtOrderName	String(1,40)	Y	
商户交易类型	mhtOrderType	String(2)	Y	01 普通消费
商户订单币种类型	mhtCurrencyType	String(3)	Y	156 人民币
商户订单交易金额	mhtOrderAmt	String(1,22)	Y	单位(人民币)：分 整数，无小数点
商户订单详情	mhtOrderDetail	String(1,1000)	Y	
商户订单超时时间	mhtOrderTimeOut	Number(4,0)	N	60~3600 秒，默认 3600
商户订单开始时间	mhtOrderStartTime	String(14)	Y	yyyyMMddHHmmss
商户后台通知 URL	notifyUrl	String(1,200)	Y	HTTP 协议
商户字符编码	mhtCharset	定值	Y	UTF-8
渠道类型	payChannelType	定值		银联支付:11 支付宝支付:12; 微信支付:13; 微信插件支付:1310 百度支付:50 QQ 支付:25 现在支付网关界面：空串 如果传入空串，请在项目中添加单独的 view 子包
是否支持信用卡	mhtLimitPay	定值	N	默认为支持，不支持请指定 no_credit
商户保留域	mhtReserved	String(100)	N	商户可以对交易进行标记，现在支付将原样返回给商户
商户签名方法	mhtSignType	定值	Y	MD5
商户数据签名	mhtSignature	String(1,64)	Y	签名逻辑见接口附录说见 5.1 BXXX 交易的 MD5 签名逻辑说明。除如下字段外，其它字段都参与 MD5 签名。排除的有：mhtSignType, mhtSignature

附录 B

插件通知接口信息说明:

字段名称	字段 Key	必填	备注
响应码	respCode	Y	交易状态成功:00 交易状态失败:01 交易状态取消:02 交易状态未知:03
错误码	errorCode	N	失败时会返回
响应信息	respMsg	N	未知和失败时会返回

插件错误码说明:

字段名称	错误信息说明
PE001	插件不支持该渠道交易
PE002	网络连接超时
PE003	现在支付获得交易号失败
PE004	渠道方交易失败
PE005	支付插件其他未知错误
PE006	渠道方不支持此种支付方式
PE007	未安装渠道方支付用客户端
PE008	支付用渠道客户端版本过低
PE009	渠道交易结果未知
PE010	渠道方其他未知失败
PE011	现在支付接口其他失败情况
PE012	商户未添加渠道子包
PE013	商户未添加视图控件子包