现在支付

中小开发者商户客户端

接入指南

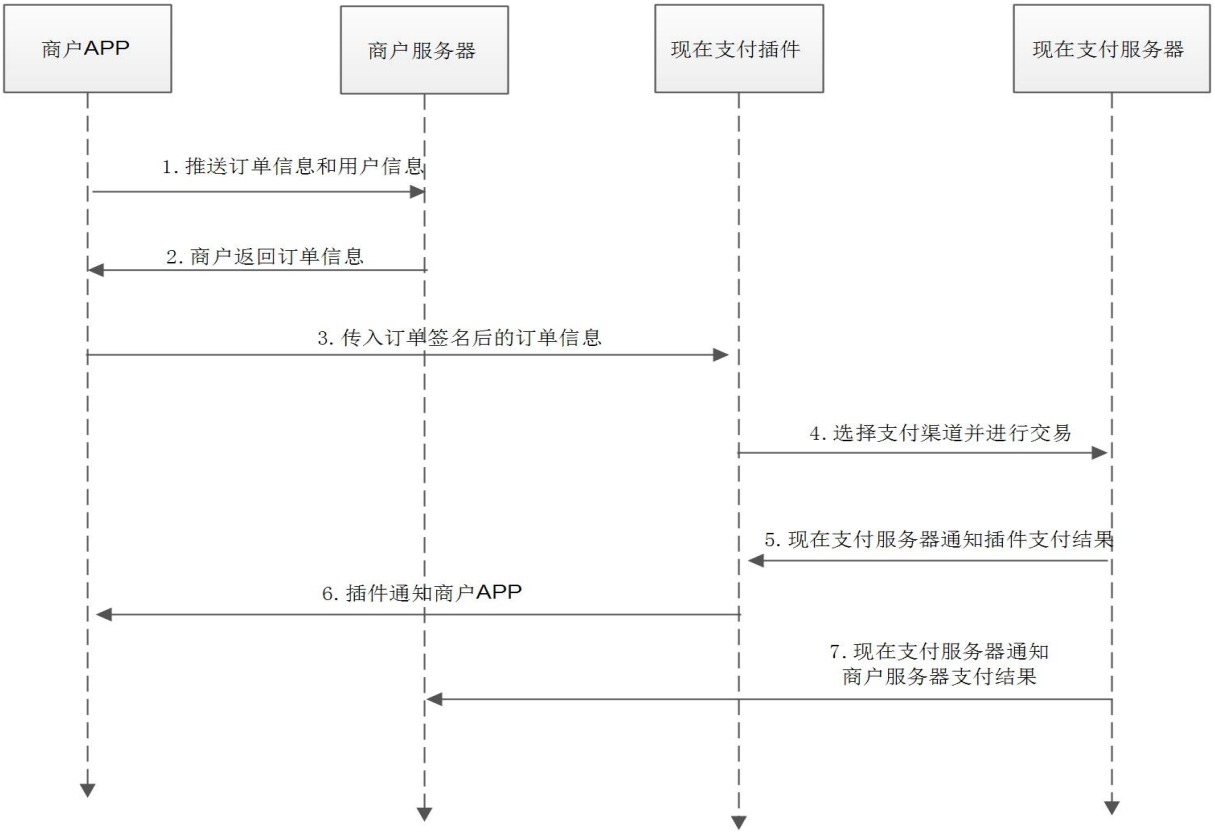
V 2.0.0

目录

一、概述

现在支付控件包括银联支付和支付宝支付，主要为开发者的手机客户端提供安全、便捷的支付服务，目前支付控件支持 Android 和 iOS 两个平台，用户通过输入银行卡号或支付宝账号等有效信息完成支付。

1.1 支付流程介绍:



步骤说明:

1.商户 APP 向商户服务器发送订单信息以及账户信息。

2.商户 APP 将符合插件调起接口规范的信息传入调起方法，并调起现在支付支付插件。

3.用户在插件中支付渠道的选择并完成支付操作。

4.支付完成后，现在支付支付插件接收服务器发送的支付结果通知。

5.现在支付支付插件通知商户 APP 支付情况。(交易状态以商户后台收到的支付结果通知为准)

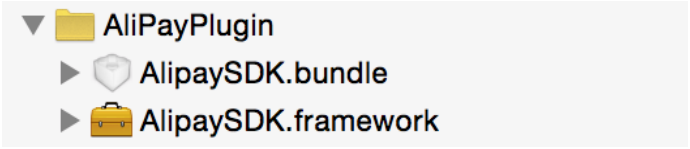
6.支付成功后，现在支付服务器通知商户服务器交易信息。

注意:
为了安全考虑，推荐商户服务器收到信息后，根据插件调起接口规范说明（见附录 A）组合信息，并对指定字段进行 MD5 签名。

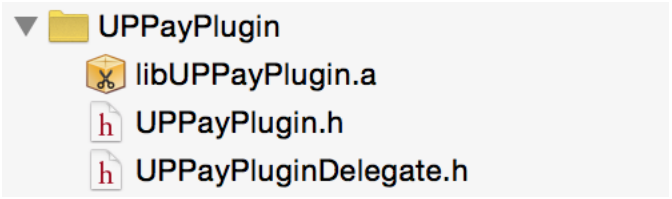
二、iOS 客户端接入

2.1 文件引用

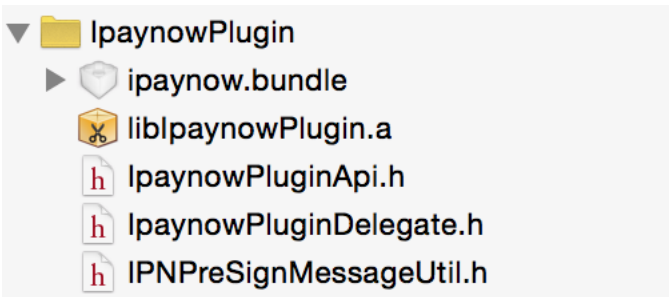
1.添加 AliPayPlugin 包：
其中包括 AlipaySDK.framework、AlipaySDK.bundle。



添加 UPPayPlugin 包:
其中包括 libUPPayPlugin.a、UPPayPlugin.h、UPPayPluginDelegate.h 文件。

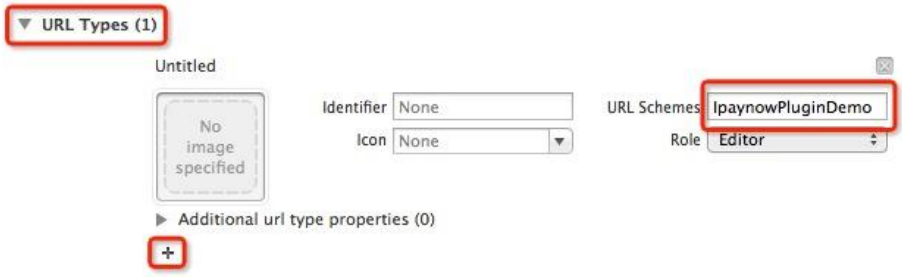


添加 IpaynowPlugin 包：
其中包括 ipaynow.bundle、libIpaynowPlugin.a、IpaynowPluginApi.h、IpaynowPluginDelegate.h 及 IPNPreSignMessageUtil.h。



2.2 工程设置

在工程的 Build Settings 中找 info，设置 URL Types，添加自定义 URL Scheme。



URL Scheme 在回调结果使用，建议起名稍复杂一些，尽量避免同其他程序冲突。

在工程的 Build Settings 中找到 Other Linker Flags 中添加-ObjC 宏。

2.3 调用支付接口

第一步：使用插件中的 IPNPreSignMessageUtil 工具类生成待签名方法：对类中的字段进行赋值,调用 generatePreSignMessage()方法进行待签名串的生成。 (若生成结果为null,则说明有必传参数没有赋值)

```
IPNPreSignMessageUtil *preSign=[[IPNPreSignMessageUtil alloc]init];
preSign.appId=@"1408709961320306";
preSign.mhtOrderNo=[dateFormatter stringFromDate:[NSDate date]];
preSign.mhtOrderName=@"手机插件测试用例";
preSign.mhtOrderType=@"01";
preSign.mhtCurrencyType=@"156";
preSign.mhtOrderAmt=@"100";
preSign.mhtOrderDetail=@"关于订单验证接口的测试";
preSign.mhtOrderStartTime=[dateFormatter stringFromDate:[NSDate date]];
preSign.notifyUrl=@"http://localhost:10802/";
preSign.mhtCharset=@"UTF-8";
preSign.mhtOrderTimeOut=@"3600";
preSign.mhtReserved=@"test";
preSign.consumerId=@"IPN00001";
preSign.consumerName=@"IpaynowCS";
preSign.payChannelType=@"11";

NSString *originStr=[preSign generatePresignMessage];
```

(mhtOrderTimeOut、mhtReserved、consumerId、consumerName和payChannelType为选发字段,若有必选字段没有赋值则返回nil)
补充说明：通过payChannelType字段可指定跳转到某支付渠道。

第二步：请求后台服务器对待签名串进行签名。

```
NSURL* url = [NSURL URLWithString:kSignURL];
NSMutableURLRequest * urlRequest=[NSMutableURLRequest requestWithURL:url];
[urlRequest setHTTPMethod:@"POST"];
urlRequest.HTTPBody=[presignStr dataUsingEncoding:NSUTF8StringEncoding];
NSURLConnection* urlConn = [[NSURLConnection alloc] initWithRequest:urlRequest delegate:self];
[urlConn start];
```

(除mhtSignature字段外还需要加入mhtSignType字段，示例代码中在后台已处理)

第三步：第一步生成的待签名串与第二步服务器生成的签名串拼接起来,传入插件调起方法中。

```
NSString* data = [[NSMutableString alloc] initWithData:mData encoding:NSUTF8StringEncoding];
NSString* payData=[_presignStr stringByAppendingString:@"&"];
payData=[payData stringByAppendingString:data];
[IpaynowPluginApi pay:payData AndScheme:@"TestPlugin" viewController:self delegate:self];
```

NSString *data
主要包含商户的订单信息，key=“value”形式，以&连接。
NSString *scheme
商户程序注册的 URL protocol，供支付完成后回调商户程序使用。
UIViewController*viewController
商户应用程序调试手机支付的当前 UIViewController。
id<IpaynowPluginDelegate>delegate
实现 IpaynowPluginDelegate 方法的 UIViewController.

第四步：实现插件通知接口

接收通知接口应用内结果通知接口为 IpaynowPluginDelegate，包含如下方法：

```
-(void)IpaynowPluginResult:(NSString*)result;
```

控件支付结果将以字符串的形式作为回调函数参数(NSString *)result 返回。

应用间结果通知接口（用于接收支付宝通知）如下：

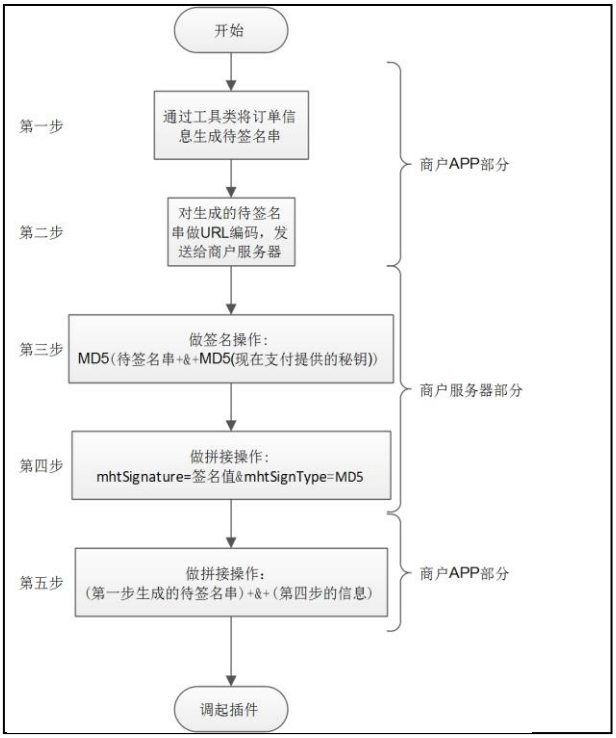
```
+ (BOOL)handleOpenUrl:(NSURL*)url;
```

在 IpaynowPluginApi.h 中添加 handleOpenUrl: (NSURL*)url,通过在 AppDelegate.m 中实现来完成独立返回 url 异步通知。

注：选择微信支付后会跳转到微信客户端进行支付，支付完成需要手动切回 App,故请开发者慎重选择开启微信支付。
微信支付需要在 AppDelegate.m 中添加[IpaynowPluginApi willEnterForeground]方法。

三、Android 客户端接入

android 版简便接入流程介绍:



详细介绍:

第一步：使用插件中的 PreSignMessageUtil 工具类生成待签名方法:

对类中的字段进行复制，调用 generatePreSignMessage()方法进行待签名串的生成。

(若生成结果为 null，则说明有必传参数没有赋值)

```
PreSignMessageUtil preSign=new PreSignMessageUtil();
preSign.appId="1410868994004446";
preSign.mhtOrderNo=new SimpleDateFormat("yyyyMMddHHmmss",Locale.CHINA).format(new Date());
preSign.mhtOrderName="支付样例-手机版";
preSign.mhtOrderType="01";
preSign.mhtCurrencyType="156";
preSign.mhtOrderAmt="100";
preSign.mhtOrderDetail="关于支付的演示";
  preSign.mhtOrderTimeOut="3600";
preSign.mhtOrderStartTime=new SimpleDateFormat("yyyyMMddHHmmss",Locale.CHINA).format(new Date());
preSign.notifyUrl="http://localhost:10802/";
preSign.mhtCharset="UTF-8";
  preSign.mhtReserved="test";

preSignStr=preSign.generatePreSignMessage();
```

(白色字段为选发字段,若有必选字段没有被设置则返回 null)

第二步:使用插件中的 PluginTools 类的 urlEncode()方法对待签名串进行 urlEncode 编码。

```
GetMessage gM = new GetMessage();
gM.execute("paydata="+MerchantTools.urlEncode(preSignStr));
```

第三步与第四步见后台样例代码（后台服务器接收到数据需要先做 UTF-8 的 url 解码再进行签名）

第五步:第一步生成的待签名串与第四步服务器生成的签名串拼接起来，传入插件调起方法中。

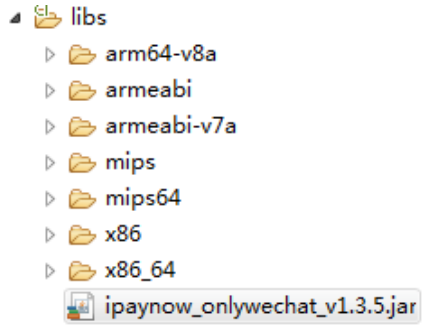
```
String needcheckmsg = HttpUtil.post(GETORDERMESSAGE_URL, msg);
needcheckmsg=MainActivity.preSignStr+"&"+needcheckmsg;
```

3.1 接入所需包介绍:(支持单独调起插件)

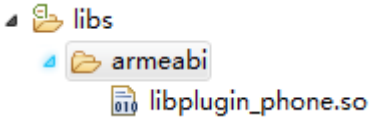
1. ipaynow_plugin_phone_onlywechat_v2.0.2b.jar（现在支付 jar 包）
2. libplugin_phone.so(现在支付 so 文件)

接入步骤介绍:

a).将 ipaynow_plugin_phone_onlywechat_v2.0.2b.jar 文件拖进项目的 libs 文件夹中



c).将 libplugin_phone.so 文件夹拖进 libs 文件夹下的 xxx 目录下,其中 xxx 为 armeabi,armeabi-v7a,mips,x86 之一。



3.2 插件接口介绍:

a).调用初始化接口:

WechatPayPlugin.getInstance().init(Context context);

b).调起插件支付接口:(必须在主线程调用，子线程调用的话没动静呦)

WechatPayPlugin.getInstance()

.pay(String req)

该接口需要传入参数为:

req:经过商户服务器签名后的符合现在支付接口规范的请求信息。

```
WechatPayPlugin.getInstance()
.setPayLoading(progressDialog)//自定义进度条
.setCallResultActivity(act)//传入回调用的Activity对象
.setCallResultReceiver(MainActivity.this)//传入继承了通知接口的类
.pay(result);//传入请求数据
```

具体 requestMessage 传入信息见附录 A

c).插件通知接口:

1)在 Activity 中接收通知:

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

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

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

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
  if (data == null) {
      return;
  }
  String respCode = data.getExtras().getString("respCode");
  String errorCode=data.getExtras().getString("errorCode");
  String respMsg = data.getExtras().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:"+respMsg);
  }else if(respCode.equals("03")) {
    temp.append("交易状态:未知").append("\n").
    append("错误码:").append(errorCode).
    append("原因:"+respMsg);
  }
  Toast.makeText(this, temp.toString(), Toast.LENGTH_LONG).show();
```

2)在非 Activity 类中接收通知:

第一步：调用.setCallResultReceiver(MainActivity.this)//传入继承了通知接口的类

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

```
@Override
public void onIpaynowTransResult(ResponseParams resp) {
  String respCode = resp.respCode;
  String errorCode=resp.errorCode;
  String respMsg =resp.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:"+respMsg);
  }else if(respCode.equals("03")) {
    temp.append("交易状态:未知").append("\n").
    append("错误码:").append(errorCode).
    append("原因:"+respMsg);
  }
  Toast.makeText(this, temp.toString(), Toast.LENGTH_LONG).show();
}
```

插件通知信息详见附录 C 列表

3)监听 Activity 销毁:

```
@Override
protected void onDestroy() {
  super.onDestroy();
  WechatPayPlugin.getInstance().onActivityDestroy();
}
```

特殊说明:

1.签名部分:

推荐使用插件中的 **PreSignMessageUtil** 类进行待签名串的生成，只需将该类的属性赋值，并调用 **generatePreSignMessage()**方法，即可生成符合规范的待签名串。客户端需要将该工具类生成的串发送至商户服务器进行 **MD5** 签名**(**详见附录**)**。

2.单独调起插件部分:

若需要跳过网关，直接调起插件，则需要将 **payChannelType** 赋值为指定渠道的标识即可。

注意:

如果商户应用是横屏的话，需要在 **androidManifest.xml** 中，修改下 **PayMethodActivity** 以及

Activity 配置:

```
<activity

android:name="com.ipaynow.wechatpay.plugin.inner_plugin.wechat_plugin.acti
vity.WeChatNotifyActivity"
            android:theme="@android:style/Theme.NoDisplay"
            android:configChanges="orientation|screenSize"
            android:screenOrientation="behind">
</activity>
```

权限配置:

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

代码混淆:

```
-libraryjars  libs/ipaynow_plugin_phone_v2.0.2b.jar

-keep public class com.ipaynow.wechatpay.plugin.utils.MerchantTools{
    <fields>;
    <methods>;
}
-keep public class com.ipaynow.wechatpay.plugin.utils.PreSignMessageUtil{
    <fields>;
    <methods>;
}
-keep public class com.ipaynow.wechatpay.plugin.api.WechatPayPlugin{
    <fields>;
    <methods>;
}
```

```
-keep public class
com.ipaynow.wechatpay.plugin.manager.route.dto.RequestParams{
    <fields>;
    <methods>;
}
```

定制化功能说明:

1.商户可以选择是否显示支付确认框:

说明:

支付确认框被设计用于在以下情况弹出:

1.用户请求支付且未登陆时，会出现登陆后跳出微信的情况，这时会弹出该弹框提示用户是否要完成支付。

2.用户请求支付且此时因为账户在其他设备登陆被踢掉，会在微信客户端内部弹出风险警告并退出微信，这时会提示用户是否继续完成支付。

定制:开发者可选择是否弹窗，不弹窗会直接返回开发者 APP，并发送通知。

设置方法为:

.setShowConfirmDialog(boolean isshow);

参数说明:isshow 设置为 true 时为弹出该确认弹框，设置为 false 时为不弹出该确认弹窗.

2.商户可以自定义加载微信以及退出微信时的进度框:

当商户不设置时默认进度框如图下:

图一

图二

说明:该进度条为进入微信以及退出微信时执行通讯时需要，默认样式为上图。

定制: 商户若觉得不符合自己 APP 风格，可以通过以下方法对进度框进行定制。

设置方法:

```
.setCustomDialog(IpaynowLoading view);
```

参数说明:

view 该参数为商户实现了 IpaynowLoading 抽象类的实例对象。

自定义实例:

```
public class LoadingDialog extends IpaynowLoading {
private TextView tips_loading_msg;
public LoadingDialog(Context context) {
      super(context);
}
@Override
protected void onCreate(Bundle savedInstanceState) {//用户加载布局文件
      super.onCreate(savedInstanceState);
      this.setContentView(R.layout.view_tips_loading);
      tips_loading_msg = (TextView) findViewById(R.id.tips_loading_msg);
}
@Override
public void setLoadingMsg(String msg) {//用于现在支付插件设置进度框内容之用(必须实现)
   tips_loading_msg.setText(msg);
}
}
```

注意:商户可以将获取服务器签名时的进度条传入，这样插件可以复用该进度条，并自行 dismiss。具体参见 Demo

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		微信支付:13;
商户是否支持信用卡支付	mhtLimitPay	定值	N	指定不能使用信用卡支必须填 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

第一步：对参与 MD5 签名的字段按字典升序排序后，分别取值后并排除值为空的字段键值对，最后组成 key1=value1&key2=value2....keyn=valuen "表单字符串"。
第二步：对 MD5 密钥进行加密得到"密钥 MD5 值"。
第三步：最后对 第一步中得到的表单字符串&第二步得到的密钥 MD5 值 做 MD5 签名
PS : MD5 密钥是用户在注册应用的时候生成的， 每个应用一个 MD5 密钥。
样例:

appId=888888888888888&mhtCharset=UTF-8&mhtOrderNo=20140821161747&mhtOrderName=%E9%99%B6%E6%A0%91%E5%BC%BA&mhtOrderType=01&mhtCurrencyType=156&mhtOrderAmt=1&mhtOrderDetail=%E5%85%B3%E4%BA%8E%E8%AE%A2%E5%8D%95%E9%AA%8C%E8%AF%81%E6%8E%A5%E5%8F%A3%E7%9A%84%E6%B5%8B%E8%AF%95&mhtOrderStartTime=20140821161747&notifyUrl=http%3A%2F%2Flocalhost%3A10802%2F&mhtSignature=72e3b9fea03b81b88224fe0eab1459d9&mhtSignType=MD5

使用简便流程注意:

如用安卓文档中推荐的简便接入流程的话，服务器只要根据商户 APP 上送的待签名字符串根据以下公式生成签名值即可:

签名公式:MD5（待签名串+"&"+MD5（现在支付提供的秘钥））;

生成签名值后，根据以下公式拼接好发送给商户 APP:

拼接公式: mhtSignature=（签名公式得到的值）+"&"+ mhtSignType=MD5;

样例: mhtSignature=1519adb35c04e0b962d8ca68476d9d56&mhtSignType=MD5

商户使用简便流程时，商户后台需要先做 UTF-8 的 url 解码后再做签名处理

附录 C

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	未知和失败时会返回,详情见错误码表
响应信息	respMsg	N	未知和失败时会返回,详情见错误码表