Mobile Payment Control Access Guide

3.3.0

2019-10-09 Published 2019-10-25 Implemented

THIS PAGE INTENTIONALLY LEFT BLANK.

Version Information

Ver. No.	Date	Remarks
2. 0.0	2013-01-08	First draft
2. 0.1	2015-07-29	Certain errors are corrected. Android upgrade 3.0.7 control data.bin is moved to the folder “assets”. The explanation on jar integration mode is added. The notes on browser invocation control are deleted, and it is recommended that the wap product should be directly used for browser mode. FAQ is added.
3.0.0	2015-12-09	Invocation interface is unified as startPay UPPayWapActivity is added libuptsmaddon.so is added
3.1.0	2017-08-02	Check interface status of X-Pay is added. Appoint X-Pay payment interface is added.
3.1.1	2018-01-12	Add the part of invocation interface of X-Pay (Huawei Pay, Mi Pay, etc.)
3.1.2	2018-02-23	Add details in section 4.2
3.2.0	2018-11-23	Add the interface of detect UnionPay APP installation (checkInstalled method deleted) Update section 4.3 Proguard
3.3.0	2019-10-25	Modify demo code Modify AndroidManifiest.xml part. Add the solution to solve the “transparent dialog” problem of Android Q dialog

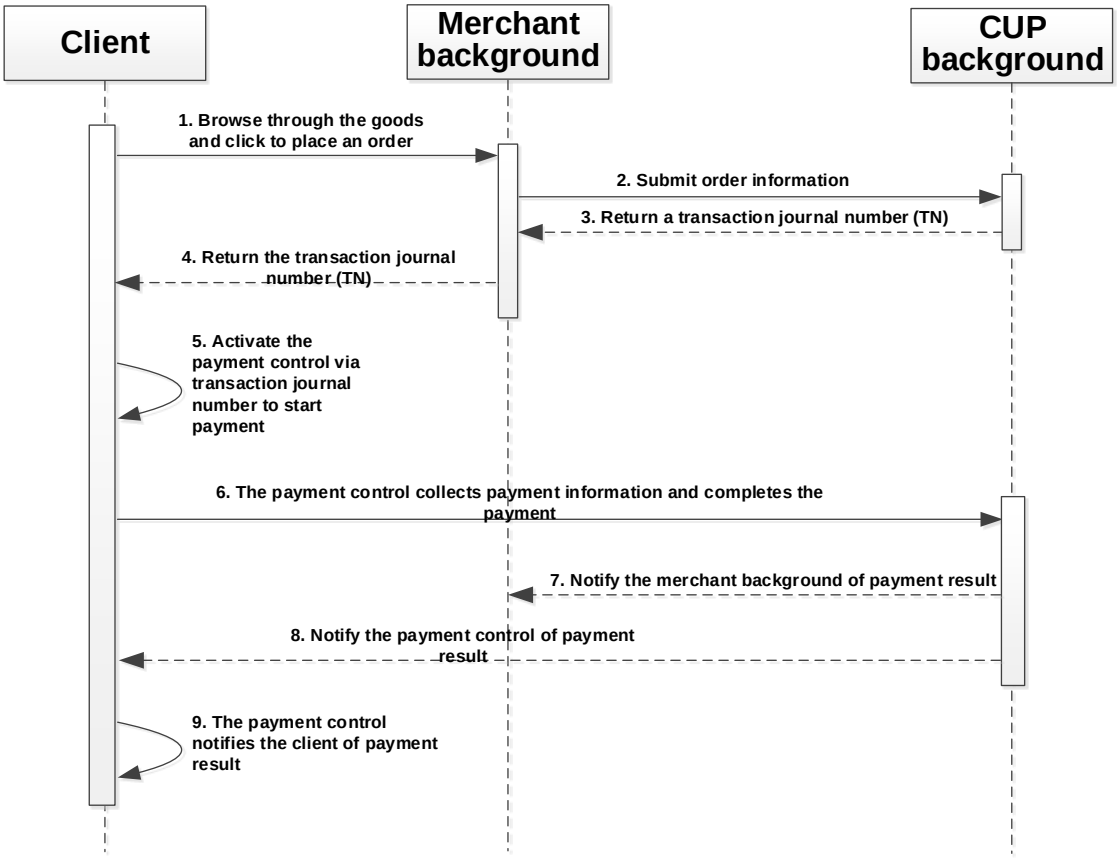
Table of Contents

1 Overview

UP mobile payment control ("payment control") is primarily intended to provide cooperating merchants' mobile client with secure and convenient payment service. The user completes payment by entering such elements as bank card number, mobile phone number, password (debit card and prepaid card) or CVN2, expiry date (credit card) and verification number in the payment control.

2 Description of payment process

The process of conducting a transaction via payment control is illustrated below:



Notes to the flowchart:

(1) The user clicks in the client to buy the goods, and the client issues an order generation request to the merchant back-end;

(2) After receipt of the order generation request, the merchant back-end organizes and pushes order information to UP back-end;

(3) After receiving and verifying the order information, UP back-end generates the corresponding transaction number (i.e., TN) and returns that transaction number to the merchant back-end (response element: transaction number, etc.);

(4) After receiving the transaction number, the merchant back-end returns the transaction number to the client;

(5) The client invokes the payment control via the transaction number (TN);

(6) After the user entering the relevant payment information in the payment control, the payment control issues a payment request to UP back-end;

(7) After successful payment, UP back-end notifies the payment result to the merchant back-end;

(8) UP notifies the payment result to the payment control;

(9) The payment control displays the payment result and returns the payment result to the client;

Note: This document mainly focuses on the realization of parts (5) and (9) in the above process.

Devices currently supported by each platform are detailed below:

Android platform SDK chiefly applies to Android 2.3 and higher versions of terminal devices;

3 Test account number

Card number and mobile phone number information provided for testing purpose (such information is provided for testing only, and no transaction actually happens)

Test card (issued in mainland China) information description:

Curr.	PAN	Attr.	Expired Date	ID Type	ID number	CVN2	Cell Phone	PIN
RMB	62228212 34560017	Credit	YYMM3312	01	331081198210112 257	123	(+86) 13012345678	111111
RMB	62231649 91230014	Debit	YYMM3312	01	331081198210112 257	123	(+86) 13012345678	111111

（Click “obtain verification code” button first on the payment page before inputting OTP）

Test card (issued overseas) information description:

Curr.	PAN	Attr.	Expired Date	ID Type	ID number	CVN2	Cell Phone	PIN
HKD	62509460 00000016	Debit	YYMM3312	01	331081198210112 257	123	(+852) 11112222	111111
HKD	62509470 00000014	Credit	YYMM3312	01	331081198210112 257	123	(+852) 11112222	111111

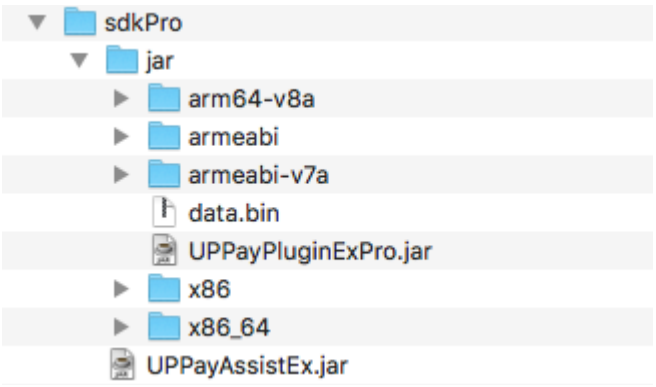
（Click “obtain verification code” button first on the payment page before inputting OTP）

4 Android client

This section deals with the description of SDK interface and its interface invocation details and requires that the reader should have certain Android programming experience.

4.1 Description of SDK kit

After getting the development kit provided by UnionPay, the merchant developer shall check the directory of the SDK, i.e., upmp_android/sdkProSDK. Files mentioned below are all under that directory:

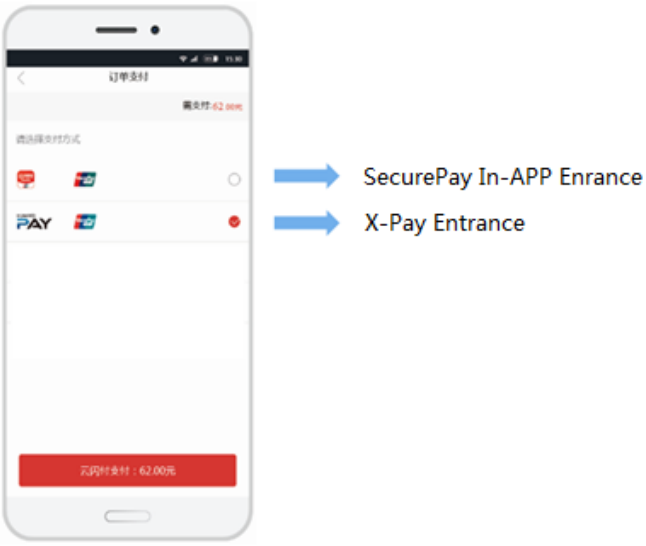


The jar directory contains jar kit, so files (supportive of arm,armv7,x86, x86_64 and arm64-v8a platforms) and resource files required for merchant integration.

UPPayAssistEx.jar defines the interface required for invocation of payment control.

4.2 Interface description

For the merchant APP check-out UI, SecurePay In-APP Entrance/Button shall always be displayed and available. “X-Pay” (eg. Huawei Pay, Samsung Pay, etc.) shall be displayed according to the exact status of consumer’s mobile phone.



1. Payment interface

upmp_android/UPPayAssistEx.jar defines the interface for activating the payment control, and that interface is defined below:

public static int startPay(Activity activity, String spId, String sysProvider, String orderInfo, String mode)

Parameter description:

activity —— Active object used for activating the payment control

spId —— Reserved for use, input null here

sysProvider —— Reserved for use, input null here

orderInfo —— Order information is transaction number, namely TN, obtained by merchant back-end from UP back-end.

mode —— UP back-end environment identifier, "00" means initiating a transaction in UP's formal environment,"01"means initiating a transaction in UP's test environment

2. Interface to check "X-Pay" status

This interface can be used to check if "X-Pay" (eg. Huawei Pay, Samsung Pay, etc.) is available on the mobile phone. The merchant APP shall display the related "X-Pay" button on the check-out UI dynamically.

public static int getSEPayinfo(Context context,UPQuerySEPay InfoCallback callback)

Parameter description:

context —— to obtain context of active object used for payment control start-up
callback —— it is defined as follows：

```
public interface UPQuerySEPayInfoCallback {
public void onResult(String SEName, String seType, int cardNumbers,
        Bundle reserved);

public void onError(String SEName, String seType, String errorCode,
        String errorDesc);
}
```

It needs to implement two methods respectively:

public void onResult(String SEName, String seType, int cardNumbers,Bundle reserved);
This method is called under normal circumstances, parameters are as follows：
SEName —— the name of X-Pay, seen in table 1
seType —— the type corresponding to the name of X-Pay, seen in table 1
cardNumbers —— the number of cards
reserved —— reserved fields

public void onError(String SEName, String seType, String errorCode,String errorDesc);
This method is called when any abnormal condition is detected, parameters are as follows:
SEName —— the name of X-Pay, seen in table 1
SeType —— the type corresponding to the name of X-Pay, seen in table 1
errorCode —— error code and corresponding reason:
1. X-Pay not installed/TSM control version outdated *ERROR_NOT_SUPPORT* = **"01"**
2. X-pay not supported by hardware *ERROR_NOT_SUPPORT* = **"01"**
3. X-pay not enabled *ERROR_NOT_READY* = **"02"**
4. No card available *ERROR_NOT_READY* = **"02"**
5. TSM control not installed *ERROR_NOT_SUPPORT* = **"04"**
6. Timeout *ERROR_NOT_READY* = **"10"**

errorDesc —— string corresponding to the error code

Return values:

UPSEInfoResp.*SUCCESS*—— obtain SEPay status successfully
UPSEInfoResp.PARAM_ERROR — fail, one or more parameters are null

Table 1 Correspondence between SEName and seType

SEName	seType
Samsung Pay	02
Huawei Pay	04
Meizu Pay	27
Le Pay	30
Mi Pay	25
OPPO Pay	29
vivo Pay	33
Smartisan Pay	32

3. Interface to confirm X-Pay payment

public static int startSEPay(Context context, String spId,String sysProvider, String orderInfo, String mode, String seType)

Parameter description:

context ——to obtain context of active object used for payment control start-up
spId —— reserved to use, input null here
sysProvider ——reserved to use, input null here
orderInfo ——Order information is transaction number, namely TN, obtained by merchant back-end from UP back-end.
mode ——UP back-end environment identifier, "00" means initiating a transaction in UP's formal environment,"01"means initiating a transaction in UP's test environment.
seType ——the type corresponding to the name of X-Pay, seen in table 1
Return values:
0

4. Interface to check if UnionPay APP is installed

upmp_android/UPPayAssistEx.jar defines the interface to check if UnionPay APP is installed, and that interface is defined below:

public static boolean checkInstalled (Context context)

Parameter description:

activity —— Context environment used for activating the payment control

Return values:

true —— The control apk has been installed on the device

false — The control apk has not been installed on the device

4.3 Add SDK kit

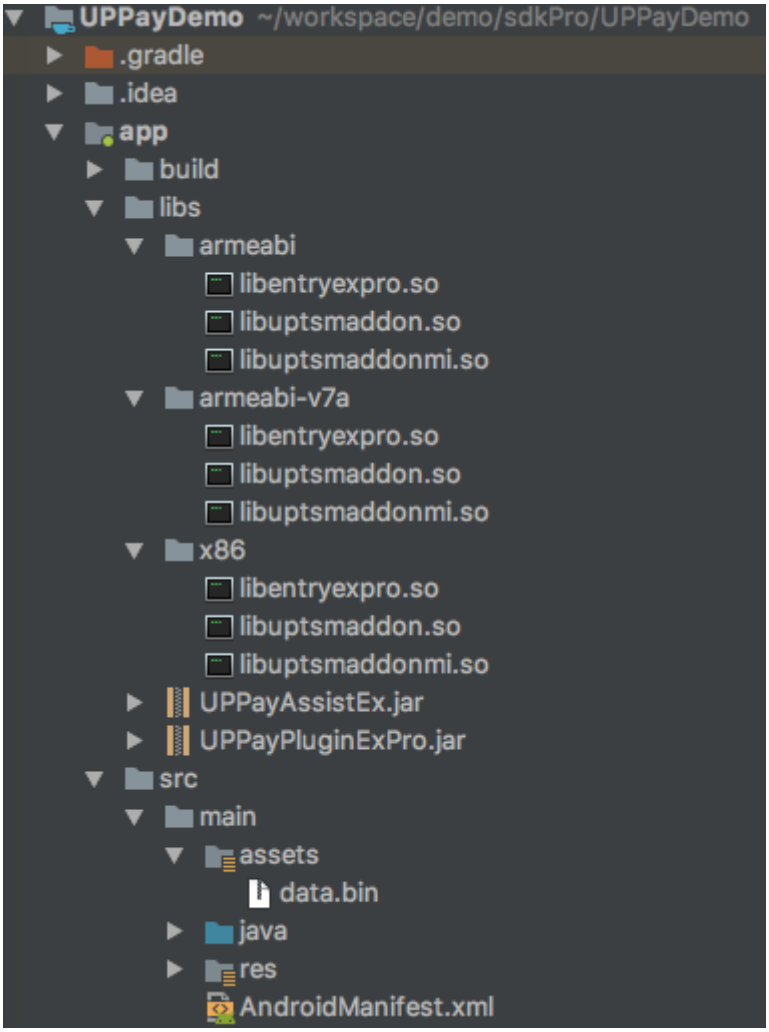
Add resource files:

1. Copy upmp_android/sdkPro/jar/data.bin to the Project's assets/ directory;
2. Copy upmp_android/sdkPro/jar/xxx/libentryexpro.so, upmp_android/sdkPro/jar/xxx/libuptsmaddon.so and upmp_android/sdkPro/jar/xxx/libuptsmaddonmi.so to the Project's libs/xxx/ directory, and xxx refers to one of armeabi-v7a, armeabi, arm64-v8a, x86 and x86_64.

arm64-v8a(x86_64) is a library file aimed at arm64(x86 64) architecture optimization. After introduction of the Project, arm64(x86 64) model will have higher performance, but the finally generated program kit will grow bigger. **Note: If the Project uses other .so libraries, then all .so libraries will have the arm64-v8a (x86_64) version.**

3. Copy upmp_android/sdkPro/UPPayAssistEx.jar to the Project's libs/ directory;
4. Copy upmp_android/sdkPro/jar/UPPayPluginExPro.jar to the Project's libs/ directory;

As shown below:



arm64-v8a(x86_64) is a library file aimed at arm64(x86 64) architecture optimization. After introduction of the Project, arm64(x86 64) model will have higher performance, but the

finally generated program kit will grow bigger. **Note: If the Project uses other .so libraries, then all .so libraries will have the arm64-v8a (x86_64) version.**

5. In the Project's file named AndroidManifest.xml, register the Activity used for the payment control, as following:

```
<application>
<!-- The Project's other configurations are omitted here …-->
<uses-library
      android:name="org.simalliance.openmobileapi"
      android:required="false"/>
 <activity
      android:name="com.unionpay.uppay.PayActivity"
      android:label="@string/app_name"
      android:screenOrientation="portrait"
      android:configChanges="orientation|keyboardHidden"
      android:excludeFromRecents="true"
      android:windowSoftInputMode="adjustResize"/>

 <activity
      android:name="com.unionpay.UPPayWapActivity"
      android:configChanges="orientation|keyboardHidden"
      android:screenOrientation="portrait"
      android:windowSoftInputMode="adjustResize"/>
</application>
```

Attention: If dialogTheme is NOT set to for the application to display, dialog would probably become transparent on Android Q. The solution is to set a defined dialogTheme for com.unionpay.uppay.PayActivity and com.unionpay.UPPayWapActivity.

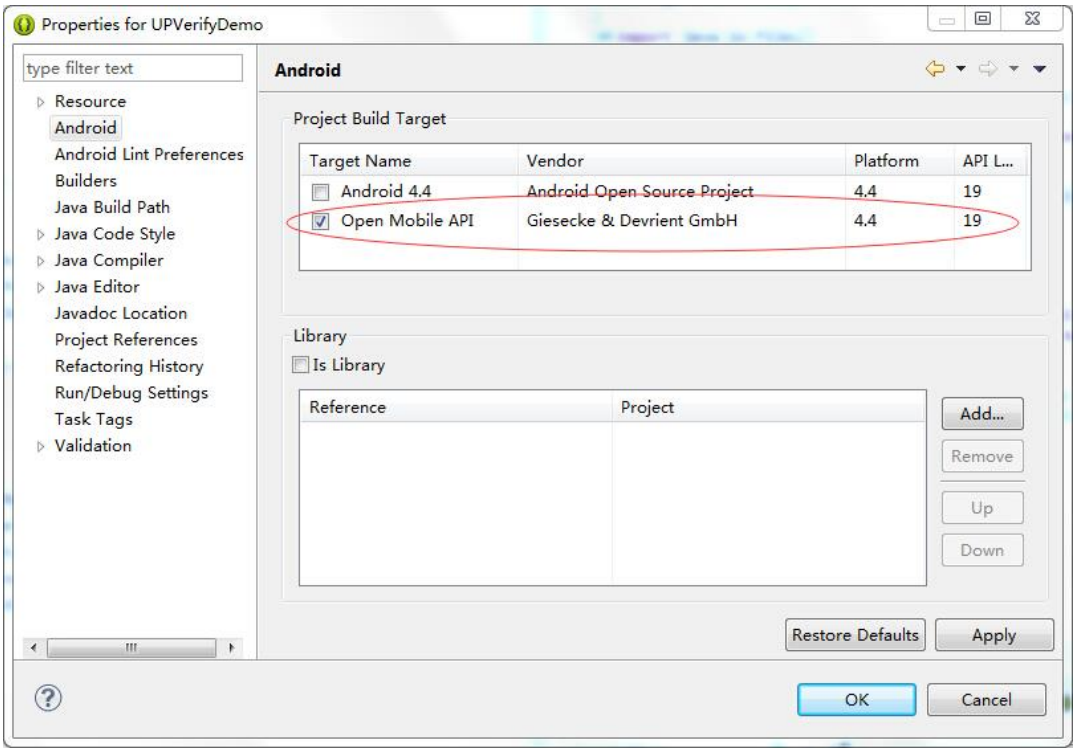
For example:

```
<uses-permission
   <style name="UPPay">
   <item name="android:dialogTheme">@android:style/Theme.Dialog
   </item>
</style>
```

Meanwhile, add permission of UnionPay payment control:

```
<uses-permission
   android:name="android.permission.INTERNET"/>
<uses-permission
   android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission
   android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permissionandroid:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permissionandroid:name="android.permission.READ_PHONE_STATE"/>
<uses-permissionandroid:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission android:name="android.permission.NFC" />
<uses-feature android:name="android.hardware.nfc.hce"/>
<uses-permissionandroid:name="org.simalliance.openmobileapi.SMARTCARD" />
```

Note: The target during version compilation shall be Open Mobile API. And level19 or higher levels are recommended for using eclipse, as following:



When using Android Studio, openmobileapi.jar can be put in the libs folder:

```
provided files('libs/org.simalliance.openmobileapi.jar')
```

6. Proguard rules

```
//Add the following rules in proguard file
-dontwarn com.unionpay.**
-keep class com.unionpay.** {*;}
-keep class org.simalliance.openmobileapi.** {*;}
```

Note：Please add the above Proguard rules while developing.

4.4 Invocation of payment control

1. Invocation of payment interface

a) Import UPPayAssistEx into the code file to invoke payment control. As shown below:

```
import com.unionpay.UPPayAssistEx;
```

b) Then the payment control may be invoked as following:

```
// “00” – UnionPay’s formal environment
// “01” – UnionPay’s testing environment, where no real transaction occurs
String serverMode = "01";
UPPayAssistEx.startPay (activity, null, null, tn, serverMode);
```

c) Return of the payment control

After the payment is completed, merchant APP shall obtain the payment control’s payment result and add the corresponding processing logic. It is okay to simply realize the onActivityResult() method in Activity invocation, and the signature information and payment result will be returned to merchant APP client after the payment is completed.

Whether merchant order is successfully paid or not shall be subject to the payment result received by merchant back-end from the UPOP, and the result returned by the payment control here serves as a reference only.

After receiving the payment result by the control, it is suggested to ignore the information and verify the signature by merchant back-end. Merchant back-end should initiate inquiry to UPOP if UPOP payment result is missing, so that the order result shown in the merchant APPclient could be consistent with back-end record, and order status in back-end could be updated in time.

If the payment result returned by the control is still used, it is recommended to verify the signature by merchant back-end. The signature verification shall not be done in the merchant APP, otherwise, it may cause signature verification failure.

Sample code is shown below:

```
protected void onActivityResult( int requestCode,
        int resultCode, Intent data) {
    if( data == null ){
        return;
    }
    String msg = "";
    String str =  data.getExtras().getString("pay_result");
    if( str.equalsIgnoreCase(“success”) ){
```

```
                //Display payment result
                 showResultDialog("Successful payment!");
                //Please check the back-end notification before display the
payment successful result.
    }else if( str.equalsIgnoreCase("fail") ){
        showResultDialog("Failed payment! ");
    }else if( str.equalsIgnoreCase("cancel") ){
        showResultDialog("Current payment cancelled!");
    }
}
```

2. Invoke the interface to check if UnionPayAPP is installed

```
if(UPPayAssistEx.checkInstalled(context))
{
    //When it is made sure that UnionPay Apk has been installed on the user's mobile phone, the
merchant client may undergo the corresponding personalized processing
}
```

3. Invoke the interface to check X-Pay payment status

```
UPQuerySEPayInfoCallback callback = new UPQuerySEPayInfoCallback() {
@Override
public void onResult(String SEName, String seType, int cardNumbers,
Bundle reserved) {
// This will be returned when X-Pay is ok
}
@Override
public void onError(String SEName, String seType, String errorCode,
String errorDesc) {
// This will be returned when X-Pay is not available
}
};
int ret = UPPayAssistEx.getSEPayInfo(activity, callback);
```

4. Invoke interface to confirm X-Pay payment

Before invoking the interface to confirm X-Pay payment (startSEPay()), the interface to check X-Pay payment status (getSEPayInfo()) must be invoked to get seType. The method to invoke startSEPay() is the same as the method to invoke startPay().

5 Notes on modification of old version of SDK merchant

For an original sdk access merchant, the following modifications shall be added after update of the relevant documents based on original modifications:

1. Add upmp_android/sdkPro/jar/xxx/libuptsmaddon.so and upmp_android/sdkPro/jar/xxx/libuptsmaddonmi.so
2. Add UPPayWapActivity to the Project's AndroidManifest.xml file. The addition is shown below:

```
<application>
<!--The Project's other configurations are omitted here …-->
 <activity
      android:name="com.unionpay.UPPayWapActivity"
      android:configChanges="orientation|keyboardHidden"
      android:screenOrientation="portrait"
      android:windowSoftInputMode="adjustResize"/>
</application>
```

3. The invocation interface is changed to payment interface startPay in section 4.2, but the original startPayByJAR interface is still reserved. The detail is shown below:

public static int startPayByJAR (Activity activity, Class<?> payCls, String spId, String sysProvider, String orderInfo, String mode)

Parameter description:

activity —— Active object used for activating the payment control

payCls —— The class represented by the payment plug-in, where it is okay to fill in with "PayActivty.**class**"

spId —— Reserved for use, input null here

sysProvider —— Reserved for use, input null here

orderInfo —— Order information is transaction number, namely TN, obtained by merchant background from UP back-end.

mode —— UP back-end environment identifier, "00" means initiating a transaction in UP's formal environment,"01"means initiating a transaction in UP's test environment

6 Summary of frequently asked questions

For more details, please refer to https://open.unionpay.com Help Center -FAQ