vivo International Cross-Promotion Payment SDK Integration Guide

Version no.: 1.6.3.0

Revision History

|  |  |  |
| --- | --- | --- |
| Build number | Revised on | Description |
| V1.0.0.0 | 2018-5-30 | Creation |
| V1.1.0.0 | 12/25/2019 | Added app payment methods |
| V1.1.0.1 | 1/8/2020 | Bug fixes and improvements |
| V1.3.0.0 | 4/22/2020 | 1. Added support for multiple countries with one package 2. Added the interface for IP whitelist query 3. Interface and interface parameter adjustments |
| V1.4.0.0 | 7/15/2020 | 1. Optimized payment process and payment path 2. Added virtual currency payment |
| V1.6.0.0 | 11/10/2020 | 1. Integrated the cashier into the SDK 2. Optimized the payment logic |
| V1.6.2.0 | 02/2021 | 1. Optimized the SDK and fixed known bugs  2. Optimized the cashier |
| V1.6.3.0 | 03/2021  06/2021 | 1. Optimized the cashier UI and text  2. Fixed known bugs  3. Added the RSA content |

Contents

[vivo International Cross-Promotion Payment SDK Integration Guide 1](#_Toc26482)

[1. SDK Invocation 5](#_Toc3447)

[1.1 Preparations 5](#_Toc7226)

[1.2 Integration steps 7](#_Toc5030)

[1.2.1 Dependencies connection 7](#_Toc4020)

[1.2.2 Asset integration 9](#_Toc19525)

[1.2.3 SDK obfuscation 9](#_Toc15702)

[1.3 API interfaces 9](#_Toc2824)

[1.3.1 API initialization (required) 9](#_Toc16304)

[1.3.2 vivo account login API 10](#_Toc1833)

[1.3.3 Cashier payment API (required) 11](#_Toc1118)

[1.3.4 SMS payment foreground API (discarded) 14](#_Toc17938)

[1.4 Notes 15](#_Toc30831)

[1.5 Appendix 15](#_Toc20636)

[1.5.1 Payment order parameter table 15](#_Toc14651)

[1.5.2 Signature parameter table 18](#_Toc12056)

[1.5.3 Payment callback message table 20](#_Toc29891)

[1.5.4 Country code list 20](#_Toc19123)

[1.5.5 Payment notes 21](#_Toc15869)

[2. Payment Server 26](#_Toc1789)

[2.1 Preparations 27](#_Toc31951)

[2.2 Payment process 27](#_Toc15510)

[2.3 Asynchronous callback notification 28](#_Toc7803)

[2.4 Order query interface 30](#_Toc17996)

[2.4.1 Interface description 30](#_Toc4809)

[2.4.2 Interface URLs 30](#_Toc7220)

[2.4.3 Request parameters 30](#_Toc17210)

[2.4.4 Response parameters 31](#_Toc5252)

[2.5 IP whitelist query interface 33](#_Toc23807)

[2.5.1 Interface description 33](#_Toc24629)

[2.5.2 Interface URLs 33](#_Toc19345)

[2.5.3 Request parameters 33](#_Toc18868)

[2.5.4 Response parameters 34](#_Toc28952)

[2.6 Signature and verification 35](#_Toc11824)

[2.6.1 Signature algorithm 35](#_Toc21258)

[2.6.2 RSA signature 35](#_Toc11848)

[3. Account server (optional) 37](#_Toc25757)

[3.1 Preparations 37](#_Toc14623)

[3.2 openToken check process 37](#_Toc9738)

[3.3 openToken check interface 38](#_Toc27939)

[3.3.1 Interface description 38](#_Toc18104)

[3.3.2 Interface URLs 38](#_Toc25616)

[3.3.3 Request parameters 39](#_Toc10677)

[3.3.4 Response parameters 39](#_Toc27630)

[3.4 Signature and verification 40](#_Toc26427)

[3.4.1 Signature algorithm 40](#_Toc32029)

[3.4.2 Key information 41](#_Toc31500)

[3.4.3 Signed string 41](#_Toc32175)

[3.4.4 Signed result 41](#_Toc3811)

[3.4.5 Notes 41](#_Toc9373)

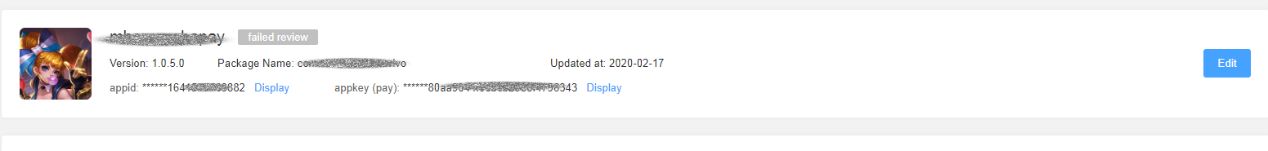
[4. FAQ 41](#_Toc2392)

# 1. SDK Invocation

## 1.1 Preparations

With the international cross-promotion payment SDK, you can quickly and easily enable the checkout function in your app. Before connecting, you need to make the following preparations:

1. Both connection with our vivo account and payment requires parameters such as **appId** and **appKey**, which can be obtained when you create games on the developer platform.

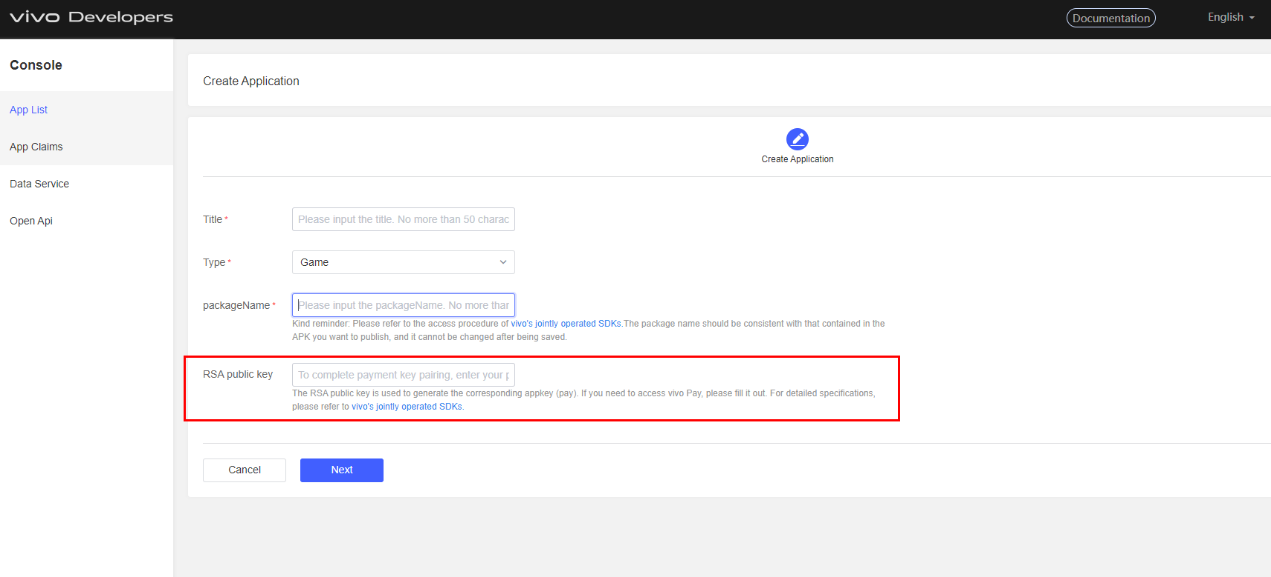


1. Payment testing is **currently only available for vivo international phones**. Otherwise, it may prompt that payment is not supported for this country or the payment does not respond. If you do not have relevant international phones available, please use our vivo cloud testing platform or contact our business staff. If you are unsure which models are international phones, please contact our business or technical staff for confirmation.
2. Communicate with our business staff to determine the supported countries and payment methods. Only after the support is implemented by our operation staff can you start testing properly. Otherwise, you may be prompted with "Payment is not supported for this app" when you test the payment.
3. Signatures and other methods referred to in the document can be found under the utils directory.
4. **The current SDK version is dependent on the Android X library**. If your game still uses the Support library, please upgrade it to the Android X library or contact our technical staff for relevant information before connecting this SDK.

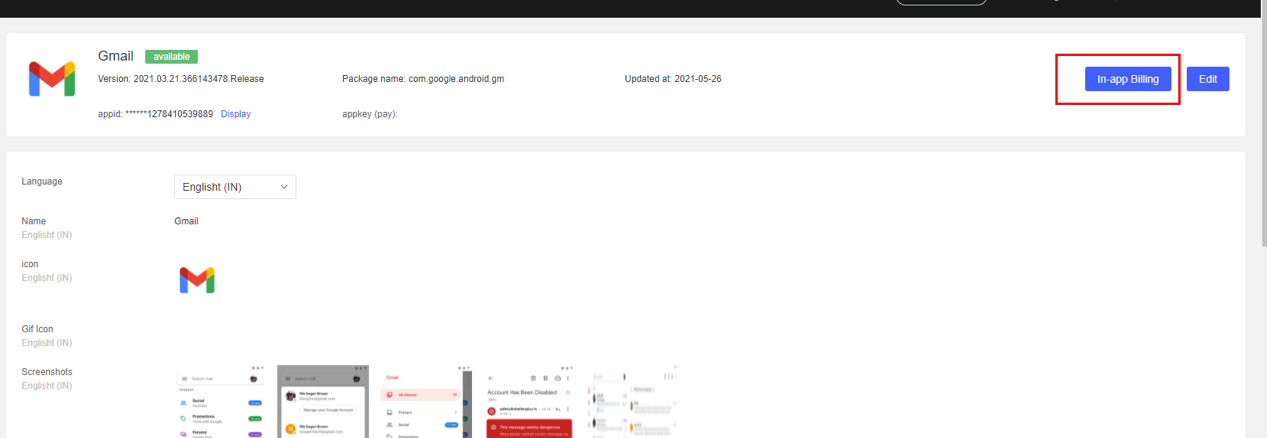
Notes:

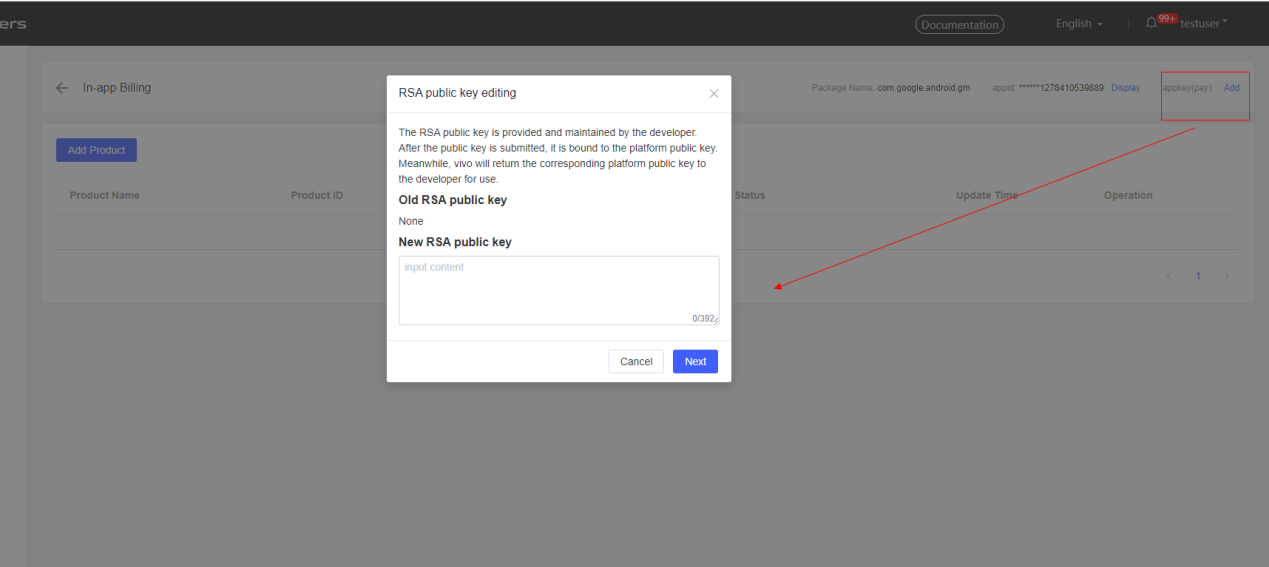
* When you connect this SDK for the first time, you need to upload your RSA public key when creating your game or creating in-app products on the developer platform. The "appkey" field displayed on the platform is the RSA public key for payment. When connecting the payment module, please use the RSA signature. For more details about the RSA signature, refer to section 2.6 Signature and verification (please ignore the MD5 coding examples and descriptions below).
* If you have already connected this SDK using the MD5 algorithm in the signature, you can still use the MD5 algorithm for countersigning and verification.

You can upload the RSA public key when creating a new app as follows:



If your app does not have an appkey, please follow the following steps to add one:





**1.2 Integration steps**

### 1.2.1 Dependencies connection

1. Import the AAR package in the "libs" directory.

2. **Connect to missing open source library dependencies (preferably the latest version) based on the actual situation of your project. This step is not mandatory.**



**Notes**:

Please note that you may need to configure obfuscation files for some open source libraries, such as "eventbus".

**When you add the dependencies for the open source library "okhttp" and "okio", the obfuscation should be configured as:**



**When you add the dependencies for the open source library "gilede", the obfuscation should be configured as:**



**When you add the dependencies for the open source library "eventbus", the obfuscation should be configured as:**



**When you add the dependencies for the open source library "gons", the obfuscation should be configured as:**



### 1.2.2 Asset integration

Import relevant assets under the assets directory.

1. Copy all the files in the "assets" folder into the project's "assets" directory.

### 1.2.3 SDK obfuscation



## 1.3 API interfaces

**All the API interface methods contained in the SDK must be called in the main thread.** Stopwatch methods contained in the SDK will be automatically executed in non-UI threads, so you don't need to worry about thread switching in CP.

### 1.3.1 API initialization (required)



| **Parameter** | **Description** |
| --- | --- |
| context | The context object of Context |
| appId | The appId of the app; this can be obtained from the developer platform |

The SDK initialization method should be called as early as possible to make sure the SDK initialization has completed before other API interfaces are called. If there are multiple processes, make sure initialization happens in the main process.

**It is recommended to initialize in onCreate of Application.**

**Coding example**:



### 1.3.2 vivo account login API



| **Parameter** | **Description** |
| --- | --- |
| activity | The current Activity object |
| vivoAccountCallback | Account information callback |

Currently, the international payment SDK supports the connection of payment with vivo account and payment without account. You can decide based on your business needs. If you choose payment without account, then skip the account API related content as there is no need to call this API. If you choose payment with vivo account, then call this account login API before calling the payment API. CP will use this interface to complete vivo account login and get the token and other information of the account. If a user has already logged in to your vivo account, this method will help refresh the login status. **(Call the login API to ask the user to log in to their vivo account or refresh their login status before calling the payment API. This will minimize the disturbance of account login during the payment process, thus increasing the payment rate. This is because if the payment amount does not meet the requirements or if the user wants to pay with virtual currency, they will need to log in to their account before further operation.)**

After login, CP will then pass the account openToken acquired through the callback to the account server for login and account validity verification. **(For more details, please refer to relevant documents about the server.**)

**Coding example:**



**1.3.3 Cashier payment API (required)**



| **Parameter** | **Description** |
| --- | --- |
| activity | The object of the current activity |
| payInfo | The VivoPayInfo object that encapsulates basic order information. See the Appendix for details. |
| callback | The result that is called back after the payment is completed. Please verify with the CP Server after receiving the payment result. |

By calling the cashier payment interface, you can call the payment cashier to make the payment. When the payment is completed, the Client will receive a callback. The Server then needs to query the payment result, and that final result shall prevail. For information about the Server, refer to **Payment Server**.

The payment information "payInfo" needs to be implemented and encapsulated by CP. **For parameters required for payment orders, please refer to section 1.5.1.**

**Coding example for creating payment orders**:

**If you have not connected to a vivo account yet, you may choose not to call the setUID method when creating an order.**

After creating a payment order, call the cashier payment interface to make the payment. You can use the following code snippet to call payment:



**1.3.4 SMS payment foreground API (discarded)**



| **Parameter** | **Description** |
| --- | --- |
| activity | The object of the current activity |
| payInfo | The VivoPayInfo object that encapsulates basic order information. See the Appendix for details. |
| callback | The result that is called back after the payment is completed. Please verify with the CP Server after receiving the payment result. |
| payType | VivoConstants.VIVO\_SMS\_PAYMENT |

This interface can directly call the SMS payment page so there is no need to redirect from the cashier.

**This interface has been discarded in version 1.3.0.0. If you have connected to this interface, please discard payment calling of this interface and ensure the adaptability.**

**1.4 Notes**

1. The earliest Android version supported by the SDK is 22, so make sure the international phone runs Android 22 or above.
2. **TargetSDKVersion is recommended to be 29 or above.**

**1.5 Appendix**

**1.5.1 Payment order parameter table**

| **Name** | **Type** | **Meaning** | **Required?** | **Description** |
| --- | --- | --- | --- | --- |
| appId | String | appId | Required | The ID of the app; this can be obtained from the developer platform |
| productId | String | Product ID | Required | The product ID generated by the app itself and synced to the developer platform for maintenance |
| partnerOrderId | String | Order serial number | Required | Generated by the app itself and should be unique for each order |
| expireTime | String | The expiration date of each order | Optional | Generated by the app itself in the format of yyyyMMddHHmmSS |
| extInfo | String | Passthrough parameter | Required | Generated by the app itself and will be returned during Server callback |
| partnerOpenid | String | The Openid called back from the account | Required | Can be obtained after account login is called (set as "null" if you don't connect to the account login API) |
| notifyUrl | String | Callback URL | Required | The URL generated by the app itself and used to receive the payment callback result from the Server |
| sign | String | Signature verification | Required | Obtained via the signature parameter (see the signature parameter table for details) |
| signType | String | Algorithm type for signature verification | Required | Both MD5 and RSA algorithms are supported, though we recommend using RSA as it provides better security. |
| VivoRoleInfo | VivoRoleInfo | Role information | Optional | Game character information generated by the app itself |

VivoRoleInfo roleInfo = new VivoRoleInfo();

roleInfo.setServiceAreaName(serviceName);//Server name

roleInfo.setRoleName(roleName);//Character name

roleInfo.setServiceAreaName(serviceName);//Server ID

roleInfo.setRoleId(roleId);//Character ID

roleInfo.setRoleGrade(rolegrade);//Character level

**Note:**

If you don't want to maintain products or complete transactions via the productID on the developer platform, you can instead upload the product name and its price in its local currency. In this case, the entry that allows the user to switch the country of payment will be hidden from the cashier screen. For more details, refer to section **1.5.5 Payment notes.**

**1.5.2 Signature parameter table**

| **Name** | **Meaning** | **Required?** | **Description** |
| --- | --- | --- | --- |
| appId | appId | Required | The ID of the app; this can be obtained from the developer platform |
| productId | Product ID | Required | The product ID generated by the app itself and synced to the developer platform for maintenance |
| partnerOrderId | Order serial number | Required | Generated by the app itself and should be unique for each order |
| expireTime | The expiration date of each order | Optional | Generated by the app itself in the format of yyyyMMddHHmmSS |
| extInfo | Passthrough parameter | Required | Generated by the app itself and will be returned during Server callback |
| partnerOpenid | The Openid called back from the account | Optional | Can be obtained after account login is called (set as "null" if you don't connect to the account login API) |
| notifyUrl | Callback URL | Required | The URL generated by the app itself and used to receive the payment callback result from the Server |

When signing using the above parameters, if the value of a parameter is "null" or empty, do not put it in the signature; otherwise signature verification may fail.

**Coding example**:

String cpOrder = UUID.randomUUID().toString();

Map<String, String> params = new HashMap<>();

params.put("appId", Constants.getAppId());

params.put("productId",productId);

params.put("partnerOrderId", cpOrder);

params.put("expireTime", expireTime);

params.put("extInfo", Constants.APP\_EXT\_INFO);

params.put("partnerOpenid", uid);

params.put("notifyUrl", Constants.APP\_NOTIFY\_URL);

//Countersign parameter, Constants.APP\_PRIVATE\_RSA\_KEY is your RSA private key

String sign = VivoSignUtils.getVivoRsaSign(params,Constants.APP\_PRIVATE\_RSA\_KEY)

**1.5.3 Payment callback message table**

If you call the cashier payment API or SMS payment foreground API to make the payment, a payment callback will be received after the payment is completed. See the detailed callback message table below:

| **code** | **Message** |
| --- | --- |
| 0 | Payment successful |
| 1 | Payment canceled |
| 2 | Payment failed |
| 3 | Internal error |
| 4 | Parameter error |
| 5 | Payment timeout |
| 7 | Processing payment... |

**Note:**

When the payment is being processed, you can use the **order query interface** on the Server to obtain the payment processing status.

**1.5.4 Country code list**

| **Country** | **Country code** |
| --- | --- |
| India | IN |
| Thailand | TH |
| Indonesia | ID |
| Philippines | PH |
| Malaysia | MY |
| Vietnam | VN |

**1.5.5 Payment notes**

**If you don't want to maintain products or complete transactions via the productID on the developer platform, you can instead upload the product name and its price in its local currency.**

Here is an example of how to create payment orders:





**Payment order parameter table:**

| **Name** | **Type** | **Meaning** | **Required?** | **Description** |
| --- | --- | --- | --- | --- |
| appId | String | appId | Required | The ID of the app; this can be obtained from the developer platform |
| productName | String | Product name | Required | Product name |
| partnerOrderId | String | Order serial number | Required | Generated by the app itself and should be unique for each order |
| productPrice | String | Product price | Required | This price is the price in the local currency. The SDK acquires the user's country code and converts the price accordingly. |
| extInfo | String | Passthrough parameter | Required | Generated by the app itself and will be returned during Server callback |
| partnerOpenid | String | The Openid called back from the account | Not  Required | Can be obtained after account login is called (set as "null" if you don't connect to the account login API) |
| notifyUrl | String | Callback URL | Required | The URL generated by the app itself and used to receive the payment callback result from the Server |
| sign | String | Signature verification | Required | Obtained via the signature parameter (see the signature parameter table for details) |
| signType | String | Algorithm type for signature verification | Required | Both MD5 and RSA algorithms are supported, though we recommend using RSA as it provides better security. |
| VivoRoleInfo | VivoRoleInfo | Role information | Optional | Game character information generated by the app itself |

VivoRoleInfo roleInfo = new VivoRoleInfo();

roleInfo.setServiceAreaName(serviceName);//Server name

roleInfo.setRoleName(roleName);//Character name

roleInfo.setServiceAreaName(serviceName);//Server ID

roleInfo.setRoleId(roleId);//Character ID

roleInfo.setRoleGrade(rolegrade);//Character level

**Payment signature parameter table**

| **Name** | **Meaning** | **Required?** | **Description** |
| --- | --- | --- | --- |
| appId | appId | Required | The ID of the app; this can be obtained from the developer platform |
| productName | Product name | Required | Product name |
| partnerOrderId | Order serial number | Required | Generated by the app itself and should be unique for each order |
| productPrice | Product amount | Required | This price is the price in the local currency. The SDK acquires the user's country code and converts the price accordingly. |
| extInfo | Passthrough parameter | Required | Generated by the app itself and will be returned during Server callback |
| partnerOpenid | The Openid called back from the account | Optional | Can be obtained after account login is called (set as "null" if you don't connect to the account login API) |
| notifyUrl | Callback URL | Required | The URL generated by the app itself and used to receive the payment callback result from the Server |

When signing using the above parameters, if the value of a parameter is "null" or empty, do not put it in the signature; otherwise signature verification may fail.

**2. Payment Server**

## 2.1 Preparations

1. Communicate with our business staff who will input the partner information and app information as well as generate the appID and appKey.
2. The appID and appKey are required to access the Payment Server, wherein the appKey is used to ensure transaction security and must not be leaked. It is not recommended to store the appKey on your Client.
3. Communicate with our business staff to determine the supported countries and payment methods. The our operation staff will implement the support accordingly.

## 2.2 Payment process

vivo payment provides the following two services to its partners:

1. **Asynchronous callback notification:**

After the payment is completed, we will notify the partner of the successful payment via callback. The callback notification URL comes from the parameter notifyUrl of the Client's payment interface.

1. **Transaction query interface:**

The partner can query the transaction information via this interface.

The vivo payment process is shown as follows:



## 2.3 Asynchronous callback notification

After the payment is completed, vivo Payment Server will send the transaction result to the notifyUrl provided by the Content Provider (CP) when submitting the payment. The CP is recommended to change the order's transaction status according to the asynchronous notification result. This callback notification will be sent up to 17 times until the CP Server gives a successful response. The vivo Server will stop sending notifications if it receives 17 failed responses. In this case, you will need to correct the order's payment results according to [2.4 Order query interface](#_四、订单查询接口).

Successful response: The http response code is 200, and the response message reads "success".

Failed response: The http response code is not 200, and the response message reads "fail".

Result notification parameters are shown as follows:

|  |  |  |
| --- | --- | --- |
| Name | Meaning | Description |
| appId | App identifier | The app identifier assigned by vivo |
| vivoOrderNum | vivo's transaction order number |  |
| partnerOrderId | Partner's order number |  |
| productId | Product ID | The product ID used when placing the order |
| orderAmount | Order amount | The order amount, correct to up to three decimal places |
| currency | Currency unit | India = INR  Indonesia = IDR  Thailand = THB  Philippines = PHP  Malaysia = MYR  United States = USD |
| partnerOpenid | vivo account identifier |  |
| extInfo | CP passthrough parameter |  |
| paymentTime | Payment completion time | Format: yyyyMMddHHmmss |
| sign | Signature | For signature rules, see: [2.6 Signature and verification](#_六、签名与验签) |

**Example of "sign" generation (MD5):**

**to\_lower\_case(md5\_hex(appId=XXX&currency=XXX&extInfo=XXX&orderAmount=XXX&partnerOpenid=XXX&partnerOrderId=XXX&paymentTime=XXX&productId=XXX&vivoOrderNum=XXX&to\_lower\_case(md5\_hex(appSecret))))**

**Refer to section 2.6.2 for the RSA signature.**

## 2.4 Order query interface

### 2.4.1 Interface description

This interface is for the partner to query the vivo payment details.

### 2.4.2 Interface URLs

International: https://pay.vivoglobal.com/api/partner/query

India: https://in-pay.vivoglobal.com/api/partner/query

### 2.4.3 Request parameters

Request type: POST

|  |  |
| --- | --- |
| Content type | : application/x-www-form-urlencoded |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Meaning | Length | Description |
| signType | Signature type | 15 | Supports MD5 and RSA. RSA is recommended. |
| sign | Signature information | 64 | For signature rules, see: [2.6 Signature and verification](#_六、签名与验签) |
| appId | App identifier | 50 | The app identifier assigned by vivo |
| vivoOrderNum | vivo's transaction order number | 25 | Pass at least one of vivoOrderNum and partnerOrderId |
| partnerOrderId | Partner's order number | 100 | Pass at least one of vivoOrderNum and partnerOrderId |

**Example of "sign" generation (MD5):**

**sign=to\_lower\_case(md5\_hex(appId=XXX&partnerOrderId=XXX&vivoOrderNum=XXX&to\_lower\_case(md5\_hex(appSecret))))**

**Please refer to section 2.6.2 for the RSA signature.**

### 2.4.4 Response parameters

The vivo Server will return a message in JSON format after successful processing. Here is the JSON data structure:

|  |  |  |
| --- | --- | --- |
| Name | Meaning | Description |
| responseCode | Response code | 1000 = Query succeeded  1001 = Parameter [XXX] is incorrect  1002 = Signature is incorrect  1003 = Order amount is incorrect  1004 = Order does not exist  1005 = App does not exist  2000 = Other exceptions |
| responseMsg | Response message | The response message corresponding to the response code |
| When respCode = 1000, the following parameters are valid | | |
| tradeStatus | Transaction status | Completed  Waiting  Failure |
| appId | App identifier | The app identifier assigned by vivo |
| vivoOrderNum | vivo's transaction order number |  |
| partnerOrderId | Partner's order number |  |
| productId | Product ID | The product ID used when placing the order |
| orderAmount | Order amount | The order amount, correct to up to three decimal places |
| currency | Currency unit | India = INR  Indonesia = IDR  Thailand = THB  Philippines = PHP  Malaysia = MYR  United States = USD |
| partnerOpenid | vivo account identifier |  |
| extInfo | CP passthrough parameter |  |
| paymentTime | Payment completion time | Format: yyyyMMddHHmmss |
| sign | Signature | For signature rules, see: [2.6 Signature and verification](#_五、签名与验签) |

**Example of "sign" generation (MD5):**

**to\_lower\_case(md5\_hex(appId=XXX&currency=XXX&extInfo=XXX&orderAmount=XXX&partnerOpenid=XXX&partnerOrderId=XXX&paymentTime=XXX&productId=XXX&responseCode=XXX&responseMsg=XXX&tradeStatus=XXX&vivoOrderNum=XXX&to\_lower\_case(md5\_hex(appSecret))))**

**Please refer to section 2.6.2 for the RSA signature.**

## 2.5 IP whitelist query interface

### 2.5.1 Interface description

This interface is for the partner to query the Payment Server whitelist.

### 2.5.2 Interface URLs

International: https://pay.vivoglobal.com/api/partner/ipwhiteList

India: https://in-pay.vivoglobal.com/api/partner/ipwhiteList

### 2.5.3 Request parameters

Request type: POST

|  |  |
| --- | --- |
| Content type | : application/x-www-form-urlencoded |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Meaning | Length | Description |
| signType | Signature type | 15 | Currently supports MD5 |
| sign | Signature information | 64 | For signature rules, see: [2.6 Signature and verification](#_六、签名与验签) |
| appId | App identifier | 50 | The app identifier assigned by vivo |

**Example of "sign" generation (MD5):**

**sign=to\_lower\_case(md5\_hex(appId=XXX&to\_lower\_case(md5\_hex(appSecret))))**

**Please refer to section 2.6.2 for the RSA signature.**

### 2.5.4 Response parameters

The vivo Server will return a message in JSON format after successful processing. Here is the JSON data structure:

|  |  |  |
| --- | --- | --- |
| Name | Meaning | Description |
| responseCode | Response code | 1000 = Query succeeded  1001 = Parameter [XXX] is incorrect  1002 = Signature is incorrect  1005 = App does not exist  1006 = Query too frequent  2000 = Other exceptions |
| responseMsg | Response message | The response message corresponding to the response code |
| When respCode = 1000, the following parameters are valid | | |
| whiteList | The payment whitelist with items separated by commas | E.g. 119.254.239.252,119.254.239.253 |

## 2.6 Signature and verification

vivo payment will sign the key information for each request, and the recipient should also verify the signature when receiving the message.

### 2.6.1 Signature algorithm

This is defined by the signType field, and currently supports MD5 and RSA. We suggest new users use the RSA method. Only the steps for RSA connection are provided here (users who have used MD5 can keep using the MD5 signature).

### 2.6.2 RSA signature

(1) Generating RSA key pairs

Base64 in the code is java.util.Base64

/\*\*

\* Generate RSA key pairs

\* @return

\*/

public static String[] generateRSAKeyPair() throws NoSuchAlgorithmException {

KeyPairGenerator kpg = KeyPairGenerator.getInstance("RSA");

// Initialize KeyPairGenerator objects, key length

kpg.initialize(2048);

// Generate key pairs

KeyPair keyPair = kpg.generateKeyPair();

// Get the public key

Key publicKey = keyPair.getPublic();

// Get the private key

Key privateKey = keyPair.getPrivate();

String priKey = new String(Base64.getEncoder().encode(privateKey.getEncoded()));

String pubKey = new String(Base64.getEncoder().encode(publicKey.getEncoded()));

return new String[]{priKey, pubKey};

}

You need to upload the RSA public key to the developer platform (please refer to section 1.1 for details) and keep the private key yourself.

(2) Countersigning and verification

It is recommended to user the server SDK (located in the assets/server SDK directory) to complete countersigning and verification operations.

Countersigning:

// Save the parameters that will be used for the countersign

Map<String, String> params = new HashMap<>();

params.put("key1", "value1");

params.put("key2", "value2");

params.put("key3", "value3");

SignAlgorithm signAlgorithm = SignAlgorithmFactory.create("RSA");

String sign = signAlgorithm.sign(params, merchantPrivateKey); // merchantPrivateKey is your private key

Verification:

// Save the parameters that will be used for the verification

Map<String, String> params = new HashMap<>();

params.put("key1", "value1");

params.put("key2", "value2");

params.put("key3", "value3");

SignAlgorithm signAlgorithm = SignAlgorithmFactory.create("RSA");

boolean result = signAlgorithm.checkSign(params, sign, payPublicKey); // payPublicKey is your public key for payment

**3. Account server (optional)**

## 3.1 Preparations

Since RSA countersigning is not yet supported by the server (under development), currently, the token verification interface still uses MD5 countersigning. Please contact our technical staff associated with the server to obtain the MD5 key.

1. Communicate with our business staff who will input the partner information and app information as well as generate the appID and appKey.
2. The appID and appKey are required to access the Account Server, wherein the appKey is used to generate a signature and must not be leaked. It is not recommended to store the appKey on your Client.
3. After the CP Client signs in successfully, it will receive the partnerOpenid and openToken.
4. The CP Client passes the partnerOpenid and openToken to the CP Server which will use the openToken to initiate a check.
5. After receiving a response, the CP Server should match the partnerOpenid from the response with the one provided by the CP Client, and complete a check if the match is successful.

## 3.2 openToken check process

Note: Offline games cannot be checked via the Server, as the partnerOpenid obtained by the Client may be fake.



## 3.3 openToken check interface

### 3.3.1 Interface description

After the CP Client obtains the openToken, the CP Server should initiate a check on the openToken to ensure the authenticity of the information obtained by the CP Client. If the openToken check is successful, the interface will return the partnerOpenid for this openToken, and the CP Server should match this partnerOpenid with the one submitted by the CP Client and complete the check if the match is successful.

### 3.3.2 Interface URLs

India: https:// [**in-usrsys-api.vivoglobal.com/api/login/validateOpenToken**](http://in-usrsys-api.vivoglobal.com/api/login/validateOpenToken)

Russia: https:// [**ru-usrsys-api.vivoglobal.com/api/login/validateOpenToken**](http://ru-usrsys-api.vivoglobal.com/api/login/validateOpenToken)

International: https:// [**asia-usrsys-api.vivoglobal.com/api/login/validateOpenToken**](http://asia-usrsys-api.vivoglobal.com/api/login/validateOpenToken)

### 3.3.3 Request parameters

Request type: POST

| **Content type** | **: application/x-www-form-urlencoded** |
| --- | --- |
|  |  |

| **Name** | **Meaning** | **Length** | **Description** |
| --- | --- | --- | --- |
| signType | Signature type | 15 | Currently supports MD5 |
| sign | Signature information | 64 | For signature rules, see: [**3.4 Signature and verification**](file:///D:\%E6%88%91%E7%9A%84%E6%96%87%E6%A1%A3\11085273\Desktop\vivo%E5%A4%96%E9%94%80%E8%81%94%E8%BF%90%E5%B8%90%E5%8F%B7%E6%9C%8D%E5%8A%A1%E7%AB%AF%E6%8E%A5%E5%85%A5%E6%96%87%E6%A1%A3.docx#_%E5%9B%9B%E3%80%81%E7%AD%BE%E5%90%8D%E4%B8%8E%E9%AA%8C%E7%AD%BE) |
| appId | App identifier | 50 | The app identifier assigned by vivo |
| openToken | openToken | 25 | The open token is valid for 10 minutes |
| timestamp | Timestamp | 100 | The total seconds till now from 01/01/1970 00:00:00 GMT |
| nonceStr | Nonce string | 32 | A nonce string of no more than 32 digits which may only include letters and numbers |

**Example of "sign" generation:**

**sign=to\_lower\_case(md5\_hex(appId=XXX&orderAmount=XXX&partnerOrderId=XXX&vivoOrderNum=XXX&to\_lower\_case(md5\_hex(appSecret))))**

### 3.3.4 Response parameters

The vivo Server will return a message in JSON format after successful processing. Here is the JSON data structure:

| **Name** | **Meaning** | **Description** |
| --- | --- | --- |
| responseCode | Response code | 1000 = Query succeeded; 1001 = Parameter error; 1002 = Signature error; 1100 = openToken expired; 1005 = App does not exist; 2000 = Other exceptions |
| responseMsg | Response message | The response message corresponding to the response code |
| When responseCode = 1000, the following parameters are valid |  |  |
| partnerOpenid | partnerOpenid | The CP Server matches this with the partnerOpenid submitted by the CP Client and completes the check if the match is successful |

**Example of "sign" generation:**

**to\_lower\_case(md5\_hex(appId=XXX&currency=XXX&extInfo=XXX&orderAmount=XXX&partnerOpenid=XXX&partnerOrderId=XXX&paymentTime=XXX&responseCode=XXX&responseMsg=XXX&tradeStatus=XXX&vivoOrderNum=XXX&to\_lower\_case(md5\_hex(appSecret))))**

## 3.4 Signature and verification

vivo payment will sign the key information for each request, and the recipient should also verify the signature when receiving the message.

### 3.4.1 Signature algorithm

This is defined by the signType field, and currently supports MD5 and RSA. RSA is recommended.

### 3.4.2 Key information

All the fields in the message except "signType" and "sign".

### 3.4.3 Signed string

The combined result of key information and app secret information. Here is an example of one such combination:

key1=value1&key2=value2&...&keyn=valuen&to\_lower\_case(md5\_hex(appSecret))

**Please refer to section 2.6.2 for the RSA signature.**

### 3.4.4 Signed result

The result from the MD5 operation on the signed string, i.e.:

to\_lower\_case(md5\_hex(key1=value1&key2=value2&...&keyn=valuen&to\_lower\_case(md5\_hex(appSecret))))

**Please refer to section 2.6.2 for the RSA signature.**

### 3.4.5 Notes

1) The key information in the signed string should be sorted by the key value in ascending order;

2) An empty string or null value is not involved in the signature operation.

**4. FAQ**

**1) What should I do if the payment cashier prompted a parameter error?**

A: Please carefully check the parameter table and signature table in the appendix to the connection documentation against any errors. If you are unable to confirm any errors, please contact our technical personnel and send the appId and partnerOrderId to us for troubleshooting.

**2) What should I do if the cashier prompted that payment is not supported for this country?**

A: International payment currently supports payment in India, Thailand, Indonesia, Malaysia, and the Philippines. Please confirm if you made the payment using a vivo international phone from the above countries. If not, payment is not supported for this country. If you do not have relevant international phones, use our vivo cloud testing platform or contact our business and technical staff.

**3) What should I do if the game crashed when calling payment?**

A: Please confirm if the game called our "init" and other methods in the main process. If not, the above issue may occur.

**4) What should I do if the payment cashier prompted that signature verification failed?**

A: Please carefully check the signature parameter table in the appendix to the connection documentation against any errors. If you are unable to confirm any errors, please contact our technical personnel for analysis.

**5) Where can I find methods such as parameters and signatures?**

A: Relevant tool methods can be found under the "utils" directory. You can directly call them after copying them to the "utils" package of the target project.

**6) What is the returned value of the device country code?**

A: The obtained device country code is a two-digit code following the ISO 3166-1 standard. For example, IN for India, ID for Indonesia, PH for the Philippines, TH for Thailand, and MY for Malaysia.

**7) I have connected to the vivo account, but why did it prompt that login failed due to a parameter error when calling the login interface?**

A: It could be an issue with the appId. Please confirm if the appId is correct. If so, it is probably because the appId data has not synced. Please contact our technical personnel for assistance.

**8) I have connected to the vivo account, but why can't I use the phone on the vivo cloud testing platform to sign in to my account or there was no response when I tapped account login?**

A: The cloud testing platform might have account restrictions. If you encountered such issue or you cannot sign out of the current account on the phone, please contact our technical personnel to erase account data.

**9) I have connected to the vivo account, but why did it prompt that the login status expired when making a payment?**

A: This is an account mechanism issue and is being optimized. You can sign out of your vivo account and sign in again via the phone settings.

**10) I have connected to the vivo account and signed in successfully, but why did I receive the callback code for canceled login instead of successful login?**

A: Please confirm if you were using a vivo domestic (China) phone for login testing, which could lead to this issue.

**11) Why did it prompt "Payment is currently not supported for this app" when making a payment?**

A: This is because payment has not been enabled for this app in the background. Please contact our business staff for assistance.

**12) What is the currency used when passing product amount?**

A: We currently provide an exchange rate conversion service. You can use USD for product information on the developer platform. When making a payment, pass the corresponding product ID to us, and we will automatically obtain the product information from the developer platform and convert the amount into local currency based on the country code to process the payment.

**13) If I connect to the vivo account, are there interfaces for account logout and cancellation?**

A: There are no additional interfaces for account logout, but you can perform relevant account operations in the phone's vivo account settings.