Citcon Pay Android SDK Documentation

Version 2.4.1

I. Introduction

Citcon's Android SDK was designed for online merchants to integrate Citcon payment solutions effortlessly into their own Android apps. By using the SDK, merchant developers can focus on business logics without having to understand the plumbing of payment transactions. The payment experience will be totally transparent and seamless to end consumers. This version of the SDK supports payments through Alipay, AlipayHK, WeChat Pay, Union Pay, KakaoPay, Dana.

The Citcon Android development SDK was developed in JAVA and targets Android 4.1 and above. The SDK is distributed as an Android Archive Library (aar) in Maven Center.

II. The Citcon Pay Framework

This section details the main components of the Citcon Pay Framework for Android development. The commonly used header files and their purposes will be listed here, and a step-by-step example of integrating the framework in a demo merchant app will be shown in the next section.

1. Library identity
    a. Package Name: citcon.sdk
    b. Current version: 2.4.1
    c. com.citcon.sdk:mobile:2.4.1

2. Required dependencies
    - com.android.support:appcompat-v7:25.3.1
    - com.android.support:multidex:1.0.1
    - com.citcon.sdk:alipay:15.8.03
    - com.citcon.sdk:unionpay:3.3.0

3. Class references
    a. **CPaySDK**

    The CPaySDK class performs the most common payment related tasks: set up merchant token, send order to Citcon Pay and query the status of a specific transaction.

Name	**Description**

requestOrder	Sends order information to Citcon and initiates the payment transaction. Has an unique callback: gotOrder.
inquireOrder	Query the status of a specific payment transaction. Has an unique callback: inquiredOrder.
inqureOrderByRef	Query by reference id , currency and vendor
gotOrder	Callback of requestOrder. This method handles the outcome of the payment transaction initiation and continues the process by either Alipay or WeChat.
setMode	Set SDK running mode. If not set, default mode is `CPayMode`.PROD.

Method summary

b. CPayOrder

Represents the order and payment information merchant wants to send to Citcon Pay for processing.

Constructor Property summary

Name	Type	Description	Example
referenceId	String	A reference merchant creates and assigns to the transaction.	123xyz
subject	String	A customer-defined description of the transaction	Gift for Mom

body	String	A more detailed customer-defined description of the transaction	A Blu-ray player and a few great movies
amount	String	Total charge amount of the transaction in cents.	245 ($2.45)
currency	String	The type of currency defined in a three-letter code	USD or CAD or CNY
vendor	String	The name of vendor that will process the payment. Only "Alipay" is supported in this version of the SDK	alipay , wechatpay, alipay_hk, kakaopay, dana, upop
ipnUrl	String	The URL of a page Citcon Pay should post transaction status to. Normally this should be a page on the merchant's website.	http://www.xyz.com/notify.php
callbackUrl	String	The URL of a page Citcon Pay should redirect customer to after the payment transaction has completed.	http://xyz.com/confirm.php
allowDuplicate	boolean	Flag to control duplicate orders.	true / false
ext	hashMap	Customized key, value <String, String>	"reference2", "123456789"

c. CPayOrderResult

Holds the status and message for a transaction returned by Citcon Pay service. An instance of the CPayOrderResult class can be inspected in the callback handler of the requestOrder method of CPaySDK.

Property summary

Name	Type	Description	Example
mRedirectUrl	String	Variable used in the WeChat sequence. Will only be populated when paying by WeChat.	
mOrderId	String	ID of the Order from the payment process.	123xyz
mOrder	CPayOrder	The actual Order object.	
mSignedString	String	Variable returned by the Order initiation process. Used in the Alipay sequence.	
mOrderSpec	String	Variable returned by the Order initiation process. Used in the Alipay sequence.	
mMessage	String	Detailed description of the status of a transaction	Transaction succeeded
mStatus	String	The status code for the result	0 (refer to 4.Status Code)
mCurrency	String	Order currency.	USD, CNY

d. CPayInquireResult

Holds the detailed status information for a transaction. An instance of the CPayInquireResult class can be inspected in the callback handler of the inquireOrder method of CPaySDK.

Property summary

Name	Type	Description	Example

mId	String	An unique identifier of the transaction. This ID is generated by Citcon Pay.	123456679
mReference	String	A reference identifying the transaction. This ID is generated by merchant	Abc123
mType	String	The type of the transaction: “charge” or “refund”	charge
mAmount	String	The total amount of the transaction in cents	225 ($2.45)
mCurrency	String	The name the of the currency in a three-letter code	USD or CAD or CNY
mTime	String	The timestamp for the transaction	2017/06/22 1:23:12 PM
mStatus	String	The status of the transaction	success
mNote	String	The note of the transaction.	

4. Status code

Status Code	Description
0	Success
-1	Bad signature, unregistered AppId, wrong AppId, mismatched AppId or some other error.

-2	The customer cancels the payment, switches back to the app, the customer cannot pay, and so on.
8000	Order is being processed
4000	Payment failed
5000	Duplicate request
6001	The customer cancels the payment
6002	Network connection error
6004	Unknown payment result
others	Other payment error

5. Notification

After the payment is completed, the Citcon SDK will automatically query immediately to ensure that the correct results are obtained. Therefore, you can also register a BroadcastReceiver to receive the results of an automatic query. The type of the broadcast is the string "CPAY_INQUIRE_ORDER", and the additional push result object key is "inquire_result".

III. Example

In this section, a demo merchant app making Alipay payments through Citcon Pay is demonstrated step-by-step using JAVA. The source code of this demo app is also provided as part of the SDK distribution package.

1. Create a new Android Studio Application project.
2. Modify the AndroidManifest.xml of your project, add the following for wechat pay:

<application ...>

<activity-alias

android:name=".wxapi.WXPayEntryActivity"

```
        android:exported="true"
        android:targetActivity="sdk.PaymentActivity" />
    </application>

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

3. Modify the app-level build.gradle:

```
dependencies {
        implementation 'com.citcon.sdk:mobile:2.2.0'
        implementation 'com.citcon.sdk:aliplay:15.8.03'
        implementation 'com.citcon.sdk:unionpay:3.3.0'
}
```

4. In your calling activity, add the following:

```
   @Override
public void onResume()
{
        super.onResume();
        CPaySDK.getInstance(MainActivity.this, AUTH_TOKEN).onResume();
        CPaySDK.setMode(CPayMode.PROD);
}
```

The AUTH_TOKEN is the merchant token and needs to be populated accordingly.

5. Initiate a payment transaction after the lifecycle phase at step 5 has run:

```
CPayOrder order = new CPayOrder("1ZLLJULOCRW3LAU",
      "Test",
      "This is a test transaction",
      "200",
      "USD",
      "alipay",
      "http://www.xyz.com/listen.php",
      "http://www.xyz.com/confirmation.php",
      true);
 CPaySDK.getInstance().requestOrder(order, new OrderResponse<CPayOrderResult>()
```

```
{
    @Override
    public void gotOrderResult(final CPayOrderResult orderResult)
    {
        if(orderResult != null)
        {
                    // TODO: add your codes
            }
    }});
```

6. Call CPaySDK onResume () in onResume () of the activity of you app to ensure that Citcon SDK can callback your app correctly.

```
@Override
public void onResume() {
        …
        CPaySDK.getInstance(Activity.this, TOKEN).onResume();
}
```

7. Register a broadcast receiver to receive the result of inquire (optional):

```
// create a new receiver
BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
                CPayInquireResult result = (CPayInquireResult) intent.getSerializableExtra("in-
quire_result");
                if (result != null) {
                        // TODO: add your codes
                }
        }
}
// register the receiver onResume()
@Override
public void onResume() {
        super.onResume();
        …
        // register receiver
```

```
        IntentFilter filter = new IntentFilter();
        filter.addAction("CPAY_INQUIRE_ORDER");
        CPaySDK.getInstance().registerReceiver(receiver, filter);
}
// remember to unregister receiver
@Override
public void onPause() {
        supper.onPause();
        …
        // unregister
        CPaySDK.getInstance().unregisterReceiver(receiver);
}
```