Citcon Pay Android SDK Documentation

Version 2.7.0

Version No.	Modify Activity	Modify Description	Editor	Modify Date
2.3.1	Creation	Alipay and WeChat Pay	Raymond Zhuang	
2.4.1	Update	Alipay HK, Dana, Kakao Pay and UnionPay added; Added currency type: IDR, HKD, KRW Added 'ext' for extra parameters; Added 'inqureOrderByRef ' to query by reference id ; Use implementation to import Lib(Android); Use Cocapods to import Lib(iOS); Added new changes of infoplist for new payments(iOS);	Raymond Zhuang	
2.5.1	Update	New Builder to initialize CPayOrder (Android); KCP added;	Raymond Zhuang	
2.5.2	Update	CN pay accelerator added; SBPS added;	Raymond Zhuang	
2.5.3	Update	Update to SDK API 31; Replaced Broadcast with LiveData;	Raymond Zhuang	
2.5.4	Update	Support TOSS; New "ext", "instalment", "card issuer", "receipt Type" fields; Changed "goods" field;	Raymond Zhuang	
2.7.0	Update	Support Cashapp; New "autoCapture", "deepLink";	Andy	9/20/2024

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, TOSS.

The Citcon Android development SDK was developed in JAVA and targets Android 8 and above. The SDK is distributed as an Android Archive Library (aar) in Maven Center.

With this payment solution, the merchant's application will present a payment button when a consumer completes the payment and checks out.

- The user clicks the payment/checkout button. After being redirected to Wallets app/H5 page/Browser, the user can log in, and then complete the payment.
- Once the payment is completed, the user is redirected back to the merchant app with the payment result. The merchant could check the result and make decision on how to move forward.
- In the meantime, an asynchronous notification will be sent to the merchant with the payment result. The notification is reliable with build-in retry mechanism.

II. Target audience

This document targets at the technical person who are intending to integrate with the Citcon's Android app payment solution.

III. Terminologies

1. **Request**
   A process of transmitting data in the form of character string required by client to recipient.

2. **Return**
   Citcon returns processed result data in the form of character string to client directly.

3. **Notification**
   Asynchronous notification from Citcon server to merchant. Citcon server takes the

initiative to notify and feeds the processed result back to merchant's website after the data received has been processed by Citcon.

4. **Observation**
   Asynchronous notification from Citcon SDK to merchant APP. After the received data is processed by Citcon SDK, the Citcon SDK will actively notify and feedback the processing results to the merchant's APP.

5. **H5 Payment**
   H5 payment uses the H5 page appears in the browser or Webview embedded in APP to complete payment.

6. **Native Payment**
   Native payment calls wallet App for Native page to complete payment.

7. **Browser Payment**
   Browser payment calls the device browser page to complete payment.

IV. Supported currency

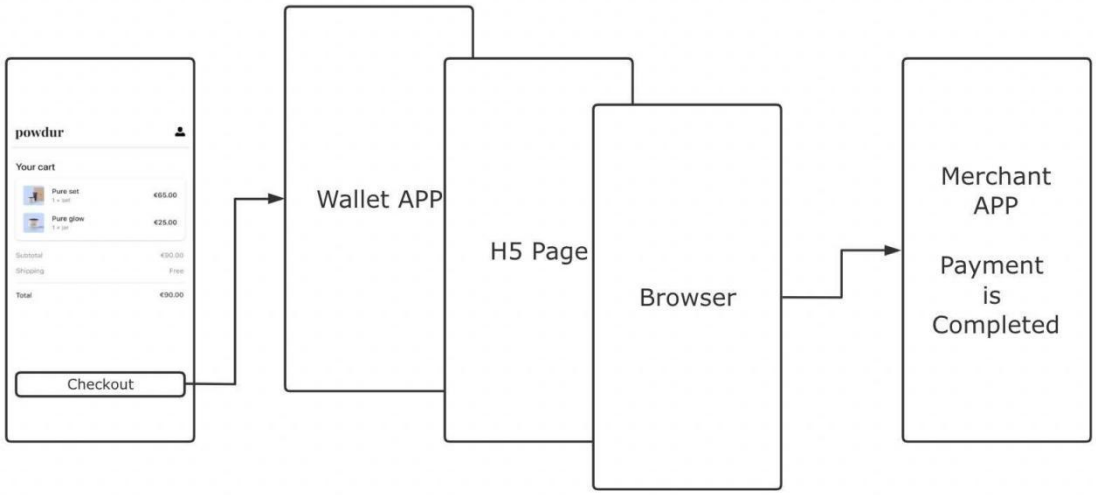
Citcon mainly supports the following currencies:

US Dollar, Chinese Yuan, Singapore Dollar, Japanese Yen, Canadian Dollar, Australian Dollar, Euro, New Zealand Dollar, British Pound, Thai Baht, Hong Kong Dollar, Swiss Franc, Swedish Krona, Danish Krone, and Norwegian Krone, etc.

V. Payment flow and user experience

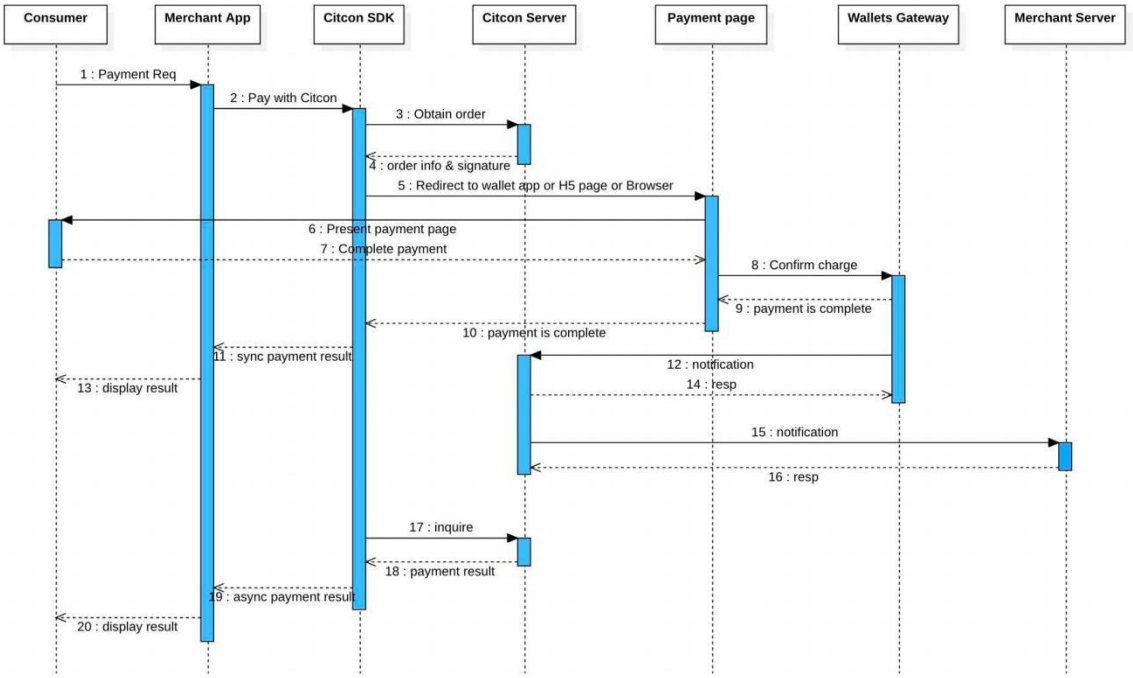
The following figure illustrates the workflow of In-app payment.

1. Customer checks out in merchant app and choose to pay with Citcon SDK provides.
2. Merchant app sends a transaction request to Citcon.
3. Citcon SDK integrated in merchant app calls wallet app/H5 page/browser.
4. Customer completes payment in wallet app/H5 page/browser.
5. Wallet app/H5 page/browser returns to merchant app.
6. Merchant app receives the payment result processed by Citcon SDK.



VI. Interaction process

Function process:



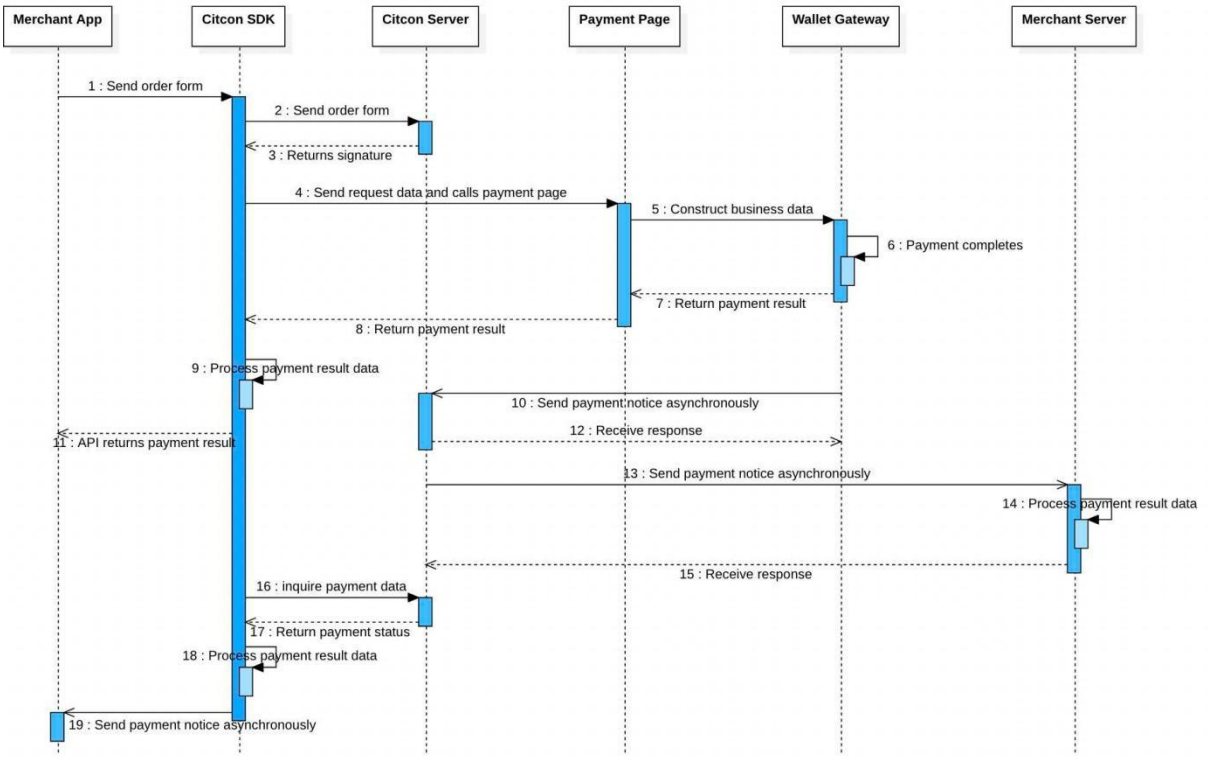
Key steps in the payment process are explained below:

Step 2: Pay with Citcon: this message is referring to the payment target requestOrder provided by SDK, which send order information to call the Citcon SDK interface – see "Request Parameters Description" for more details on order format.

Step 11: Synch payment result: payment API called by merchant's app in step2 returns a final payment result (a synchronous response) – see "Synchronous Response Parameters".

Step19: Asynch payment result: Citcon SDK sends an asynchronous notification to merchant' app (note: step 19 may happen before step 11, depending on wallet). – see "Asynchronous Notification Parameters".

VII. Data interaction



1. **Construct order data and sign**
   Citon server side generates digital signature and a set of data for Citcon mobile payment SDK using the Citcon payment development API.

2. **Send request data**
   Send the constructed data to Payment page.

3. **Payment page process request data**
   Payment page will send payment request data, in accordance with the business and payment policy, to wallet's payment server. The wallet's payment server will conduct security check and other verifications after receiving the payment request data. If and only if all the verification passes the security check, the payment request will be processed.

4. **Return the processed result data**
   Once a transaction/payment has been processed, Citcon will feed the processed data back to the merchant's client and server in two ways respectively.

a. On the client side, Citcon SDK directly feeds the processed result data back to the merchant's client. Also, the Citcon SDK sends an asynchronous notification with the processed payment result data.

b. On the server side, Citcon payment server initiates a notification using the page path set by the merchant under the parameter *ipnUrl* (if the merchant has not set the page path, this operation will not be conducted)

5. **Processing of the returned data by merchant**
After obtaining Citcon returned result data at the client's response receiving module or server asynchronous notification receiving module, merchant can process the received data taking into account the seller's own business logic (e.g., order update, automatically top-up the user's account, etc.) Merchant must use asynchronous notification as a payment's final result.

VIII. 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: 2.7.0
    c. com.citcon.sdk:mobile:2.7.0

2. Required dependencies
    - com.android.volley:volley:1.2.1
    - com.alipay.sdk:alipaysdk-android:15.8.17
    - com.citcon.sdk:unionpay:3.4.0
    - com.tencent.mm.opensdk:wechat-sdk-android:6.8.28
    - app.cash.paykit:core:2.5.0

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	Type	Description
mInquireResult	MutableLiveData<CPayInquireResult>	Observable CPayInquireResult

After the payment is completed, the Citcon SDK will automatically query immediately to ensure that the correct results are obtained(optional for AliPay and Wechat). Therefore, observe mInquireResult .

Method summary:

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.
setToken	Set SDK running token.

b. CPayOrder.Builder

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

Builder Property summary

Type	Method	Description	Example
CPayLaunchType	setLaunchType	KCP/SBPS = URL Other payments = OTHERS	URL/OTHERS
String	setReferenceId	A reference merchant creates and assigns to the transaction.	123xyz
String	setSubject	A customer-defined description of the transaction	Gift for Mom
String	setBody	A more detailed customer-defined description of the transaction	A Blu-ray player and a few great movies
String	setAmount	Total charge amount of the transaction in cents.	245 ($2.45)
String	setCurrency	The type of currency defined in a three letter code	USD, CAD, CNY, HKD, KRW, IDR, JPY
String	setVendor	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, card, payco, naverpay, banktransfer
String	setIpnUrl	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
String	setCallbackUrl	The URL of a page Citcon Pay should redirect customer to after the payment transaction has completed.	http://xyz.com/confirm.php
boolean	setAllowDuplicate	Flag to control duplicate orders.	true / false

HashMap	setExt	Customized key, value <String, String>	“reference2”, “123456789”
String	setCallbackFailUrl	New field for KCP/ SBPS	
String	setCallbackCancelUrl	New field for KCP/ SBPS	
Locale	setCountry	New field for KCP/ SBPS	
String	setNote	New field for KCP/ SBPS	
String	setSource	New field for KCP/ SBPS	
boolean	setAutoCapture	New field for KCP/ SBPS	
Goods	setGoods	New field for KCP/ SBPS	
Consumer	setConsumer	New field for KCP/ SBPS	
boolean	enableCNPayAcceleration	Use Chinese endpoint to accelerate connection(AliPay/UnionPay/ WechatPay)	
String	setInstallment	New field for TOSS	
String	cardIssuer	New field for TOSS	
String	receiptType	New field for TOSS	
String	setDeepLink	New field for CashApp	citcon://cpay.sdk

c. Goods

Property summary

Name	Type	Description
name	String	Name of good
taxable_amount	Integer	Taxable amount
tax_exempt_amount	Integer	Tax exempt amount
total_tax_amount	Integer	Total amount
total_discount_code	String	
sku	String	
category	String	
total_amount	String	
unit_amount	Integer	
quantity	Integer	
description	String	
product_type	String	
url	String	
unit_tax_amount	Integer	
total_discount_amount	Integer	
total_tax_rate	Integer	

d. Consumer

Property summary

Name	Type	Description
first_name	String	
last_name	String	
phone	String	
email	String	
reference	String	

zip	String	
country	String	
city	String	
street	String	
registration_time	Long	
registration_ip	String	
risk_level	String	
first_interaction_time	Long	
total_transaction_count	Long	

e. 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, CAD, CNY, HKD, KRW, IDR, JPY

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

IX. 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" />

        <activity android:name="activity.DemoActivity"
                android:exported="true"
                android:launchMode="singleTop"
                >
                ...
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>

                <intent-filter>
                        <action android:name="android.intent.action.VIEW" />
                        <category android:name="android.intent.category.DEFAULT" />
                        <category android:name="android.intent.category.BROWSABLE" />
                        <data android:scheme="citcon" android:host="cpay.sdk" />
                </intent-filter>
        </activity>

</application>

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

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

3. Modify the app-level build.gradle:

```
dependencies {
        implementation 'com.citcon.sdk:mobile:2.7.0'
}
```

4. In your calling activity, add the following:

```
@Override
public void onCreate()
{
    super.onCreate();
    CPaySDK.initInstance(DemoActivity.this, AUTH_TOKEN);
    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 4 has run:

```
CPayOrder order = new CPayOrder.Builder()
.setLaunchType(CPayLaunchType.URL)
.setReferenceId(1ZLLJULOCRW3LAU)
.setSubject("Test")
.setBody("This is a test transaction")
.setAmount("200")
.setCurrency("KRW")
.setVendor("payco")
.setIpnUrl("http://www.xyz.com/listen.php")
.setCallbackUrl("http://www.xyz.com/confirmation.php")
.setAllowDuplicate(true)
.setDeepLink("citcon://cpay.sdk")
.setSource("app_h5")
.setAutoCapture(true)
.setCountry(Locale.KOREA)
```

```
.setNote("note dddd")
.setCallbackFailUrl("https://exampe.com/fail")
.setCallbackCancelUrl("https://exampe.com/cancel")
.setConsumer("John","Doe","6145675309", "test@test.com","consumer-000")
.setGoods("Battery Power Pack", 0,0,0)
.build();
```

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().onResume();
}
```

7. Observe mInquireResult to receive the result of inquire:

```
// optional if only use WeChatPay and Alipay.
CPaySDK.mInqureResult.observe(this, new Observer<CPayInquireResult> {
  // TODO
  }
);
```

8. CashApp payment
   - Create order:

```
CPayOrder.Builder orderBuilder = new CPayOrder.Builder()
.setReferenceId("123")
.setAmount("8")
.setCurrency("USD")
.setCountry("US")
.setAutoCapture(true)
.setSource("app_native")
.setNote("test order")
```

```
.setIpnUrl("https://www.merchant.com/ipn")
.setCallbackUrl("https://www.merchant.com/success")
.setCallbackFailUrl("https://www.merchant.com/fail")
.setCallbackCancelUrl("https://www.merchant.com/cancel")
.setVendor("cashapppay")
.setLaunchType(CPayLaunchType.OTHERS)
.setDeepLink("citcon://cpay.sdk");
```

- Button UI (optional):

  a. Add the payment-buttons package dependency in Merchant App's build.gradle file:

```
dependencies {
        implementation 'app.cash.paykit:core:2.5.0'
}
```

  b. Add the following code to your app's layout XML

```
// Light style
<app.cash.paykit.core.ui.CashAppPayButton
  style="@style/CAPButtonStyle.Light"
  android:id="@+id/cashapp_button"
  android:layout_height="54dp"
  android:layout_width="match_parent"
  />

// Dark style
<app.cash.paykit.core.ui.CashAppPayButton
  style="@style/CAPButtonStyle.Dark"
  android:id="@+id/cashapp_button"
  android:layout_height="54dp"
  android:layout_width="match_parent"
  />
```

  c. Handle the button event:

```
findViewById(R.id.cashapp_button).setOnClickListener(v -> {
```

```
        // TODO
});
```