Citcon Pay SDK (Android)

Version 4.23.00

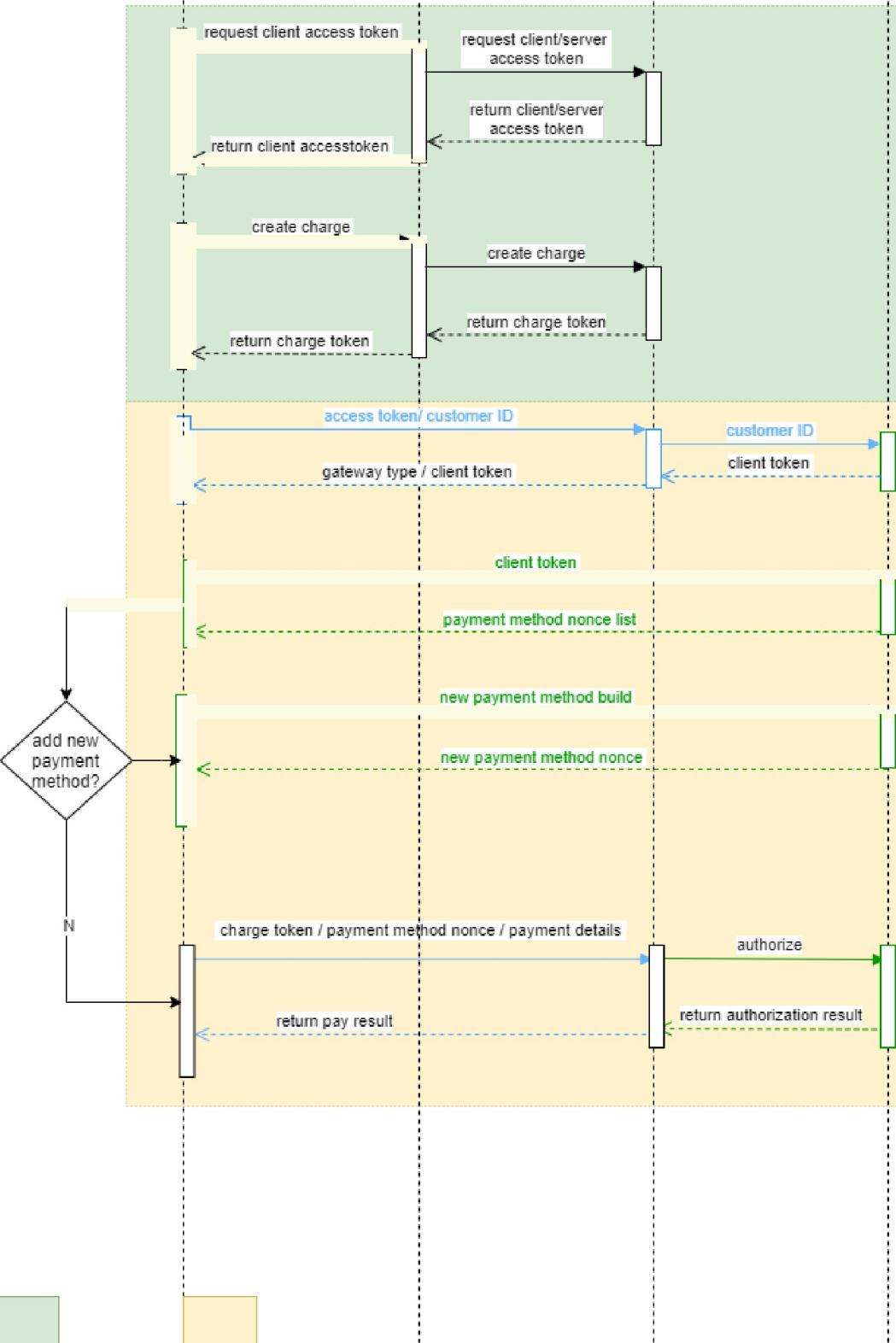
**@2023 Copyright Citcon**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version No.** | **Modify Activity** | **Modify Description** | **Editor** | **Modify Date** |
| 1.0.0 | Creation | Braintree | Raymond Zhuang | Dec 2021 |
| 3.1.2 | Update | UnionPay, Alipay and WeChat Pay | Raymond Zhuang | Apr 2022 |
| 4.22.07 | Update | Integrated UPI & Online in one ; TOSS, FOMO | Raymond Zhuang | Aug 2022 |
| 4.22.09 | Update | Support Alipay+ Pay | Andy | Oct 2022 |
| 4.23.00 | Update | Add non-Citcon WeChat SDK broadcast callback | Andy | Jan 2023 |

1. **Introduction**

Citcon UPI SDK was designed for online merchant to integrate Citcon payment solutions eﬀortlessly into their own app. By using the SDK, merchant developers can focus on businesss logic without having to understand the plumbing of payment transactions. This version of the SDK supports credit cards, PayPal , Venmo, AliPay, WeChatPay , UnionPay, TOSS and FOMO.

An exhibition of the payment transaction ﬂow will help developers better understand the integration process at the conceptual level, as illustrated in Figure.1 blow.



Merchant Clie nt

Merchant Server

Citcon Server

Vendor.°Gate '7ay Server

APP scope Nlo bile SD K scope



***Fig.1 - Payment transaction ﬂow from the merchant app to payment processor through Citcon***

This version of the SDK targets Android 5.0 and above. It is distributed as an imported dependency.

1. **Integration Steps**

This section will show you step-by-step how to integrate Citcon UPI SDK. The source code of the demo has also been included in the distribution package.

!!!Attention: Do not use “\_“(underscore) in project name. (for example: ~~demo\_app~~).

1. **Setup your project build.gradle at the end of repositories:**



1. **Add the dependency in your app build.gradle**

implementation ‘com.citcon.sdk:cpay:4.23.00’

1. **Get Mandatory Parameter from backend server**

Generate Access Token and Charge Token on your server. The reference has to be get from your server as well. those 3 parameters are mandatory to be set up in next step.

1. **Initialize SDK request**

Braintree Kotlin Example:

fun buildRequest(type: CitconPaymentMethodType): CPayRequest { return CPayRequest.PaymentBuilder

.accessToken(mAccessToken.value!!)

.chargeToken(mChargeToken.value!!)

.reference(mReference)

.consumerID(mConsumerID)

.request3DSecureVeriﬁcation(true)

.consumer(demo3DSsetup()!!)

.paymentMethod(CPayMethodType.UNKNOWN)//credit card

.build(CPayENVMode.UAT)

}

Braintree Java Example:

CPayRequest request = CPayRequest.PaymentBuilder.INSTANCE

.accessToken(accessToken)

.chargeToken(chargeToken)

.reference(reference)

.consumerID(consumerID)

.request3DSecureVeriﬁcation(false)

.paymentMethod(CPayMethodType.PAYPAL)

.build(CPayENVMode.UAT);

UPI: AliPay/WeChatPay/UnionPay Java Example:

CPayRequest request = CPayRequest.UPIOrderBuilder.INSTANCE

.accessToken(mAccessToken)

.reference(mReference)

.consumerID(consumerID)

.currency(mCurrency)

.amount(mAmount)

.callbackURL(callbackURL)

.ipnURL(IPNURL)

.mobileURL("https://exampe.com/mobile")

.cancelURL("https://exampe.com/cancel")

.failURL("https://exampe.com/fail")

.setAllowDuplicate(true)

.paymentMethod(CPayMethodType.ALI)//AliPay

.country(Locale.CANADA)

.setExpiry(SystemcurrentTimeMillis() + mTimeout)

.build(CPayENVMode.UAT);

UPI: AliPay/WeChatPay/UnionPay Kotlin Example:

fun buildDropInRequest(type: CitconPaymentMethodType): CPayRequest { return CPayRequest.UPIOrderBuilder

.accessToken(mAccessToken.value!!)

.reference(mReference)

.consumerID(mConsumerID)

.currency(mCurrency)

.amount(mAmount)

.callbackURL(mCallback)

.ipnURL("https://exampe.com/ipn")

.mobileURL("https://exampe.com/mobile")

.cancelURL("https://exampe.com/cancel")

.failURL("https://exampe.com/fail")

.setAllowDuplicate(true)

.paymentMethod(CPayMethodType.WECHAT)

.country(Locale.US)

.setExpiry(SystemcurrentTimeMillis() + mTimeout)

.build(CPayENVMode.UAT)

}

ONLINE: AliPay/WeChatPay/UnionPay Java Example:

CPayRequest request = CPayRequest.CPayOrderBuilder.INSTANCE

.token(mToken)

.reference(mReference)

.currency(mCurrency)

.amount(mAmount)

.paymentMethod(CPayMethodType.ALI)

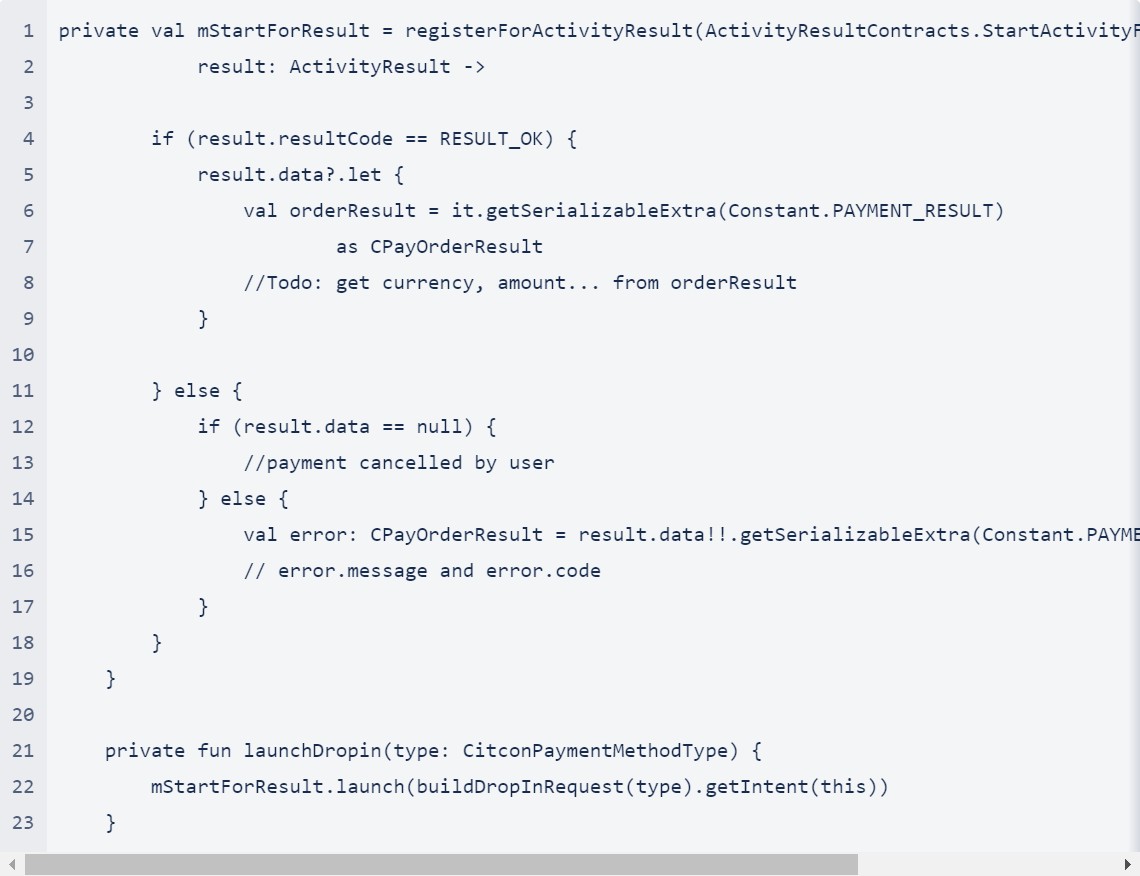
.setAllowDuplicate(true)

.build(CPayENVMode.UAT);

## Start Citcon Payment UI\*

Parameter: CPayRequest.getIntent() Java Example:

startActivityForResult(request.getIntent(this), DROP\_IN\_REQUEST);

\* onActivityResult has been deprecated recently, new implemented method is in this Kotlin example:

## Get Payment Result

Java Example:

public void onActivityResult(int requestCode, int resultCode, Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (resultCode == RESULT\_OK) {

CPayResult order = (CPayResult) data.getSerializableExtra(Constant.PAYMENT\_RESULT);

} else {

String message; if(data != null) {

CPayResult error = (CPayResult) data.getSerializableExtra(Constant.PAYMENT\_RESULT);

} else {

//payment cancelled by user";

}

}

}

\*\* onActivityResult has been deprecated recently, new implemented method is in this Kotlin example

private val mStartForResult = registerForActivityResult(ActivityResultContracts.StartActivityForResult()) { result: ActivityResult ->

if (result.resultCode == RESULT\_OK) { result.data?.let {

val orderResult = it.getSerializableExtra(Constant.PAYMENT\_RESULT) as CPayResult

//Todo: get currency, amount... from orderResult

}

} else {

if (result.data == null) {

//payment cancelled by user

} else {

val error: CPayResult = result.data!!.getSerializableExtra(Constant.PAYMENT\_RESULT) as CPayResult

// error.message and error.code

}

}

}

private fun launchDropin(type: CitconPaymentMethodType) { mStartForResult.launch(buildDropInRequest(type).getIntent(this))

}

## Using 3DS

In order to use 3DS2, you will need to create a CPayConsumer object with relevant customer and transaction data in order to minimize the need for issuing banks to present authentication challenges to customers

Java Example:

private CPayConsumer demo3DSsetup() {

CPayBillingAddr billingAddr = CPayRequest.BillingAdressBuilder.INSTANCE

.city("Chicago")

.state("IL")

.street("555 Smith St")

.postCode("12345")

.country("US")

.build();

return CPayRequest.ConsumerBuilder.INSTANCE

.ﬁrstName("Alex")

.lastName("Smith")

[.email("google@gmal.com](mailto:google@gmal.com)")

.phone("1112223344")

.billingAddress(billingAddr)

.build();

}

1. **No-Citcon Wechat SDK Broadcast**

Sometimes our integrators have already integrated WeChatpay SDK, and they want to keep their own WeChatpay module. In this case, there will be a conflict in the callback class registration of WeChat payment.

1. define a simple broadcast class

|  |
| --- |
| package com.citconpay.cupay;  import android.annotation.SuppressLint;  import android.content.BroadcastReceiver;  import android.content.Context;  import android.content.Intent;  import android.os.Bundle;  import android.util.Log;  import android.widget.Toast;  import com.tencent.mm.opensdk.constants.ConstantsAPI;  import com.tencent.mm.opensdk.modelpay.PayResp;  public class WXHandlerBroadcastReceiver extends BroadcastReceiver {  private static final String TAG = "WXHandlerBroadcastReceiver";  @SuppressLint("LongLogTag")  @Override  public void onReceive(Context context, Intent intent) {  // onReceive: The function cannot do time-consuming things, reference value: within 10s  //jump to activity  //Intent i = new Intent(context, MainActivity.class);  //i.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);  //context.startActivity(i);    //get content  String stage = intent.getStringExtra("stage");  Bundle object = intent.getBundleExtra("object");  Log.d(TAG, "---> onReceive: "+ stage +",object: "+ object);  if ("onCreate".equals(stage)) {  Bundle savedInstanceState = object;  if (savedInstanceState != null) {  }  } else if ("onNewIntent".equals(stage)) {  Intent newIntent = object.getParcelable("intent");  if (newIntent != null) {  }  } else if ("onReq".equals(stage)) {  if (object != null) {  }  } else if ("onResp".equals(stage)) {  if (object != null) {  int type = object.getInt("\_wxapi\_command\_type");  if (type == ConstantsAPI.COMMAND\_PAY\_BY\_WX) {  PayResp resp = new PayResp();  resp.fromBundle(object);  if (resp.prepayId != null) {  }  }  }  }  Toast.makeText(context, "Broadcast Receiver: "+stage, Toast.LENGTH\_SHORT).show();  }  } |

1. register broadcast

|  |
| --- |
| IntentFilter filter = new IntentFilter();  filter.setPriority(1000);  filter.addAction(Constant.CUPAT\_WXHANDLE\_BROADCAST\_ACTION);  registerReceiver(new WXHandlerBroadcastReceiver(), filter); |

or in **application** element of **AndroidManifest.xml**

|  |
| --- |
| <receiver android:name=".WXHandlerBroadcastReceiver" android:exported="true">  <intent-filter>  <action android:name="citcon.cupay.wxhandle" />  </intent-filter>  </receiver> |

# The Citcon UPI SDK

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

**Class references**

1. **CPayDropInRequest:**

The **CPayRequest** class set up most parameters to start up Citcon Payment UI.

*Class Methods:*

* + **accessToken** (Mandatory) accessToken(String accessToken)
    - Parameters:
      * accessToken - Which gets from the server is used to identify merchants.
  + **chargeToken** (Mandatory) chargeToken(String chargeToken)
    - Parameters:
      * accessToken - Which gets from the server is used to identify transactions.
  + **consumerID** (Mandatory) accessToken(String consumerID)
    - Parameters:
      * consumerID - This ID is used to identify users on vaulting payment methods.
  + **reference** (Mandatory) reference(String reference)
    - Parameters:
      * reference - This is the unique transaction reference id.
  + **paymentMethod** (Mandatory) paymentMethod(**CitconPaymentMethodType** type)
    - Parameters:
      * type - this type is used to bring up diﬀerent payment method UI.

### collectDeviceData (Optional)

collectDeviceData(boolean collectDeviceData)

* + - Parameters:
      * collectDeviceData - true if Drop-in should collect and return device data for fraud prevention.

### paypalRequest (Optional)

paypalRequest(PayPalRequest request)

* + - Parameters:
      * request - The PayPal Request PayPalRequest for the transaction. If no amount is set, PayPal will default to the billing agreement (Vault) ﬂow. If the amount is set, PayPal will follow the one time payment (Checkout) ﬂow.
  + disablePayPal **(Optional)**

disablePayPal()

* + - Disables PayPal in Drop-in.
  + disableVenmo **(Optional)**

disableVenmo()

* + - Disables Venmo in Drop-in.
  + disableCard **(Optional)**

disableCard()

* + - Disables Card in Drop-in.
  + request3DSecureVeriﬁcation **(Optional)**

request3DSecureVeriﬁcation(boolean requestThreeDSecure)

If 3D Secure has been enabled in the control panel and an amount is speciﬁed in amount(String) or a ThreeDSecureRequest is provided, Drop-In will request a 3D Secure veriﬁcation for any new cards added by the user.

* + - Parameters:
      * requestThreeDSecure - true to request a 3D Secure veriﬁcation as part of Drop- In, false to not request a 3D Secure veriﬁcation. Defaults to false.
  + maskCardNumber **(Optional)**

maskCardNumber(boolean maskCardNumber)

* + - Parameters:
      * maskCardNumber - true to mask the card number when the ﬁeld is not focused. See CardEditText for more details. Defaults to false.
  + maskSecurityCode **(Optional)**

maskSecurityCode(boolean maskSecurityCode)

* + - Parameters:
      * maskSecurityCode - true to mask the security code during input. Defaults to false.
  + getIntent **(Optional)**

public Intent getIntent(android.content.Context context)

* + - Get an Intent that can be used in FragmentActivity.startActivityForResult(Intent, int) to launch DropInActivity and the Drop-in UI.
    - Parameters:
      * context - Returns:

Intent containing all of the options set in CPayDropInRequest.

* + isPayPalEnabled **(Optional)**

public boolean isPayPalEnabled()

* + getPayPalRequest **(Optional)**

public com.braintreepayments.api.models.PayPalRequest getPayPalRequest()

* + isVenmoEnabled **(Optional)**

public boolean isVenmoEnabled()

* + isCardEnabled **(Optional)**

public boolean isCardEnabled()

* + getGooglePaymentRequest **(Optional)**

public com.braintreepayments.api.models.GooglePaymentRequest getGooglePaymentRequest()

* + getCardholderNameStatus **(Optional)**

public int getCardholderNameStatus()

* + isSaveCardCheckBoxShown **(Optional)**

public boolean isSaveCardCheckBoxShown()

1. **CPayMethodType: enum**

Type deﬁnition of payment method.

|  |  |
| --- | --- |
| **Name** | **Description** |
| PAYPAL | Paypal |
| CREDIT | Credit card. (Eg. American Express, Discover, Visa, Mastercard and so on) |
| PAY\_WITH\_VENMO | Venmo |
| ALI | AliPay |
| WECHAT | WeChat Pay |
| UNIONPAY | Union Pay |
| ALI\_HK | AliPay HongKong |
| KAKAO |  |
| SAMSUNG |  |
| LG |  |
| LPAY |  |
| TOSS |  |
| BANKTRANSFER |  |
| ATOME |  |
| PAYNOW |  |
| NETSPAY |  |

|  |  |
| --- | --- |
| GRABPAY |  |
| SHOPEEPAY |  |
| DANA |  |
| GCASH |  |
| RABBIT LINE PAY |  |
| TNG |  |
| BPI |  |
| TRUE MONEY |  |
| ALIPAY+ |  |
|  |  |

1. **CPayBillingAddr:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Required-/ Optiona-l/Conditi- onal** | **Description** | **Example** |
| Street | String? | O | Street name |  |
| Street2 | String? | O | Street name |  |
| City | String? | O |  |  |
| State | String? | O |  |  |
| Zip | String? | O | Post code |  |
| Country | String? | O |  |  |

1. **CPayConsumer:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Required-/ Optiona-l/Conditi- onal** | **Description** | **Example** |
| reference | String? | O | Transaction id |  |
| ﬁrstName | String? | O |  |  |
| lastName | String? | O |  |  |
| phone | String? | O |  |  |
| email | String? | O |  |  |
| billingAddress | CPayBillingAddr | O |  |  |

1. **CPayResult: (Kotlin data class)**

Holds the result and message for a transaction returned by Citcon Pay service.

*Property summary*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Required-/ Optiona-l/ Conditi-onal** | **Description** | **Example** |
| resultCode | Int | R | resultCode in onActivityResult | RESULT\_OK |
| message | String | C | Detailed description of the status of a transaction. The message maybe null when result is "success". | bad request |
| result | String | R | The result of the transaction | success |
| code\*\* | String | R | The status code for the result | 0 |
| transactionId | String | C | An unique identiﬁer of the transaction. This ID is generated by Citcon Pay. It will be null when result is fail. | 123xyz |
| amount | String | C | The total amount of an transaction. It will be null when result is fail. | 10 |
| currency | String | C | Currency type deﬁned by three- letter code. | USD |
| reference | String | C | A reference identifying the transaction. This ID is generated by merchant. | 456wyz |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | String | C | The timestamp for the transaction. | 12321321312 |
| paymentMetho d | CitconPaymen tMehodType | R | Payment method type of this order | PAYPAL |
| country | String | C | Two letter country code | US |

**\*\*Status code**

|  |  |
| --- | --- |
| **Code** | **Description** |
| 0 | Success |
| -1 | Fail. For details, please refer to the message. |
| -2 | Cancel |
| -3 | Load conﬁguration fail |
| 4000 | Bad request |
| 4010 | Unauthorized |
| 4100 | Duplicate request |
| 4101 | Unknown error |
| 4102 | Invalid header |
| 4103 | Not found |
| 5100 | Internal server error |
| others | Other payment error |