Citcon Pay iOS SDK Documentation

Version 2.6.1

Version Information

Ver. No.	Date	Remarks
2.6.1		Add new payment methods to support toss.
2.5.3		Add new payment methods to support Korean and Japanese wallets.

Table of Contents

Introduction

The Citcon's iOS app payment solution provides a convenient, safe, and reliable payment services to third-party applications. By using the SDK, merchant developers can focus on business logics without having to understand the pluming of payment transactions. The payment experience will be totally transparent and seamless to end consumers.

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.

Target audience

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

Terminologies

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

2. Return
   Citcon returns processed result data in the form of character string to iOS 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.

Supported currency

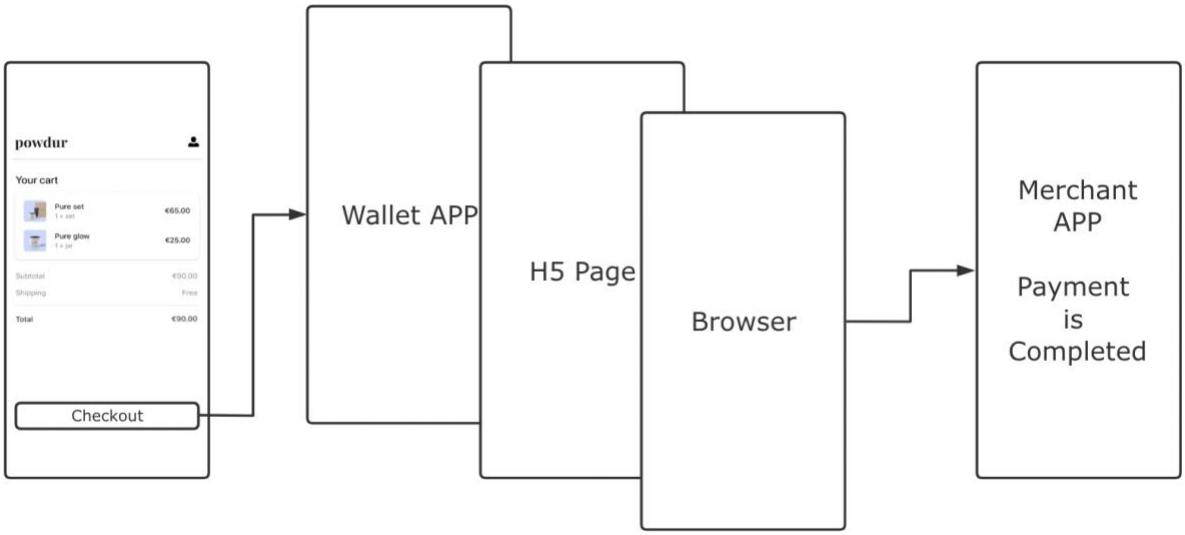
Citcon mainly supports the following currencies:

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.

Payment flow and user experience

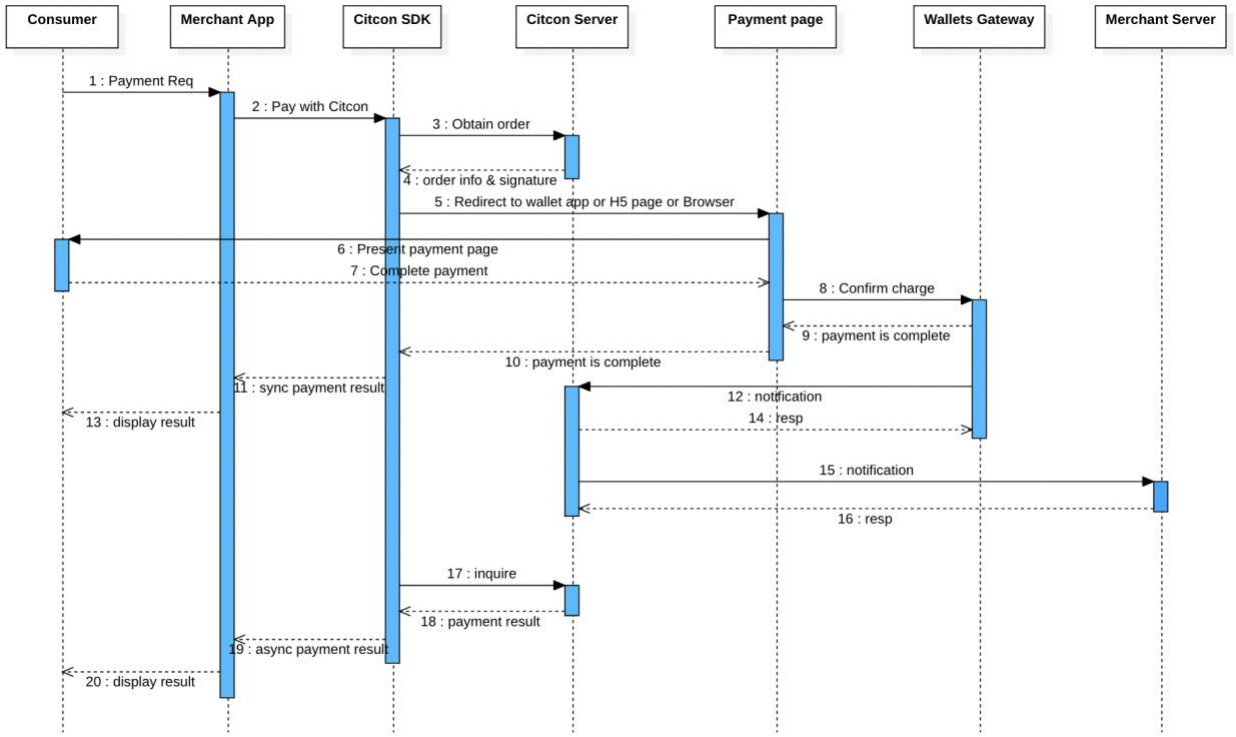
The following figure illustrates the workflow of In-app payment.

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.



Interaction process

Function process



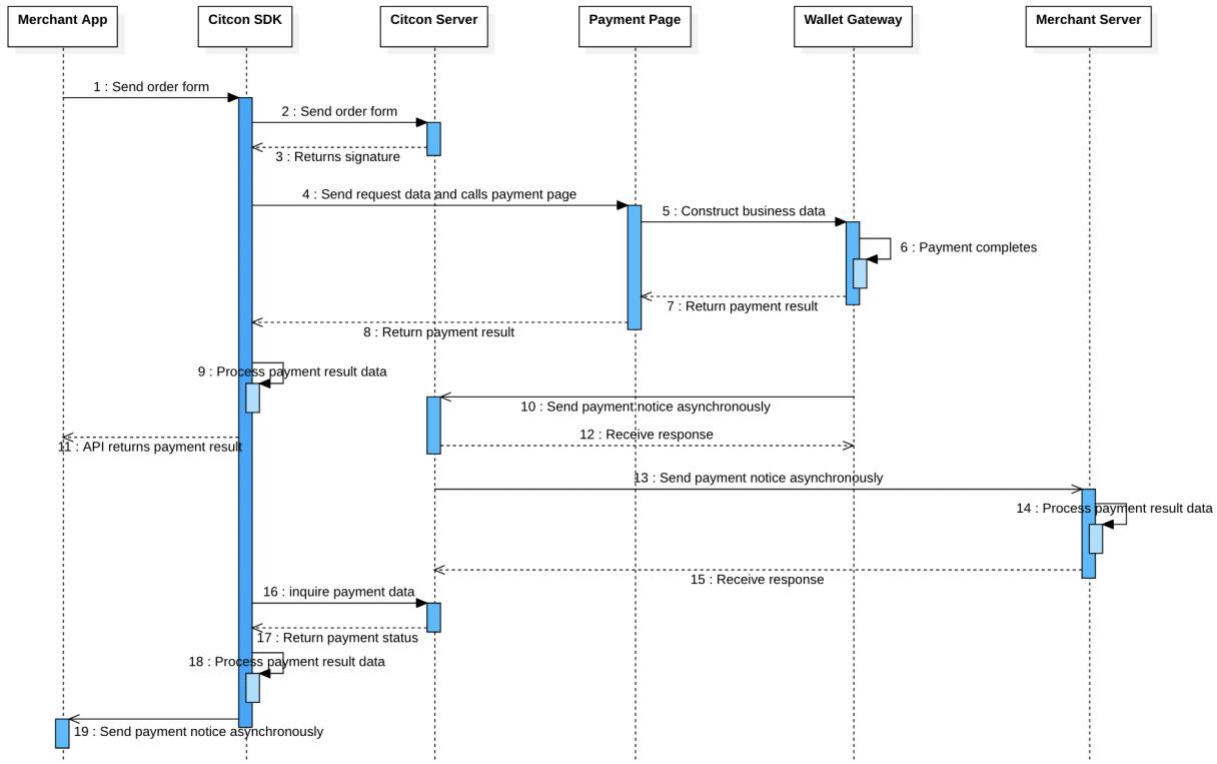
Key steps in the payment process are explained below (take the payment process on iOS platform as an example):

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".

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.

SDK overview

The client-side SDK can facilitate your integration with Citcon. This section details the main components of the Citcon Pay Framework for iOS development.

API name: CPayManager

Description: Citcon SDK provides payment function.

Citcon API provides merchants with order payment function. Methods provided by API are detailed in the table below.

Method name	Method description
+ (void)setupTokenKey	Set the token generated by Citcon.
+ (void)setupMode	Set runtime mode for Citcon SDK.
+ (void)requestOrder	Pay and get result via callback.
+ (void)inquireResult	Query via transaction id and get result via callback.
+ (void)inquireResultByRef	Query via reference id and get result via callback.
+ (NSString *)getVersion	Returns the version of Citcon SDK.
+ (BOOL)processOpenUrl	Client-side processing method processes the Payment side returned *url*.
+ (BOOL)processUserActivity	Client-side processing method processes the Payment side returned *url*.
+ (BOOL)isWechatInstalled	Check if the device has the WeChat app installed.
+ (BOOL)isAlipayInstalled	Check if the device has the Alipay app installed.
+ (BOOL)isAlipayHKInstalled	Check if the device has the AlipayHK app installed.
+ (BOOL)isKakaoPayInstalled	Check if the device has the Kakao app installed.

+ (BOOL)isUnionPayInstalled	Check if the device has the UnionPay app installed

- Express order payment iOS
- Express order inquiry iOS
- Processing Client Side Returned URL
- Callback API
- Notification
- Request
- Response

Express order payment iOS

Method name: Pay method

Method prototype: + (void)requestOrder:(CPayOrder *_Nonnull)order completion:(CPayOrderResultBlock _Nonnull)completion;

Method function: Pay method provides merchants with express order payment function.

Parameter name	Parameter description
CPayOrder *order	App payment request parameters contain merchant's order information.
(CPayOrderResultBlock)callback	Express pay SDK callback function returns with payment result. Please refer to "synchronous response parameter" for more details on the relevant payment results.

Payment parameters examples are shown below, see "CPayOrder" for parameters description.

```
CPayOrder * order = [CPayOrder new];
order.referenceId = @"CPayDemoTest_1647011058000000";
order.subject = @"subject";
order.body = @"body";
order.amount = @"328";
```

```
order.currency = @"USD";
order.vendor = @"wechatpay";
order.ipnUrl = @"https://ipn.merchant.com";
order.allowDuplicate = @"yes";
// If you use alipay_hk/kakaopay/dana/kcp/sbps, set as URLScheme
order.callbackUrl = @"cpaydemo.citconpay.com://";
order.callbackFail = @"cpaydemo.citconpay.com://";
order.cancelUrl = @"cpaydemo.citconpay.com://";
// required for upop payment
order.controller = self;
// (required) your app scheme for alipay payment, set in the Info.plist
order.scheme = @"cpaydemo.citconpay.com";
// (required) your app universal link if you are using wechat pay
order.universalLink = @"https://dev.citconpay.com/cpaydemo";
// If you use kcp/sbps, set following values
order.country = @"US";
order.firstName = @"John";
order.lastName = @"Doe";
order.phone = @"6145675309";
order.email = @"test.sam@test.com";
order.consumerReference = @"consumer-reference-000";
order.taxableAmount = 0;
order.taxExemptAmount = 0;
order.totalTaxAmount = 0;
order.isAccelerateCNPay = NO;
order.extra = @"{\"key\":\"value\"}";
```

Express order inquiry iOS

Method name: inquire method by *transaction id*

Method prototype:

- + (void)inquireResult:(NSString *_Nonnull)transactionId order:(CPayOrder*_Nonnull)order method:(NSString *_Nonnull)method completion:(CPayCheckResultBlock _Nonnull)completion;

Method function: The query method for the client to query Citcon orders.

Parameter name	Parameter description
NSString *transactionId	The serial number assigned by Citcon to identify a trade in the Citcon system.
CPayOrder *order	The original order used for the Citcon SDK payment request.
NSString * method	Indicates how to query. "real" means querying from the vendor's gateway. Default value is “real”. e.g., real
(CPayCheckResultBlock)callback	Express pay SDK callback function returns with inquire result. Please refer to “CPayCheckResult” for more details on the relevant inquire results.

Method name: inquire method by *reference id*

Method prototype:

- + (void)inquireResultByRef:(NSString *_Nonnull)referenceId currency:(NSString *_Nonnull)currency method:(NSString *_Nonnull)method vendor:(NSString *_Nonnull)vendor isAccelerated:(BOOL)acceleration completion:(CPayCheckResultBlock _Nonnull)completion;

Method function: The query method for the client to query Citcon orders.

Parameter name	Parameter description
NSString *referenceId	The serial number assigned by merchant.
NSString *currency	Settlement currency type defined by three-letter code. **NOTE:** It should be from the same order as the reference id.

NSString *method	Indicates how to query. "real" means querying from the vendor's gateway. Default value is “real”. e.g., real
NSString *vendor	The name of vendor that will process the payment. **NOTE:** It should be from the same order as the reference id. e.g., alipay
NSString *acceleration	Indicates the acceleration of CNY payments. **NOTE:** It should be from the same order as the reference id. e.g., YES
(CPayCheckResultBlock)callback	Express pay SDK callback function returns with inquire result. Please refer to “CPayCheckResult” for more details on the relevant inquire results.

Processing client side returned URL

Method name: processing client method

Method prototype:

- + (BOOL)processOpenUrl:(UIApplication *_Nonnull)application url:(NSURL *_Nonnull)url standbyCallback:(CPayOrderResultBlock _Nonnull)callback;

Method function: Client-side processing method processes the wallet app/H5 page/browser returned *url*.

Note:

These two methods must be implemented, otherwise when calls Payment page, the payment result cannot be synchronously returned.

Parameter name	Parameter description
NSURL *url	*url* returned by wallet app/H5 page/browser.
(CPayOrderResultBlock)callback	Called after Citcon SDK processes the payment result.

Note:

Please call this method in - (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation in AppDelegate. In iOS9.0 and above versions (including iOS9.0), please call this method in - (BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options. See demo for details.

Method name: processing client method

Method prototype:

- + (BOOL)processUserActivity:(nonnull NSUserActivity *)userActivity;

Method function: Client-side processing method processes the wallet app/H5 page/browser returned *userActivity*.

Parameter name	Parameter description
NSUserActivity *userActivity	*userActivity* returned by WeChat or other wallet apps opened via *universalLink.*

Note:

Please call this method in - (BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> * _Nullable))restorationHandler. See demo for details.

Callback API

Prototype:

- typedef void (^CPayOrderResultBlock) (CPayOrderResult * result);

Description: Definition of *callbackBlock* used by the payment API.

After the payment is completed, it will return the payment result synchronously via *callbackBlock*.

Result returned needs to use value of the *resultStatus* and *result* fields of *CPayOrderResult* to determine the payment result. When verifications of *resultStatus* = *0*, payment is confirmed to be successful, otherwise, it may be regarded as failure. For circumstances of low security level, payment result can be determined by checking *resultStatus* only. See CPayOrderResult for details.

Parameter name	Parameter description
CPayOrderResult * result	Payment result returned via callback.

An example of a successful order payment is shown below:

```
{
   resultStatus = @"0";
   status = @"success";
   message = @"success";
}
```

Prototype:

- typedef void (^CPayCheckResultBlock) (CPayCheckResult * result);

Description: Definition of *callbackBlock* used by the inquire API.

After the inquiry is completed, it will return the transaction information synchronously via *callbackBlock*. See CPayOrderResult for details.

Parameter name	Parameter description
CPayCheckResult * result	Query result returned via callback and asynchronous notification.

An example of a successful order payment is shown below:

```
{
   status = @”success”;
   type = @”charge”;
   transactionId = @”txnId”;
   referenceId = @”refId”;
   currency = @”USD”;
   amount = @”123”;
}
```

Notification

Prototype:

- static NSString * const kOrderPaymentFinishedNotification = @"CPAY_INQUIRE_ORDER";
- [[NSNotificationCenter defaultCenter] addObserver: self selector:@selector(cpay_inquired_order:) name:kOrderPaymentFinishedNotification object:nil];

After the payment is completed, it will return the payment result asynchronously via iOS notification.

Parameter name	Parameter description
NSNotification *notification	*notification.object* as query result returned via callback.

Result returned needs to use value of the *status* field of *CPayCheckResult* to determine the payment result. When verification of *status = 0*, payment is confirmed to be successful. The Citcon SDK will send a notification after checking the status of the order through the Citcon server and confirming the payment result. See CPayCheckResult for details.

An example of a successful order payment is shown below:

```
{
    status = @"0";
    type = @"charge";
    transactionId = @"txnId";
    referenceId = @"refId";
    currency = @"USD";
    amount = @"123";
}
```

Note:

The payment result will be returned to the merchant app through synchronous callback or asynchronous notification only when the user does not interrupt the payment process.

Otherwise, the Citcon SDK will not return any results even if the user completes the payment. (e.g., user kills wallet app or switches apps manually).

Request

CPayOrder

The order contains parameters used to construct the business payment data.

Parameter	Description
referenceId String Required	A reference id generated by merchant and assigns to the transaction. e.g., a3852167-9d10-4e8f-adf1-aef975667a87
amount String Required	Total charge amount of the transaction in cents. e.g., 258
currency String Required	Settlement currency type defined by three-letter code. See currency for details.

	e.g., USD
vendor String Required	The name of vendor that will process the payment. e.g., alipay
subject String	A description of the transaction. e.g., Gift for Mom
body String	A more detailed description of the transaction. e.g., A Blu-ray player and a few great movies
ipnUrl String Required	The URL for receiving asynchronous notifications after the payments is completed. e.g., https://notify.merchant.com
scheme String Required	URL protocol registered by merchant program is for the use of merchant callback program after payment has been completed. For more information about Scheme, see Defining a URL Scheme in iOS. e.g., cpaydemo.citconpay.com
universalLink String Required	Universal Links allow you to connect to deep links in you iOS app and are supported in iOS 9.2 or later. For more information about Universal Links, see Support Universal Links. **NOTE:** If you use WeChat Pay, you must register Universal Links on the WeChat Open Platform. e.g., https://www.merchant.com/apps/
controller UIViewController * Required	The view controller used by the merchant's app for invoking wallet payment control. e.g., self

callbackUrl String Required	After the payment is completed, the wallet app/H5 page/browser is redirected to this URL. **NOTE:** It is invalid for some vendors (such as WeChat Pay, Alipay, UnionPay, etc.), but must be set as the *URLScheme* of the merchant's app for other vendors (such as alipay_hk, kakaopay, etc.) to correctly return to the merchant's app. e.g., cpaydemo.citconpay.com://
callbackFail String	After the payment is failed, the wallet app/H5 page/browser is redirected to this URL. **NOTE:** The same as the *callbackUrl*, it is only valid for some vendors and must be set to the *URLScheme* of the merchant's app to correctly return the merchant's app. If you don't set this value, it will default to the same as *callbackUrl*. e.g., cpaydemo.citconpay.com://
cancelUrl String	After the payment is canceled, the wallet app/H5 page/browser is redirected to this URL. **NOTE:** The same as the *callbackUrl*, it is only valid for some vendors and must be set to the *URLScheme* of the merchant's app to correctly return the merchant's app. If you don't set this value, it will default to the same as *callbackUrl*. e.g., cpaydemo.citconpay.com://
allowDuplicate BOOL	Flag to control duplicate orders. e.g., YES
transCurrency String	Transaction currency type defined by three-letter code. See currency for details. e.g., USD

extra String	Merchant custom information. e.g., “{\”key\”:\”value\”}”
isAccelerateCNPay BOOL	Indicates the acceleration of CNY payments. e.g., YES
country String	The country of the billing address. Defined by tow-letter code. Use upper case. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., US
firstName String	The first name of billing address. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., John
lastName String	The last name of billing address. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., Doe
phone String	The phone number of billing address. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., 6145675309

email String	The email of billing address. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., test.sam@test.com
consumerReference String	The serial number assigned by merchant to identify a user in the merchant system. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) e.g., GUID-270f4530-a169-4253-be43-8dccaee407c8
taxableAmount String	Taxable amount in cents. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) Default is 0. e.g., 0
taxExemptAmount String	Tax exempt amount in cents. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) Default is 0. e.g., 0
totalTaxAmount String	Total tax amount in cents. **NOTE:** Required if you pay via vendors that require this parameter. (e.g., KCP & SBPS) Default is 0. e.g., 0

transactionId String	The serial number assigned by Citcon to identify a trade in the Citcon system. **NOTE:** Assigned only after Citcon has successfully created an order. Empty if order creation failed. e.g., P0023477472-35fe8730f1463bcaaa34
installmentId String	This field represents the maximum value of the installment plan.

Notes:

1. Different Citcon runtime environments may use different *tokens* for the same merchant. Token is usually associated with supported currencies and vendors.
   If your order creation fails, check that your runtime environment matches the specified *token*.
2. Some parameters are required when paying with a specific supplier. See the table above for more details.
   If the order creation fails, check that all required parameters for the specified *vendor* are correct.

Response

Synchronous response

After the synchronous notification is processed by Citcon SDK, the payment result will be synchronously feed back to the merchant app.

The data returned by the synchronous notification must be verified by the merchant on the server side. After the verification is passed, the payment can be considered successful. In some cases, the synchronous result cannot be received correctly. Then, the payment result can be completely dependent on the asynchronous notification received in merchant server.

Note:

Both the synchronous notification and asynchronous notification can be used as the payment completion certificate. The asynchronous notification will be surely sent to the merchant client and server from Citcon.

However, to simplify the integration process, merchant can use either the synchronous result or asynchronous result as a notification of the end of payment.

CPayOrderResult

Holds the synchronization result of the transaction returned by Citcon. Obtain via callback.

Parameter	Description
message String	Parameter reserved. In most cases, no value for this parameter. e.g., Create payment failed.
resultStatus String Required	Status code, which is returned from Citcon SDK. For more details, see Codes Returned to Client End. e.g., 0
result String Required	Status description, which is returned from Citcon SDK. This value is used to indicate the status of the payment. For more details, see Status Returned to Client End. e.g., success
order String Required	The original order used for the Citcon SDK payment request. **NOTE:** The transaction id included in the order is assigned by the Citcon SDK once the payment is created successfully. e.g., order

Asynchronous response

Asynchronous responses include client-side and server-side.

After processing the request data, Citcon will notify merchant's website of the processed result data in a server-actively-notifying manner.

After processing the data returned by Citcon's SDK automatic query, if the merchant APP has registered an observer to receive the notification, Citcon will notify the merchant APP of the processed result data.

These processed result data are the asynchronous response parameters of the client.

CPayCheckResult

Holds the transaction results returned by the query. It will return via callback and notification.

Parameter	Description
status String Required	The value of status is used to indicate the result of the payment. **NOTE:** "success" means the payment is successful, otherwise it fails. e.g., success
transactionId String Required	The serial number assigned by Citcon to identify a trade in the Citcon system. **NOTE:** Assigned only after Citcon has successfully created an order. Empty if order creation failed. e.g., P0023477472-35fe8730f1463bcaaa34
referenceId String Required	A reference id generated by merchant and assigns to the transaction. e.g., a3852167-9d10-4e8f-adf1-aef975667a87
type String Required	The type of the transaction: "charge" or "refund". e.g., charge
amount String Required	Total charge amount of the transaction in cents. e.g., 258
currency String Required	Settlement currency type defined by three-letter code. See currency for details. e.g., USD

time String	The time the order was created. e.g., 2017/06/22 1:23:12 PM
refund_status String	Indicates the refund status of the transaction. e.g., success
refunded_amount String	The actual amount in cents refunded to the customer, which is may or may not be the same as the charge amount. e.g., 121
note String	Additional information for the transaction.

Notification

Name	Description
kOrderPaymentFinishedNotification	The value of this notification is “CPAY_INQUIRE_ORDER”. Will be sent after payment is completed.

Notification trigger condition

Trigger condition name	Description
TRADE_FINISHED	Trade is completed successfully.

Codes returned to Client end

Return code	Description
0	Successful order payment.

-1	Bad signature, unregistered appid, wrong appid, mismatched app id or some other error.
-2	The user cancels the payment, switch back to the merchant's app, the user cannot pay, etc.
8000	Under processing, unknown payment result (payment might have been made successfully), please inquiry order payment status in sellers' orders list. (This code returns from server side.)
4000	Failed order payment.
5000	Duplicate request.
6001	Canceled by user during the process.
6002	Error in network connection.
6004	Unknown payment result (payment might have been made successfully), please inquiry order payment status in sellers' orders list. (Usually caused by network issue, causing Alipay client cannot receive response from server side)
others	Other payment errors.

Status Returned to Client end

Name	Description
success	Indicates that the payment is successful.
fail	Indicates that the payment is failed.
cancel	Indicates that the payment has been cancelled. In most cases it is returned by the synchronous result. This is behavior that occurs on the client side and may not be logged, using *NULL* or *fail* instead.
NULL	In most cases, *NULL* means the payment failed. Because in some cases no notification was received from the vendor to update the

	status of the transaction. (e.g., consumer kills wallet app, cancels, etc.)

Client-side integration

iOS SDK setup

Getting started with the iOS SDK requires 6 steps:

1. Install the SDK in your app
2. Configure your app
3. Initialize the SDK in your app
4. Set your return URL
5. Set up the payment request
6. Receive the payment result

Refer to demo for more details.

Install the SDK in your app

Citcon Pay iOS Framework requires a minimum deployment target of iOS 9+ and Xcode 12+, and can be installed with CocoaPods, or by manually integrating the framework.

CocoaPods

1. If you haven't already done so, install a recent version of CocoaPods.
2. If you don't have an existing Podfile, run the following command to create one:
   ```
   $ pod init
   ```
3. Add this line to your Podfile:
   ```
   pod 'CPay'
   ```
4. Run the following command:
   ```
   $ pod install
   ```
5. From now on, use the `.xcworkspace` file to open your project in Xcode, instead of the `.xcodeproj` file.

6. Navigate to “General” pane of your target and find the “Frameworks, Libraries, and Embedded Content” dropdown. Click the "+" and select CPay.xcframework in Pods from the dropdown options. Switch CPay.xcframework from “Do Not Embed” to “Embed and Sign”.

Manual

1. Visit the Citcon iOS repository on Github and navigate to the latest release.
2. Download the mobile_sdk_ios-${version}.zip file attached to the Github release.
3. Unzip the file, the drag and drop the XCFramework into your Xcode project.
4. If the framework’s symbols are unable to be loaded, navigate to “General” pane of your target and find the “Frameworks, Libraries, and Embedded Content” dropdown. Switch CPay.xcframework from “Do Not Embed” to “Embed and Sign”.

Configure your app

To prepare your app to work with Citcon iOS SDK, make a few changes to your info.plist file in Xcode.

LSApplicationQueriesSchemes

If it doesn’t exist, click “+” to add new property LSApplicationQueriesSchemes.

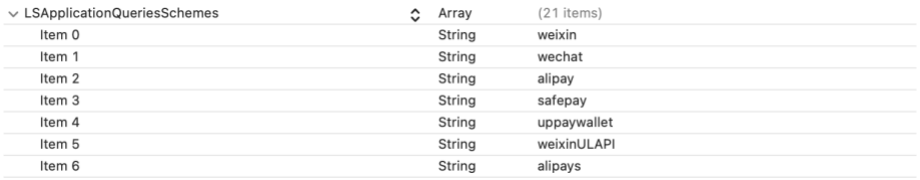
Under LSApplicationQueriesSchemes, add the following wallet-related items:

Wallet-Related	Items
WeChat Pay	weixin wechat weixinULAPI
Alipay	alipay safepay alipays
Union Pay	uppaywallet uppaysdk uppayx1 uppayx2 uppayx3

Alipay Hong Kong	alipayhk
Kakao Pay	kakaokompassauth kakaolink
GCash	gcash
Naver	naversearchapp
Line Pay	line linepay
Rakuten Pay	rakutenpay
Payco	payco
Paypay	paypay

NOTE:

You don't need to add all items to LSApplicationQueriesSchemes, just the items related to the wallet you want to use.



Register URL Type

Click on your project in the Project Navigation and navigate to “Add Target” -> “Info” -> “URL Type”.

Click “+” to add some new URL types.

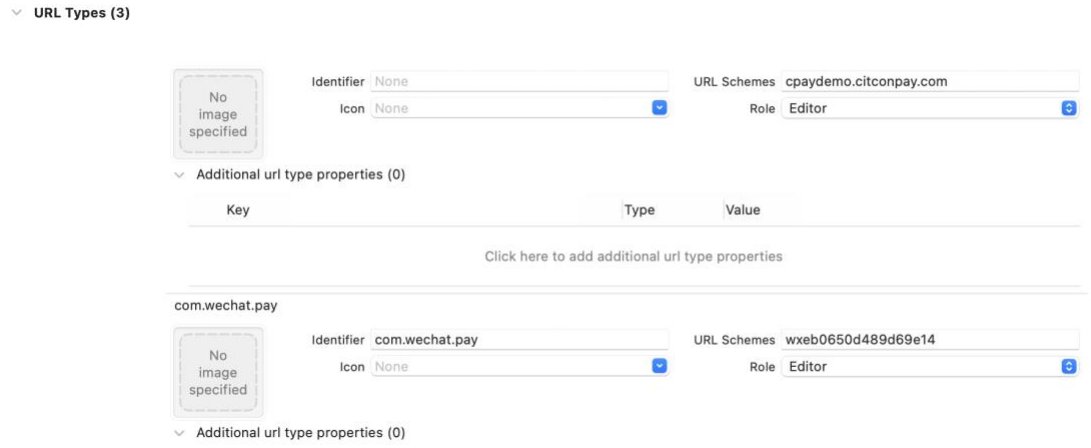
Under URL Schemes, enter your app switch return URL scheme as following

Wallet-Related	URL Schemes
WeChat Pay	The content of URL Schemes for WeChat Pay is assigned by the WeChat Open Platform. You must register an app on the

	Open Platform to obtain a wxappid from WeChat. And set it as URL Schemes for WeChat Pay. e.g., wxeb0650d489d69e14
Alipay	The content of Alipay URL Schemes is assigned by the merchant. It must be the same as the "*scheme*" field of CPayOrder. e.g., cpaydemo.citconpay.com

NOTE:

You don't need to add all items to URL Schemes, just the items related to the wallet you want to use.



Set up Universal Link

Universal Links allow you to connect to deep links in your iOS app and are supported in iOS 9.2 or later. When a Universal Link is accessed, iOS redirects the link directly to the deep link in your app. If your app is not installed, it opens a URL for your website in a browser instead. For more information about Universal Link, see Support Universal Links.

For Citcon, Universal Link is primarily used for WeChat Pay as a way to redirect to the wallet to complete the payment, return to the merchant app and include the payment result.

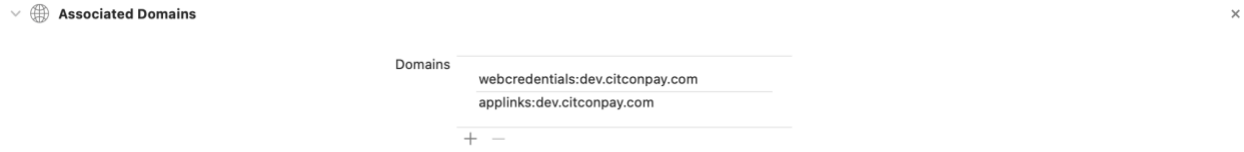
If you use WeChat Pay, you must set up your Universal Link on the WeChat Open Platform.

1. Turn on the “Associated Domains” option of your AppID in Apple Developer Center.

2. Edit and reinstall the profile which associated with the AppID that you turn on “Associated Domains”.

```
{
  "applinks": {
    "apps": [],
    "details": [
    {
      "appID": "9JA89QQLNQ.com.apple.wwdc",
      "paths": [ "/wwdc/news/", "/videos/wwdc/2015/*"]
    },
    {
      "appID": "ABCD1234.com.apple.wwdc",
      "paths": [ "*" ]
    }]
  }
}
```

3. Add “applinks:yourdomain” to your Xcode project. Set up the URL to the location where you uploaded the file “apple-app-site-association” with *https*.



4. Enter the URL set by your applinks + path (“apple-app-site-association”) to ensure you can navigate to your app from Safari.

Initialize the SDK in your app

Token

Citcon is responsible for generating an access token, which contains the authorization and configuration details that your client needs to initialize the client SDK.

Follow the code below to set the token

```
[CPayManager setupTokenKey:@"${your token}"];
```

Runtime mode

Different runtime environments use different access tokens assigned by Citcon. Citcon provides four different runtime environments (`DEV`, `UAT`, `PROD`, `QA`).

Usually, merchants only need UAT and PROD. UAT is often used by merchants for integration testing.

NOTE:

Make sure you are using the correct access token associated with the runtime environment.

Follow the code below to set the runtime mode

```
[CPayManager setupMode:CPayModeUAT];
```

Set your return URL

Please call this method in `- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication annotation:(id)annotation` in AppDelegate.

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url
options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {
    [CPayManager processOpenUrl:app url:url standbyCallback:^(CPayOrderResult *result) {
        // TODO:
    }];
    return YES;
}
```

In iOS9.0 and above versions (including iOS9.0), please call this method in `-(BOOL)application:(UIApplication *)app openURL:(NSURL *)url options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options`.

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url sourceApplication:(NSString *)sourceApplication
annotation:(id)annotation {
    [CPayManager processOpenUrl:app url:url standbyCallback:^(CPayOrderResult *result) {
        // TODO:
    }];
    return YES;
}
```

Please call this method in `- (BOOL)application:(UIApplication *)application continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> * _Nullable))restorationHandler`.

```
- (BOOL)application:(UIApplication *)app continueUserActivity:(NSUserActivity *)userActivity restorationHandler:(void
(^)(NSArray<id<UIUserActivityRestoring>> * _Nullable))restorationHandler {
    [CPayManager processUserActivity:userActivity];
    return YES;
}
```

Set up the payment request

Follow the code below to create an order for payment

```
    CPayOrder * order = [CPayOrder new];
    order.referenceId = @"CPayDemoTest_1647011058000000";
    order.subject = @"subject";
    order.body = @"body";
    order.amount = @"328";
    order.currency = @"USD";
    order.vendor = @"wechatpay";
    order.ipnUrl = @"https://ipn.merchant.com";
    order.allowDuplicate = @"yes";
    // If you use alipay_hk/kakaopay/dana/kcp/sbps, set as URLScheme
    order.callbackUrl = @"cpaydemo.citconpay.com://";
    order.callbackFail = @"cpaydemo.citconpay.com://";
    order.cancelUrl = @"cpaydemo.citconpay.com://";
```

```
// required for upop payment
order.controller = self;
// (required) your app scheme for alipay payment, set in the Info.plist
order.scheme = @"cpaydemo.citconpay.com";
// (required) your app universal link if you are using wechat pay
order.universalLink = @"https://dev.citconpay.com/cpaydemo";
// If you use kcp/sbps, set following values
order.country = @"US";
order.firstName = @"John";
order.lastName = @"Doe";
order.phone = @"6145675309";
order.email = @"test.sam@test.com";
order.consumerReference = @"consumer-reference-000";
order.taxableAmount = 0;
order.taxExemptAmount = 0;
order.totalTaxAmount = 0;
order.isAccelerateCNPay = NO;
order.extra = @"{\"key\":\"value\"}";
```

Use the order created in the previous step to pay through Citcon.

```
[CPayManager requestOrder:order completion:^(CPayOrderResult * payResult) {
    // TODO:
    }];
```

Receive the payment result

Follow the code below to receive synchronous payment result

```
[CPayManager requestOrder:order completion:^(CPayOrderResult * payResult) {
    // TODO: payResult is asynchronous payment result
    }];
```

After the payment is completed, the Citcon SDK will send a notification named `kOrderPaymentFinishedNotification` with the payment result. Integrators can register an observer to receive this notification.

Follow the code below to receive asynchronous payment result

```
// Register an observer to receive the notification of automatically query
[[NSNotificationCenter defaultCenter] addObserver:self
                                         selector:@selector(cpay_inquired_order:)
                                             name:kOrderPaymentFinishedNotification
                                           object:nil];
```

SEL for processing payment results

```
- (void)cpay_inquired_order:(NSNotification *)notification {
    CPayCheckResult *checkResult = (CPayCheckResult *) notification.object;
    // TODO: checkResul is asynchronous payment result
}
```