



# Citcon UPI SDK iOS Documentation

Version 2.0.7

## Version Information

Ver. No.	Date	Remarks
2.0.7	2022-09-19	Optimize payment flow and fix bug.
2.0.4	2022-08-11	Add new payment methods to support toss and fomo.
2.0.2	2022-05-06	Optimize performance.
2.0.0	2022-04-18	Add new payment methods to support WechatPay/Alipay/UnionPay.
1.0.0	2021-9-30	Add new payment methods to support card, PayPal and Venmo.

## Table of Contents

INTRODUCTION	5
Target audience	5
Terminologies	5
Supported currency	6
PAYMENT FLOW AND USER EXPERIENCE	7
INTERACTION PROCESS	8
Function process	8
Data interaction	9
SDK OVERVIEW	11
Express order payment iOS	12
Express order inquiry iOS	12
Processing client side returned URL	13
Callback API	15
Notification	18
Request	19
Response	32
Synchronous response	32
Asynchronous response	36
Codes returned to Client end	40
Status Returned to Client end	40

CLIENT-SIDE INTEGRATION	41
iOS SDK setup	41
Install the SDK in your app	42
Configure your app	42
Initialize the SDK in your app	44
Set your return URL	45
Receive the payment result	46

# 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 plumbing 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

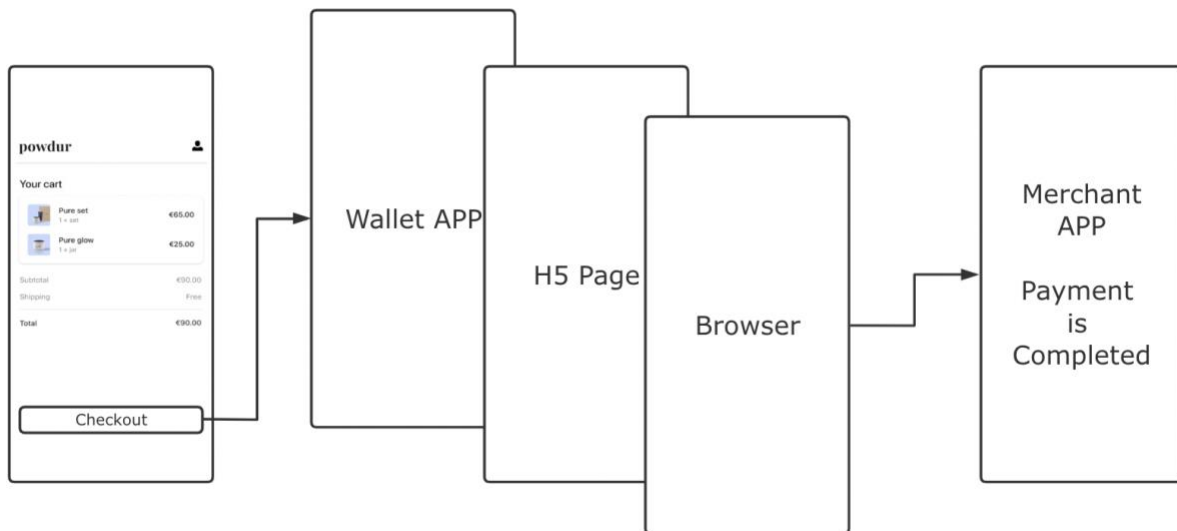
Citcon mainly supports the following currencies:

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

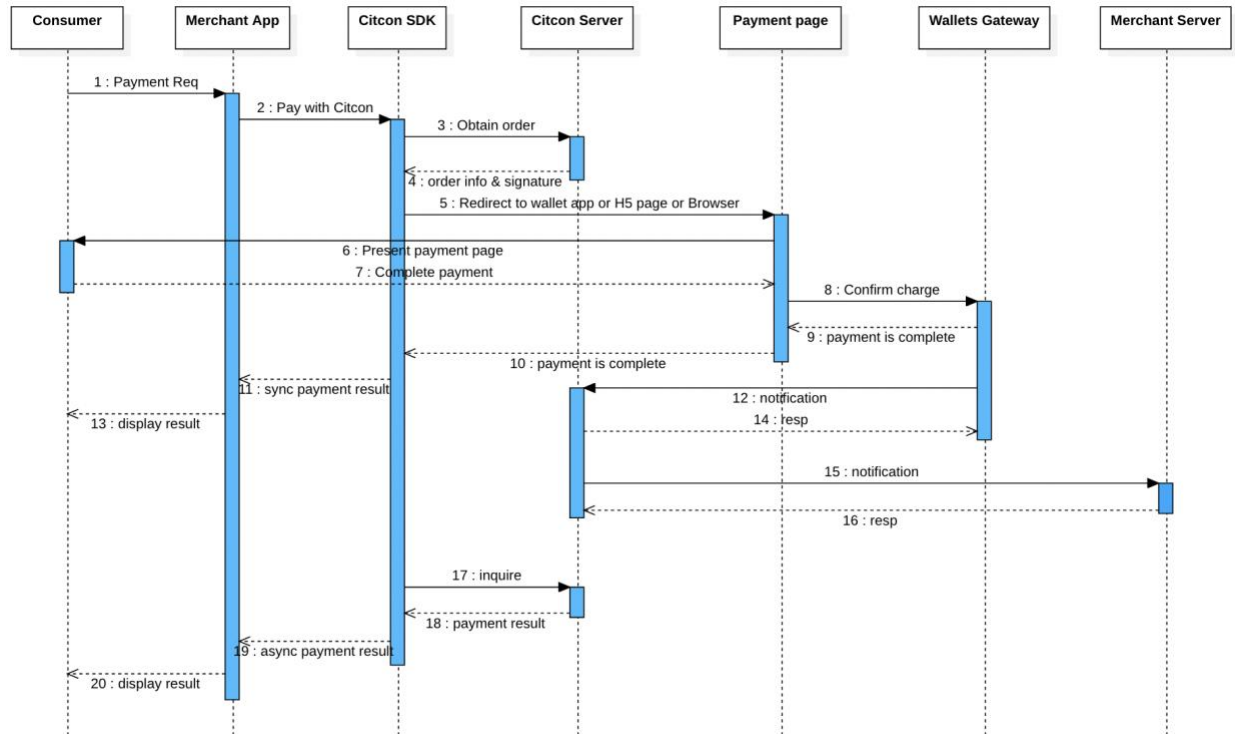
The following figure illustrates the workflow of In-app payment.

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):

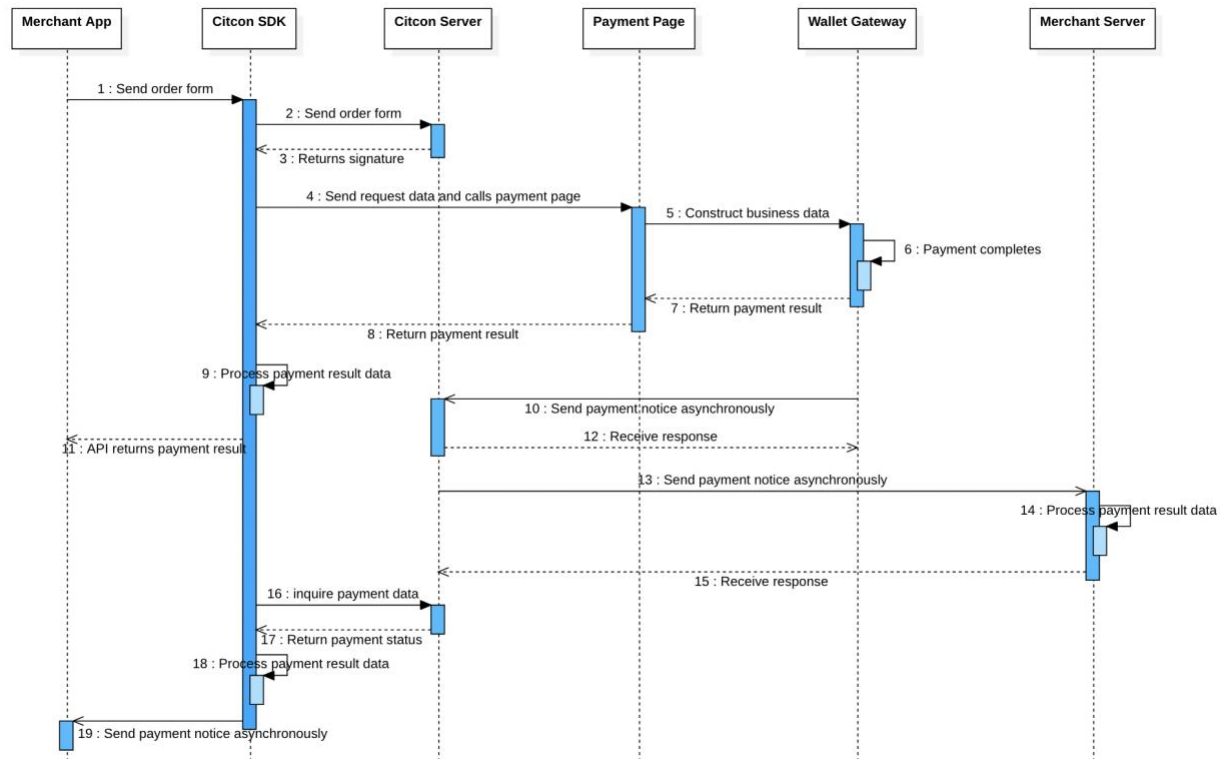
Step 2: Pay with Citcon: this message is referring to the payment target *requestOrder* provided by SDK, which send order information to call the Citcon SDK interface – see “[Request Parameters Description](#)” for more details on order format.

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

Citcon 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
static func sharedInstance() -> CPayManager	Get an instance of the CPayManager class.
func setMode(_ mode: CPayENVMode) -> Bool	Used to set the runtime environment. See CPayENVMode for details.
func setAccessToken(_ token: String) -> Bool	Set up a Citcon-assigned access token to access the API.
func requestOrder(_ order: CPayRequest, callback: @escaping CPayOrderCallback) -> Bool	Pay and get result via callback.
func inquireOrder(_ transId: String, callback: @escaping CPayCheckCallback) -> Bool	Query via transaction id and get result via callback.
func getVersion() -> String	Returns the version of Citcon SDK.
func handleScene(_ URLContexts: Set<UIOpenURLContext>) -> Void	Client-side processing method processes the Payment side returned <i>url</i> .
func handleUrl(_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool	Client-side processing method processes the Payment side returned <i>url</i> .
func handleUniversalLink(_ userActivity: NSUserActivity) -> Void	Client-side processing method processes the Payment side returned <i>url</i> .

- [Express order payment iOS](#)
- [Express order inquiry iOS](#)
- [Processing Client Side Returned URL](#)
- [Callback API](#)
- [Notification](#)
- [Request](#)
- [Response](#)
- [Codes returned to Client End](#)
- [Status returned to Client End](#)

## Express order payment iOS

**Method name:** Pay method

**Method prototype:** func requestOrder(\_ order: CPayRequest, callback: @escaping CPayOrderCallback) -> Bool

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

Parameter name	Parameter description
order: CPayRequest	App payment request parameters contain merchant's order information. See " <a href="#">CPayRequest</a> " for parameters description.
callback: @escaping CPayOrderCallback	Express pay SDK callback function returns with payment result. Please refer to " <a href="#">synchronous response parameter</a> " for more details on the relevant payment result.

## Express order inquiry iOS

**Method name:** inquire method by transaction id

**Method prototype:** func inquireOrder(\_ transId: String, callback: @escaping CPayCheckCallback) -> Bool

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

Parameter name	Parameter description
transId: String	The serial number assigned by Citcon to identify a trade in the Citcon system.
callback: @escaping CPayCheckCallback	Express pay SDK callback function returns with inquire result. Please refer to " <a href="#">CPayCheck</a> " for more details on the relevant inquire results.

## Processing client side returned URL

**Method name:** processing client method

**Method prototype:** func handleScene(\_ URLContexts: Set<UIOpenURLContext>) -> Void

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

**Note:**

**This method must be implemented, otherwise when call payment page, the payment result cannot synchronously returned.**

Parameter name	Parameter description
URLContexts: Set<UIOpenURLContext>	<i>url</i> returned by wallet app/H5 page/browser.

**Note:**

Please call this method in - (void)scene:(UIScene \*)scene  
openURLContexts:(NSSet<UIOpenURLContext \*> \*)URLContexts in SceneDelegate if you are using SceneDelegate in iOS 13 and above.

**Method name:** process client method

**Method prototype:** func handleUrl(\_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool

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

**Note:**

**This method must be implemented, otherwise when call payment page, the payment result cannot synchronously returned.**

Parameter name	Parameter description
app: UIApplication	The instance of application.
url: URL	<i>url</i> returned by wallet app/H5 page/browser.
options: [UIApplication.OpenURLOptionsKey : Any] = [:]	<i>options</i> returned by wallet app/H5 page/browser.

**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:** process client method

**Method prototype:** func handleUniversalLink(\_ userActivity: NSUserActivity) -> Void

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

**Note:**

**This method must be implemented, otherwise when call payment page, the payment result cannot synchronously returned.**

Parameter name	Parameter description
userActivity: NSUserActivity	<i>userActivity</i> returned by WeChat or other wallet apps opened via <i>universalLink</i> .

**Note:**

Please call this method in - (BOOL)application:(UIApplication \*)application continueUserActivity:(NSUserActivity \*)userActivity restorationHandler:(void (^)(NSArray<id<UIUserActivityRestoring>> \* \_Nullable))restorationHandler. In iOS 13.0 and above versions (including iOS 13.0), please call this method in - (void)scene:(UIScene \*)scene continueUserActivity:(NSUserActivity \*)userActivity if you are using SceneDelegate.

## Callback API

**Prototype:** public typealias CPayOrderCallback = (\_ result: CPayResult?) -> Void;

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

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

Parameter name	Parameter description
result: CPayResult?	Payment result returned via callback. See " <a href="#">CPayResult</a> " for parameters description.

Result returned needs to use value of the *status* and *data.status* fields of *CPayResult* to determine the payment result. When verifications of *status* = "success" and *data.status* = "\$*{success\_status}*", 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 *data.status* only.

An example of a successful sync notification of payment result is shown below:

```
{
  "status": "success",
  "app": "citcon_upi",
  "version": "v0.1.1",
  "data": {
    "object": "charge",
    "id": "8b3ac410a2d911ec8410e9f493c7d3ef",
    "reference": "sdk_digit_1647181572.353276",
    "amount": 1,
    "amount_captured": null,
    "amount_refunded": null,
```

```

    "currency": "USD",
    "time_created": 1647181576000,
    "time_captured": null,
    "auto_capture": false,
    "status": "succeeded",
    "country": "US",
    "payment": {
      "method": "paypal"
    }
  }
}

```

**Prototype:** public typealias CPayCheckCallback = (\_ result: CPayCheck?) -> Void;

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

After the inquiry is completed, it will return the payment results synchronously and asynchronously via *callbackBlock*.

Parameter name	Parameter description
result: CPayCheck?	Parameter result via callback and asynchronous notification. See " <a href="#">CPayCheck</a> " for parameters description.

Result returned needs to use value of the *status* and *data.status* fields of *CPayResult* to determine the payment result. When verifications of *status* = "success" and *data.status* = "\${success\_status}", 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 *data.status* only.

An example of a successful async notification of payment result is shown below

```

{
  "status": "success",
  "app": "citcon_upi",
  "version": "v0.1.1",

```



```

"data":{
  "id":"8b3ac410a2d911ec8410e9f493c7d3ef",
  "object":"charge",
  "amount":1,
  "currency":"USD",
  "status":" succeeded",
  "time_canceled":"","
  "expiry":null,
  "time_created":1647181576000,
  "country":"US",
  "reference":"sdk_digit_1647181572.353276",
  "amount_captured":null,
  "amount_refunded":null,
  "time_captured":null,
  "auto_capture":false,
  "payment":{
    "method":"paypal",
    "token":null,
    "type":"","
    "data":{
      "pan":"","
      "expiry":""
    }
  },
  "exchange":{
    "amount":null,
    "currency":null,
    "rate":null
  },
  "captures":{
    "data":[]
  },
  "refunds":{
    "data":[]
  }
}

```

```

    },
    "chargebacks":{
        "data":[]
    }
}
}
}

```

## Notification

### Prototype:

- static func NTFY\_ASYNC() -> String;
- `[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onAsyncResult:) name:[CPayRuntimeInst NTFY_ASYNC] object:nil];`

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

Parameter name	Parameter description
NSNotification *notification	<i>notification.object</i> as query result returned via callback. Actually notification.object is an instance of CPayCheck. See " <a href="#">CPayCheck</a> " for parameters description.

Result returned needs to use value of the *status* and *data.status* fields of *CPayResult* to determine the payment result. When verifications of *status* = "success" and *data.status* = "*success\_status*", payment is confirmed to be successful, otherwise, it may be regarded as failure. The Citcon SDK will send a notification after checking the status of the order through the Citcon server.

### 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

### CPayRequest

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

Parameter	Description
<b>transaction</b> CPayTransaction Required	<p>A <i>transaction</i> contains parameters for trade. Such as <i>reference</i>, <i>amount</i>, <i>currency</i>, etc. See <a href="#">CPayTransaction</a> for details.</p> <p><b>NOTE:</b> It is automatically assigned when the <i>CPayRequest</i> is created, which means it can be used directly.</p> <pre>e.g., order.transaction.reference = @"a3852167-9d10-4e8f-adf1-aef975667a87"</pre>
<b>consumer</b> CPayConsumer	<p>A <i>consumer</i> contains parameters used to represent the consumer. Such as <i>reference</i>, <i>firstName</i>, <i>lastName</i>, <i>billingAddress</i>, etc. Except for some wallets (e.g., <a href="#">card</a>, <a href="#">PayPal</a>, <a href="#">Venmo</a>), in most cases you don't need to use <i>consumer</i>. <a href="#">Required</a> for those wallets. See <a href="#">CPayConsumer</a> for details.</p> <p><b>NOTE:</b> Must be allocated via the merchant before use.</p> <pre>e.g., order.consumer = [CPayConsumer new];</pre>
<b>payment</b> CPayPayment Required	<p>A <i>payment</i> contains parameters indicating the type of trade, such as <a href="#">wechatpay</a>, <a href="#">alipay</a>, etc. See <a href="#">CPayPayment</a> for details.</p> <p><b>NOTE:</b> Must be allocated via the merchant before use.</p> <pre>e.g., order.payment = [CPayPayment new];</pre>

<b>goods</b> CPayGoods	<p>A <i>goods</i> contains parameters including <i>product</i> information and <i>shipping</i> address. In most cases, merchants do not need to use it. See <a href="#">CPayGoods</a> for details.</p> <p><b>NOTE:</b> Must be allocated via the merchant before use.</p> <pre>e.g., order.goods = [CPayGoods new];</pre>
<b>installments</b> CPayInstallments	<p>A <i>installments</i> contains only one parameter, indicating the number of installments. In most cases, you cannot use <i>installments</i> except for wallets that support Installment Payments. See <a href="#">CPayInstallments</a> for details.</p> <p><b>NOTE:</b> Must be allocated via the merchant before use.</p> <pre>e.g., order.installments = [CPayInstallments new];</pre>
<b>urls</b> CPayUrls Required	<p>A <i>urls</i> contains parameters for all urls used by trade. Such as <i>ipn</i>, <i>success</i>, etc. See <a href="#">CPayUrls</a> for details.</p> <p><b>NOTE:</b> It is automatically assigned when the <i>CPayRequest</i> is created, which means it can be used directly.</p> <pre>e.g., order.urls.ipn = @"https://ipn.merchants.com/"</pre>
<b>scheme</b> String	<p>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 <a href="#">Defining a URL Scheme in iOS</a>.</p> <p><b>NOTE:</b> If you use Alipay, you must set up it.</p> <pre>e.g., com.citcon.citconpay</pre>
<b>universalLink</b> String	<p>Universal Links allow you to connect to deep links in your iOS app and are supported iOS 9.2 or later. For more information about Universal Links, see <a href="#">Support Universal Links</a>.</p>

	<p><b>NOTE:</b> If you use WeChat Pay, you must set up Universal Links on the WeChat Open Platform.</p> <p>e.g., <code>https://www.merchant.com/apps/</code></p>
<p><b>controller</b> UIViewController</p>	<p>The view controller passed by the merchant's app for invoking wallet payment controller.</p> <p>e.g., <code>self</code></p>
<p><b>request3DSecureVerification</b> Bool</p>	<p>Indicates whether card payments use 3DS. Invalid for payment methods that do not supports card.</p> <p>e.g., <code>NO</code></p>
<p><b>unionpayMode</b> String</p>	<p>Indicates which mode to use when using UnionPay. "01" is development mode, "00" is production mode. Default value is "00".</p> <p>e.g., <code>00</code></p>
<p><b>chargeToken</b> String Required</p>	<p><i>chargeToken</i> is generated by merchants requesting Citcon. We strongly recommend that merchants obtain it through the merchant's own server rather than the client.</p> <p>e.g., <code>62d5c1e0242011eda300f59379b812ce</code></p>

## CPayTransaction

The *transaction* contains parameters for trade.

Parameter	Description
<p><b>reference</b> String Required</p>	<p>A serial number assigned by merchant to identify a trade in the merchant system.</p> <p><b>NOTE:</b> The value of <i>reference</i> should be unique to the merchant. Otherwise, the creation of the order may fail.</p> <p>e.g., <code>a3852167-9d10-4e8f-adf1-aef975cb7a87</code></p>

<b>amount</b> Int Required	Field to pass the amount int cents with tax included your customer are going to pay.  e.g., 128
<b>currency</b> String Required	Settlement currency type defined by three-letter code. Use upper case. For more information about supported currencies, see <a href="#">Supported Currencies</a> .  e.g., USD
<b>country</b> String Required	The country code defined by two-letter code. Use upper case. Indicates the country related to the order indicating the country payment is being processed in.  e.g., US
<b>autoCapture</b> Bool	Indicates to confirm that this charge should be captured without the need of the merchant sending a separate capture transaction. Default is <i>false</i> .  <b>NOTE:</b> It is invalid for some wallets such as <code>wechatpay</code> , <code>alipay</code> , and <code>union pay</code> .  e.g., false
<b>note</b> String	Merchant-defined additional information.  e.g., note

## CPayConsumer

A *consumer* contains parameters used to represent the consumer.

Parameter	Description
<b>reference</b> String	A serial number assigned by merchant to identify a consumer in the merchant system.  <b>NOTE:</b> It must be assigned by the merchants for some wallets. Usually used for some payment methods to obtain

	<p>relevant vaulting account. Such as <code>Paypal</code>, <code>card</code>, etc.  <b>Required</b> if using vault.</p> <p>e.g., <code>GUID4530-a169-4253-be43-8dc6a3e407c8</code></p>
<p><b>firstName</b> String</p>	<p>The first name of cardholder.</p> <p><b>NOTE:</b>  <b>Required</b> if using card 3D Secure.</p> <p>e.g., <code>John</code></p>
<p><b>lastName</b> String</p>	<p>The last name of cardholder.</p> <p><b>NOTE:</b>  <b>Required</b> if using card 3D Secure.</p> <p>e.g., <code>Doe</code></p>
<p><b>phone</b> String</p>	<p>The phone number of cardholder.</p> <p><b>NOTE:</b>  <b>Required</b> if using card 3D Secure.</p> <p>e.g., <code>5551234567</code></p>
<p><b>email</b> String</p>	<p>The email of cardholder.</p> <p><b>NOTE:</b>  <b>Required</b> if using card 3D Secure.</p> <p>e.g., <code>test@email.com</code></p>

**Notes:**

1. We do not store any sensitive information and it does not pass through Citcon's servers. Any vaulting accounts are obtained through the API of third-party wallets that meets security, reliability, and PCI DSS compliance requirements.

**CPayBillingAddr**

The parameters included in *billingAddr* include the billing address information for credit card.

Parameter	Description
<b>street</b> String	Represents the street number and street name related to the billing address.  <b>NOTE:</b> Required if using 3D Secure.  e.g., 555 Smith St
<b>street2</b> String	Represents the unit number related to the billing address.  <b>NOTE:</b> Required if using 3D Secure.  e.g., #2
<b>city</b> String	Represents the city, town, or village related to the billing address.  <b>NOTE:</b> Required if using 3D Secure.  e.g., Chicago
<b>state</b> String	Represents the State or Province related to the billing address.  <b>NOTE:</b> Required if using 3D Secure.  e.g., IL
<b>zip</b> String	Represents Zip Code related to billing address.  <b>NOTE:</b> Required if using 3D Secure.  e.g., 12345
<b>country</b> String	The country code defined by two-letter code. Use upper case.



	<b>NOTE:</b> Required if using 3D Secure. e.g., US
--	--

**NOTE:**

For security and sensitivity reason, we do not currently use or store this information. Usually they are required by third-part wallets with security guarantees, so it is safe to use for the SDK.

**CPayPayment**

A *payment* contains parameters indicating the type of trade.

Parameter	Description
<b>method</b> String Required	Indicates the consumer-selected payment method. Use lowercase letters. e.g., wechatpay
<b>indicator</b> String	Merchant identifier to indicate the source of a customer transaction. Usually, the SDK doesn't use it. e.g., authenticated
<b>requestToken</b> Bool	This field would inform Citcon that the merchant wishes to request a token it relates to the payment information being provided in the transaction. Usually, the SDK doesn't use it. Default is <i>false</i> . <b>NOTE:</b> A <i>token</i> will only be created on successful authorization. If the authorization is declined, then no token or vault services will be completed. A <i>token</i> can only be requested for the following payment methods. card, PayPal, Venmo e.g., false

<b>token</b> String	<p><i>token</i> is assigned by Citcon when the merchant has requested Citcon token and vault a specific card used by a customer. When a token is created and provided to a merchant, it can be used in subsequent transaction requests when the token is available. Usually, the SDK doesn't use it. Default is empty.</p> <p>e.g., ""</p>
<b>client</b> String [] Required	<p>Indicates the type of payment request. Merchants don't care about it, the Citcon SDK will automatically fill it.</p> <p><b>NOTE:</b> The values must contain "<i>mobile_native</i>" for the SDK. And the merchant doesn't need to assign it.</p> <p>e.g., ["mobile_native"]</p>
<b>expiry</b> Int	<p>Order timeout time is an absolute time.</p> <p>e.g., 60000.</p>
<b>data</b> CPayPaymentData	<p>This structure contains additional payment information such as card number, cvv, cardholder, etc. See <a href="#">CPayPaymentData</a>.</p>
<b>billingAddress</b> CPayBillingAddr	<p>The billing address of cardholder. See <a href="#">CPayBillingAddr</a> for details.</p> <p><b>NOTE:</b> Required if using card 3D Secure.</p>

### CPayPaymentData

A *data* contains parameters including *card* information.

Parameter	Description
<b>firstName</b> String	<p>Cardholder's first name.</p> <p>e.g., John.</p>

<b>lastName</b> String	Cardholder's last name.  e.g., Smith.
<b>cvv</b> String	Card Verification Value.  e.g., 123.
<b>pan</b> String	Represents the card number of either credit card or debit card.  e.g., 1230000000045.
<b>expiry</b> String	The expiration date of the card. MM/YY  e.g., 12/22.

### CPayGoods

A *goods* contains parameters including *product* information and *shipping* address.

Parameter	Description
<b>goods</b> CPayProduct []	<i>goods</i> array is represented as part of the goods object. See <a href="#">CPayProduct</a> for details.
<b>shipping</b> CPayShipping	<i>shipping</i> represents the shipping details related to the overall purchase of goods. See <a href="#">CPayShipping</a> for details.

### CPayProduct

A *product* contains parameters including product information.

Parameter	Description
<b>name</b> String	The name of the goods being purchased as part of the transaction.  e.g., Toy

<b>sku</b> String	A serial number assigned by merchant. Represents the SKU data related to the items being purchased.  e.g., 3264571
<b>url</b> String	<i>url</i> represents the URL that the products being purchased as part of this transaction can be found.  e.g., <a href="https://global.nichel.com/toy/3264571">https://global.nichel.com/toy/3264571</a>
<b>quantity</b> Int	<i>quantity</i> represents the quantity items being purchased related to product <i>name</i> .  e.g., 12
<b>totalAmount</b> Int	<i>totalAmount</i> represents the total amount for the item being purchased including tax and discount.  e.g., 32618
<b>unitAmount</b> Int	<i>unitAmount</i> represents the amount of the individual items related to the product <i>name</i> . Includes tax but excludes discount.  e.g., 24713
<b>totalTaxRate</b> Int	The tax rate the consumer is being charged for the item being purchased.  e.g., 12
<b>totalTaxAmount</b> Int	<i>totalTaxAmount</i> represents the total tax amount being charged for the item.  e.g., 28
<b>totalDiscountAmount</b> Int	<i>totalDiscountAmount</i> represents the total discount amount related to the item.  e.g., 128
<b>taxableAmount</b> Int	Represents the amount of the item that is taxable.  e.g., 0

<b>taxExemptAmount</b> Int	Represents the amount of the item that is tax exempted. e.g., 0
-------------------------------	--

**NOTE:**

Usually, the SDK doesn't use it.

**CPayShipping**

A shipping contains parameters that indicates where your shipment is going.

Parameter	Description
<b>firstName</b> String	The first name of the consumer the items are being shipped to as per shipping details. e.g., John
<b>lastName</b> String	The last name of the consumer the items are being shipped to as per shipping details. e.g., Smith
<b>phone</b> String	The phone number of the consumer the items are being shipped to as per shipping details. e.g., 5551234567
<b>email</b> String	The email address of the consumer the items are being shipped to as per shipping details. e.g., test@email.com
<b>street</b> String	The street number and street name related to the shipping address. e.g., 555 Smith St
<b>street2</b> String	The unit number related to the shipping address. e.g., #2

<b>city</b> String	Represents the city, town, or village related to the shipping address.  e.g., Chicago
<b>state</b> String	The State or Province related to the shipping address.  e.g., IL
<b>zip</b> String	The Zip Code related to the shipping address.  e.g., 123456
<b>country</b> String	The two-letter country code related to the shipping address.  e.g., US

**NOTE:**

Usually, the SDK doesn't use it.

**CPayInstallments**

A *installments* contains only one parameter, indicating the number of installments.

Parameter	Description
<b>id</b> String	This field represents the maximum value of the installment plan.

**CPayUrls**

A *urls* contains parameters for all urls used by trade.

Parameter	Description
<b>ipn</b> String	The URL for receiving asynchronous notifications after the payment is completed.

	e.g., <code>https://ipn.merchant.com/</code>
<b>success</b> String	After payment successful, the webpage is redirect to this URL.  e.g., <code>https://success.merchant.com/</code>
<b>fail</b> String	After payment failed, the webpage is redirect to this URL.  e.g., <code>https://fail.merchant.com/</code>
<b>cancel</b> String	After payment canceled, the webpage is redirect to this URL.  e.g., <code>https://cancel.merchant.com/</code>

## Notes:

1. In most cases, the simplest *order* contains only *transaction*, *payment*, *urls*. But for some cases, such as cardholder is enrolled in *3D Secure*, you should include parameters such as billing address in the *order*. See demo for details.
2. Some specific parameters are usually used for specific payment methods. E.g., *universalLink* for `wechatpay`, *scheme* for `alipay`, etc. See the tables above for more details.
3. Most of the above will not be used in the SDK but are reserved for future expansion.

An example of a successful order payment is shown below

```
CPayRequest *order = [CPayRequest new];
order.transaction.reference = @"sdk_wechatpay_1647325284.794511";
order.transaction.amount = 1;
order.transaction.currency = @"USD";
order.transaction.country = @"US";
order.transaction.note = @"";

order.payment = [CPayPayment new];
order.payment.method = @"wechatpay";
```

```

order.urls.ipn = @"https://ipn.merchants.com/";
order.urls.success = @"https://success.merchants.com/";
order.urls.cancel = @"https://cancel.merchants.com/";
order.urls.fail = @"https://fail.merchants.com/";

order.universallink = @"https://universallink.com/apps";

```

## 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.

### CPayResult

Holds the synchronization result of the transaction returned by Citcon. Obtain via [callback](#).

Parameter	Description
<b>status</b> String Required	Payment request status. Indicates whether the payment request succeeded or failed. It does not mean that the payment is successful. See <a href="#">Status Returned to Client End</a> for details.  <b>NOTE:</b> Merchant should verify <i>data.status</i> to check if payment was successful.



	e.g., success
<b>version</b> String Required	The version of API. e.g., v0.1.1
<b>data</b> CPayResultData Required	<i>data</i> contains more details of the transaction. The value of <i>data.status</i> needs to be verified by the merchant to determine the payment result. See <a href="#">CPayResultData</a> for more details.

## CPayResultData

Holds more details on payment result returned via synchronous notification.

Parameter	Description
<b>code</b> String	<i>code</i> used to indicate the type of error when a payment fails. See <a href="#">Codes Returned to Client End</a> for more details. <b>NOTE:</b> If payment successful, <i>code</i> has no value. e.g., 4000
<b>message</b> String	<i>message</i> used to store the description of the error. <b>NOTE:</b> If payment successful, <i>message</i> has no value. e.g., duplicate request
<b>id</b> String	A unique serial number assigned by Citcon to identify the transaction. <b>NOTE:</b> If payment fails, <i>id</i> has no value. e.g., Q0000323853-04e0cfafae47ffcd41c9
<b>object</b> String	Indicates the transaction type.

	<p><b>NOTE:</b> If payment fails, <i>object</i> has no value.</p> <p>e.g., <code>charge</code></p>
<p><b>reference</b> String</p>	<p>A serial number assigned by the merchant when sending the payment request. It should be unique.</p> <p><b>NOTE:</b> If payment fails, <i>reference</i> has no value.</p> <p>e.g., <code>365fa7c8-499c-487c-af7a-8812affe84b9</code></p>
<p><b>amount</b> Int</p>	<p>The total amount in cents of the transaction.</p> <p><b>NOTE:</b> If payment fails, <i>amount</i> has no value.</p> <p>e.g., <code>138</code></p>
<p><b>amountCaptured</b> Int</p>	<p>Amount captured in cents.</p> <p><b>NOTE:</b> Invalid for some wallets that do not support capture charge, such as <code>wechatpay</code>, <code>alipay</code>, and <code>union pay</code>. If payment fails, <i>amountCaptured</i> has no value.</p> <p>e.g., <code>0</code></p>
<p><b>amountRefunded</b> Int</p>	<p>Amount refunded in cents.</p> <p><b>NOTE:</b> If payment fails, <i>amountRefunded</i> has no value.</p> <p>e.g., <code>10</code></p>
<p><b>currency</b> String</p>	<p>The currency of the transaction defined by three-letter code. For more information about supported currencies, see <a href="#">Supported Currencies</a>.</p> <p><b>NOTE:</b> If payment fails, <i>currency</i> has no value.</p> <p>e.g., <code>USD</code></p>

<b>country</b> String	<p>The country code of the transaction defined by the two-letter code.</p> <p><b>NOTE:</b> If payment fails, <i>country</i> has no value.</p> <p>e.g., US</p>
<b>autoCapture</b> String	<p>Indicates whether the transaction is automatically captured.</p>
<b>status</b> String	<p>The transaction status represents the result of payment. The merchant should verify if <i>status</i> = “<i>\${success_status}</i>” to determine payment result. For more <i>status</i> Citcon supported, see <a href="#">Status Returned to Client End</a>.</p> <p><b>NOTE:</b> If payment fails, <i>status</i> has no value.</p> <p>e.g., succeeded</p>
<b>timeCreated</b> Int	<p>Timestamp when the transaction was created.</p> <p><b>NOTE:</b> If payment fails, <i>timeCreated</i> has no value.</p> <p>e.g., 1647423168</p>
<b>timeAuthorized</b> Int	<p>Timestamp when the transaction was authorized.</p> <p><b>NOTE:</b> If payment fails, <i>timeAuthorized</i> has no value.</p> <p>e.g., 1647423168</p>
<b>timeCaptured</b> Int	<p>Timestamp when the transaction was captured.</p> <p><b>NOTE:</b> If payment fails, <i>timeCaptured</i> has no value.</p> <p>e.g., 1647423168</p>
<b>chargeToken</b> String	<p>The token assigned by Citcon used to confirm charge. Invalid in SDK.</p>

**NOTE:**

If payment fails, chargeToken has no value.

e.g., null

## 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.

## CPayCheck

Holds the asynchronization result of the transaction returned by Citcon. Obtain via [callback](#) and [notification](#).

Parameter	Description
<b>status</b> String Required	status only represents whether the result of payment request was a successful or failure. It does not mean that the payment is successful. See <a href="#">Status Returned to Client End</a> for details.  <b>NOTE:</b> Merchant should verify <i>data.status</i> to check if payment was successful.  e.g., success
<b>version</b> String Required	The version of API.  e.g., v0.1.1
<b>app</b> String	Indicates what the API is.  e.g., citcon_upi

Required	
<b>data</b> String Required	<i>data</i> contains more details of the transaction. The value of <i>data.status</i> needs to be verified by the merchant to determine the payment result. See <a href="#">CPayCheckData</a> for more details.

## CPayCheckData

Holds more details of payment result returned via asynchronous notification.

Parameter	Description
<b>code</b> String	<i>code</i> used to indicate the type of error when a payment fails. See <a href="#">Codes Returned to Client End</a> for more details.  <b>NOTE:</b> If payment successful, <i>code</i> has no value.  e.g., 4000
<b>message</b> String	<i>message</i> used to store the description of the error.  <b>NOTE:</b> If payment successful, <i>message</i> has no value.  e.g., duplicate request
<b>id</b> String	A unique serial number assigned by Citcon to identify the transaction.  <b>NOTE:</b> If payment fails, <i>id</i> has no value.  e.g., Q0000323853-04e0cfafae47ffcd41c9
<b>object</b> String	Indicates the transaction type.  <b>NOTE:</b> If payment fails, <i>object</i> has no value.  e.g., charge

<b>amount</b> Int	<p>The total amount in cents of the transaction.</p> <p><b>NOTE:</b> If payment fails, <i>amount</i> has no value.</p> <p>e.g., 1</p>
<b>currency</b> String	<p>The currency of the transaction defined by three-letter codes. For more information about supported currencies, see <a href="#">Supported Currencies</a>.</p> <p><b>NOTE:</b> If payment fails, <i>currency</i> has no value.</p> <p>e.g., USD</p>
<b>status</b> String	<p>The transaction status represents the result of payment. The merchant should verify if status="<i>success_status</i>" to determine payment result. For more status Citcon supported, see <a href="#">Status Returned to Client End</a>.</p> <p><b>NOTE:</b> If payment fails, status has no value.</p> <p>e.g., succeeded</p>
<b>timeCanceled</b> Int	<p>Timestamp of transaction cancellation.</p> <p><b>NOTE:</b> If payment fails or cancellation did not happen, <i>timeCanceled</i> has no value.</p>
<b>expiry</b> String	<p>Timestamp of transaction expiration.</p> <p><b>NOTE:</b> If payment fails, <i>expiry</i> has no value.</p>
<b>timeCreated</b> Int	<p>Timestamp of transaction creation.</p> <p><b>NOTE:</b> If payment fails, <i>timeCreated</i> has no value.</p> <p>e.g., 1649814984000</p>

<b>country</b> String	<p>The country code of the transaction defined by two-letter codes.</p> <p><b>NOTE:</b> If payment fails, <i>country</i> has no value.</p> <p>e.g., US</p>
<b>reference</b> String	<p>A unique serial number assigned by merchant.</p> <p>e.g., 365fa7c8-499c-487c-af7a-8812affe84b9</p>
<b>amountCaptured</b> Int	<p>Amount captured in cents.</p> <p><b>NOTE:</b> Invalid for some wallets that do not support capture charges, such as <code>wechatpay</code>, <code>alipay</code> and <code>union pay</code>. It might be <i>null</i>.</p> <p>e.g., null</p>
<b>amountRefunded</b> Int	<p>Amount refunded in cents.</p> <p><b>NOTE:</b> If payment fails, <i>amountRefunded</i> has no value.</p> <p>e.g., 10</p>
<b>timeCaptured</b> Int	<p>Timestamp when the transaction was captured.</p> <p><b>NOTE:</b> Invalid for some wallets that do not support capture charges. It might be <i>null</i>.</p> <p>e.g., null</p>
<b>autoCapture</b> Bool	<p>Indicates whether the transaction was automatically captured.</p> <p>e.g., true</p>
<b>payment</b> CPayCheckDataPayment	<p><i>payment</i> holds details of payment. See <a href="#">CPayCheckDataPayment</a> for more details.</p>

## CPayCheckDataPayment

Holds payment details contained in the *data* of the payment result returned via asynchronous notification.

Parameter	Description
<b>method</b> String	Indicates payment method of the transaction. <code>e.g., wechatpay</code>

## Notification trigger condition

Trigger condition name	Description
TRADE_FINISHED	Trade is completed successfully.

## Codes returned to Client end

Return code	Description
0	Success.
-1	Common error type. Such as bad request, failed to create payment and so on.
-2	Cancellation.
6001	Cancellation.
other	Other errors. See the returned message for details.

## Status Returned to Client end

Return code	Description
succeeded	Indicates that the payment is successful.
failed	Indicates that the payment is failed.



initiated	Indicates that the transaction has been initialized.
authorized	Indicates that the payment has been authorized.
pending	Indicates that the payment is pending. Typically, this status indicates a payment failure.
canceled	Indicates that the payment has been canceled.
expired	Indicates that the payment has expired.

## 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 11+ and Xcode 12+, and can be installed by manually integrating the framework.

### Manual

1. Visit the Citcon iOS repository on Github and navigate to the latest release.
2. Download the `citcon_upi_sdk_ios-${version}.zip` file attached to the Github release.
3. Unzip the file, then drag and drop the XCFramework into your Xcode project.
4. If the framework's symbols are unable to load, navigate to "General" pane of your target and find the "Frameworks, Libraries, and Embedded Content" dropdown. Switch \*.xcframework from "Do Not Embed" to "Embed and Sign".

#### NOTE:

The three .xcframeworks are CardinalMobile.xcframework, CPaySDK.xcframework and PPriskMagnes.xcframework, which you have to drag and drop into your Xcode project.

## 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 uppay sdk uppayx1 uppayx2 uppayx3
Card PayPal Venmo	\${your-bundle-id}.payments com.venmo.touch.v2

**NOTE:**

You don't need to add all items to LSAApplicationQueriesSchemes, just the items related to 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. <code>e.g., wxeb0650d489d69e14</code>
Alipay	The content of Alipay URL Schemes is assigned by the merchant. It must be the same as the "scheme" field of CPayRequest. <code>e.g., com.citcon.citconpay</code>
Card	<code>\${your-bundle-id}.payments</code>

**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 iOS SDK, 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.

## Initialize the SDK in your app

### Import SDK

In order to use the SDK, first you should import the SDK header files using the following code.

```
#import <CPaySDK/CPaySDK-Swift.h>
```

### Access-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 sharedInstance] setAccessToken:@"${your access 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 sharedInstance] setMode:CPayENVModePROD];
```

## Set your return URL

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

```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url sourceApplication:(nullable NSString *)sourceApplication annotation:(nonnull id)annotation {
    return [[CPayManager sharedInstance] handleUrl:app open:url options:@{}];
}
```

In iOS 9.0 and above versions (including iOS 9.0), please call this method in -

```
(BOOL)application:(UIApplication *)app openURL:(NSURL *)url
options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options
```

```
- (BOOL)application:(UIApplication *)app
    openURL:(NSURL *)url
    options:(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {
    return [[CPayManager sharedInstance] handleUrl:app open:url options:options];
}
```

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

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

## Set up the payment request

Follow the code below to create an WeChat Pay order for payment.

```
CPayRequest *order = [CPayRequest new];
order.transaction.reference = @"wechatpay_ref_123erg0234s";
order.transaction.amount = 1;
```

```

order.transaction.currency = @"USD";
order.transaction.country = @"US";
order.transaction.note = @"note";

order.payment = [CPayPayment new];
order.payment.method = @"wechatpay";

order.urls.ipn = @"https://ipn.com/";
order.urls.success = @"https://success.com/";
order.urls.cancel = @"https://cancel.com/";
order.urls.fail = @"https://fail.com/";

```

**Note:**

Different payment methods may use different parameters, see Demo for details.

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

```

[[CPayManager sharedInstance] requestOrder:order callback:^(CPayResult * _Nullable resp) {
    if ([resp.status isEqualToString:@"success"]) {
        NSLog(@"Txn Id: %@", resp.data.id);
        NSLog(@"Ref Id: %@", resp.data.reference);
        NSLog(@"Amount: %ld", (long)resp.data.amount);
        NSLog(@"Currency: %@", resp.data.currency);
        NSLog(@"PaymentMethod: %@", resp.data.payment.method);
        NSLog(@"AutoCapture: %@", resp.data.autoCapture ? @"true" : @"false");
        NSLog(@"Txn status: %@", resp.data.status);
        NSLog(@"TimeCreated: %ld", (long)resp.data.timeCreated);
        NSLog(@"ChargeToken: %@", resp.data.chargeToken);
    }
}];

```

## Receive the payment result

Follow the code below to receive synchronous payment result.

```

[[CPayManager sharedInstance] requestOrder:order callback:^(CPayResult * _Nullable resp) {
    // resp is the CPayResult instance of the synchronization result of the payment
}];

```

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

Follow the code below to receive asynchronous payment result.

```
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(onAsyncResult:)  
name:[CPayRuntimeInst NTFY_ASYNC] object:nil];
```

`SEL` for processing payment result.

```
- (void)onAsyncResult:(NSNotification *)notification {  
    CPayCheck *resp = [notification object];  
}
```