Citcon UPI SDK iOS Documentation

Version 2.0.8

Version Information

Ver. No.	Date	Remarks
2.0.8	2022-10-14	Add new payment methods to support alipay+
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

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

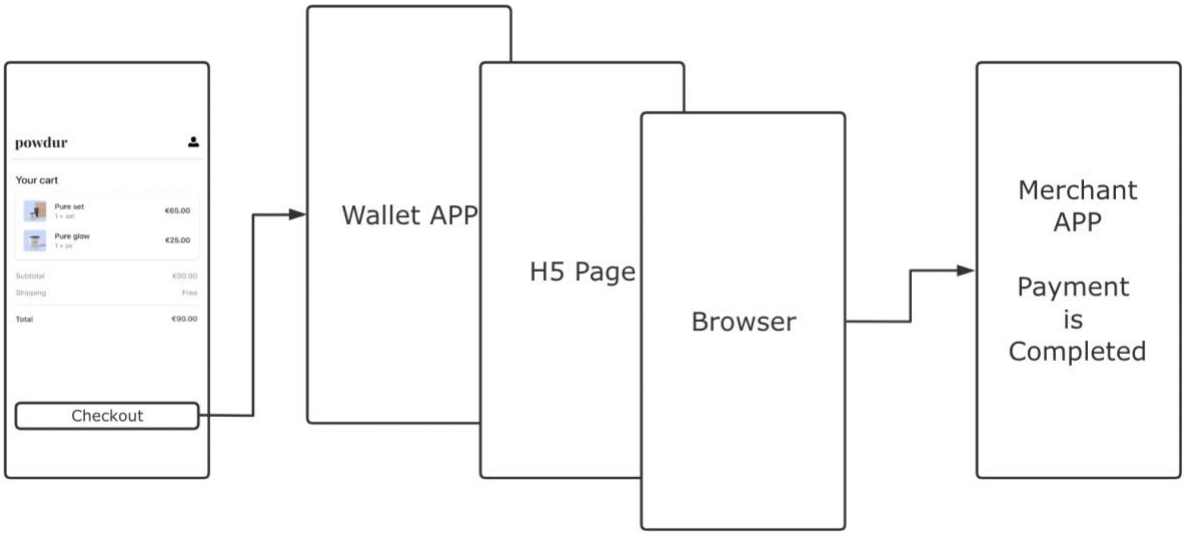
Citcon mainly supports the following currencies:

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

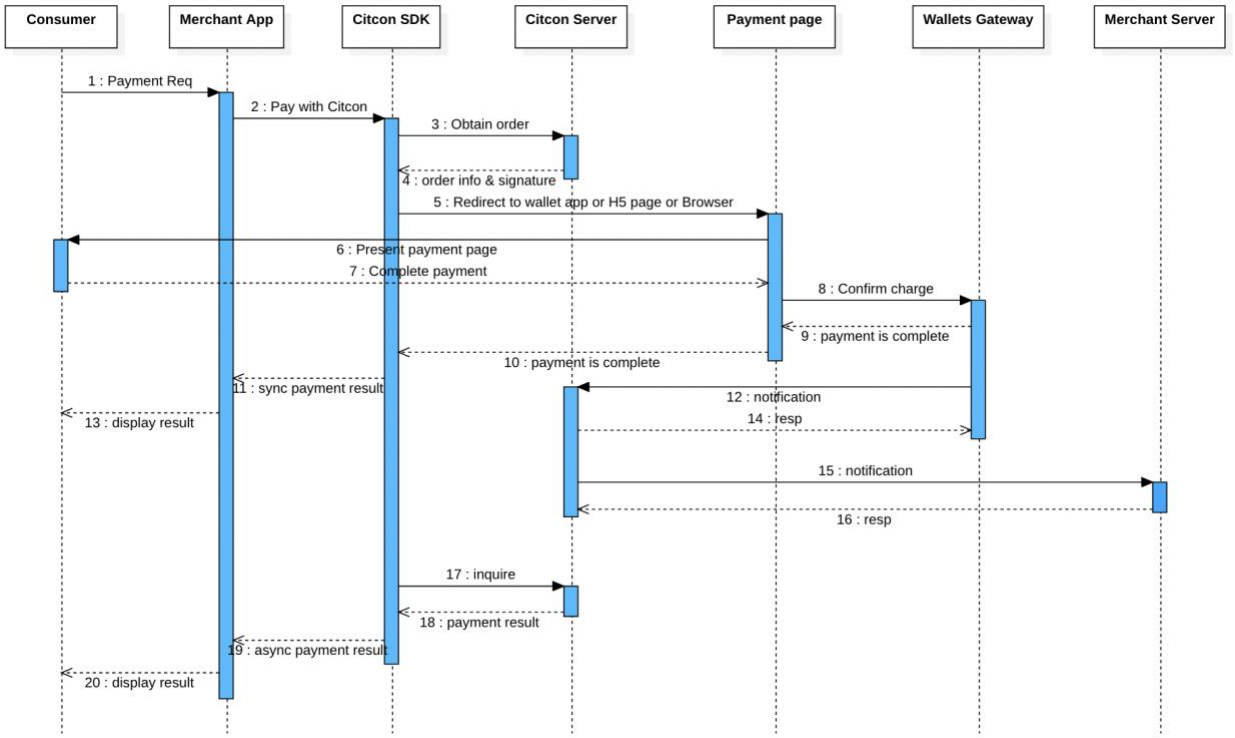
The following figure illustrates the workflow of In-app payment.

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



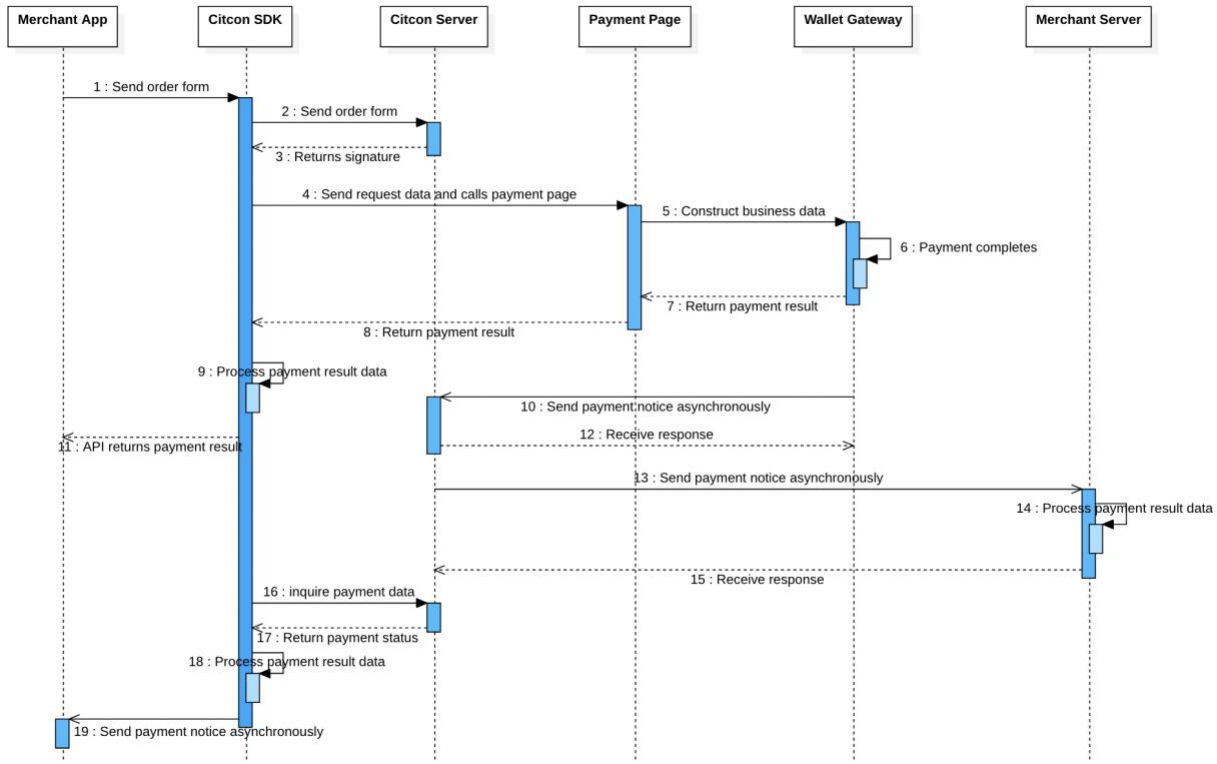
Key steps in the payment process are explained below (take the payment process on iOS platform as an example):

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**
   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 sharedInst() -> 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 *url*.
func handleUrl(_ app: UIApplication, open url: URL, options: [UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool	Client-side processing method processes the Payment side returned *url*.
func handleUniversalLink(_ userActivity: NSUserActivity) -> Void	Client-side processing method processes the Payment side returned *url*.

Express order payment iOS

Method name: Pay method

Method prototype: func requestOrder(_ order: CPayRequest, callback: @escaping CPayOrdertCallback) -> 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 "CPayRequest" for parameters description.
callback: @escaping CPayOrderCallback	Express pay SDK callback function returns with payment result. Please refer to "synchronous response parameter" 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 "CPayCheck" 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>	*url* 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	*url* returned by wallet app/H5 page/browser.
options: [UIApplication.OpenURLOptionsKey : Any] = [:]	*options* 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	*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. 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 "CPayResult" 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 "CPayCheck" 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	*notification.object* as query result returned via callback. Actually notification.object is an instance of CPayCheck. See "CPayCheck" 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
transaction CPayTransaction Required	A *transaction* contains parameters for trade. Such as *reference*, *amount*, *currency*, etc. See CPayTransaction for details. **NOTE:** It is automatically assigned when the *CPayRequest* is created, which means it can be used directly. e.g., order.transaction.reference = @"a3852167-9d10-4e8f-adf1-aef975667a87"
consumer CPayConsumer	A *consumer* contains parameters used to represent the consumer. Such as *reference*, *firstName*, *lastName*, *billingAddress*, etc. Except for some wallets (e.g., card, PayPal, Venmo), in most cases you don't need to use *consumer*. Required for those wallets. See CPayConsumer for details. **NOTE:** Must be allocated via the merchant before use. e.g., order.consumer = [CPayConsumer new];
payment CPayPayment Required	A *payment* contains parameters indicating the type of trade, such as wechatpay, alipay, etc. See CPayPayment for details. **NOTE:** Must be allocated via the merchant before use. e.g., order.payment = [CPayPayment new];

goods CPayGoods	A *goods* contains parameters including *product* information and *shipping* address. In most cases, merchants do not need to use it. See CPayGoods for details. **NOTE:** Must be allocated via the merchant before use. e.g., order.goods = [CPayGoods new];
installments CPayInstallments	A *installments* contains only one parameter, indicating the number of installments. In most cases, you cannot use *installments* except for wallets that support Installment Payments. See CPayInstallments for details. **NOTE:** Must be allocated via the merchant before use. e.g., order.installments = [CPayInstallments new];
urls CPayUrls Required	A *urls* contains parameters for all urls used by trade. Such as *ipn*, *success*, *etc*. See CPayUrls for details. **NOTE:** It is automatically assigned when the *CPayRequest* is created, which means it can be used directly. e.g., order.urls.ipn = @"https://ipn.merchants.com/"
scheme String	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. **NOTE:** If you use Alipay, you must set up it. e.g., com.citcon.citconpay
universalLink String	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 Support Universal Links.

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

CPayTransaction

The *transaction* contains parameters for trade.

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

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

CPayConsumer

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

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

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

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

	NOTE: 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
method String Required	Indicates the consumer-selected payment method. Use lowercase letters. e.g., wechatpay
indicator String	Merchant identifier to indicate the source of a customer transaction. Usually, the SDK doesn't use it. e.g., authenticated
requestToken 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 *false*. **NOTE:** A *token* will only be created on successful authorization. If the authorization is declined, then no token or vault services will be completed. A *token* can only be requested for the following payment methods. card, PayPal, Venmo e.g., false

token String	*token* 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. e.g., “”
client String [] Required	Indicates the type of payment request. Merchants don't care about it, the Citcon SDK will automatically fill it. **NOTE:** The values must contain "*mobile_native*" for the SDK. And the merchant doesn't need to assign it. e.g., [“mobile_native”]
expiry Int	Order timeout time is an absolute time. e.g., 60000.
data CPayPaymentData	This structure contains additional payment information such as card number, cvv, cardholder, etc. See CPayPaymentData.
billingAddress CPayBillingAddr	The billing address of cardholder. See CPayBillingAddr for details. **NOTE:** Required if using card 3D Secure.

CPayPaymentData

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

Parameter	Description
firstName String	Cardholder's first name. e.g., John.

lastName String	Cardholder's last name. e.g., Smith.
cvv String	Card Verification Value. e.g., 123.
pan String	Represents the card number of either credit card or debit card. e.g., 1230000000045.
expiry 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
goods CPayProduct []	*goods* array is represented as part of the goods object. See CPayProduct for details.
shipping CPayShipping	*shipping* represents the shipping details related to the overall purchase of goods. See CPayShipping for details.

CPayProduct

A *product* contains parameters including product information.

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

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

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

city String	Represents the city, town, or village related to the shipping address. e.g., Chicago
state String	The State or Province related to the shipping address. e.g., IL
zip String	The Zip Code related to the shipping address. e.g., 123456
country 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
id String	This field represents the maximum value of the installment plan.

CPayUrls

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

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

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

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
status String Required	Payment request status. Indicates whether the payment request succeeded or failed. It does not mean that the payment is successful. See Status Returned to Client End for details. **NOTE:** Merchant should verify *data.status* to check if payment was successful.

	e.g., success
version String Required	The version of API. e.g., v0.1.1
data CPayResultData Required	*data* contains more details of the transaction. The value of *data.status* needs to be verified by the merchant to determine the payment result. See CPayResultData for more details.

CPayResultData

Holds more details on payment result returned via synchronous notification.

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

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

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

	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
status 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 Status Returned to Client End for details. **NOTE:** Merchant should verify *data.status* to check if payment was successful. e.g., success
version String Required	The version of API. e.g., v0.1.1
app String	Indicates what the API is. e.g., citcon_upi

Required	
data String Required	*data* contains more details of the transaction. The value of *data.status* needs to be verified by the merchant to determine the payment result. See CPayCheckData for more details.

CPayCheckData

Holds more details of payment result returned via asynchronous notification.

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

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

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

CPayCheckDataPayment

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

Parameter	Description
method String	Indicates payment method of the transaction. e.g., wechatpay

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, the drag and drop the XCFramework into your Xcode project.
4. If the framework's symbols are unable to loaded, 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 uppaysdk uppayx1 uppayx2 uppayx3
Card PayPal Venmo	${your-bundle-id}.payments com.venmo.touch.v2

NOTE:

You don't need to add all items to LSApplicationQueriesSchemes, 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. 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 CPayRequest. e.g., com.citcon.citconpay
Card	${your-bundle-id}.payments

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 sharedInst] 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 sharedInst] 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 sharedInst] 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 sharedInst] 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 sharedInst] 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 sharedInst] 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 sharedInst] 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];
}
```