Citcon Pay iOS SDK Documentation

Version 2.1.4

I. Introduction

Citcon's iOS development SDK was designed for online merchants to integrate Citcon payment solutions effortlessly into their own iOS apps. By using the SDK, merchant developers can focus on business logics without having to understand the plumbing of payment transactions. The payment experience will be totally transparent and seamless to end consumers. This version of the SDK only supports payments through Alipay and WeChatPay. Union Pay, credit cards and other payment methods will be added to future versions. The version of the SDK can process Alipay/WeChat Pay transactions using the following currencies: USD, CAD and RMB. Other currencies will be supported by future versions.

Although it's not necessary for merchant's engineering team to have in-depth knowledge of how Citcon Pay is integrated under the hood, an exhibition of the payment transaction flow will help developers better understand the integration process at the conceptual level, as illustrated in Figure.1 below.

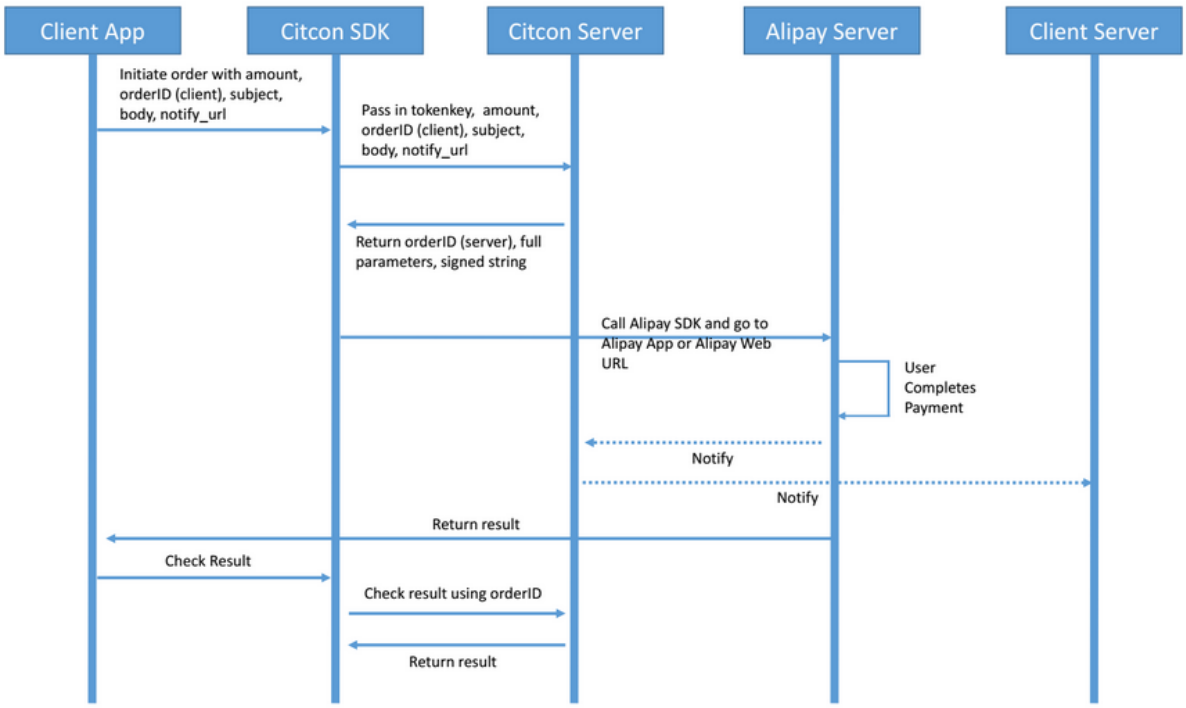


Fig. 1 – Payment transaction flow from merchant app to payment processors (Alipay/ WeChat Pay) through Citcon

This version of the SDK targets iOS 9.0 and above. Because it is distributed as a standard framework and therefore language-independent, merchant developers can program in either Objective-C or Swift to integrate with it.

II. The Citcon Pay Framework

This section details the main components of the Citcon Pay Framework for iOS development. The commonly used header files and their purposes will be listed here, and A step-by-step

example of integrating the framework in a demo merchant app will be shown in the next section.

1. Framework identity
    a. Name: CPay
    b. Bundle identifier: com.cpay.cpay
    c. Current version: 2.0

2. Required frameworks or libraries
    - AlipaySDK (included by Citcon)
    - CoreGraphics
    - CoreMotion
    - CoreTelephony
    - Foundation
    - Security
    - SystemConfiguration
    - UIKit
    - WeChatAuthSDK

3. Class references
    a. CPayManager :NSObject

    The CPayManagerclass performsthe most common payment related tasks: set up merchant token, send order to Citcon Pay and query the status of a specific transaction.

    Class methods:

Name	**Description**
+ setupTokenKey	Set up merchant token (provided by Citcon)
+ setupMode	Set up SDK running environment, default CPayModePROD

+ requestOrder	Sends order information to Citcon and initiates the payment transaction. It is recommended to check the order result (CPayOrderResult) in the callback function (completion).
+ inquireResult	Inquire about the status of a payment transaction
+ processOpenUrl	Handle the callback function of payment vendor

b. CPayOrder: NSObject

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

Property summary

Name	Type	Description	Example
referenceId	NSString	A reference merchant creates and assigns to the transaction.	123xyz
Amount	NSString	Total charge amount of the transaction in cents	245, e.g. $2.45
Currency	NSString	The type of currency defined in a three-letter code	USD or CAD or CNY
Vendor	NSString	The name of vendor that will process the payment	alipayor wechatpay
Subject	NSString	A customer-defined description of the transaction	Gift for Mom
Body	NSString	A more detailed customer-defined description of the transaction	A Blu-ray player and a few great movies
scheme	NSString	Your app scheme for alipay payment, set in the Info.plist.	cpaydemo.citconpay.com
ipnUrl	NSString	The URL of a page Citcon Pay should post transaction status to. Normally this should be a page on the merchant's website.	http://www.xyz.com/notify.php

callbackUrl	NSString	The URL of a page Citcon Pay should redirect customer to after the payment transaction has completed. Normally this should be a page on the merchant's website.	http://xyz.com/confirm.php
allowDuplicate	BOOL	Flag to control duplicate orders.	YES/NO
transactionId	NSString	An unique identifier of the transaction. This ID is generated by Citcon Pay.	123456789

c. CPayOrderResult :NSObject

Holds the status and message for a transaction returned by Citcon Pay service. An instance of the CPayOrderResult class can be inspected in the callback handler of the requestOrder method of CPayManager.

Property summary

Name	Type	Description	Example
Message	NSString	Detailed description of the status of a transaction	Transaction succeeded
Result	NSString	The result of the transaction	success
resultStatus	NSInteger	The status code for the result	0 (refer to 4.Status Code)
order	CPayOrder	Order object containg order details	

d. CPayCheckResult :NSObject

An instance of the CPayCheckResult class holds the details of a payment transaction's state at any time the inquireResult method of CPayManager is called. A common practice is to inspect the properties in the callback handler of inquireResult.

Property summary

Name	Type	Description	Example
transactionId	NSString	An unique identifier of the transaction. This ID is generated by Citcon Pay.	123456679
referenceId	NSString	A reference identifying the transaction. This ID is generated by merchant	Abc123
Type	NSString	The type of the transaction: "charge" or "refund"	charge
Amount	NSString	The total amount of the transaction in cents	245, e.g. $2.45
Currency	NSString	The name the of the currency in a three-letter code	USD
Time	NSString	The timestamp for the transaction	2017/06/22 1:23:12 PM
Status	NSString	The status of the transaction	success
refunded_amount	NSString	The actual amount in centsrefunded to the customer, which is may or may not be the same as the charge amount	179, e.g. $1.79
refund_status	NSString	The status of a "refund" transaction	success
Note	NSString	Additional information for the transaction	

4. Status code

Status Code	Description
0	Success
-1	Bad signature, unregistered AppId, wrong AppId, mismatched AppId or some other error.
-2	The customer cancels the payment, switches back to the app, the customer cannot pay, and so on.
8000	Order is being processed
4000	Payment failed
5000	Duplicate request
6001	The customer cancels the payment
6002	Network connection error
6004	Unknown payment result
others	Other payment error

5. Notification

After the payment is completed, the Citcon SDK will automatically query immediately to ensure that the correct results are obtained. And post a notification as soon as the automated query is complete. Therefore, you can also add observers to receive notifications. The notification is kOrderPaymentFinishedNotification.

III. Example

In this section, a demo merchant app making Alipay payments through Citcon Pay is demonstrated step-by-step using Objective-C. The source code of this demo app is also provided as part of the SDK distribution package.

1. Create a new XCode project → Choose the iOS template → Choose Single View Application --> Choose options for the project.
2. Modify the configurations of your project as the following steps:

   2.1 Add the KeyValues to the info.plist of your project

   <key>LSApplicationQueriesSchemes</key>

   <array>

   <string>weixin</string>

   <string>wechat</string>

   <string>alipay</string>

   <string>safepay</string>

   </array>

   <key>NSAppTransportSecurity</key>

   <dict>

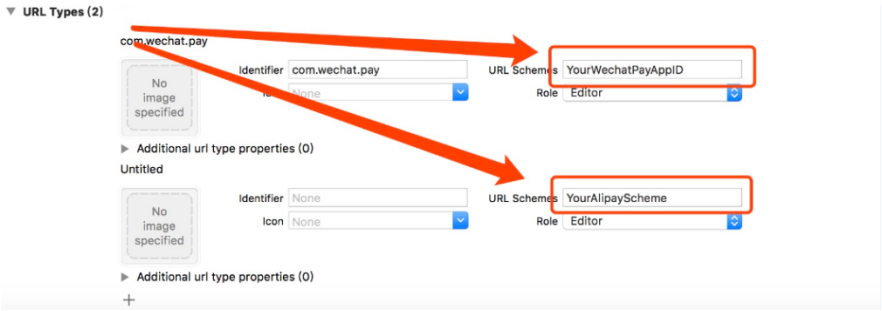
   <key>NSAllowsArbitraryLoads</key>

   <true/>

   </dict>

   that allow wechat or alipay come back your app after payment

   2.2. Add the URL Types to the configuration of your project



Make sure the value of URL Schemes of WechatPayAppID is the same as your Wechat pay AppID

and the value of URL Schemes of YourAlipayScheme is the same as the field ' scheme' of the CPayOder which your wants to send to Citcon Pay for processing.

3. Import the CPay.framwork.

4. Add Run Script Phase in Build Phases as below
   Shell: /bin/sh
   bash "./Frameworks/CPay.framework/strip-frameworks.sh"

   Important: This step will allow the merchant app to embed the framework for development on both simulator and actual device.

5. Set up merchant token and SDK running environment in appDelegate

```
- (BOOL)application:(UIApplication*)applicationdidFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {

[CPayManagersetupTokenKey:@"abc1234"];

[CPayManagersetupMode:CPayModePROD];

//... other initializations
}
```

6. Handle callback in appDelegate

```
- (BOOL)application:(UIApplication*)appopenURL:(NSURL *)url options:
(NSDictionary<UIApplicationOpenURLOptionsKey,id> *)options {
BOOL processed = [CPayManagerprocessOrder:app url:url
standbyCallback:^(CPayOrderResult *result) {
//optional further processing
}];

if (processed) {
return YES;
}

//optional further processing

return YES;
}
```

7. Initiate a payment transaction

```
CPayOrder * order = [CPayOrdernew];
order.referenceId = @"1ZLLJULOCRW3LAU";
order.subject = _@"Test";
order.body = @"This is a test transaction";
order.amount = @”2.00”;
order.currency = @”USD”;
```

```
order.vendor = “alipay”        ;
order.ipnUrl = “http://www.xyz.com/listen.php”;
order.callbackUrl = http://www.xyz.com/confirmation.php;
order.scheme = @" paydemo.citconpay.com";

[CPayManagerrequestOrder:ordercompletion:^(CPayOrderResult * payResult) {

// Interpret the result in callback handler


};
```

(Optional) Inquire status of the transaction and displays the status information in the merchant app

```
[CPayManagerrequestOrder:ordercompletion:^(CPayOrderResult * payResult) {

[CPayManagerinquireResult:order.transactionIdmethod:@"real"completion:^(CPayCheck
Result *checkResult) {

        NSMutableString * result = [NSMutableStringnew];

        [result appendFormat:@"- ORDER RESULT\n"];
        [result appendFormat:@"    STATUS: %d\n", (int)payResult.resultStatus];
        [result appendFormat:@"    MESSAGE: %@\n", payResult.message];
        [result appendFormat:@"\n"];
        [result appendFormat:@"- CHECK RESULT\n"];
        [result appendFormat:@"    TRANSACTION_ID: %@\n",
checkResult.transactionId];
        [result appendFormat:@"    REFERENCE_ID  : %@\n", checkResult.referenceId];
        [result appendFormat:@"    TYPE: %@  TIME: %@\n", checkResult.type,
checkResult.time];
        [result appendFormat:@"    STATUS: %@  AMOUNT: %@ (%@)\n",
checkResult.status, checkResult.amount, checkResult.currency];
        [result appendFormat:@"    REFUND_STATUS: %@  AMOUNT: %@\n",
checkResult.refund_status, checkResult.refunded_amount];
        [result appendFormat:@"    NOTE: %@\n", checkResult.note];

    }];
};
```

8. Add observer to receive notification (optional)

```
- (void)completeAutoInquire:(NSNotification *)notification {

    CPayCheckResult *checkResult = (CPayCheckResult *) notification.object;
    // TODO: add your code
}

// Add observer before you request order
[[NSNotificationCenter defaultCenter]addObserver:self
selector:@selector(completeAutoInquire:) name:kOrderPaymentFinishedNotification
object:nil];
```