Mobile Payment Control Access Guide

3.3.1

2019-10-09 Published **2019-10-25 Implemented**

THIS PAGE INTENTIONALLY LEFT BLANK.

Version Information

Ver. No.	Date	Remarks
2.0.0	2013-01-08	First draft
2.1.0	2015-09-16	1. Certain errors are corrected 2. The notes on browser invocation control are deleted, and it is recommended that the wap related information should be consulted for browser mode.
3.0.0	2015-12-09	1. UPPaymentControl is used to substitute the old version of payment control 2. Interface description and invocation plug-in description are modified 3. “4.4 Description of project configuration” is added 4. The notes on modification of old version of UnionPay merchant are added
3.3.1	2019-10-25	1. Modify the demo code. 2. Modify the url scheme demo code.

Table of Contents

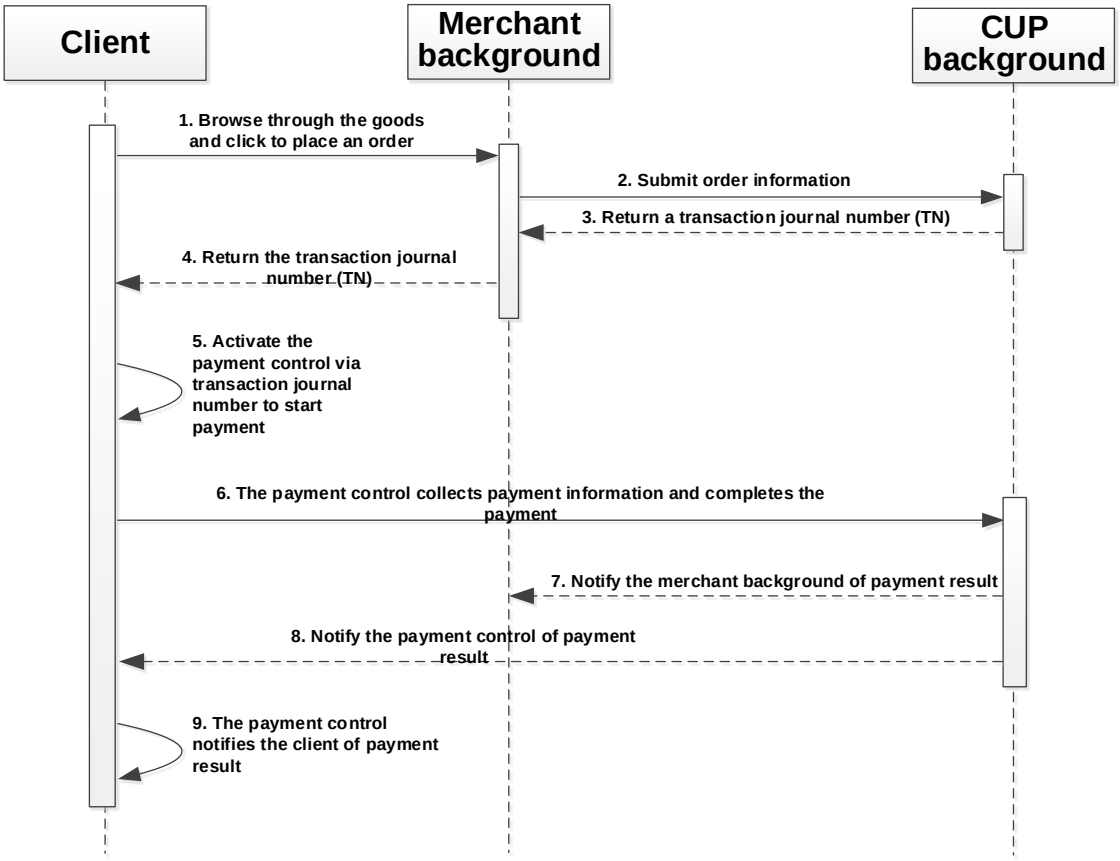
1 Overview

UNIONPAY mobile payment control ("payment control") is primarily intended to provide cooperating merchants' mobile client with secure and convenient payment service. The user completes payment by entering such elements as bank card number, mobile phone number, password (debit card and prepaid card) or CVN2, expiry date (credit card) and verification number in the payment control.

(Warm Tip: If you are a merchant using China UnionPay's old version of payment control, you may simply skip other chapters and transform the Project according to section 5 of this document.)

2 Description of payment process

The process of conducting a transaction via payment control is illustrated below:



Notes to the flowchart:

(1) The user clicks in the client to buy the goods, and the client issues an order generation request to the merchant background;

(2) After receipt of the order generation request, the merchant background organizes and pushes order information to UNIONPAY background;

(3) After receiving and verifying the order information, UNIONPAY background generates the corresponding transaction journal number (i.e., TN) and returns that transaction journal number to the merchant background (response element: transaction journal number, etc.);

(4) After receiving the transaction journal number, the merchant background returns the transaction journal number to the client;

(5) The client invokes the payment control via the transaction journal number (TN);

(6) After the user entering the relevant payment information in the payment control, the payment control issues a payment request to UNIONPAY background;

(7) After successful payment, UNIONPAY background notifies the payment result to the merchant background;

(8) UNIONPAY notifies the payment result to the payment control;

(9) The payment control displays the payment result and returns the payment result to the client;

Note: This document mainly focuses on the realization of parts (5) and (9) in the above process.

iOS version of payment control applies to iOS 6.0 and higher versions of terminal devices.

3 Test account number

Card number and mobile phone number information provided for testing purpose (such information is provided for testing only, and no transaction actually happens)

China Merchants Bank debit card: 6226090000000048 Mobile phone number: 18100000000 Password: 111101 SMS verification number: 123456 (click and get a verification number before inputting) Certificate type: 01 identity card Certificate number: 510265790128303 Name: Zhang San
Huaxia Bank credit card: 6226388000000095 Mobile phone number: 18100000000 cvn2: 248 Expiry date: 1219 SMS verification number: 123456 (click and get a verification number before inputting) Certificate type: 01 identity card Certificate number: 510265790128303 Name: Zhang San

4 iOS client

This section deals with the description of SDK interface, Xcode engineering configuration and its interface invocation details and requires that the reader should have certain iOS programming experience.

4.1 SDK description

After getting the development kit provided by China UnionPay, the merchant developer shall check the directory where SDK file belongs, i.e., upmp_iphone/paymentcontrol. Files mentioned below are all under that directory. UnionPay payment control static library ("UPPaymentControl") consists of the following two files:

UPPaymentControl.h

libPaymentControl.a

4.2 Interface description

1. Payment interface

```
- (BOOL)startPay:(NSString*)tn
        fromScheme:(NSString*)schemeStr
          mode:(NSString*)mode
   viewController:(UIViewController*)viewController
```

All parameters are defined below:

Parameter name	**Type**	**Meaning**
tn	NSString*	Required; Transaction journal number, which is a transaction certificate generated by UNIONPAY background and issued to merchant background after the merchant background delivers order information to UNIONPAY background;
schemeStr	NSString *	Required; Merchant-defined protocol, a protocol defined by the merchant to guide the return of the payment control after the invoked payment interface com-

		pletes payment. For details, refer to the definition of URL Type in step 2 in section 4.4;
mode	NSString*	Required; Access mode, which indicates how the merchant invokes the payment control. The parameter provides the following two optional values: "00" means access to production environment (required for official version); "01" means access to development and test environment (required for test version);
viewController	UIViewController*	Required; View controller initiated, the view controller used by the merchant's application program for invoking UnionPay mobile payment control;
Return value	BOOL	YES: successful invocation of payment control; NO: failed invocation of payment control;

2. Check if the interface of UnionPay App is installed

```
- (BOOL)isPaymentAppInstalled
```

This function has no input parameter and its main function is check if the UnionPay payment App is installed on the user's mobile phone. YES will be returned if UnionPay payment App is already installed on the user's mobile phone.

Parameter name	Type	Meaning
Return value	BOOL	YES: UnionPay payment APP is installed; NO: UnionPay payment APP is not installed;

3. Interface for returned result

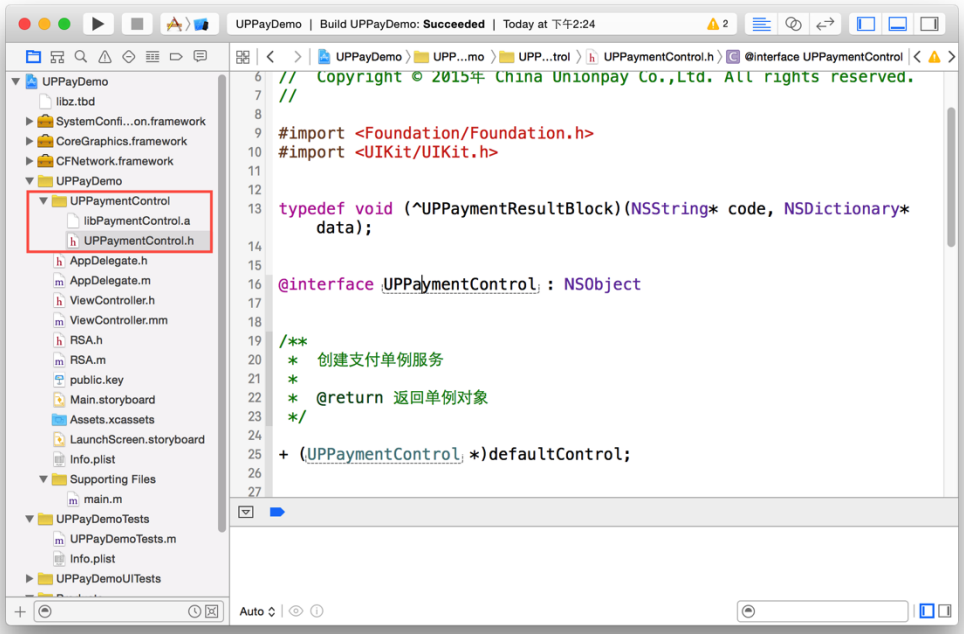
```
- (void)handlePaymentResult:(NSURL*)url
  completeBlock:(UPPaymentResultBlock)completionBlock;
```

Relevant parameters are defined below:

Parameter name	Type	Meaning
url	NSURL*	Required; Payment result url, which, after input, is parsed by SDK and returned to the merchant client via completionBlock;
completionBlock	Block	Required; Result handling method defined by the merchant client, which incorporates two input parameters namely code and data. This method will be detailed in paragraph 3 of section 4.5 of this document;

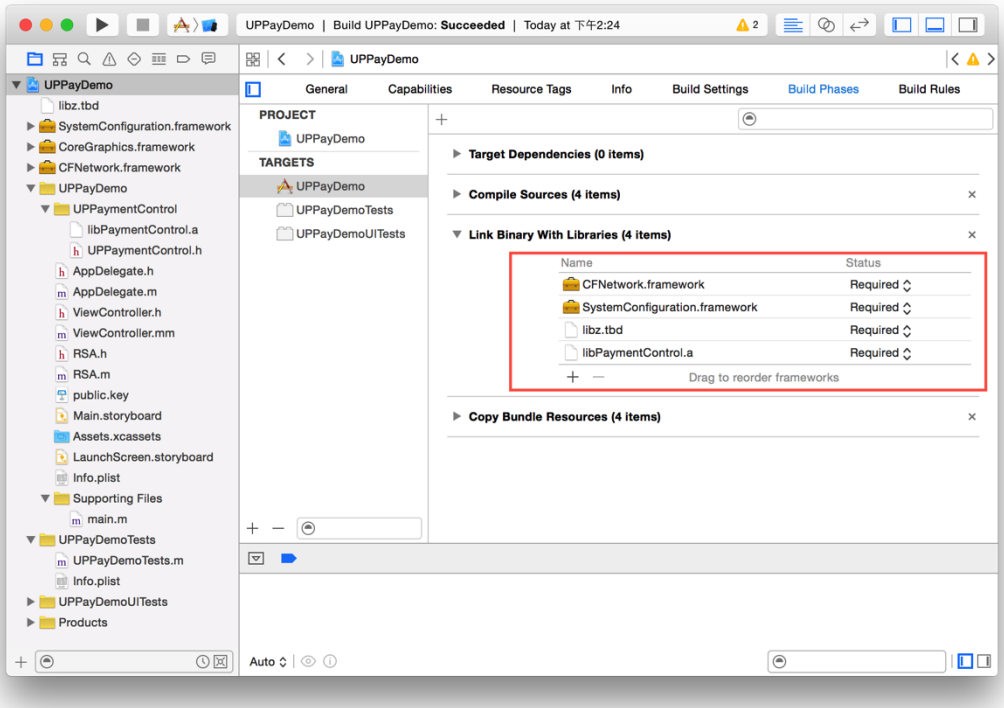
4.3 Add SDK kit

To use UPPaymentControl, it is necessary to add the UPPaymentControl.h file under the directory of paymentcontrol/inc and the libPaymentControl.a file under the directory of paymentcontrol/libs to the project applied by the merchant. The result of such addition is shown below:

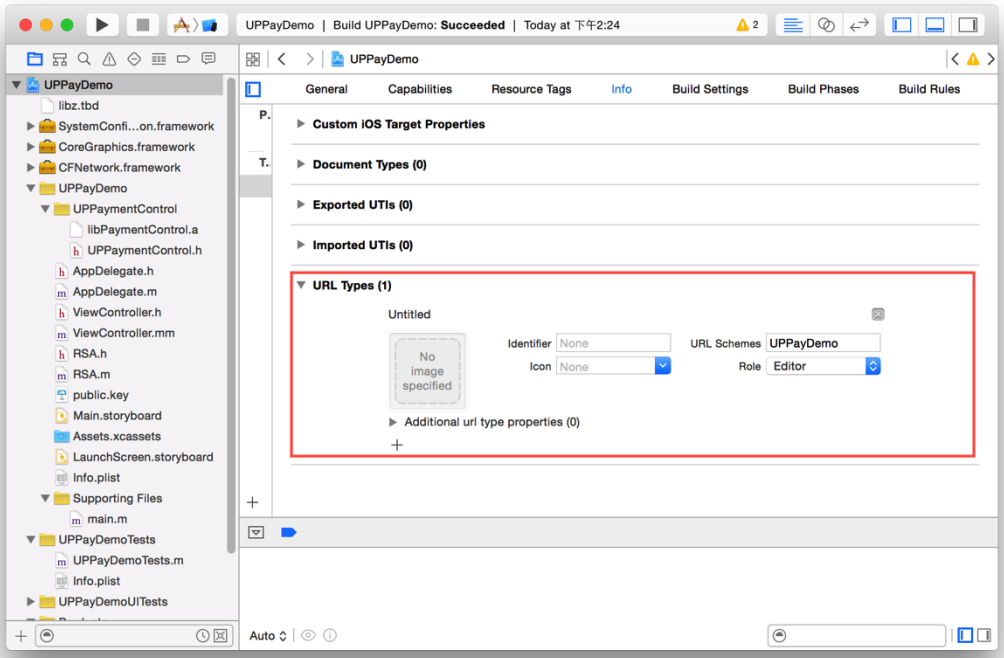


4.4 Project configuration

1. To use UPPaymentControl, it is necessary to add CFNetwork.framework, SystemConfiguration.framework, libz and libPaymentControl.a to the Project. The result of such addition is shown below:

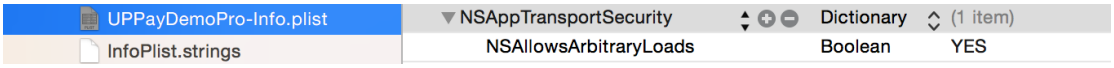


2. Add a URL Types callback protocol (in UPPayDemo project use “UPPayDemo” as the protocol) to the project configuration info.plist in order to return to the merchant client after completion of payment.



3. http request setting

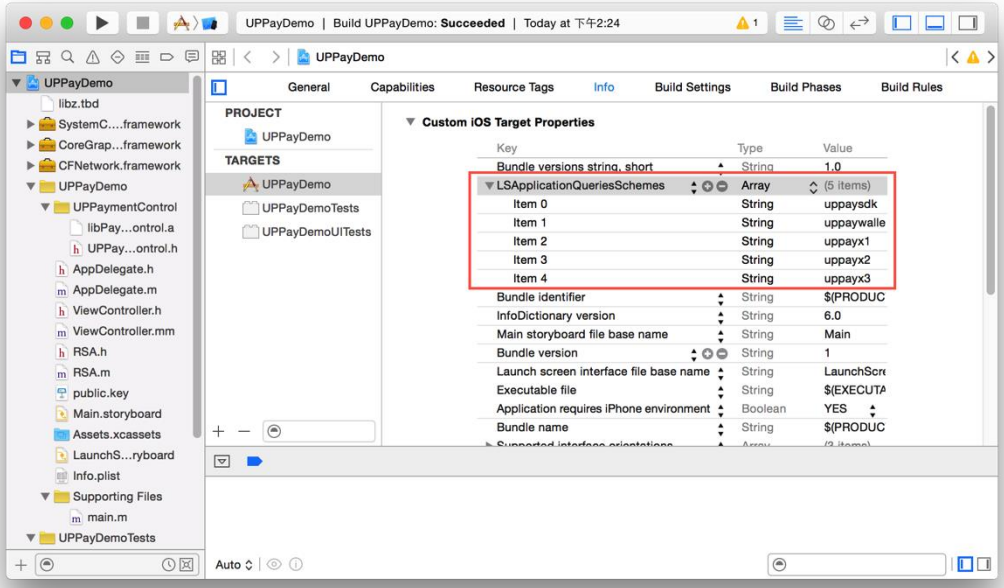
To conduct http request in versions following Xcode7.0, it is necessary to add NSAppTransportSecurity Dictionary to the plist file corresponding to the Project and meanwhile set the attribute value of NSAllowsArbitraryLoads at YES. Specific setting may be performed with reference to the following screenshot:



Please delete this setting before implementation under production environment. Please delete this setting before the official release of your APP to Apple.

4. Add protocol whitelist

For the purpose of development in versions following Xcode7.0, it is necessary to add LSApplicationQueriesSchemesArray to the plist file corresponding to the Project and add 5 items including uppaysdk, uppaywallet, uppayx1, uppayx2 and uppayx3. Specific setting may be performed with reference to the following screenshot:



Alternatively, simply add the following codes to the plist file:

```
<key>LSApplicationQueriesSchemes</key>
<array>
        <string>uppaysdk</string>
        <string>uppaywallet</string>
        <string>uppayx1</string>
        <string>uppayx2</string>
        <string>uppayx3</string>
</array>
```

4.5 Invocation plug-in

Header file UPPaymentControl.h is cited among code files requiring invocation of payment control interfaces.

(Note: If the value of the Project's compile source as option is not Objective–C++, the type of all files citing the foregoing header file shall be changed as .mm)

1. Invocation of payment interface

The merchant App gets tn from the merchant server. When tn is not null, the payment interface is invoked.

```
// When the gotten tn is not null, the payment interface is invoked
if (tn != nil&& tn.length>0)
{
        [[UPPaymentControldefaultControl]
startPay:tn
            fromScheme:@"UPPayDemo"
            mode:self.tnMode
  viewController:self];
  }
```

2. Check if UnionPay APP interface invocation is installed

```
if([[UPPaymentControldefaultControl] isPaymentAppInstalled])
{
// When it is made sure that UnionPay APP has been installed on the user's mobile phone, the merchant client may undergo the corresponding personalized processing
  }
```

3. Invocation of interface for returned result

The payment control result handling function handlePaymentResult: completeBlock: needs to be invoked in the method of application: openURL: sourceApplication: annotation: in the Project's AppDelegate file.

The payment control result handling function handlePaymentResult: completeBlock: comprises two parameters. Parameter 1url is payment result string, whose url content is parsed according to the method of handlePaymentResult: completeBlock:; parameter 2completionBlock is the result handling method defined by the merchant APP, which comprises two input parameters including code and data. Code represents payment result and its values will be "success", "fail" or "cancel", which stands for successful payment, failed payment and cancelled payment. Data is currently remained as reservation..

The payment result returned by the payment control is only for reference. The exact payment result shall be in the back-end notification returned from UPOP.

After invocation of the payment interface, sample code of result handling method is shown below:

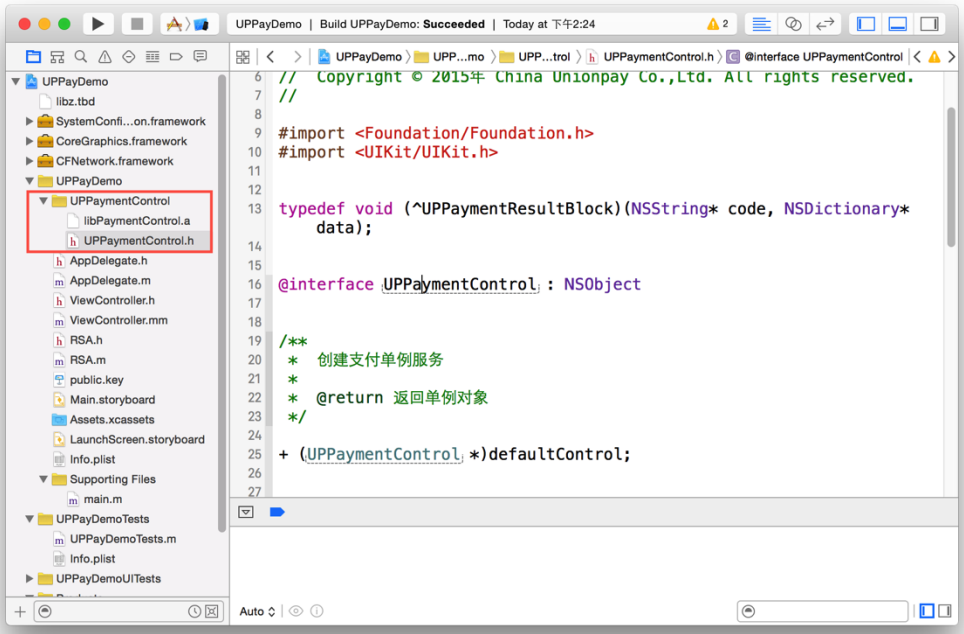
```
- (BOOL)application:(UIApplication *)app openURL:(NSURL *)url op-
```

```
tions:(NSDictionary<NSString*, id>*)options {
[[UPPaymentControl defaultControl] handlePaymentResult:url completeBlock:^(NSString *code, NSDictionary *data) {
if([code isEqualToString:@"success"]) {
//if code is “success”,  please check the back-end notification before displaying the pay-
  mentresult
}
else if([code isEqualToString:@"fail"]) {
//payment failed
}
else if([code isEqualToString:@"cancel"]) {
//payment cancelled
}
}];
return YES;
}
```

5 Notes on modification of old version of SDK merchant

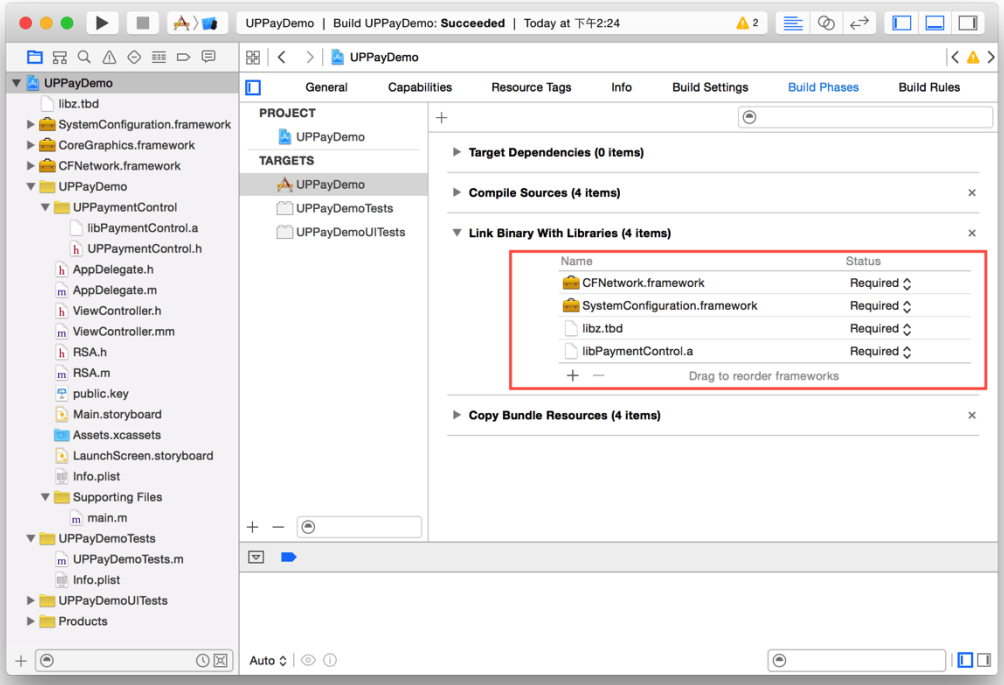
5.1 Add SDK kit

For any merchant using an old version of UNIONPAY payment control, please delete the old version of UNIONPAY payment control from the Project and then add the UPPaymentControl.h file under the new directory of paymentcontrol/inc and the libPaymentControl.a file under the directory of paymentcontrol/libs to the project applied by the merchant. The effect of such addition is shown below:

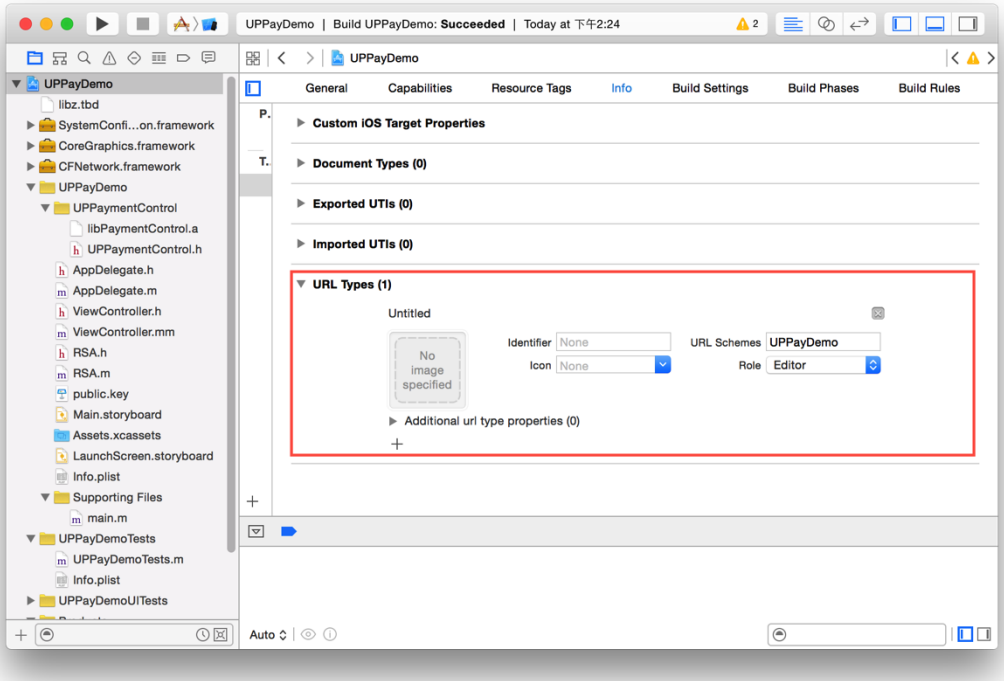


5.2 Project configuration

1. For any merchant using an old version of UNIONPAY payment control, please delete the third-party library corresponding to old version of payment control from the project configuration and then add CFNetwork.framework, SystemConfiguration.framework, libPaymentControl.a and libz to the project. The effect of such addition is shown below:

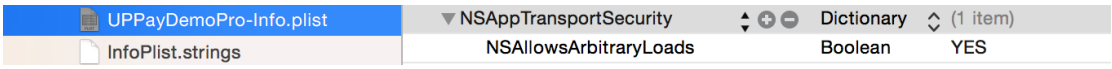


2. In the project configuration info.plist add a URL Types callback protocol (in UPPayDemo project use “UPPayDemo” as the protocol) in order to return to the merchant client after the payment control completes payment.



3. http request setting

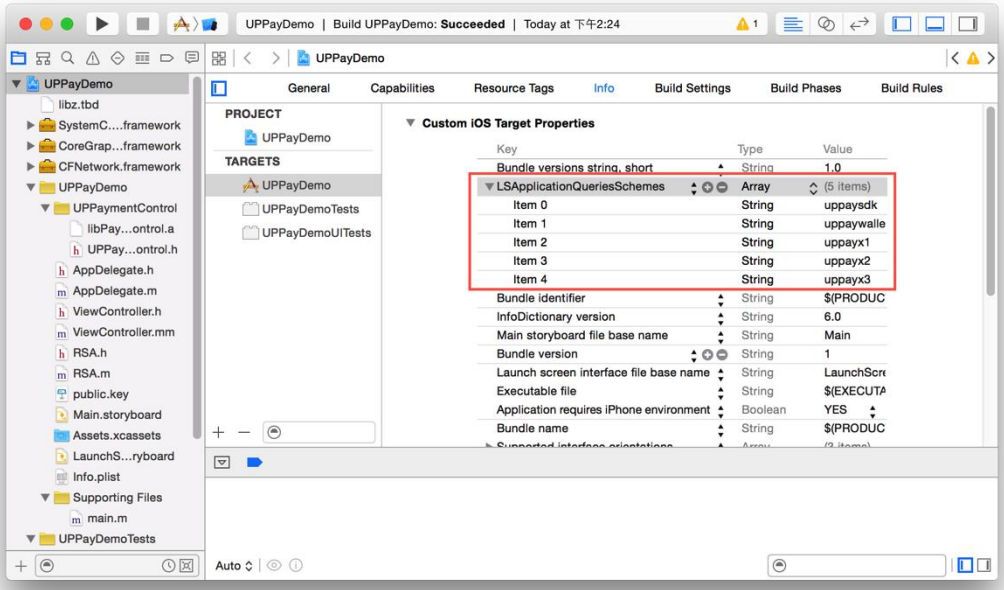
To conduct http request in versions following Xcode7.0, it is necessary to add NSAppTransportSecurity Dictionary to the plist file corresponding to the Project and meanwhile set the attribute value of NSAllowsArbitraryLoads at YES. Specific setting may be performed with reference to the following screenshot:



Please delete this setting before implementation under production environment. Please delete this setting before the official release of your APP to Apple.

4. Add protocol whitelist

In the plist file corresponding to the Project, add LSApplicationQueriesSchemesArray and add 5 items including uppaysdk, uppaywallet, uppayx1, uppayx2 and uppayx3. Specific setting may be performed with reference to the following screenshot:



Alternatively, simply add the following codes to the plist file:

```
<key>LSApplicationQueriesSchemes</key>
<array>
        <string>uppaysdk</string>
        <string>uppaywallet</string>
        <string>uppayx1</string>
        <string>uppayx2</string>
        <string>uppayx3</string>
</array>
```

5.3 Interface description and plug-in invocation

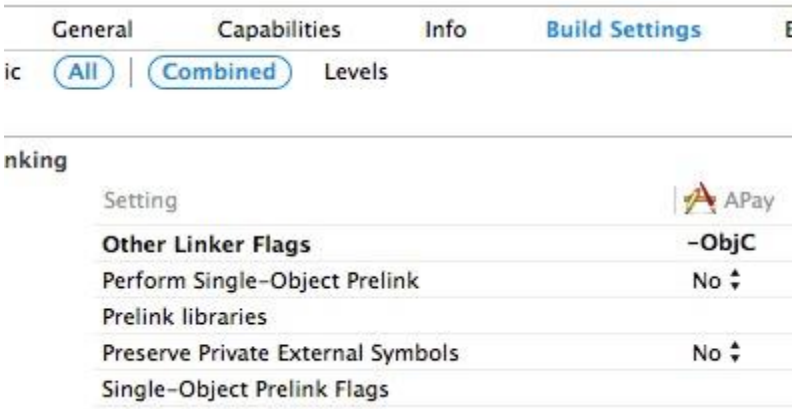
Please refer to 4.2 Interface description and 4.4 Notes on plug-in invocation.

6 Summary of frequently asked questions

For more information, refer to https://open.unionpay.com Help Center -FAQ

6.1 How to add –ObjC macro

Select project targets——》build settings ->Linking->other linker flags



6.2 Undefined for architecture XXX prompt error during compilation

For instance:

```
Undefined symbols for architecture arm64:
  "___cxa_begin_catch", referenced from:
      ___clang_call_terminate in libPaymentControl.a(UPMPService.o)
  "operator new[](unsigned long)", referenced from:
      UPPasswordTool::getBlockPublicKeyWithModulues(char const*, char const*, char const*, char const*, char const*,
  char**) in libPaymentControl.a(UPPasswordTool.o)
      UPXProguardUtil::proguardSeed(char**) in libPaymentControl.a(UPXProguardUtil.o)
      UPChannelExpress::UPChannelExpress() in libPaymentControl.a(UPChannelExpress.o)
      UPChannelExpress::setPublicKey(char const*, char const*, char const*, char const*, char const*, char const*) i
  libPaymentControl.a(UPChannelExpress.o)
      UPRSAUtil::setPublicKey(char const*, char const*, char const*, char const*, char const*, char const*) in
  libPaymentControl.a(UPRSAUtil.o)
      UPXCryptUtil::randomSessionKey(char**) in libPaymentControl.a(UPXCryptUtil.o)
      UPXCryptUtil::setPublicKey(char const*) in libPaymentControl.a(UPXCryptUtil.o)
      ...
```

1. Since payment control involves mixed programming of C, C++ and OC, link error may occur after the merchant project introduces the header file UPPaymentControl.h. This problem may be resolved through one of the following three methods:

   a) Change the suffix name of all source files involving citation of UPPaymentControl.h as .mm;

   b) If the merchant does not want to modify the suffix name of source file, it may add to the project a null class inherited from NSObject and change the suffix name of file.m as .mm. The method is: new file->Objective-C class-> Class name self-defined -> Save -> Change suffix name as .mm;

   c) Set the value of the project's "compile source as" option at Objective–C++;

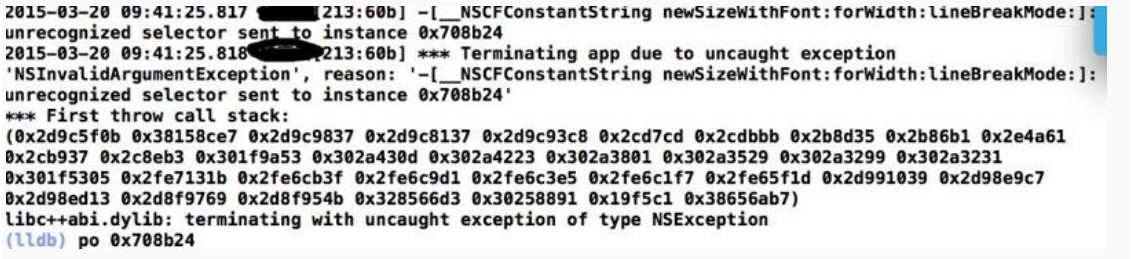
2. Because the self-defined library file libPaymentControl.a is added to the UPPayDemo project, during compilation of the project, it is necessary to check the route setting of Framework Search Paths, Header Search Paths and Library Search Paths in project configuration Search Paths. In addition, check if some uncertain paths are redundant.

3. Modify the attribute value of C++ Language Dialect and C++ Standard Library under the tag of Build Setting -> Apple LLVM compiler Language in the xcode project as Compiler Default. If the error is still unresolved, try modifying each place with addition of –ObjC macro as -force_load+space character+control path, e.g., -force_load $(PROJECT_DIR)/ libPaymentControl.a. If there is still an error report, the context shall be that the file libPaymentControl.a cannot identify an exception. For instance, ld: file not found: /Users/apple/Desktop/Communication 2/ libPaymentControl.a clang: error: linker command failed with exit code 1 (use -v to see invocation). Please make sure that the file libPaymentControl.a does exist in this path, and the possible cause is a mismatch of context path.

6.3 Control crash exception 'NSInvalidArgumentException', reason: '-[__NSCFConstantString newSizeWithFont: details omitted]'

'NSInvalidArgumentException', reason: '-[__NSCFConstantString newSizeWithFont: the following details omitted]: unrecognized selector sent to instance 0x[random number]'

See the figure below:



The above problem occurs because the place with an addition of –ObjC macro is not correctly configured. If –ObjC configuration problem cannot be resolved, try eliminating –ObjC and substituting it with -force_load+space character+control path, e.g., -force_load $(PROJECT_DIR)/ libPaymentControl.a.

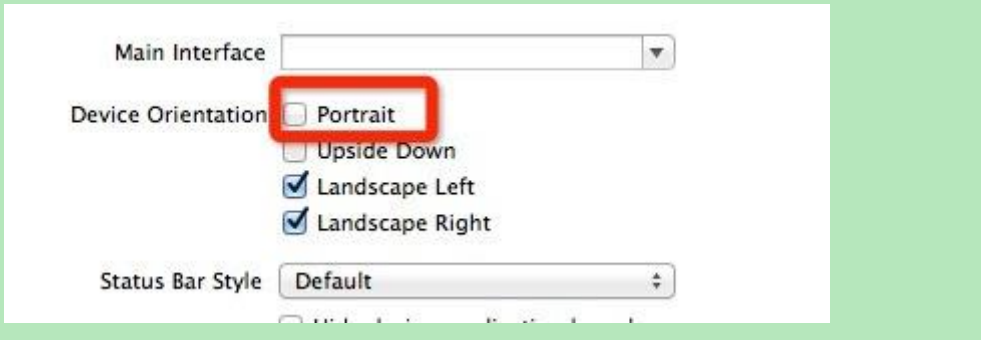
6.4 Orientation exception

The detailed phenomenon is control breakdown, and error message is: Terminating app due to uncaught exception ‘UIApplicationInvalidInterfaceOrientation’, reason: ‘Supported orientations has no common orientation with the application, and shouldAutorotate is returning YES’

Please don’t change device orientation setting. Iphone control can support portrait orientation only, while ipad control supports both landscape and portrait orientation.

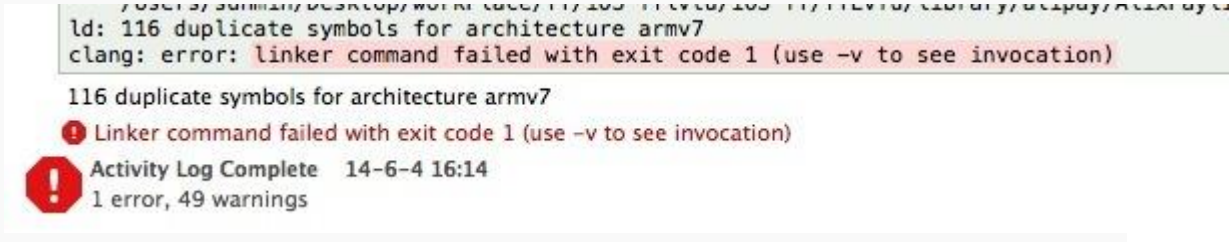
If app itself is landscape, please control landscape and portrait orientation via code and do

not change the configuration.



6.5 Conflict among files of other third-party static libraries (e.g., alipay and WeChat) duplicate symbols for architecture XXX

For instance:



Please try modifying the place with addition of –ObjC macro as -force_load+space character+control path, e.g., -force_load $(PROJECT_DIR)/ libPaymentControl.a. Such operation does not affect the functionality of other payment channels like alipay.

6.6 Indefinite loading of control interface

1. The logic of invocation control shall be executed in the main thread, and it is best not to conduct UI operation in a sub-thread. Indefinite loading wait is actually caused by sub-thread deadlock.
2. Please refer to “Add –ObjC macro” in the section of “Add SDK kit” in the Control Use Guide iOS and it is specially noted that capital and small letters should not be mixed up. (If this operational step is problematic, the alternative correction may be: -force_load+space character+control path, e.g., -force_load $(PROJECT_DIR)/ libPaymentControl.a).
3. Build Setting -> Apple LLVM compiler Language-C++:

6.7 Attempt to present XX on XX while a presentation is in progress!

Warning: Attempt to present <UPNavController: 0x8bc24d0> on <tttViewController: 0x8b9a390> while a presentation is in progress!

Invocate our plug-in only upon completion of animation, and simultaneous appearance of two presents will trigger off a warning.