FLASHiZ
iPhone library module
Documentation

I. Context

The objective of this document is to describe the technical integration and usage of the "iPhone library" module for implementation by Flashiz' professional clients that need to integrate a payment function in their iOS applications.

The «iPhone library» has been set up by FLASHiZ to produce a solution of integration of the payment process in iOS Apps designed for iPhones. Clients iOS apps don't fit with the standard process (scan of a QR code) and therefore this library allows a payment without changing from the original app.

The «iPhone library» module has been developed as an intermediary between the client[1] IOS application and FLASHiZ' servers in native objectif C language

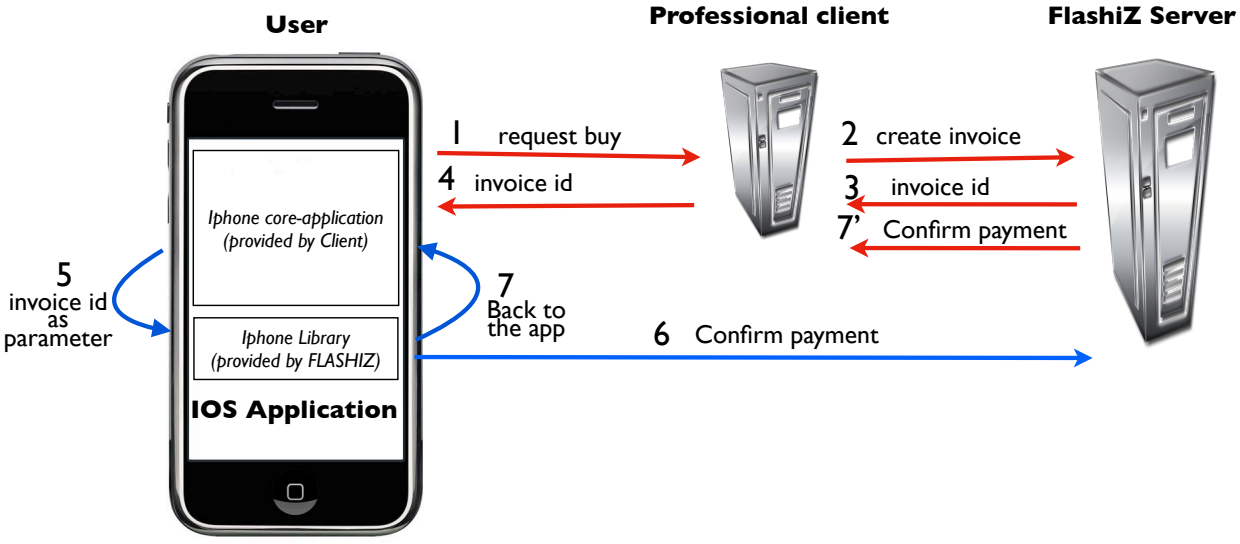
- to keep the level of security of the original FLASHiZ iOS app
- to authorize the definition and usage of PIN codes
- to facilitate the deployment (presented as a package).

This document presents FlashiZ iOS framework, specifically

- Framework installation
- API usage

II. Presentation

The process is carried out according to the following:



Red arrows in the diagram above are the ones concerned in the web API module (detailed in another document). The blue arrows are managed by «iPhone library» module that is treated this document.

1. The client iOS application send to the professional client server an invoice creation request.

[1] The client represents the professional that sells goods or services to a user

2.The professional client server transmits request to the Flashiz server to initiate the transaction (standard API).

3.The Flashiz server returns an invoice id to the professional client server

4.The professional client server transmits the invoice id to the client iOS application

5.The client iOS application makes use of the invoice id as a parameter to Flashiz «iPhone library»

6.The Flashiz «iPhone library» creates the client process acceptance (user's authentication and confirmation)

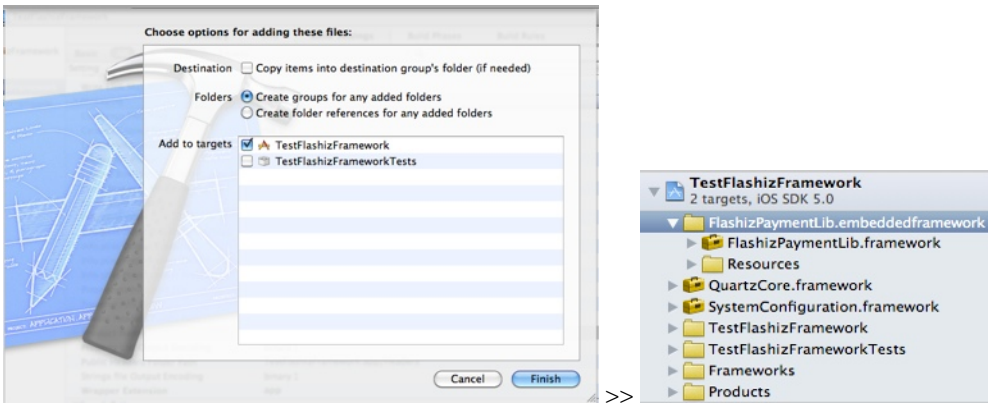
7.The Flashiz «iPhone library» adverts the application of the success or cancellation of the order

7'. The Flashiz server confirms the payment the professional client server (standard API)

III. Framework installation

The installation of the framework is performed in two steps:

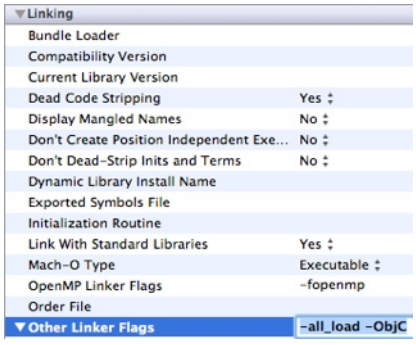
- Drag & drop the **FlashizPaymentLib.embeddedFramework** directory (contained in the provided zip file) on the targeted project. The framework files can optionally be copied in the project directory if needed.



Remark : The framework should appear as a directory in the project folder, containing a standard Framework.

- To build correctly, it is required to add the following option to the linker command line: -all_load –ObjC.

This is done via the "Other Linker Flags" of the Linking category in the Build Settings.



- all libraries are included in the package itself, you don't have to add them but the Framework needs other Frameworks to work properly: QuartzCore and SystemConfiguration

IV. API usage

A. *Overview*

The payment process is performed via two important Objective-C items:

- FlashizFacade : It is the main object of the API.
- FlashizPaymentDelegate : Protocol allowing the calling application to get events on the payment process

Here is the definition of the FlashizFacade object.

```
@property(nonatomic,retain) NSString* invoiceId;
-(id) init;
-(id) initWithEnvironment:(FlashizEnvironment) env;
-(id) initWithEnvironment:(FlashizEnvironment) env lang:(NSString*)lang;
-(void) executePaymentForInvoiceId:(NSString*) invoiceId;
@end
```

The facade object has to be initialized on a defined FLASHiZ environment (Flashiz Server). Two environments are available:

- FE_PRODUCTION: production environment
- FE_TEST: test environment (corresponding to the pre-production environment).

The init method without argument initializes the API on the production environment.

The calling application can register for payment events with the following delegate:

```
@protocol FlashizPaymentDelegate <NSObject>
-(void) paymentAcceptedForInvoice:(NSString*) invoiceId;
-(void) paymentCanceledForInvoice:(NSString*) invoiceId;
@end
```

The registering can be done by setting the delegate property of the façade object:

```
flashizFacade.delegate = self;
```

To be able to integrate with the calling application, the parent view controller has to be defined through the parentViewController object:

```
facade.parentViewController = self;.
```

B. Complete example

```
FlashizFacade* facade = [[FlashizFacade alloc] initWithEnvironment:FE_TEST];
facade.parentViewController = self;
facade.delegate = self;
self.flashizFacade = facade;
 [facade executePaymentForInvoiceId:invoiceId];
 [facade release];
```

C. Example to force another language

```
FlashizFacade* facade = [[FlashizFacade alloc] initWithEnvironment:FE_TEST lang:@"fr"];
facade.parentViewController = self;
facade.delegate = self;
self.flashizFacade = facade;
 [facade executePaymentForInvoiceId:invoiceId];
 [facade release];
```

Language codes supported :

en (English)
fr (French)
de (German)
es (Spanish)