FIO SDK for iOS

Software Design Specification

|  |  |
| --- | --- |
| Document Number: | 1.0 |
| Creation Date: | October 24, 2018 |
| Author: | Shawn Arney |
| Last Updated: | October 24, 2018 |
| Version: | 1.0.1 |
| File Name: | FIOSDK\_IOS.docx |



Dapix, Inc

Denver, CO

Dapix.io

Table of Contents

1. About This Document 3
   1. Document History 3
   2. Reference Documentation 3
   3. Approval History 3
   4. Keywords 3
2. SDK Conventions and Installation 4
   1. SDK Source Code Workspace and Projects 4
   2. SDK Coding Methodology and Conventions 4
      1. Class Conventions 4
   3. Installing the SDK 5
      1. Building the SDK Framework 5
      2. Adding the FIOSDK.framework to an existing iOS Project 5
3. SDK Methods 8
   1. Initialization before usage 8
      1. Parameters 8
      2. Examples 8
   2. isFioNameValid 8
      1. Parameters 8
      2. Method Returned Value 8
      3. Examples 8
   3. getAddressByFioName 9
      1. Parameters 9
      2. Returned Objects 9
      3. Examples 9
   4. getFioNameByAddress 10
      1. Parameters 10
      2. Returned Objects 10
      3. Examples 10
   5. registerFioName 11
      1. Parameters 11
      2. Returned Objects 11
      3. Examples 11

# About This Document

This document describes the methodology for implementing the FIO SDK in an iOS application.

## Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Author | Date | Description |
| 0.1 | Shawn Arney | October 24, 2018 | Initial version |

## Reference Documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Title | Author | Ref. Number | Revision | Creation Date |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## Approval History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Approval Authority | Approval Status | Approval Date |
| 0.1 |  | No version currently approved |  |

## Keywords

|  |  |
| --- | --- |
| Word | Description |
| Currency Code | This is the Currency/Token Code. i.e. “BTC”, “ETH”, “FIO” |

# SDK Conventions and Installation

## SDK Source Code Workspace and Projects

The FIOSDK workspace is comprised of two projects. There is a sample app with code linking to the FIO SDK framework to demonstrate its usage. And the FIO SDK framework project. There are also unit tests available within the FIO SDK framework project, testing all functionality of the SDK.

|  |  |  |
| --- | --- | --- |
|  | File Name | Description |
| FIOSDK Workspace | FIOSDK.xcworkspace | The Workspace for the FIOSDK |
| FIOSDK | FIOSDK.xcodeproj | The FIOSDK Framework Project |
| FIOSDK Sample App | FIOSDKSample.xcodeproj | The FIOSDK Sample App |

## SDK Coding Methodology and Conventions

The FIO SDK Coding Methodology and Conventions follow swift patterns and conventions. For usage in Swift language based iOS applications.

### Class Conventions

Singleton’s are used for accessing SDK functionality. Using the sharedInstance() convention.

2.1.2.Examples

FIOSDK.sharedInstance().isFioNameValid(fioName: "test.brd")

2.1.3.Caller Style

Method calls are asynchronous and follow the swift closure style for these calls.

2.1.4.Examples

FIOSDK.sharedInstance().getFioNameByAddress(publicAddress: receiveAddress,

currencyCode:"ETH", completion: {result, error in ()

print (result.name)

})

2.1.3.Error Handling

Errors are returned within the closure for asynchronous methods. These are returned in the error object within the closure. The type of error and status can be determined by evaluating the error?.kind property.

2.1.2.Examples

if (error?.kind == FIOError.ErrorKind.Success){

print(“success”)

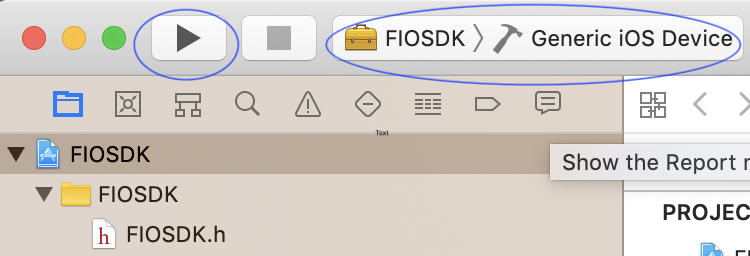
}

## Installing the SDK

### Building the SDK Framework

The FIO SDK can be built by opening up the FIO SDK project file in Xcode. And building the FIOSDK project. Make sure to select “Generic iOS Device” for the build only device. Follow these steps to build:

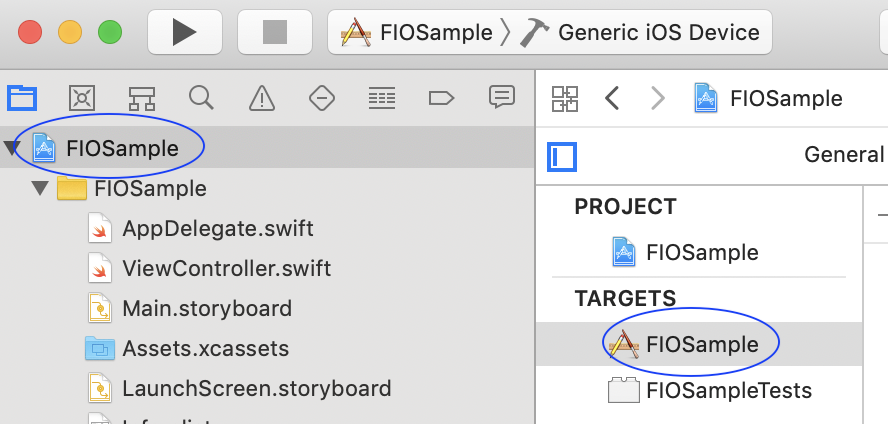
1. Open the FIOSDK project in xCode
2. Select the “Generic iOS Device” from the dropdown for build only device
3. Build the FIOSDK project by clicking the Run Icon
4. The FIOSDK.framework file will be built and ready for use, in the “\FIOSDK\” folder.



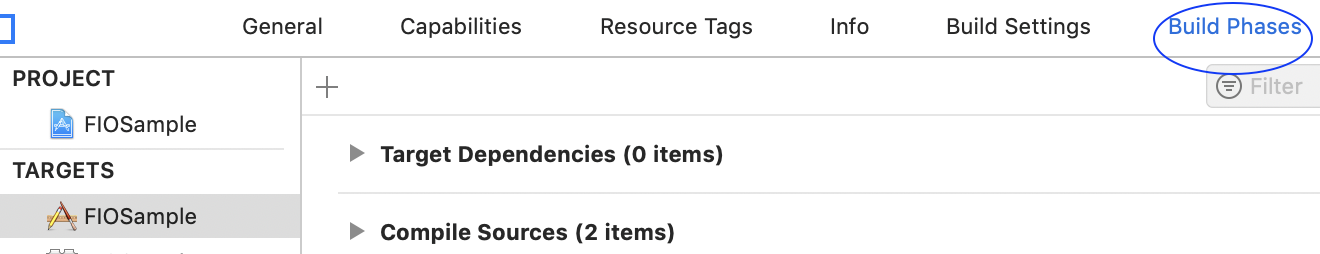
### Adding the FIOSDK.framework to an existing iOS Project

The FIOSDK built above can be added to an existing iOS project. By following these steps:

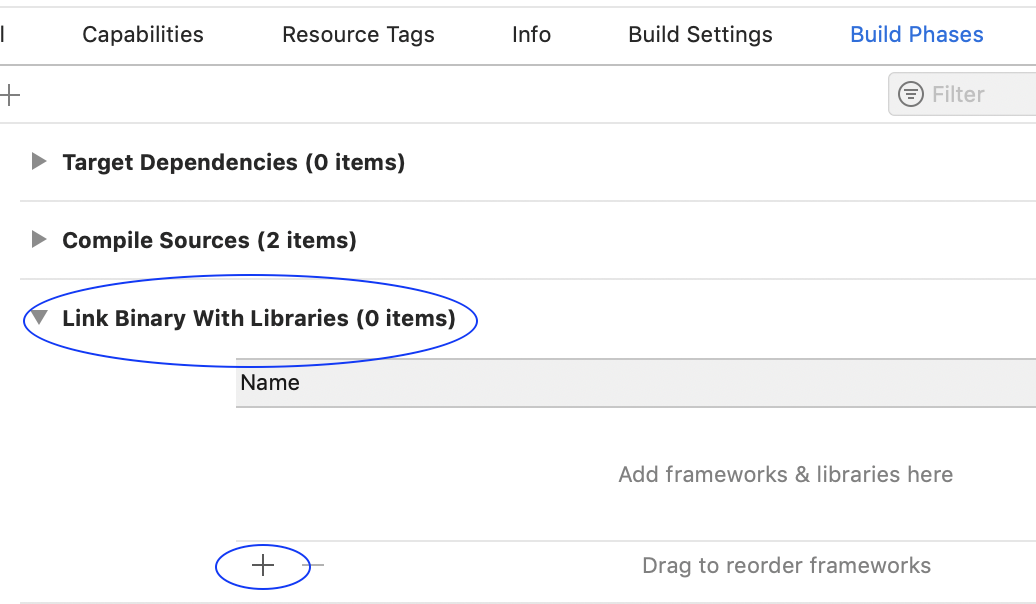
1. Open your existing iOS Project in Xcode, to add the SDK.
2. Navigate to your project file on the left pane in Xcode, by selecting the “Show Project Navigator” icon and then selecting your project file on the list on the left pane.
3. Select your target application, as listed under targets, on the right pane.



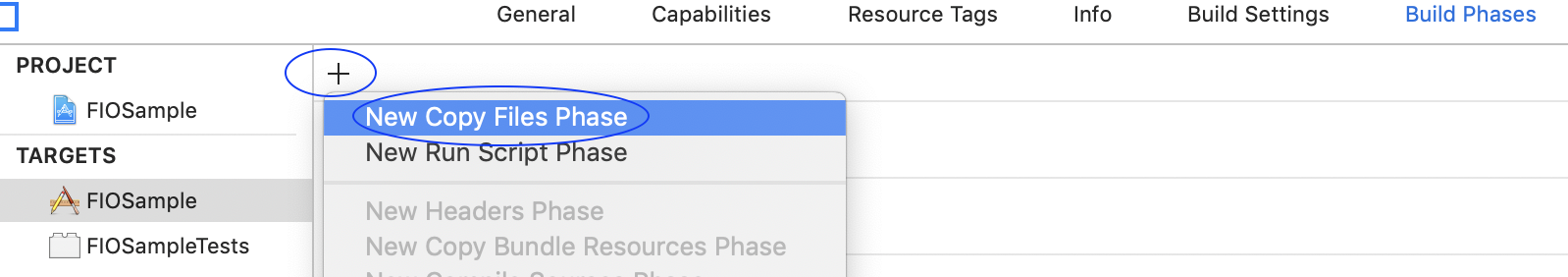
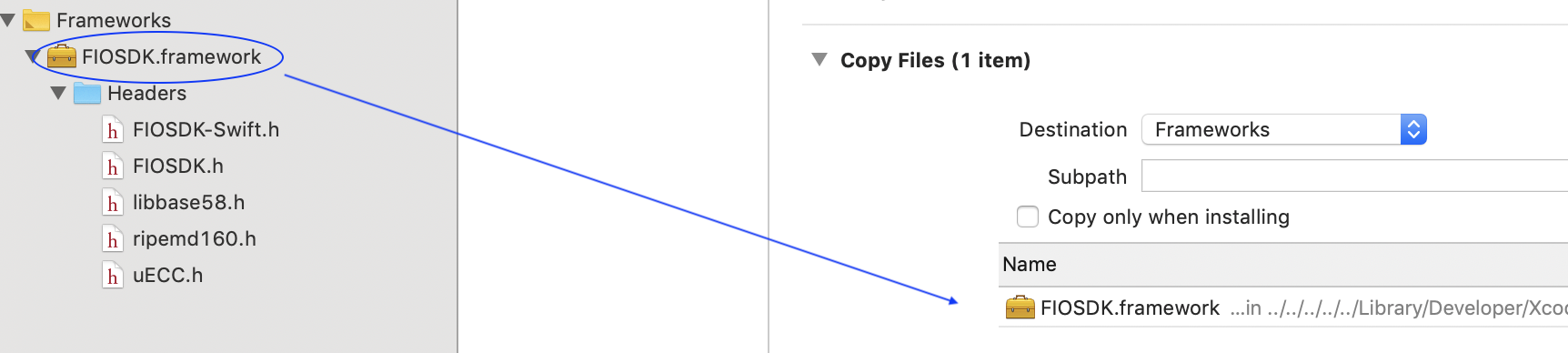
1. Select Build Phases tab, from the right pane



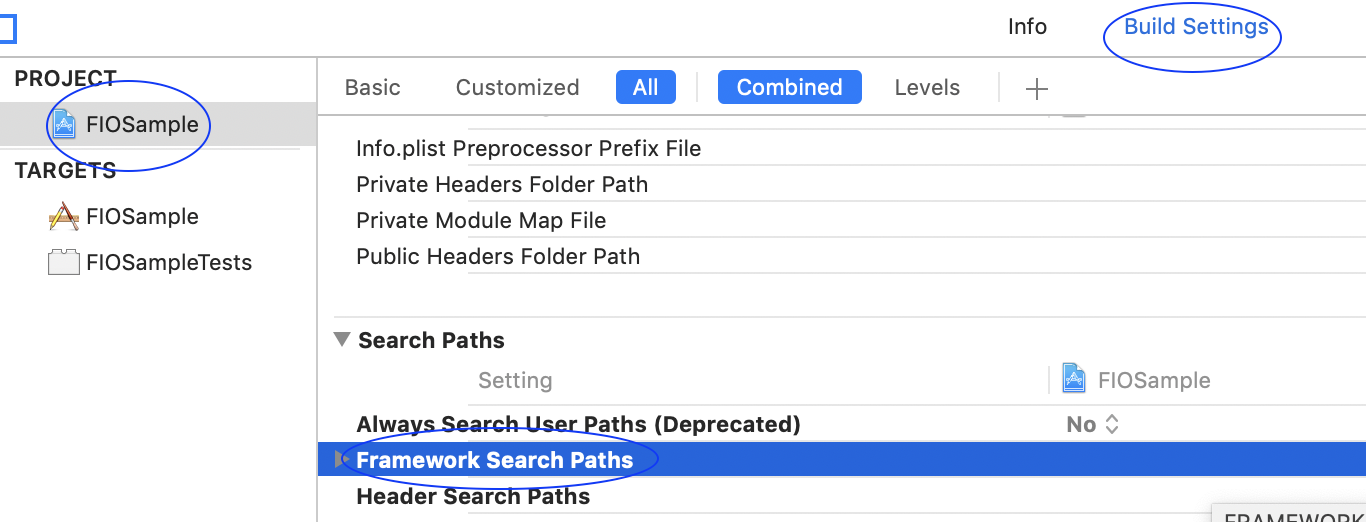
1. Select the “Link Binary With Libraries” option, then select the “+” option to add the SDK



1. Choose the “Add Other” button, to select and choose the “FIOSDK.framework” file
2. The FIOSDK.framework should now be available on the left pane with your project.
3. Create a “New Copy File Phase” by click the “+” button on the right pane. And selecting “New Copy Files Phase” option. Make sure to select “Frameworks” as the Destinaion. And then drag the FIOSDK.framework file from the left pane to the right pane.



1. Point your “Framework Search Paths” to the FIOSDK.framework path. By Selecting the project on the right pane and under the “Build Settings” tab, select the “Framework Search Paths”



# SDK Methods

## Initialization before usage

This method is used to initialize the FIO SDK sharedInstance with the SDK’s account name, private key, and FIO url. This is typically done at application startup. And these values can be dynamically changed at anytime, using any sharedInstance method call.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| accountName | Account Name | This is the account name used for a distinct wallet application. This doesn’t change for different users of the app. It is unique per wallet application. |
| privateKey | Private Key | This is the private key used for a distinct wallet application. This doesn’t change for different users of the app. It is unique per wallet application. |
| url | Url | This is the URL to the FIO Block chain. |

### Examples

The following example initializes the FIO SDK

\_ = FIOSDK.sharedInstance(accountName: "exchan532",

privateKey: "5KDQzVMaD19UdYsrA2PNK3qEP7zNbUf8D41ZVKqGzZ117PdM5Ap",

url: "http://9.14.221.124:8889/v1")

## isFioNameValid

This method determines if the FIO Name is Valid. Checking to see if the name follows the standard rules for FIO names, such as correct length, and valid characters. This DOES NOT check to see if the FIO Name is already registered. Please use the getAddressByFioName() method for determining previous registrations.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioName | FIO Name | This is the FIO Name to test for validity. |

### Method Returned Value

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| Returned Value | Boolean | Returns true if valid, false if NOT valid |

### Examples

The following example uses the Fio Name parameter to determine if the fio name is valid.

let isValid = FIOSDK.sharedInstance().isFioNameValid(fioName: “test.brd")

## getAddressByFioName

This method gets the public address that is associated with a Fio name and currency/token code.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioName | FIO Name | This is the FIO Name associated with the public address to be retrieved. |
| currencyCode | Currency Code | This is the currency/token code, associated with the public address to be retrieved. i.e. “FIO”,”BTC”,”ETH” |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | AddressByNameResponse | AddressByNameResponse struct with the following properties   * address :String - the public receive address * isRegistered :Bool - is the Fio Name registered? * isDomain :Bool - is this a domain? |
| error | FIOError | FIOError object with the following properties   * kind :ErrorKind - the type of error. i.e. success, failure * localizedDescription :String - error description |
|  |  |  |

### Examples

The following example gets the address associated with the Fio name and currency/token.

FIOSDK.sharedInstance().getAddressByFioName(fioName: "myfioname.brd", currencyCode: "ETH", completion: { results, error in ()

if (error?.kind == FIOError.ErrorKind.Success){

DispatchQueue.main.async {

self.receiveAddress.text = results.address

}

}

}})

## getFioNameByAddress

This method gets the FIO name that is associated with a public address and currency/token code.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| publicAddress | Public Address | This is the Public Address associated with the FIO Name to be retrieved. |
| currencyCode | Currency Code | This is the currency/token code, associated with the FIO Name to be retrieved. i.e. “FIO”,”BTC”,”ETH” |
|  |  |  |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | NameByAddressResponse | NameByAddressResponse struct with the following properties   * name :String - the Fio Name |
| error | FIOError | FIOError object with the following properties   * kind :ErrorKind - the type of error. i.e. success, failure * localizedDescription :String - error description |
|  |  |  |

### Examples

The following example gets the FIO Name associated with the address and currency/token.

FIOSDK.sharedInstance().getFioNameByAddress(publicAddress: “0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”, currencyCode: "ETH", completion: {result, error in ()

print (result.name)

})

## registerFioName

This method registers a FIO Name. And adds public addresses and currency/token codes that are associated with this newly registered FIO Name.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioName | FIO Name | This is the FIO Name to be registered. |
| publicReceiveAddresses | Dictionary of Address and Currency Code | This is a dictionary of public receive addresses and currency/token codes. To be associated with the FIO Name being registered.  The Format is:  Dictionary<String,String>  i.e.  Dictionary of Addresses and Currency Codes  example  [“0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”:”ETH"] |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| error | FIOError | FIOError object with the following properties   * kind :ErrorKind - the type of error. i.e. success, failure * localizedDescription :String - error description |
|  |  |  |

### Examples

The following example registers a new Fio Name with the associated public receive address and currency code.

let addresses:Dictionary<String,String> = [“0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”:"ETH"]

FIOSDK.sharedInstance().registerFioName(fioName: ("mynewfioname.brd"), publicReceiveAddresses: addresses, completion: {error in ()

if (error?.kind == FIOError.ErrorKind.Success){

print (“successful”)

}

})