FIO SDK for iOS

Software Design Specification

|  |  |
| --- | --- |
| Document Number: | 1.0 |
| Creation Date: | October 24, 2018 |
| Author: | Shawn Arney |
| Last Updated: | April 09, 2019 |
| Version: | 1.0.2 |
| File Name: | FIOSDK\_IOS.docx |



Dapix, Inc

Denver, CO

Dapix.io

Table of Contents

1. About This Document 4
   1. Document History 4
   2. Reference Documentation 4
   3. Approval History 4
   4. Keywords 4
2. SDK Conventions and Installation 5
   1. SDK Source Code Workspace and Projects 5
   2. SDK Coding Methodology and Conventions 5
      1. Class Conventions 5
   3. Installing the SDK 6
      1. Building the SDK Framework 6
      2. Adding the FIOSDK.framework to an existing iOS Project 6
3. SDK Methods 9
   1. Initialization before usage 9
      1. Parameters 9
      2. Examples 9
   2. isFioNameValid 9
      1. Parameters 10
      2. Method Returned Value 10
      3. Examples 10
   3. getPublicAddress 10
      1. Parameters 10
      2. Returned Objects 10
      3. Examples 11
   4. getFioNames 11
      1. Parameters 11
      2. Returned Objects 11
      3. Examples 11
   5. registerFioName 12
      1. Parameters 12
      2. Returned Objects 12
      3. Examples 12
   6. isAvailable 13
      1. Parameters 13
      2. Returned Objects 13
      3. Examples 13
   7. privatePubKeyPair 13
      1. Parameters 14
      2. Returned Objects 14
      3. Examples 14
   8. wipePrivPubKeys 14
      1. Examples 14
   9. addPublicAddress 14
      1. Parameters 15
      2. Returned Objects 15
      3. Examples 15
   10. getPendingFioRequests 15
       1. Parameters 16
       2. Returned Objects 16
       3. Examples 16
   11. getFIONameDetails 17
       1. Parameters 17
       2. Returned Objects 17
       3. Examples 17
   12. getTokenPublicAddress 17
       1. Parameters 18
       2. Returned Objects 18
       3. Examples 18
   13. requestFunds 18
       1. Parameters 19
       2. Returned Objects 19
       3. Examples 19
   14. rejectFundsRequest 19
       1. Parameters 20
       2. Returned Objects 20
       3. Examples 20
   15. getSentFioRequests 20
       1. Parameters 20
       2. Returned Objects 20
       3. Examples 21
   16. recordSendAutoResolvingWith 21
       1. Parameters 22
       2. Returned Objects 22
       3. Examples 23
   17. recordSend 23
       1. Parameters 23
       2. Returned Objects 23
       3. Examples 24
   18. getFIOPublicAddress 24
       1. Returned Objects 24
       2. Examples 25
   19. getFIOBalance 25
       1. Parameters 25
       2. Returned Objects 25
       3. Examples 25
   20. transferFIOTokens 25
       1. Parameters 26
       2. Returned Objects 26
       3. Examples 26

# 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 |
| 0.2 | Shawn Arney | December 14, 2018 | Addition of call |
| 0.3 | Vitor Navarro | April 09, 2019 |  |

## Reference Documentation

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

## Approval History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Approval Authority | Approval Status | Approval Date |
| 0.2 |  | No version currently approved |  |

## Keywords

|  |  |
| --- | --- |
| Word | Description |
| Currency Code | This is the Currency/Token Code. i.e. “BTC”, “ETH”, “FIO” |
| FIO Address | A easy-to-remember human name to public addresses on a specific blockchain. i.e. john.fio, sara.brd |
| FIO Domain | The domain portion of the FIO Address, i.e. john.**fio**,sara.**brd** |
| FIO Name | Either FIO Domain or FIO Address |

# 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”)

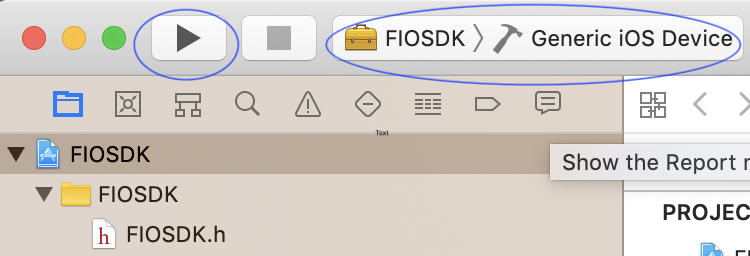
}

## 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:

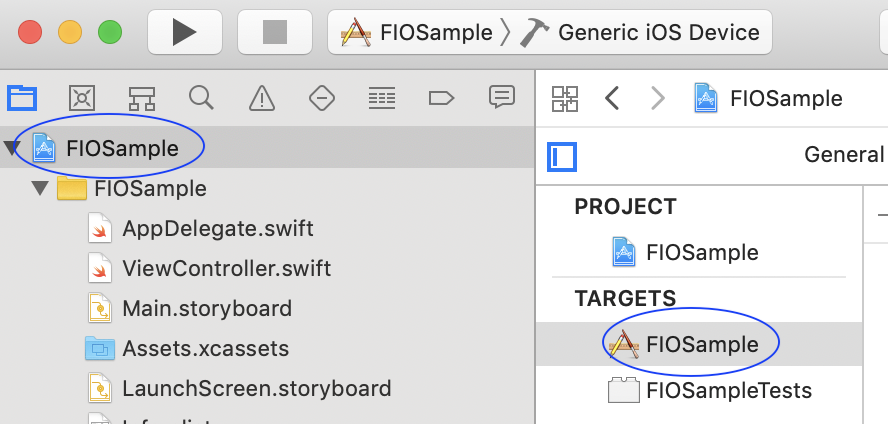
1. Open the FIOSDK project in xCode
2. Select the “Generic iOS Device” from the dropdown for build only device
3. Build the FIOSDK project by clicking the Run Icon
4. The FIOSDK.framework file will be built and ready for use, in the “\FIOSDK\” folder.



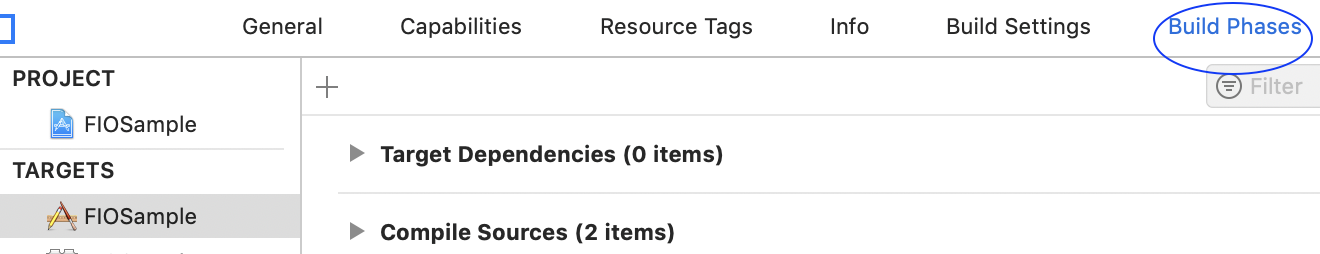
### Adding the FIOSDK.framework to an existing iOS Project

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

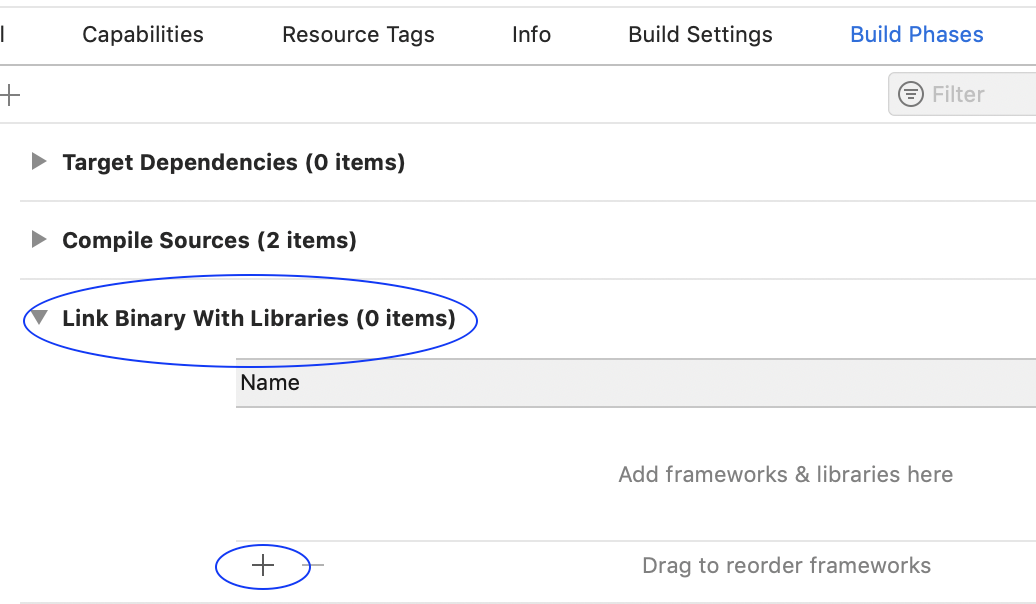
1. Open your existing iOS Project in Xcode, to add the SDK.
2. Navigate to your project file on the left pane in Xcode, by selecting the “Show Project Navigator” icon and then selecting your project file on the list on the left pane.
3. Select your target application, as listed under targets, on the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



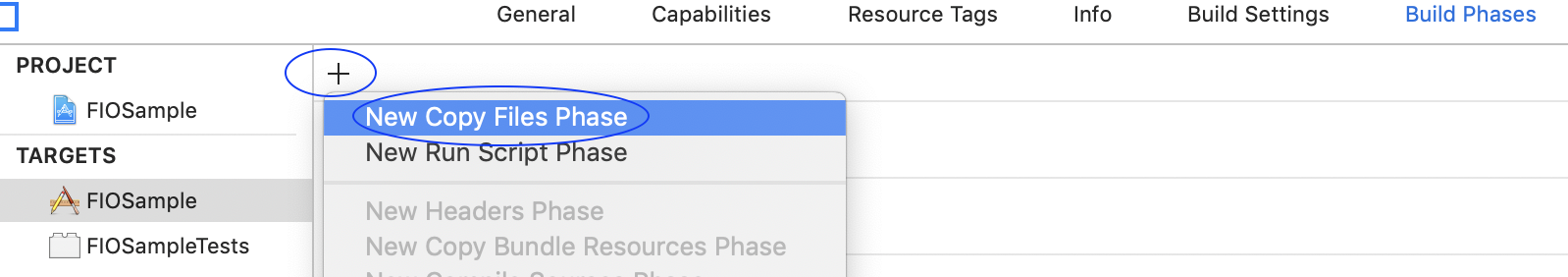
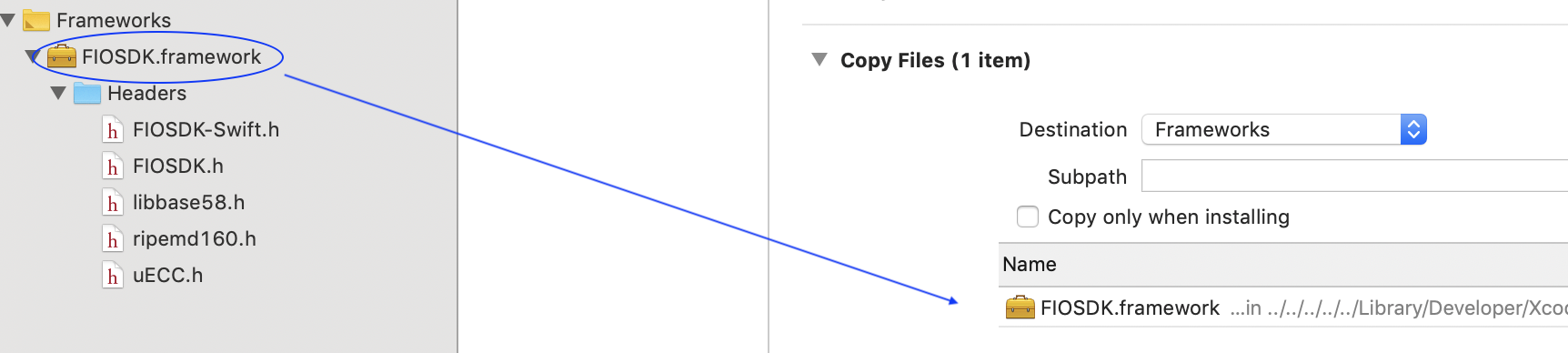
1. Select Build Phases tab, from the right pane



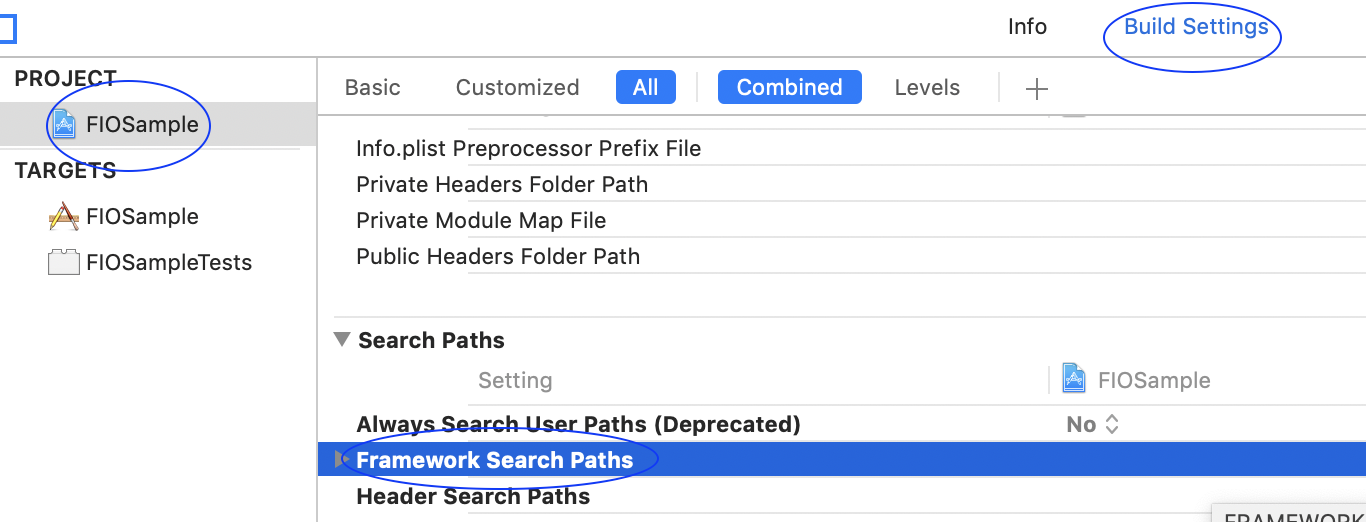
1. Select the “Link Binary With Libraries” option, then select the “+” option to add the SDK



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. |
| publicKey | Public Key | This is the public 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. |

**Note**

Private and public key have a very specific format for working, I.e 5KDQzVMaD1iUdYDrA2PNK3qEP7zNbUf8D41ZVKqGzZ117PdM5Ap/EOS6D6gSipBmP1KW9SMB5r4ELjooaogFt77gEs25V9TU9FrxKVeFb. To garantee proper setup we provide the method privatePubKeyPair(), described in section 3.7.

### Examples

The following example initializes the FIO SDK

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

privateKey: "5KDQzVMaD19UdYsrA2PNK3qEP7zNbUf8D41ZVKqGzZ117PdM5Ap",

publicKey: “EOS6D6gSipBmP1KW9SMB5r4ELjooaogFt77gEs25V9TU9FrxKVeFb",

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")

## getPublicAddress

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

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioAddress | FIO Address | This is the FIO Address associated with the public address to be retrieved. |
| tokenCode | Token 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 | PublicAddressResponse | PublicAddressResponse struct with the following properties   * fioAddress :String - The FIO Address sent * tokenCode :String - The Token code sent * publicAddress :String - the public address associated with the FIO Address and Token Code |
| 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().getPublicAddress(fioAddress: "myfioname.brd", currencyCode: "ETH", completion: { results, error in ()

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

DispatchQueue.main.async {

self.receiveAddress.text = results.address

}

}

}})

## getFioNames

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

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| publicAddress | Public Address | This is the Public Address associated with the FIO Names to be retrieved. |
|  |  |  |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | FioNamesResponse | FioNamesResponse struct with the following properties   * publicAddress: String - The address to which was requested the results * domains: [FioDomainResponse] - A list of domains and their expiration times * Addresses: [FioAddressResponse] - A list of FIO addresses and their expiration times |
| 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 Names associated with an address.

FIOSDK.sharedInstance().getFioNames(publicAddress: “0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”, completion: {result, error in ()

print (result.publicAddress)

print (result.domains)

print (result.addresses)

})

## 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. i.e. john.brd |
| 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  [”ETH”:“0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”] |

### 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> = [“ETH":“0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”]

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

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

print (“successful”)

}

})

## isAvailable

This method determines if a FIO Address or FIO Domain exists.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioAddress | FIO Address | This is the FIO Address or FIO Domain to check |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| isAvailable | Bool | Is the FIO Address or FIO Domain Registered? |
| 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 checks to see if a passed in FIO Address is already registered and available.

FIOSDK.sharedInstance().isAvailable(fioAddress:"mynewfioname.brd") { (isAvailable, error) in

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

if (isAvailable == true){

print ("is available")

}

}

}

## privatePubKeyPair

This method creates a private and public key based on a mnemonic, it does store both keys in keychain. Use it to setup FIOSDK properly.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| forMnemonic | Mnemonic | The text to use in private/public key generation. |
|  |  |  |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | (privateKey, publicKey) | A tuple containing both private and public keys to be used in FIOSDK setup. |
|  |  |  |

### Examples

The following example generate private and public keys.

let keyPair = FIOSDK.privatePubKeyPair(forMnemonic: “a seed phrase goes here")

print(keyPair.privateKey)

print(keyPair.publicKey)

## wipePrivPubKeys

This method remove private and public keys from keychain. It may throw keychain access errors while doing so.

### Examples

The following example wipes private public keys.

FIOSDK.wipePrivPubKeys()

## addPublicAddress

Assign a public address for a combination of FIO Address and currency/token code.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioAddress | FIO Address | The FIO Address to assign the public address to. i.e. john.fio |
| chain | Currency/Token Code | The currency code to assign the public address to. i.e. ETH, FIO |
| publicAddress | Public Address | The public address to be assign to the combination of FIO Address + Token Code |

### 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 add a public address to a FIO Address and Token Code.

FIOSDK.sharedInstance().addPublicAddress(fioAddress: “john.brd”, chain: “ETH”, publicAddress: “0xc39b2845E3CFAdE5f5b2864fe73f5960B8dB473B”) { error in

if error.kind == .Success { print(“Success”) }

else { print(“Error”) }

}

## getPendingFioRequests

Pending requests call polls for any pending requests sent to a receiver.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioPublicAddress | Public Address | The Public Address to which is desired a fetch of the pending requests. |
|  |  |  |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | PendingFioRequestsResponse | PendingFioRequestsResponse struct with the following properties   * fioPubAdd: String - The address to which was requested the results * requests: [PendingFioRequest] - A list of pending requequests   + PendingFioRequest:     - fundsRequestId: String - The request ID     - fromFioAddress: String - The FIO Address that is sending funds     - toFioAddress: String - The FIO Address that is receiving funds     - toPublicAddress: String - The Public Address connected to the token that will receive the funds.     - amount: String - The value being transfered     - tokenCode: String - Currency that is being used. i.e. ETH, FIO, BTC     - metadata: Metadata - A model containing notes related to the transfer request.     - timeStamp: TimeInterval: Time of the request. |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().getPendingFioRequests(fioPublicAddress: fioPublicAddress) { (pendingRequests, error) in

guard error?.kind == FIOError.ErrorKind.Success, let pendingRequests = pendingRequests, !pendingRequests.requests.isEmpty else {

return

}

print(pendingRequets)

}

## getFIONameDetails

Return details regarding a given FIO Name.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioAddress | FIO Address | FIO Address for which to get details for. john.fio |
|  |  |  |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | FioAddressResponse | FioAddressResponse struct with the following properties   * address: String - Same address as the one provided * expiration: Date - Expiration date of registered FIO Address |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().getFIONameDetails(fioAddress: “john.fio”) { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response.address)

print(response.expiration)

}

## getTokenPublicAddress

Returns a public address for the specified token registered under a FIO public address.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| forToken | Currency/Token Code | Token code for which public address is to be returned, e.g. "ETH". |
| withFIOPublicAddress | FIO Public Address | The public address registered for FIO Address. |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | TokenPublicAddressResponse | TokenPublicAddressResponse struct with the following properties   * fioAddress: String - The FIO Address for the provided FIO Public Address * tokenPublicAddress: String - The Currency/Token Code public address |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().getTokenPublicAddress(forToken: “BTC”, withFIOPublicAddress: "htjonrkf1lgs") { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response.fioAddress)

print(response.tokenPublicAddress)

}

## requestFunds

Creates a new funds request from one FIO Address to another FIO Address.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| from | FIO Address | The FIO Address from funds sender |
| to | FIO Address | The FIO Address from funds receiver |
| toPublicAddress | Public Address | Public address of user receiving funds. |
| amount | Currency amount | Value to be requested. |
| tokenCode | Currency/Token Code | Currency/Token code of the currency being requested. |
| metadata | Extra information | Contains the: memo or hash or offlineUrl (they are mutually excludent, fill only one) |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | RequestFundsResponse | RequestFundsResponse struct with the following properties   * fundsRequestId: String - The request ID |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().requestFunds(from: “john.fio”, to:”sara.fio”, toPublicAddress: “htjonrkf1lgs”, amount: “10.2542”, tokenCode: “FIO”, metadata: RequestFundsRequest.MetaData(memo: “memo”)) { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response?.fundsRequestId)

}

## rejectFundsRequest

Reject a funds request made from one FIO Address to another FIO Address.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fundsRequestId | Funds Request ID | The ID of created funds request to be rejected. |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | RejectFundsRequestResponse | RejectFundsRequestResponse struct with the following properties   * fioReqID: String - The rejected funds request ID * status: Status - Either rejected or unknown |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().rejectFundsRequest(fundsRequestId: “10”) { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response?.status)

}

## getSentFioRequests

Get all requests sent by the given public address. Usually made with requestFunds.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| publicAddress | FIO Public Address | FIO public address which to query for sent requests. |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | SentFioRequestResponse | SentFioRequestResponse struct with the following properties   * fioPublicAddress: String - The public address used to get sent requests. * requests: [SentFioRequest] - The list of sent requests   + SentFioRequest     - fundsRequestId: String - The request ID     - fromFioAddress: String - The sender FIO Address     - toFioAddress: String - The receiver FIO Address     - toPublicAddress: String - The receiver public address     - amount: String - The currency amount requested     - tokenCode: String - The currency/token code to which request was sent     - metadata: MetaData - Extra info about the request     - timeStamp: TimeInterval - The request time stamp     - status: String - The request status |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().getSentFioRequests(publicAddress: “htjonrkf1lgs”) { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response)

}

## recordSendAutoResolvingWith

Register a transaction on blockhain, it does auto resolve from (requestor) FIO address and to (requestee) token public address parameters. Must be called after any transaction when recordSend is already not called.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| toFIOAdd | FIO Address | FIO address that is receiving currency. (requestee) |
| andFromPubAdd | From FIO Public Address | FIO public adress from user that sent funds |
| amountSent | Currency Amount Sent | The amount sent from one user to another |
| forTokenCode | Currency/Token Code | The currency/token code for that transaction. i.e “FIO”, “ETH”, “BTC" |
| obtID | Other Blockchain transaction | The transaction ID (OBT) representing the transaction from one blockchain to another one. |
| fioReqID | FIO Request ID | The FIO request ID to register the transaction for. Only required when approving transaction request. |
| memo | Memo | The note for the transaction |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | RecordSendResponse | RecordSendResponse struct with the following properties   * fioObtID: String - * fromFIOAdd: String - * toFIOAdd: String - * fromPubAdd: String - * toPubAdd: String - * amount: String - * tokenCode: String - * chainCode: String - * status: String - * obtID: String - * metadata: MetaData - * fioReqID: String? - |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().recordSendAutoResolvingWith(toFIOAdd: "john.fio", andFromPubAdd: "hjrtu45dos45", amountSent: 23.4567, forTokenCode: "FIO", obtID: "0xFF", fioReqID: "10", memo: "Sample memo") { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response)

}

## recordSend

Register a transaction on blockhain, it does auto resolve from (requestor) FIO address and to (requestee) token public address parameters. Must be called after any transaction when recordSendAutoResolvingWith is already not called.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioReqID | FIO Request ID | The FIO request ID to register the transaction for. Only required when approving transaction request. |
| fromFIOAdd | From FIO Address | FIO address that is sending currency. (requestor) |
| toFIOAdd | FIO Address | FIO address that is receiving currency. (requestee) |
| fromPubAdd | From FIO Public Address | FIO public address related to the token code being sent by from user (requestor) |
| toPubAdd | To FIO Public Address | FIO public address related to the token code being received by to user (requestee) |
| amount | Currency Amount Sent | The amount sent from one user to another |
| fromTokenCode | Currency/Token Code | The currency/token code sent in transaction. i.e “FIO”, “ETH”, “BTC" |
| toTokenCode | Currency/Token Code | The currency/token code received in transaction. i.e “FIO”, “ETH”, “BTC" |
| obtID | Other Blockchain transaction | The transaction ID (OBT) representing the transaction from one blockchain to another one. |
| memo | Memo | The note for the transaction |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | RecordSendResponse | RecordSendResponse struct with the following properties   * fioObtID: String - * fromFIOAdd: String - * toFIOAdd: String - * fromPubAdd: String - * toPubAdd: String - * amount: String - * tokenCode: String - * chainCode: String - * status: String - * obtID: String - * metadata: MetaData - * fioReqID: String? - |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().recordSend(fioReqID: "1",fromFIOAdd: "john.fio",toFIOAdd: "sara.fio",fromPubAdd: "dfr4hrty567d",toPubAdd: "jjhfrutyd34d",amount: 23.1234,fromTokenCode: "FIO",toTokenCode: "FIO",obtID: "0xFF",memo: "Sample memo") { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response)

}

## getFIOPublicAddress

The method returns current FIO pub address.

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
|  | FIO Public Address | The FIO Public Address string value. |
|  |  |  |

### Examples

The following example gets pending requests for a given FIO Public Address.

print(FIOSDK.sharedInstance().getFIOPublicAddress())

## getFIOBalance

Retrieve FIO balance tokens.

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| fioPublicAddress | FIO Request ID | The FIO public address to get FIO tokens balance for. |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | GetFIOBalanceResponse | GetFIOBalanceResponse struct with the following properties   * balance: String - The balance value in the float format. |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().getFIOBalance(fioPublicAddress: ”jjhfrutyd34d") { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response)

}

## transferFIOTokens

Transfer FIO tokens from current wallet to

### Parameters

|  |  |  |
| --- | --- | --- |
|  | Description | Discussion |
| toFIOPublicAddress | To FIO Public Address | The FIO public address that will receive funds. |
| amount | Currency Amount Sent | The amount being sent to that FIO public address |

### Returned Objects

|  |  |  |
| --- | --- | --- |
|  | Type | Description |
| result | TransferFIOTokensResponse | TransferFIOTokensResponse struct with the following properties   * status: String - The transfer status |
| 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 pending requests for a given FIO Public Address.

FIOSDK.sharedInstance().transferFIOTokens(toFIOPublicAddress:"dfr4hrty567d") { (response, error) in

guard error?.kind == FIOError.ErrorKind.Success else {

return

}

print(response)

}