Initializing Web3auth CoreKit tKey iOS SDK

Once you have installed the Web3Auth Core Kit tKey SDK, the next step is to initialize it. This involves a few steps, such as initiating the tKey SDK with the service provider and modules.

- [Configuration of Service Provider](#)
- [Initializing Service Provider](#)
- [Instantiation of tKey](#)

Configuring Service Provider[â](#)

Service Provider intKey generates a[Share A](#) , i.e., the private key share managed by a wallet service provider via their authentication flows. This share in our[wallet infrastructure](#) refers to the social login aspect, where we associate a private key share with the user's social login, enabling the seamless login experience.

To configure your service provider, you must use[CustomAuth Swift SDK](#) . Please note that this SDK is not automatically installed with tKey iOS SDK, so you must install it first.

Usage of CustomAuth Swift SDK

Installation[â](#)

Swift Package Manager[â](#)

Add the CustomAuth Swift SDK to your project[â](#)

1. In Xcode, with your app project open, navigate toFile > Add Packages
2. .
3. When prompted, add the CustomAuth Swift SDK repository:
4. https://github.com/torusresearch/customauth-swift-sdk
5. From theDependency Rule
6. dropdown, selectExact Version
7. and enter6.0.1
8. as the version.
9. When finished, Xcode will automatically begin resolving and downloading your dependencies in the background.

Cocoapods[â](#)

pod 'CustomAuth', '~> 6.0.1'

Initialization[â](#)

Initalize the SDK depending on the login you require.

import

CustomAuth

let sub =

SubVerifierDetails (loginType :

. web ,

// default .web loginProvider :

< typeOfLogin

,

// .google, clientId :

"" , verifierName :

"" , redirectURL :

"" , browserRedirectURL :

"")

let tdsdk =

CustomAuth (aggregateVerifierType :

"" ,

// singleLogin, singleIdVerifier supported aggregateVerifierName :

"" ,

// Web3Auth Custom verifier name subVerifierDetails :

[sub] , network :

. TESTNET)

SubVerifierDetails

[â](#)

Parameter Type Mandatory Description loginType SubVerifierType No loginType to be used. [web : default,installed] loginProvider LoginProviders Yes loginProvider to be used. [google ,facebook ,twitch ,reddit ,discord ,apple ,github ,linkedin ,kakao ,twitter ,weibo ,line ,wechat ,email_password , andjwt] clientId String Yes login provider's client Id. verifier String Yes Web3Auth verifier name redirectURL String Yes It refers to a url for the login flow to redirect into your app, it should have a scheme that is registered by your app, for examplecom.mycompany.myapp://redirect browserRedirectURL String No It refers to a page that the browser should use in the login flow, it should have a http or https scheme. e.g.https://scripts.toruswallet.io/redirect.html jwtParams String No Additional JWT parameters to be passed. urlSession URLSession No Custom URLSession to be used.

CustomAuth

[â](#)

Parameter Type Mandatory Description aggregateVerifierType String Yes Type of the aggregate verifier. aggregateVerifier String Yes Name of the aggregate verifier. subVerifierDetails [SubVerifierDetails] Yes Array of SubVerifierDetails. network Network Yes Network to be used. [MAINNET ,TESTNET ,CYAN ,AQUA]

Initializing Service Provider[â](#)

1. triggerLogin()
2. returns a promise that resolve with a Dictionary that contain at leastprivateKey
3. andpublicAddress
4. field.
5. Initialize the service provider with the privateKey retrived by the result oftriggerLogin()
6. .

tdsdk . triggerLogin (controller :

< UIViewController

    ? , browserType :

< method - of - opening - browser

    , modalPresentationStyle :

< style - of - modal

) . done { data in print ("private key rebuild" , data) let key = data ["privateKey"] service_provider =

try !

ServiceProvider (enable_logging :

true , postbox_key : key) } . catch { err in print (err) }

Instantiating tKey[â](#)

let thresholdKey =

try ?

ThresholdKey (storage_layer : storage_layer , service_provider : service_provider , enable_logging :

true , manual_sync :

false)

Parameters[â](#)

Parameter Type Description Mandatory metadata Metadata Metadata object containing the metadata details of tKey. No shares ShareStorePolyIdIndexMap Array of ShareStore with PolyId. No storage_layer StorageLayer Takes in the Storage Provider Instance No service_provider ServiceProvider Takes in the Service Provider Instance No local_matadata_transitions Metadata Local metadata transitions No enable_logging Bool This option is used to specify whether to enable logging or not. No manual_sync Bool manual sync provides atomicity to your tkey share. If manual_sync is true, you should sync your local metadata transitions manually to your storage_layer, which means your storage layer doesnât know the local changes of your tkey unless you manually sync, gives atomicity. Otherwise, If manual_sync is false, then your local metadata changes will be synced automatically to your storage layer. If manual_sync = true and want to synchronize manually, you need to call sync_local_metadata_transitions() manually. No

Example[â](#)

guard

let postboxkey = userData ["privateKey"]

as ?

String

else

{ alertContent =

"Failed to get postboxkey" return }

guard

let storage_layer =

try ?

StorageLayer (enable_logging :

true , host_url :

"https://metadata.tor.us" , server_time_offset :

2)

else

{ alertContent =

"Failed to create storage layer" return }

guard

let service_provider =

try ?

ServiceProvider (enable_logging :

true , postbox_key : postboxkey)

else

{ alertContent =

"Failed to create service provider" return }

guard

let thresholdKey =

try ?

ThresholdKey (storage_layer : storage_layer , service_provider : service_provider , enable_logging :

true , manual_sync :

false)

else

{ alertContent =

"Failed to create threshold key" return }

threshold_key

thresholdKey