Initializing tKey JS SDK

After installation, the next step to using Web3Auth Core Kit tKey SDK is to Initialize the SDK. The Initialization takes a few steps, including initiating the tKey SDK with the service provider and modules.

- Configuration of Service Provider
- Configuration of Storage Layer
- Configuration of Modules
- Instantiation of tKey
- Initialisation of Service Provider

note These are the most critical steps where you must pass on different parameters according to the need of your project.

Configuring Service Providerâ

Service Provider intKey is used for generating aShare A , i.e. the private key share managed by a wallet service provider via their authentication flows. This share in ourwallet 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.

@tkey/service-provider-sfa

â

In the Web3Auth Core Kit tKey SDK setup, you have to use the Web3Auth SFA Service Provider (@tkey/service-provider-sfa), which is basically a wrapper around theSingle Factor Auth Web library implementing Web3Auth's Social Login feature as a service provider and enabling you to associate the social login of the user as one of the shares.

Web3AuthOptions

â

- Table
- Interface

Parameter Type Description Mandatory clientId string Web3Auth Client ID. Obtain yourclientId from theWeb3Auth Developer Dashboard . Yes web3AuthNetwork? TORUS_LEGACY_NETWORK_TYPE Web3Auth Network to use for login. @defaultValue mainnet No enableLogging? boolean Setting it to true will enable logs. No export

interface

Web3AuthOptions

{ /* * *Client id for web3auth. * You can obtain your client id from the web3auth developer dashboard. * You can set any random string for this on localhost.* / clientId :

string ; /* * *Web3Auth Network to use for login * @defaultValue mainnet*/ web3AuthNetwork ? :

TORUS_LEGACY_NETWORK_TYPE ;

/* * *setting to true will enable logs * * @defaultValue false*/ enableLogging ? :

boolean ; } warning Please note that the some of the parameters mentioned in the type description within the SFA Web SDK are not valid for the service provider. We have only mentioned the ones that are applicable.

Usageâ

const clientId =

"BPi5PB_UiIZ-cPz1GtV5i1I2iOSOHuimiXBI0e-Oe_u6X3oVAbCiAZOTEBtTXw4tsluTITPqA8zMsfxIKMjiqNQ" ;

// get from https://dashboard.web3auth.io

const chainConfig =

{ chainId :

"0x1" , rpcTarget :

"https://rpc.ankr.com/eth" , displayName :

"mainnet" , blockExplorer :

"https://etherscan.io/" , ticker :

"ETH" , tickerName :

"Ethereum" , } ;

const web3AuthOptions :

any

=

{ clientId ,

// Get your Client ID from the Web3Auth Dashboard chainConfig , web3AuthNetwork :

"sapphire_mainnet" ,

// ["sapphire_mainnet", "sapphire_devnet"] } ;

// Configuration of Service Provider const serviceProvider =

new

SFAServiceProvider ({ web3AuthOptions }) ;

Configuring Storage Layerâ

The Storage Layer refers to the Metadata Storage, which is used to store the metadata information of the shares generated by the tKey SDK.

@tkey/storage-layer-torus

â

info By default, we recommend using the Web3Auth (Torus) Metadata Storage URL as your default storage. However, if you want to own the storage layer, you can contactour team to help you set that up.

TorusStorageLayerArgs

â

ThisTorusStorageLayer constructor takes an object withTorusStorageLayerArgs as input. It contains the following parameters:

- Table
- Type Reference

Parameter Type Description Mandatory enableLogging? boolean Enables logging on the console No hostUrl? string URL for connecting to your metadata server. Usehttps://metadata.tor.us for connecting to the Torus Metadata Server No serverTimeOffset? number Time offset for letting user connect to metadata server No declare

type

TorusStorageLayerArgs

=

{ enableLogging ? :

boolean ; hostUrl ? :

string ;

// use https://metadata.tor.us for connecting to the Torus Metadata Server serverTimeOffset ? :

number ; } ;

Usageâ

import

TorusStorageLayer

from

"@tkey/storage-layer-torus" ;

const storageLayer =

new

TorusStorageLayer ({ hostUrl :

"https://metadata.tor.us"

}) ;

Configuring Modulesâ

Modules extend the tKey SDK's capability to provide additional functionality for managing your tKey shares. Out of the multiple modules available, you can choose the module you want to include in your implementation. Here is the list of available modules:

Share B (Storage Modules)â

Storage Modules are used to store the tKey shares in the user's device storage. You can choose any of the following modules:

Web Storage Moduleâ

Adds the capability to add or remove a share from the local and file storage.

@tkey/web-storage

â

import

{

WebStorageModule

}

from

"@tkey/web-storage" ; const webStorageModule =

new

WebStorageModule ( ) ;

React Native Storage Moduleâ

Adds the capability to add or remove a share from the React Native Encrypted Store/ Expo Secure Store.

@tkey/react-native-storage

â

- Bare React Native
- Expo React Native

import

{

ReactNativeStorageModule

}

from

"@tkey/react-native-storage" ; import

EncryptedStorage

from

"react-native-encrypted-storage" ; const reactNativeStorageModule =

new

ReactNativeStorageModule (EncryptedStorage) ; import

{

ReactNativeStorageModule

}

from

"@tkey/react-native-storage" ; import

*

as

SecureStore

from

"expo-secure-store" ; const reactNativeStorageModule =

new

ReactNativeStorageModule (SecureStore) ;

Chrome Storage Moduleâ

Adds the capability to add or remove a share from the Chrome Extension storage.

@tkey/chrome-storage

â

import

{

ChromeExtensionStorageModule

}

from

"@tkey/chrome-storage" ; const chromeStorageModule =

new

ChromeExtensionStorageModule ( ) ;

Recovery Modules (Share C)â

Recovery Modules are used to recover the tKey shares in the case user doesn't have access to their device/storage or needs additional security.

Security Questions Moduleâ

Adds the capability to add or remove a question and password as a share.

@tkey/security-questions

â

import

{

SecurityQuestionsModule

}

from

"@tkey/security-questions" ; const securityQuestionsModule =

new

SecurityQuestionsModule ( ) ;

Share Transfer Moduleâ

Adds the capability to transfer a share to another device.

@tkey/share-transfer

â

import

{

ShareTransferModule

}

from

"@tkey/share-transfer" ; const shareTransferModule =

new

ShareTransferModule ( ) ;

Share Serialization Moduleâ

Adds the capability to serialize/ deserialize a BN (private key) into a mnemonic.

@tkey/share-serialization

â

import

{

ShareSerializationModule

}

from

"@tkey/share-serialization" ; const shareSerializationModule =

new

ShareSerializationModule ( ) ;

Additional Modulesâ

These modules provide extra capability to the tKey SDK, like importing users' existing private keys or seedphrases.

Seed Phrase Moduleâ

Adds the capability to store and use seed phrases on the metadata.

@tkey/seed-phrase

â

import

{

SeedPhraseModule

}

from

"@tkey/seed-phrase" ; const seedPhraseModule =

new

SeedPhraseModule ( ) ;

Private Keys Moduleâ

Adds the capability to store extra private keys on the metadata.

@tkey/private-keys

â

import

{

PrivateKeyModule

}

from

"@tkey/private-keys" ; const privateKeyModule =

new

PrivateKeyModule ( ) ;

Instantiating tKeyâ

Import theThresholdKey

class from@tkey/default â

import

ThresholdKey

from

"@tkey/core" ;

Assign theThresholdKey

class to a variableâ

const tKey =

new

ThresholdKey (TKeyArgs) ; This ThresholdKey constructor takes an object withTKeyArgs as input.

Parametersâ

TKeyArgs

â

- Table
- Type Declaration

Parameter Type Description Mandatory enableLogging? boolean This option is used to specify whether to enable logging. No modules? ModuleMap An object mapping the modules needed to be configured with their instances within the app No serviceProvider? IServiceProvider Takes in the Service Provider Instance No storageLayer? IStorageLayer Takes in the Storage Layer Instance No customAuthArgs? CustomAuthArgs Optional, you can pass thecustomAuthArgs from Web3Auth Service Provider here as well. No manualSync? boolean Helps you turn on manual sync, which helps you configure Tkey to not constant the server for every request. You can sync at your convenience. No serverTimeOffset? number Time offset for letting user connect to metadata server No declare

type

TKeyArgs

=

{ enableLogging ? :

boolean ; modules ? :

ModuleMap ;

// pass over the modules you want to use serviceProvider ? :

IServiceProvider ;

// pass over the service provider variable you instantiated storageLayer ? :

IStorageLayer ;

// pass over the storage layer variable you instantiated customAuthArgs ? :

CustomAuthArgs ;

// you need to pass the customAuthArgs from Web3Auth Service Provider here. manualSync ? :

boolean ; serverTimeOffset ? :

number ; } ;

export

declare

type

ModuleMap

=

{ [moduleName :

string] :

IModule ; } ;

Exampleâ

- Web
- Bare React Native
- Expo React Native

```
import
ThresholdKey
from
"@tkey/core" ; import
SecurityQuestionsModule
from
"@tkey/security-questions" ; import
SFAServiceProvider
from
"@tkey/service-provider-sfa" ; import
TorusStorageLayer
from
"@tkey/storage-layer-torus" ; import
{
ShareSerializationModule
}
from
"@tkey/share-serialization" ; import
{
WebStorageModule
}
from
"@tkey/web-storage" ;
// Configuration of Service Provider const clientId =
"BEglQSgt4cUWcj6SKRdu5QkOXTsePmMcusG5EAoyjyOYKlVRjlF1iCNnMOTfpzCiunHRrMui8TlwQPXdkQ8Yxuk" ;
// get from https://dashboard.web3auth.io
const chainConfig =
{ chainId :
"0x1" , rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"mainnet" , blockExplorer :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , } ;
const web3AuthOptions :
any
=
```

```
{ clientId ,

// Get your Client ID from the Web3Auth Dashboard chainConfig , web3AuthNetwork :

"sapphire_mainnet" ,

// ["sapphire_mainnet", "testnet"] } ;

const serviceProvider =

new

SFAServiceProvider ( { web3AuthOptions } ) ;

// Configuration of Storage Layer const storageLayer =

new

TorusStorageLayer ( { hostUrl :

"https://metadata.tor.us" , } ) ;

// Configuration of Modules const webStorageModule =

new

WebStorageModule ( ) ; const shareSerializationModule =

new

ShareSerializationModule ( ) ; const securityQuestionsModule =

new

SecurityQuestionsModule ( ) ;

// Instantiation of tKey export

const tKeyInstance =

new

ThresholdKey ( { serviceProvider , storageLayer , modules :

{ securityQuestions : securityQuestionsModule , shareSerialization : shareSerializationModule , webStorage :
webStorageModule , } , } ) ; import

ThresholdKey

from

"@tkey/core" ; import

SecurityQuestionsModule

from

"@tkey/security-questions" ; import

SFAServiceProvider

from

"@tkey/service-provider-sfa" ; import

TorusStorageLayer

from

"@tkey/storage-layer-torus" ; import

{

ShareSerializationModule
```

```
}
from
"@tkey/share-serialization" ; import
{
ReactNativeStorageModule
}
from
"@tkey/react-native-storage" ; import
EncryptedStorage
from
"react-native-encrypted-storage" ;
// Configuration of Service Provider const clientId =
"BEglQSgt4cUWcj6SKRdu5QkOXTsePmMcusG5EAoyjyOYKlVRjIF1iCNnMOTfpzCiunHRrMui8TIwQPXdkQ8Yxuk" ;
// get from https://dashboard.web3auth.io
const chainConfig =
{ chainId :
"0x1" , rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"mainnet" , blockExplorer :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , } ;
const web3AuthOptions :
any
=
{ clientId ,
// Get your Client ID from the Web3Auth Dashboard chainConfig , web3AuthNetwork :
"sapphire_mainnet" ,
// ["sapphire_mainnet", "testnet"] } ;
const serviceProvider =
new
SFAServiceProvider ( { web3AuthOptions } ) ;
// Configuration of Storage Layer const storageLayer =
new
TorusStorageLayer ( { hostUrl :
"https://metadata.tor.us" , } ) ;
// Configuration of Modules const reactNativeStorageModule =
```

new

ReactNativeStorageModule (EncryptedStorage) ; const shareSerializationModule =

new

ShareSerializationModule ( ) ; const securityQuestionsModule =

new

SecurityQuestionsModule ( ) ;

// Instantiation of tKey export

const tKeyInstance =

new

ThresholdKey ({ serviceProvider , storageLayer , modules :

{ securityQuestions : securityQuestionsModule , shareSerialization : shareSerializationModule , reactNativeStorage : reactNativeStorageModule , } , }) ; import

ThresholdKey

from

"@tkey/core" ; import

SecurityQuestionsModule

from

"@tkey/security-questions" ; import

SFAServiceProvider

from

"@tkey/service-provider-sfa" ; import

TorusStorageLayer

from

"@tkey/storage-layer-torus" ; import

{

ShareSerializationModule

}

from

"@tkey/share-serialization" ; import

{

ReactNativeStorageModule

}

from

"@tkey/react-native-storage" ; import

*

as

SecureStore

from

```
"expo-secure-store" ;
// Configuration of Service Provider const clientId =
"BEglQSgt4cUWcj6SKRdu5QkOXTsePmMcusG5EAoyjyOYKlVRjIF1iCNnMOTfpzCiunHRrMui8TIwQPXdkQ8Yxuk" ;
// get from https://dashboard.web3auth.io
const chainConfig =
{ chainId :
"0x1" , rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"mainnet" , blockExplorer :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , } ;
const web3AuthOptions :
any
=
{ clientId ,
// Get your Client ID from the Web3Auth Dashboard chainConfig , web3AuthNetwork :
"testnet" ,
// ["sapphire_mainnet", "testnet"] } ;
const serviceProvider =
new
SFAServiceProvider ( { web3AuthOptions } ) ;
// Configuration of Storage Layer const storageLayer =
new
TorusStorageLayer ( { hostUrl :
"https://metadata.tor.us" , } ) ;
// Configuration of Modules const reactNativeStorageModule =
new
ReactNativeStorageModule ( SecureStore ) ; const shareSerializationModule =
new
ShareSerializationModule ( ) ; const securityQuestionsModule =
new
SecurityQuestionsModule ( ) ;
// Instantiation of tKey export
const tKeyInstance =
new
ThresholdKey ( { serviceProvider , storageLayer , modules :
```

{ securityQuestions : securityQuestionsModule , shareSerialization : shareSerializationModule , reactNativeStorage : reactNativeStorageModule , } , }) ;

Initializing Service Providerâ

You need to initialize your Service Provider within your constructor function to use it while logging your user in through the social accounts. This is done by calling theinit() function within thetKey instance'sserviceProvider property.

tKeyInstance . serviceProvider . init (params :

PrivateKeyProvider) ;

Parametersâ

PrivateKeyProvider

â

Theinit function in SFAServiceProvider needs aPrivateKeyProvider as a parameter. It can accept any of the providers mentioned in ourProviders documentation

Exampleâ

import

{

EthereumPrivateKeyProvider

}

from

"@web3auth/ethereum-provider" ;

const chainConfig =

{ chainId :

"0x1" , rpcTarget :

"https://rpc.ankr.com/eth" , displayName :

"mainnet" , blockExplorer :

"https://etherscan.io/" , ticker :

"ETH" , tickerName :

"Ethereum" , } ;

export

const ethereumPrivateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig , } , }) ;

useEffect (( )

=>

{ const

init

=

async

( )

=>

{ // Initialization of Service Provider try

{ await

(tKeyInstance . serviceProvider

as

any) . init (ethereumPrivateKeyProvider) ; }

catch

(error)

{ console . error (error) ; } } ; init ( ) ; } ,

[ ]) ;Edit this page Previous Install Next Usage