Initializing Core Kit SFA React Native SDK

After Installation, the next step to use Web3Auth Single Factor Auth Web SDK is to Initialize the SDK.

However, the Initialization is a two-step process, ie.

- [Instantiation of Web3Auth](#)
- [Initialization of Web3Auth](#)

Instantiating Web3Auth[â](#)

Import theWeb3Auth

class from@web3auth/single-factor-auth-react-native package.[â](#)

import

{

Web3Auth

}

from

"@web3auth/single-factor-auth-react-native" ;

Assign theWeb3Auth

class to a variable[â](#)

- Expo Managed Workflow
- Bare React Native Workflow

import

*

as

SecureStore

from

"expo-secure-store" ; import

{

Web3Auth

}

from

"@web3auth/single-factor-auth-react-native" ;

const web3auth =

new

Web3Auth (SecureStore ,

SdkInitOptions) ; The react native SFA Web3Auth constructor takesSecureStore and an object withSdkInitOptions as parameters for the expo-managed workflow. import

EncryptedStorage

from

"react-native-encrypted-storage" ; import

```
{

Web3Auth

}

from

"@web3auth/single-factor-auth-react-native" ;

const web3auth =

new

Web3Auth ( EncryptedStorage ,
```

SdkInitOptions) ; The react native SFA Web3Auth constructor takesEncryptedStorage and an object withSdkInitOptions as parameters for the bare workflow.

Arguments[â](#)

SdkInitOptions

[â](#)

SdkInitOptions is an object with the following properties:

- Table
- Interface

Parameter Type Description Mandatory clientId string Web3Auth Client ID. Obtain yourclientId from the[Web3Auth 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 usePnPKey? boolean Setting this to true returns the same key as PnP Web SDK, By default, this SDK returns CoreKitKey. Use this flag if you require the key returned by PnP SDK. Ideal for migrating from PnP Products to SFA No sessionTime? number Determine the session length in seconds. By default, the value is set at 86400 seconds, ie. 7 days. No export

interface

SdkInitOptions

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

boolean ; /* * setting this to true returns the same key as web sdk (i.e., plug n play key) * By default, this sdk returns CoreKitKey / usePnPKey ? :

boolean ; /* * How long should a login session last at a minimum in seconds * * @defaultValue 86400 seconds * @remarks Max value of sessionTime can be 7 * 86400 (7 days) / sessionTime ? :

number ; }

Example[â](#)

- Expo Managed Workflow
- Bare React Native Workflow

```
const web3auth =

new

Web3Auth ( SecureStore ,

{ clientId :

"WEB3AUTH_CLIENT_ID" ,
```

```
// Get your Client ID from the Web3Auth Dashboard web3AuthNetwork :

"sapphire_mainnet" , usePnPKey :

false ,

// By default, this sdk returns CoreKitKey } ) ; const web3auth =

new

Web3Auth ( EncryptedStorage ,

{ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

// By default, this sdk returns CoreKitKey } ) ;
```

Instantiating Provider[â](#)

For getting the appropriate key for your selected chain, you need to instantiate a provider. This provider is used to make calls to the selected blockchain. Currently, Web3Auth exposes the following provider packages to be integrated within the SDKs:

- [EthereumPrivateKeyProvider](#)
- : For Ethereum and EVM compatible chains (secp256k1 private key)
- [SolanaPrivateKeyProvider](#)
- : For Solana Blockchain (ed25519 private key)
- [XRPLPrivateKeyProvider](#)
- : For XRP Ledger
- [CommonPrivateKeyProvider](#)
- : For other blockchains (secp256k1 private key)

Using these providers, you can export the private keys for the respective curves of the selected blockchain. You can use these keys to even create manual providers for other blockchains supporting the curves/ have conversion packages for their blockchain curves.

- ETH
- Solana
- XRPL
- Other Chains

Usage import

{

EthereumPrivateKeyProvider

}

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ;

const privateKeyProvider =

```
new

EthereumPrivateKeyProvider ( { config :

{ / pass the chain config that you want to connect with. all chainConfig fields are required/ chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

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

"ETH" , tickerName :

"Ethereum" , } , } , } ) ; Usage import

{

SolanaPrivateKeyProvider

}

from

"@web3auth/solana-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ;

const privateKeyProvider =

new

SolanaPrivateKeyProvider ( { config :

{ / pass the chain config that you want to connect with. all chainConfig fields are required/ chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . SOLANA , chainId :

"0x1" , rpcTarget :

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

"Solana Mainnet" , blockExplorer :

"https://explorer.solana.com/" , ticker :

"SOL" , tickerName :

"Solana" , } , } , } ) ; Usage import

{

XrplPrivateKeyProvider

}

from
```

```javascript
"@web3auth/xrpl-provider" ; import
{
CHAIN_NAMESPACES
}
from
"@web3auth/base" ;
const privateKeyProvider =
new
XrplPrivateKeyProvider ( { config :
{ /* pass the chain config that you want to connect with. all chainConfig fields are required*/ chainConfig :
{ chainNamespace :
CHAIN_NAMESPACES . OTHER , chainId :
"0x1" , rpcTarget :
"https://s.altnet.rippletest.net:51234" , displayName :
"XRPL" , blockExplorer :
"https://testnet.xrpl.org" , ticker :
"XRP" , tickerName :
"XRP" , } , } , } ) ; Usage import
{
CommonPrivateKeyProvider
}
from
"@web3auth/base-provider" ; import
{
CHAIN_NAMESPACES
}
from
"@web3auth/base" ;
const privateKeyProvider =
new
CommonPrivateKeyProvider ( { config :
{ /* pass the chain config that you want to connect with. all chainConfig fields are required*/ chainConfig :
{ chainNamespace :
CHAIN_NAMESPACES . OTHER , chainId :
"0x1" , rpcTarget :
https://rpc.target.url , displayName :
"Display Name" , blockExplorer :
```

"https://chain.explorer.link" , ticker :

"TKR" , tickerName :

"Ticker Name" , } , } , }) ;

Initializing Web3Auth[â]

init()

[â]

To complete the initialization process, we need to initialize theWeb3Auth instance, which we namedweb3auth . This is achieved by calling theinit() function of the previously createdweb3auth instance, using aprivate key provider .

await web3auth . init (privateKeyProvider) ;

Example[â]

- Expo Managed Workflow
- Bare React Native Workflow

import

*

as SecureStore from

"expo-secure-store" ; import

{ Web3Auth }

from

"@web3auth/single-factor-auth-react-native" ; import

{ EthereumPrivateKeyProvider }

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ;

// Swap with a different provider if you want to use a different blockchain const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

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

```
"ETH" , tickerName :

"Ethereum" , } , } , } ) ;

const web3auth =

new

Web3Auth ( SecureStore ,

{ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

// By default, this sdk returns CoreKitKey } ) ;

await web3auth . init ( privateKeyProvider ) ; import EncryptedStorage from

"react-native-encrypted-storage" ; import

{ Web3Auth }

from

"@web3auth/single-factor-auth-react-native" ; import

{ EthereumPrivateKeyProvider }

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ;

// Swap with a different provider if you want to use a different blockchain const privateKeyProvider =

new

EthereumPrivateKeyProvider ( { config :

{ chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

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

"ETH" , tickerName :

"Ethereum" , } , } , } ) ;

const web3auth =
```

```
new

Web3Auth ( EncryptedStorage ,

{ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

// By default, this sdk returns CoreKitKey } ) ;

await web3auth . init ( privateKeyProvider ) ;
```