Initializing Core Kit SFA Node SDK

After Installation, the next step to use Web3Auth Node.js SDK is to Initialize the SDK.

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

- Instantiation of Web3Auth
- Initialization of Web3Auth

Instantiating Web3Authâ

Import theWeb3Auth

class from@web3auth/node-sdk â

const

{

Web3Auth

}

=

require ("@web3auth/node-sdk") ;

Assign theWeb3Auth

class to a variableâ

const web3auth =

new

Web3Auth (Web3AuthOptions) ; This Web3Auth constructor takes an object withWeb3AuthOptions as input.

Argumentsâ

Web3AuthOptions

â

- Table
- Interface

Parameter Type Description Mandatory clientId string Web3Auth Client ID. Obtain yourclientId from theWeb3Auth Developer Dashboard . Yes web3AuthNetwork? sapphire_mainnet orsapphire_testnet Web3Auth Networks No enableLogging? boolean Setting it to true will enable logs. No usePnPKey? boolean Setting this to true returns the same key as web sdk (i.e., plug n play key), By default, this sdk returnsCoreKitKey . 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_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 ; }

Usage Exampleâ

const web3auth =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

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

Instantiating Providerâ

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

"@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 Web3Authâ

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 .

web3auth . init ({

provider : privateKeyProvider }) ;

Exampleâ

const

Web3Auth

=

require ("@web3auth/node-sdk") ; const

EthereumPrivateKeyProvider

=

require ("@web3auth/ethereum-provider") ; const

CHAIN_NAMESPACES

=

require ("@web3auth/base") ;

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

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig :

{ chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

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

"ETH" , tickerName :

"Ethereum" , } , } , }) ;

const web3auth =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

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

web3auth . init ({

provider : privateKeyProvider }) ; Note One needs to re-initialize the SDK for each new userEdit this page Previous Install Next Authentication