No Modal React Hooks Introduction

Web3Auth provides essential React hooks for Web3Auth No Modal SDK for managing authentication, chain configuration, and user interactions within your application. These hooks can be directly imported from the@web3auth/no-modal-react-hooks package. Here's an example of how to import them:

import

{ useWeb3Auth }

from

"@web3auth/no-modal-react-hooks" ; For more information on React hooks, refer to theofficial React documentation .

This Documentation is based on version8.6.1 of Web3Auth No Modal hooks.

Available Hooksâ

- useWeb3Auth
- : Provides access to the Web3Auth context initialized via theWeb3AuthProvider
- component.

Hook Context Interfaceâ

- Table
- Interface

Parameter Description isConnected Indicates whether a user is currently logged in or not. provider The current provider, ornull if not connected. userInfo Information about the logged-in user. isMFAEnabled Indicates whether Multi-Factor Authentication (MFA) is enabled or not. isInitialized Indicates whether Web3Auth has been initialized or not. status The current status of Web3Auth. Can take the following values:NOT_READY ,READY ,CONNECTING ,CONNECTED ,DISCONNECTED ,ERRORED . enableMFA(params?) Enables Multi-Factor Authentication for the user. Returns a Promise. logout(params?) Logs out the user, with an optional cleanup parameter. Returns a Promise. addAndSwitchChain Adds and switches to a new blockchain. TakeschainConfig of typeCustomChainConfig as a parameter. Returns a Promise. addPlugin Adds a plugin to the Web3Auth instance. Takes aplugin of typeIPlugin as a parameter. getPlugin Retrieves a plugin by name. TakespluginName of typestring as parameter. Returns anIPlugin ornull . authenticateUser Retrieves the idToken for the logged-in user. Returns a Promise. addChain Adds a new blockchain configuration. TakeschainConfig of typeCustomChainConfig as a parameter. Returns a Promise. switchChain Switches to a specified blockchain by chain ID. Takesparams of type{ chainId: string } as a parameter. Returns a Promise. interface

IBaseWeb3AuthHookContext

{ /* * *Indicates whether a user is currently logged in or not.*/ isConnected :

boolean ; /* * *The current provider, or* *null* *if not connected.*/ provider :

IProvider

|

null ; /* * *Information about the logged-in user.*/ userInfo :

Partial < OpenloginUserInfo

|

null ; /* * *Indicates whether Multi-Factor Authentication (MFA) is enabled or not.*/ isMFAEnabled :

boolean ; /* * *Indicates whether Web3Auth has been initialized or not.*/ isInitialized :

boolean ; /* * *The current status of the Web3Auth adapter.*/ status :

ADAPTER_STATUS_TYPE

|

null ; /* * *Enables Multi-Factor Authentication for the user. * @param params Optional parameters for enabling MFA. * @returns A Promise that resolves when MFA is enabled.* / enableMFA (params ? :

LoginParams) :

Promise < void

; /* * Logs out the user, with an optional cleanup parameter. * @param params Optional parameters for logging out. * @returns A Promise that resolves when the user is logged out. / logout (params ? :

{ cleanup :

boolean

}) :

Promise < void

; /* * Adds and switches to a new blockchain. * @param chainConfig Configuration for the new blockchain. * @returns A Promise that resolves when the chain is added and switched. / addAndSwitchChain (chainConfig :

CustomChainConfig) :

Promise < void

; /* * Adds a plugin to the Web3Auth instance. * @param plugin The plugin to add/ addPlugin (plugin :

IPlugin) :

void ; /* * Retrieves a plugin by name. * @param pluginName The name of the plugin. * @returns The plugin instance or null if not found. / getPlugin (pluginName :

string) :

IPlugin

|

null ; /* * Retrieves the idToken for the logged-in user. * @returns A Promise that resolves with the authenticated user's information. / authenticateUser ( ) :

Promise < UserAuthInfo

; /* * Adds a new blockchain configuration. * @param chainConfig Configuration for the new blockchain. * @returns A Promise that resolves when the chain is added. / addChain (chainConfig :

CustomChainConfig) :

Promise < void

; /* * Switches to a specified blockchain by chain ID. * @param params Parameters for switching the chain. * @returns A Promise that resolves when the chain is switched. / switchChain (params :

{ chainId :

string

}) :

Promise < void

; }

Web3AuthProviderâ

TheWeb3AuthProvider component wraps the main component and injects the Web3Auth-related context into it. It takes the following properties as its context:

- Table
- Interface

Parameter Description web3AuthOptions Configuration options for Web3Auth. adapters An array of adapters for connecting to different blockchain networks. plugins An array of plugins to add additional functionality to Web3Auth. export

interface

Web3AuthProviderProps

{ config :

Web3AuthContextConfig ; }

export

type

Web3AuthContextConfig

=

{ web3AuthOptions :

IWeb3AuthCoreOptions ; adapters ? :

IAdapter < unknown

[ ] ; plugins ? :

IPlugin [ ] ; } ;

export

interface

IWeb3AuthCoreOptions

{ clientId :

string ; chainConfig ? :

CustomChainConfig ; enableLogging ? :

boolean ; storageKey ? :

"session"

|

"local" ; sessionTime ? :

number ; web3AuthNetwork ? :

OPENLOGIN_NETWORK_TYPE ; useCoreKitKey ? :

boolean ; uiConfig ? :

WhiteLabelData ; privateKeyProvider ? :

IBaseProvider < string

; } info Please check out thePnP No Modal SDK references for interfaces for the inner parameters.

Shared Methods Descriptionsâ

Once you've installed and successfully initialized Web3Auth, you can use it to authenticate your users. Further, you can use the native provider given by Web3Auth to connect the users to the respective blockchain network.

Natively, the instance ofWeb3Auth (referred to asweb3auth in our examples) returns the following functions:

- init()
-
  - Initializes the Web3Auth instance.
- await
- init
- (
-)
- ;

- Returns:
- init
- (
-)
- :
- Promise
- <
- void
-

- ;
- connectTo(walletName, loginParams?)
-
    - Connect to a specific wallet adapter.
- await
- connectTo
- (
- WALLET_ADAPTERS
- .
- OPENLOGIN
- ,
- {
- loginProvider
- :
- "google"
- ,
- }
-)
- ;
- Returns:
- connectTo
- <
- T
-

- (
- walletName
- :
- WALLET_ADAPTER_TYPE
- ,
- loginParams
- ?
- :
- T
-)
- :
- Promise
- <
- IProvider
- |
- null
-

- ;
- On successful login, theconnectTo
- function returns anIProvider
- instance. You can use this
- provider to connect your user to the blockchain and make transactions.
- provider()
-
    - Returns the native provider that can be used to make different blockchain
- transactions. Returns:
- get
- provider
- (
-)
- :
- IProvider
- |

- null
- ;
- connected()
-
    - Returnstrue
- orfalse
- depending on whether the web3auth instance is
- available or not. Returns:
- get
- connected
- (
-)
- :
- boolean
- ;
- getUserInfo()
-
    - Getting the User's Information.
- const
- userInfo
- =
- await
- getUserInfo
- (
-)
- ;
- authenticateUser()
-
    - Getting the idToken from Web3Auth.
- const
- idToken
- =
- await
- authenticateUser
- (
-)
- ;
- addChain()
-
    - Add chain config details to the connected adapter.
- await
- addChain
- (
- chainConfig
-)
- ;
- switchChain()
-
    - Switch chain as per chainId already added to chain config.
- await
- switchChain
- (
- {
- chainId
- :
- "0x1"
- }
-)
- ;
- getAdapter()
-
    - Get the connected adapter instance.
- const
- adapter
- =
- await
- getAdapter
- (

- adapterName
-)
- ;
- configureAdapter()
-
    - Configure the adapter instance.
- await
- configureAdapter
- (
- adapterConfig
-)
- ;
- clearCache()
-
    - Clear the cache.
- await
- clearCache
- (
-)
- ;
- addPlugin()
-
    - Add a plugin to Web3Auth.
- await
- addPlugin
- (
- plugin
-)
- ;
- logout()
-
    - Logging out the User.
- await
- logout
- (
-)
- ;

Example Usageâ

web3AuthProviderProps.ts import

{ Web3AuthContextConfig }

from

"@web3auth/modal-react-hooks" ; import

{ Web3AuthOptions }

from

"@web3auth/modal" ; import

{ EthereumPrivateKeyProvider }

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES ,

WEB3AUTH_NETWORK

}

from

"@web3auth/base" ; import

```
{ OpenloginAdapter }

from

"@web3auth/openlogin-adapter" ; import

{ WalletServicesPlugin }

from

"@web3auth/wallet-services-plugin" ;

const chainConfig =

{ chainId :

"0xaa36a7" ,

// for wallet connect make sure to pass in this chain in the loginSettings of the adapter. displayName :

"Ethereum Sepolia" , chainNamespace :

CHAIN_NAMESPACES . EIP155 , tickerName :

"Ethereum Sepolia" , ticker :

"ETH" , rpcTarget :

"https://rpc.ankr.com/eth_sepolia" , blockExplorerUrl :

"https://sepolia.etherscan.io" , logo :

"https://cryptologos.cc/logos/ethereum-eth-logo.png" , } ;

const privateKeyProvider =

new

EthereumPrivateKeyProvider ( { config :

{ chainConfig , } , } ) ;

const web3AuthOptions : Web3AuthOptions =

{ clientId : "BPi5PB_UiIZ-cPz1GtV5i1I2iOSOHuimiXBI0e-Oe_u6X3oVAbCiAZOTEBtTXw4tsluTITPqA8zMsfxIKMjiqNQ" ,
web3AuthNetwork :

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider : privateKeyProvider , } ;

const openloginAdapter =

new

OpenloginAdapter ( ) ;

const walletServicesPlugin =

new

WalletServicesPlugin ( { wsEmbedOpts :

{ } , walletInitOptions :

{ whiteLabel :

{ showWidgetButton :

true

}

} , } ) ;

export
```

```
const web3AuthContextConfig : Web3AuthContextConfig =
{ web3AuthOptions , adapters :
[ openloginAdapter ] , plugins :
[ walletServicesPlugin ] , } ; App.tsx import React from
"react" ; import
{ Web3AuthProvider , useWeb3Auth }
from
"@web3auth/no-modal-react-hooks" ; import
{ web3AuthContextConfig }
from
"./web3AuthProviderProps" ; import
{
WALLET_ADAPTERS
}
from
"@web3auth/base" ;
const
App
=
( )
=>
{ const
{ init , connectTo , logout , isConnected , enableMFA , addAndSwitchChain , authenticateUser , addChain , switchChain , }
=
useWeb3Auth ( ) ;
React . useEffect ( ( )
=>
{ const
initialize
=
async
( )
=>
{ await
init ( ) ; } ; initialize ( ) ; } ,
[ init ] ) ;
return
```

```
( < Web3AuthProvider config = { web3AuthContextConfig }
        < div
        { isConnected ?
( <
        < button onClick = { logout }
        Logout < / button
        < button onClick = { enableMFA }
        Enable MFA < / button
        < button onClick = { ( )
=> addAndSwitchChain ( { chainId :
"0x89" , rpcTarget :
"https://rpc.ankr.com/polygon" , displayName :
"Polygon Mainnet" , ticker :
"MATIC" , } ) }
        Add and Switch Chain < / button
        < button onClick = { authenticateUser }
        Authenticate User < / button
        < button onClick = { ( )
=> addChain ( { chainId :
"0x89" , rpcTarget :
"https://rpc.ankr.com/polygon" , displayName :
"Polygon Mainnet" , ticker :
"MATIC" , } ) }
        Add Chain < / button
        < button onClick = { ( )
=>
switchChain ( { chainId :
"0x89"
} ) }
        Switch Chain < / button
        < /
        )
:
( < button onClick = { ( )
=> connectTo ( WALLET_ADAPTERS . OPENLOGIN ,
{ loginProvider :
"google" , } ) }
```

```
        Login < / button
        ) } < / div
        < / Web3AuthProvider
        ) ; } ;
export
default App ;
```

Edit this page Previous Usage Next Whitelabel