Initializing PnP Web Modal SDK

After Installation, the next step to use Web3Auth is to Initialize the SDK. However, the Initialization is a two-step process, with an additional two steps for customizations, i.e.

- Instantiation of Web3Auth
- Configuration of Adapters
- (optional)
- Configuration of Plugins
- (optional)
- Initialization of Modal

Please note that these are the most critical steps where you need to pass on different parameters according to the preference of your project. Additionally, If you wish to customize your Web3Auth Instance, Whitelabeling and Custom Authentication have to be configured within this step.

Instantiating Web3Authâ

Import theWeb3Auth

class from@web3auth/modal â

import

{

Web3Auth

}

from

"@web3auth/modal" ;

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 Description clientId Your Web3Auth Client ID. You can get it from Web3AuthDashboard under project details. It's a mandatory field of typestring web3AuthNetwork Defines the Web3Auth network. It's a mandatory field of typeOPENLOGIN_NETWORK_TYPE . useCoreKitKey? Use CoreKit Key to get core kit key. It's an optional field with default value asfalse . sessionTime? It allows developers to configure the session management time. Session Time is in seconds, default is 86400 seconds which is 1 day.sessionTime can be max 7 days authMode? Web3Auth instance provides different adapters for different type of usages. If you are a dApp and want to use external wallets like metamask, then you can use theDAPP authMode. If you are a wallet and only want to use you own wallet implementations, then you should useWALLET authMode. uiConfig? WhiteLabel options for web3auth. It helps you define custom UI, branding, and translations for your brand app. It takesOmit as a value. storageKey? Setting to "local" will persist social login session across browser tabs. privateKeyProvider Private key provider for your chain namespace. It takesIBaseProvider as a value. interface

Web3AuthOptions

extends

Web3AuthNoModalOptions

{ /* * *web3auth instance provides different adapters for different type of usages. If you are dapp and want to * use external wallets like metamask, then you can use the* DAPP *authMode. * If you are a wallet and only want to use you own wallet implementations along with openlogin, * then you should use* WALLET *authMode. * * @defaultValue* DAPP / authMode ? :

"DAPP"

|

"WALLET" ; /* * *Config for configuring modal ui display properties*/ uiConfig ? :

Omit < UIConfig ,

"adapterListener"

; }

interface

Web3AuthNoModalOptions

{ /* * *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 ; /* * *custom chain configuration for chainNamespace * * @defaultValue mainnet config of provided chainNamespace*/ chainConfig :

Partial < CustomChainConfig

&

Pick < CustomChainConfig ,

"chainNamespace"

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

boolean ; /* * *setting to "local" will persist social login session accross browser tabs. * * @defaultValue "local"* storageKey ? :

"session"

|

"local" ; /* * *sessionTime (in seconds) for idToken issued by Web3Auth for server side verification. * @defaultValue 86400 * * Note: max value can be 7 days (86400 * 7) and min can be 1 day (86400)* / sessionTime ? :

number ; /* * *Web3Auth Network to use for the session & the issued idToken * @defaultValue mainnet*/ web3AuthNetwork ? :

OPENLOGIN_NETWORK_TYPE ; /* * *Uses core-kit key with web3auth provider * @defaultValue false*/ useCoreKitKey ? :

boolean ; /* * *WhiteLabel options for web3auth*/ uiConfig ? :

WhiteLabelData ; /* * *Private key provider for your chain namespace*/ privateKeyProvider ? :

IBaseProvider < string

; }

Adding a Private key providerâ

privateKeyProvider

â

privateKeyProvider parameter helps you to connect with various wallet SDKs. These are preconfigured RPC clients for different blockchains used to interact with the respective blockchain networks.

note It's mandatory to provideprivateKeyProvider for your corresponding chain namespace. To know more in-depth about

providers, have a look at yourProviders reference . const chainConfig =

{ chainId :

"0x1" ,

// Please use 0x1 for Mainnet rpcTarget :

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

"Ethereum Mainnet" , blockExplorerUrl :

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

"ETH" , tickerName :

"Ethereum" , logo :

"https://images.toruswallet.io/eth.svg" , } ;

const ethereumPrivateKeyProvider =

EthereumPrivateKeyProvider ({ config :

{ chainConfig } , }) ;

const web3auth =

new

Web3Auth ({ clientId :

"" ,

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

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider : ethereumPrivateKeyProvider , }) ;

Adding a Custom Chain Configurationâ

chainConfig

â

- Table
- Type Declarations

Parameter Description chainNamespace The namespace of your preferred chain. CheckoutProviders SDK Reference for understanding RPC Calls. It acceptsChainNamespaceType as a value. chainId The chain id of the selected blockchain in hexstring format. rpcTarget * RPC Target URL for the selectedchainNamespace * &chainId * . * We provide a default RPC Target for certain blockchains, but due to congestion it might be slow hence it is recommended to provide your own RPC Target URL. wsTarget Web socket target URL for the chain instring . displayName Display Name for the chain instring . blockExplorerUrl Blockchain's explorer URL instring . (eg:https://etherscan.io) ticker Default currency ticker instring for the network (e.g:ETH) tickerName Name for currency ticker instring (e.g:Ethereum) decimals? Number of decimals innumber for the currency ticker (e.g:18) logo Logo for the chain. isTestnet? Defines whether the network is testnet or not. declare

const

CHAIN_NAMESPACES :

{ readonly

EIP155 :

"eip155" ; readonly

SOLANA :

"solana" ; readonly

OTHER :

"other" ; } ;

declare

type

ChainNamespaceType

=

(typeof

CHAIN_NAMESPACES) [keyof

typeof

CHAIN_NAMESPACES] ; declare

type

CustomChainConfig

=

{ chainNamespace :

ChainNamespaceType ; /* * *The chain id of the chain*/ chainId :

string ; /* * *RPC target Url for the chain*/ rpcTarget :

string ; /* * *web socket target Url for the chain*/ wsTarget ? :

string ; /* * *Display Name for the chain*/ displayName :

string ; /* * *Url of the block explorer*/ blockExplorerUrl :

string ; /* * *Default currency ticker of the network (e.g: ETH)*/ ticker :

string ; /* * *Name for currency ticker (e.g:Ethereum)* / tickerName :

string ; /* * *Number of decimals for the currency ticker (e.g: 18)*/ decimals ? :

number ; /* * *Logo for the chain*/ logo :

string ; /* * *Whether the network is testnet or not*/ isTestnet ? :

boolean ; } ;

Whitelabelingâ

Within theuiConfig parameter, you can configure the Web3Auth Modal according to your application's requirements.

tip This is just one of the aspects of whitelabeling you can achieve with Web3Auth. To know more in-depth about how you can Whitelabel your application with Web3Auth Plug and Play Modal SDK, have a look at ourWhitelabeling SDK Reference .

uiConfig

â

- Table
- Interface

Parameter Description loginMethodsOrder? The list of login methods can be reordered with this parameter. Those methods specified will be first on the list. Default value is ["google", "facebook", "twitter", "reddit", "discord", "twitch", "apple", "line", "github", "kakao", "linkedin", "weibo", "wechat", "email_passwordless"]. modalZIndex? Z-index of the modal and iframe. The default value is99998 and accepts astring as a value. displayErrorsOnModal? Whether to show errors on Web3Auth modal. Default value istrue . loginGridCol? Number of columns to display the Social Login buttons. Default value is 3, available options are 2 or 3. primaryButton? Decides which button will be displayed as primary button in modal. Only one button will be primary and other buttons in modal will be secondary. Default value issocialLogin . Available options areexternalLogin ,socialLogin , andemailLogin . interface

UIConfig

extends

WhiteLabelData

{ /* * order of how login methods are shown * * @defaultValue* ["google", "facebook", "twitter", "reddit", "discord", "twitch", "apple", "line", "github", "kakao", "linkedin", "weibo", "wechat", "email_passwordless"] */ loginMethodsOrder ? :

string [ ] ; /* * Z-index of the modal and iframe * @defaultValue 99998 */ modalZIndex ? :

string ; /* * Whether to show errors on Web3Auth modal. * * @defaultValue* true */ displayErrorsOnModal ? :

boolean ; /* * number of columns to display the Social Login buttons * * @defaultValue* 3 */ loginGridCol ? :

2

|

3 ; /* * decides which button will be displayed as primary button in modal * only one button will be primary and other buttons in modal will be secondary * * @defaultValue *socialLogin* */ primaryButton ? :

"externalLogin"

|

"socialLogin"

|

"emailLogin" ; }

WhiteLabelData

â

whiteLabel?: WhiteLabelData;

â

The whitelabel parameter takesWhitelabelData as input. TheWhitelabelData object takes the following parameters:

- Table
- Interface

WhiteLabelData

Parameter Description appName? App name to be displayed in the User Flow Screens. It acceptsstring as a value. appUrl? App URL to be displayed in the User Flow Screens. It acceptsstring as a value. logoLight? App logo to be shown on the light background (light theme). It acceptsstring as a value. logoDark? App logo to be shown on the dark background (dark theme). It acceptsstring as a value. defaultLanguage? Default Language to use. Choose from:* en * - English * de * - German * ja * - Japanese * ko * - Korean * zh * - Mandarin * es * - Spanish * fr * - French * pt * - Portuguese * nl * - Dutch * tr * - Turkish

. Default language isen mode? Choose betweenauto ,light ordark background modes. Default isauto . theme? Used to customize the theme of the login modal with the following options 'primary' - To customize the primary color of the modal's content. It acceptsRecord as a value. tncLink? Language specific link for terms and conditions on torus-website. See (examples/vue-app) to configure e.g. tncLink:{en: "http://example.com/tnc/en", ja: "http://example.com/tnc/ja"} privacyPolicy? Language specific link for privacy policy on torus-website. See (examples/vue-app) to configure e.g.privacyPolicy: { en: "http://example.com/tnc/en", ja: "http://example.com/tnc/ja", } export

type

WhiteLabelData

=

{ /* * App name to display in the UI */ appName ? :

string ; /* * App url */ appUrl ? :

string ; /* * App logo to use in light mode */ logoLight ? :

string ; /* * *App logo to use in dark mode*/ logoDark ? :

string ; /* * *language which will be used by web3auth. app will use browser language if not specified. if language is not supported it will use "en" * en: english * de: german * ja: japanese * ko: korean * zh: mandarin * es: spanish * fr: french * pt: portuguese * nl: dutch * * @defaultValue en* / defaultLanguage ? :

LANGUAGE_TYPE ; /* *theme * * @defaultValue auto*/ mode ? :

THEME_MODE_TYPE ; /* * *Use logo loader * * @defaultValue false*/ useLogoLoader ? :

boolean ; /* * *Used to customize theme of the login modal with following options *'primary' - To customize primary color of modal's content.* / theme ? :

{ primary ? :

string ; gray ? :

string ; red ? :

string ; green ? :

string ; success ? :

string ; warning ? :

string ; error ? :

string ; info ? :

string ; white ? :

string ; } ; /* * *Language specific link for terms and conditions on torus-website. See (examples/vue-app) to configure * e.g. * tncLink: { * en: "http://example.com/tnc/en", * ja: "http://example.com/tnc/ja", * }* / tncLink ? :

Partial < Record < LANGUAGE_TYPE ,

string

; /* * *Language specific link for privacy policy on torus-website. See (examples/vue-app) to configure * e.g. * privacyPolicy: { * en: "http://example.com/tnc/en", * ja: "http://example.com/tnc/ja", * }* / privacyPolicy ? :

Partial < Record < LANGUAGE_TYPE ,

string

; } ;

Returnsâ

- Object
- : The web3auth instance with all its methods and events.

Exampleâ

- With Whitelabeling
- Without Whitelabeling

const web3auth =

new

Web3Auth ({ clientId :

"" ,

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

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET ,

// import {WEB3AUTH_NETWORK} from "@web3auth/base"; chainConfig :

```
{ chainNamespace :
CHAIN_NAMESPACES . EIP155 , chainId :
"0x1" , rpcTarget :
"https://rpc.ankr.com/eth" ,
// This is the mainnet RPC we have added, please pass on your own endpoint while creating an app } , uiConfig :
{ appName :
"W3A Heroes" , mode :
"dark" ,
// light, dark or auto loginMethodsOrder :
[ "apple" ,
"google" ,
"twitter" ] , logoLight :
"https://web3auth.io/images/web3auth-logo.svg" , logoDark :
"https://web3auth.io/images/web3auth-logo---Dark.svg" , defaultLanguage :
"en" ,
// en, de, ja, ko, zh, es, fr, pt, nl, tr loginGridCol :
3 , primaryButton :
"socialLogin" ,
// "externalLogin" | "socialLogin" | "emailLogin" } , modalZIndex :
"99998" , } ) ; const chainConfig =
{ chainId :
"0x1" ,
// Please use 0x1 for Mainnet rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"Ethereum Mainnet" , blockExplorerUrl :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , logo :
"https://images.toruswallet.io/eth.svg" , } ;
const ethereumPrivateKeyProvider =
EthereumPrivateKeyProvider ( { config :
{
chainConfig : chainConfig } , } ) ;
const web3auth =
new
Web3Auth ( { clientId :
"" ,
```

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

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET , privateKeyProvider : ethereumPrivateKeyProvider , }) ;

Configuring Adaptersâ

An adapter is a pluggable package that implements anIAdapter interface for a wallet within Web3Auth. An adapter can be plugged in and out of web3auth modal. Each adapter exposes the provider on successful user login that can be used to invoke RPC calls on the wallet and on the connected blockchain. Web3Auth's modal UI supports a set of default adapters depending on theauthMode being used.

info This step is generally optional. You don't have to configure any default adapter unless you want to override default configs for the adapter.

Only those adapters that are marked are nondefaultin this table on the Adapters Documentation are required to be configured always based on theauthMode you are using.

Configuring Openlogin Adapterâ

The default adapter of Web3Auth is theopenlogin-adapter . This adapter is a wrapper around theopenlogin library from Web3Auth and enables the social login features. For customising features of the main Web3Auth flow, likeWhitelabel ,Custom Authentication , etc. you need to customise the Openlogin Adapter.

tip Checkout theopenlogin-adapter SDK Reference for more details on different configurations you can pass for customisations.

Whitelabelingâ

whiteLabel

â

For customising the redirect screens while logging in and constructing the key, you need to pass onwhiteLabel configurations to theadapterSettings property of theopenlogin-adapter .

tip This is just one of the aspects of whitelabeling you can achieve with Web3Auth. To know more in depth about how you can Whitelabel your application with Web3Auth, have a look at ourWhitelabeling SDK Reference .

Exampleâ

import

{

OpenloginAdapter

}

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code network :

"sapphire_mainnet" ,

// Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code uxMode :

"popup" , whiteLabel :

{ appName :

"W3A Heroes" , appUrl :

"https://web3auth.io" , logoLight :

"https://web3auth.io/images/web3auth-logo.svg" , logoDark :

"https://web3auth.io/images/web3auth-logo---Dark.svg" , defaultLanguage :

"en" ,

// en, de, ja, ko, zh, es, fr, pt, nl, tr mode :

"dark" ,

// whether to enable dark mode. defaultValue: auto theme :

{ primary :

"#00D1B2" , } , useLogoLoader :

true , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (openloginAdapter) ;

Custom Authenticationâ

loginConfig

â

With Web3Auth, you have the option to configure logins using your own authentication services. For adding your own authentication, you have to first configure your verifiers in the Web3Auth Dashboard. Have a look at ourCustom Authentication Documentation for configuring that first.

Custom Authentication in Web3Auth is supported by the Openlogin Adapter, which is the default adapter for the Web3Auth SDK. For this, you need to configure theloginConfig parameter in theadapterSettings of theopenlogin-adapter package.

tip Refer to theCustom Authentication Documentation for more information.

Exampleâ

Since we're using the@web3auth/modal , ie. the Plug and Play Modal SDK, theloginConfig should correspond to the socials mentioned in the modal. Here, we're customizing Google and Facebook to be custom verified, rest of all other socials will be the default. You can customize other social logins or remove them using the whitelabeling option.

- Google
- Facebook
- Discord
- Twitch

import

OpenloginAdapter

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code uxMode :

"popup" , loginConfig :

{ // Google login google :

{ verifier :

"YOUR_GOOGLE_VERIFIER_NAME" ,

// Please create a verifier on the developer dashboard and pass the name here typeOfLogin :

"google" ,

// Pass on the login provider of the verifier you've created clientId :

"GOOGLE_CLIENT_ID.apps.googleusercontent.com" ,

// Pass on the clientId of the login provider here - Please note this differs from the Web3Auth ClientID. This is the JWT Client ID } , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (openloginAdapter) ; import

OpenloginAdapter

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code uxMode :

"popup" , loginConfig :

{ // Facebook login facebook :

{ verifier :

"YOUR_FACEBOOK_VERIFIER_NAME" ,

// Please create a verifier on the developer dashboard and pass the name here typeOfLogin :

"facebook" ,

// Pass on the login provider of the verifier you've created clientId :

"FACEBOOK_CLIENT_ID_1234567890" ,

// Pass on the clientId of the login provider here - Please note this differs from the Web3Auth ClientID. This is the JWT Client ID } , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (openloginAdapter) ; import

OpenloginAdapter

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code uxMode :

"popup" , loginConfig :

{ // Discord login discord :

{ verifier :

"YOUR_DISCORD_VERIFIER_NAME" ,

// Please create a verifier on the developer dashboard and pass the name here typeOfLogin :

"discord" ,

// Pass on the login provider of the verifier you've created clientId :

"DISCORD_CLIENT_ID_1234567890" ,

//use your app client id you got from discord } , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (openloginAdapter) ; import

OpenloginAdapter

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code uxMode :

"popup" , loginConfig :

{ // Facebook login facebook :

{ verifier :

"YOUR_TWITCH_VERIFIER_NAME" ,

// Please create a verifier on the developer dashboard and pass the name here typeOfLogin :

"twitch" ,

// Pass on the login provider of the verifier you've created clientId :

"TWITCH_CLIENT_ID_1234567890" ,

//use your app client id you got from twitch } , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (openloginAdapter) ;

Configuring External Wallet Adaptersâ

configureAdapter(ADAPTER)

â

To configure an adapter, create the instance of adapter by using its corresponding package and pass the returnedadapter instance inconfigureAdapter function.

tip Refer to theAdapters documentation to know more deeply about what adapters are available and how to configure them.

Exampleâ

If you want to configure the Torus EVM Wallet Adapter

- Import theTorusWalletAdapter
- from@web3auth/torus-evm-adapter
- package
- Create an instance of the adapter along with the configuration
- Pass the returned instance in toweb3auth.configureAdapter

import

{

TorusWalletAdapter

}

from

"@web3auth/torus-evm-adapter" ;

const torusAdapter =

new

TorusWalletAdapter ({ adapterSettings :

{ clientId ,

//Optional - Provide only if you haven't provided it in the Web3Auth Instantiation Code buttonPosition :

"bottom-left" , } , loginSettings :

{ verifier :

"google" , } , initParams :

{ buildEnv :

"testing" , } , chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

// This is the mainnet RPC we have added, please pass on your own endpoint while creating an app displayName :

"Ethereum Mainnet" , blockExplorerUrl :

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

"ETH" , tickerName :

"Ethereum" , logo :

"https://images.toruswallet.io/eth.svg" , } , }) ;

web3auth . configureAdapter (torusAdapter) ;

Subscribing the Lifecycle Eventsâ

Subscribing to events help you trigger responses based on the status of the connection of the user. An adapter emits certain events likeCONNECTED ,CONNECTING andDISCONNECTED etc during login lifecycle of a user. For example, you can use this to show an error message, if the user is not connected to the network. Generally, this is not a required step and should be done only if needed in particular cases.

info This step is totally optional. If you don't want to use any plugins, feel free to skip this section. tip If you're using theuxMode: "redirect" option within youropenlogin-adapter configuration, you need to subscribe to the event to handle the logging in implicitly. This is because, when redirected to a different application, the app state is not updated as per the login status. Using a lifecycle method to check this, one can easily handle the login status within the constructor function.

on(EVENT, CALLBACK)

â

Web3Auth provides the following lifecycle event to check the login status:

Adapter Eventsâ

- Table
- Type Declarations

Event Trigger with@web3auth/base package Trigger without package Description ADAPTER_DATA_UPDATED ADAPTER_EVENTS.ADAPTER_DATA_UPDATED "adapter_data_updated" Adapter data is updated within the dApp NOT_READY ADAPTER_EVENTS.NOT_READY "not_ready" Adapter is not yet ready for login READY ADAPTER_EVENTS.READY "ready" Adapter is ready for login CONNECTING ADAPTER_EVENTS.CONNECTING

"connecting" User is connecting to the dApp/ login process is in progress CONNECTED ADAPTER_EVENTS.CONNECTED "connected" User is logged in and connected with the dApp DISCONNECTED ADAPTER_EVENTS.DISCONNECTED "disconnected" User is logged out and disconnected from the dApp ERRORED ADAPTER_EVENTS.ERRORED "errored" There has been some error in connecting the user to the dApp declare

const

ADAPTER_EVENTS :

{ readonly

ADAPTER_DATA_UPDATED :

"adapter_data_updated" ; readonly

NOT_READY :

"not_ready" ; readonly

READY :

"ready" ; readonly

CONNECTING :

"connecting" ; readonly

CONNECTED :

"connected" ; readonly

DISCONNECTED :

"disconnected" ; readonly

ERRORED :

"errored" ; } ;

Exampleâ

```
import
{
ADAPTER_EVENTS
}
from
"@web3auth/base" ;
// subscribe to lifecycle events emitted by web3auth const
subscribeAuthEvents
=
( web3auth :
Web3Auth )
=>
{ web3auth . on ( ADAPTER_EVENTS . CONNECTED ,
( data :
CONNECTED_EVENT_DATA )
=>
{ console . log ( "connected to wallet" , data ) ; // web3auth.provider will be available here after user is connected } ) ;
```

web3auth . on (ADAPTER_EVENTS . CONNECTING ,

( )

=>

{ console . log ("connecting") ; }) ; web3auth . on (ADAPTER_EVENTS . DISCONNECTED ,

( )

=>

{ console . log ("disconnected") ; }) ; web3auth . on (ADAPTER_EVENTS . ERRORED ,

(error)

=>

{ console . log ("error" , error) ; }) ; web3auth . on (ADAPTER_EVENTS . ERRORED ,

(error)

=>

{ console . log ("error" , error) ; }) ; } ;

Login Modal Eventsâ

- Table
- Interface

Event Trigger with@web3auth/ui package Trigger without package Description INIT_EXTERNAL_WALLETS LOGIN_MODAL_EVENTS.INIT_EXTERNAL_WALLETS "INIT_EXTERNAL_WALLETS" External Wallet are initialized LOGIN LOGIN_MODAL_EVENTS.LOGIN "LOGIN" Login is triggered DISCONNECT LOGIN_MODAL_EVENTS.DISCONNECT "DISCONNECT" Disconnection is triggered MODAL_VISIBILITY LOGIN_MODAL_EVENTS.MODAL_VISIBILITY "MODAL_VISIBILITY" Indicates whether the modal is visible or not declare

const

LOGIN_MODAL_EVENTS :

{ readonly

INIT_EXTERNAL_WALLETS :

"INIT_EXTERNAL_WALLETS" ; readonly

LOGIN :

"LOGIN" ; readonly

DISCONNECT :

"DISCONNECT" ; readonly

MODAL_VISIBILITY :

"MODAL_VISIBILITY" ; } ;

Exampleâ

import

{

LOGIN_MODAL_EVENTS

}

from

"@web3auth/ui" ;

// subscribe to lifecycle events emitted by web3auth const

subscribeAuthEvents

=

(web3auth :

Web3Auth)

=>

{ // emitted when modal visibility changes. web3auth . on (LOGIN_MODAL_EVENTS . MODAL_VISIBILITY ,

(isVisible)

=>

{ console . log ("is modal visible" , isVisible) ; }) ; } ;

Configuring Pluginsâ

Plugins are essentially extensions to the core functionality of Web3Auth, allowing you to add additional features to your dApp. These features can be used to extend the UI functionalities, making your integration more interoperable, and a lot more, even having the functionality to be customised extremely and to your liking.

info This step is totally optional. If you don't want to use any plugins, feel free to skip this section.

showWalletConnectScanner()

â

Shows the Wallet Connect Scanner to connect with dApps having Wallet Connect login option. This is useful for interoperability with dApps having Wallet Connect login option.

Exampleâ

import

{

WalletServicesPlugin

}

from

"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =

new

WalletServicesPlugin ( ) ; web3auth . addPlugin (walletServicesPlugin) ;

// Add the plugin to web3auth

await walletServicesPlugin . showWalletConnectScanner ( ) ;

initiateTopup()

â

Shows the TopUp modal to select local currency and amount to top up the wallet.

Exampleâ

import

{

WalletServicesPlugin

}

from

"@web3auth/wallet-services-plugin" ;

const walletServicesPlugin =

new

WalletServicesPlugin ( ) ; web3auth . addPlugin (walletServicesPlugin) ;

// Add the plugin to web3auth

await walletServicesPlugin . showCheckout ( ) ;

// Opens the TopUp modal

Initializing Modalâ

initModal()

â

The final step in the whole initialization process is the initialize the Modal from Web3Auth.

This is done by calling theinitModal function of theweb3auth instance we created above.

await web3auth . initModal (params) ;

Argumentsâ

Theweb3auth.initModal takes anoptional params config object as input.

params ? :

{ modalConfig ? :

Record < WALLET_ADAPTER_TYPE ,

ModalConfig

; } Thisparams object further takes amodalConfig object using which you can configure the visibility of each adapter within the modal. Each modal config has the following configurations:

modalConfig

â

- Table
- Interface

modalConfig: { ADAPTER : { params } }

Parameter Description label Label of the adapter you want to configure. It's a mandatory field which acceptsstring . showOnModal? Whether to show an adapter in modal or not. Default value istrue . showOnDesktop? Whether to show an adapter in desktop or not. Default value istrue . showOnMobile? Whether to show an adapter in mobile or not. Default value istrue . Additionally, to configure the Openlogin Adapter's each login method, we have theloginMethods? parameter.

Parameter Description loginMethods? To configure visibility of each social login method for the openlogin adapter. It acceptsLoginMethodConfig as a value. initModal (params ? :

{ modalConfig ? :

Record < WALLET_ADAPTER_TYPE ,

ModalConfig

; }) :

Promise < void

;

interface

ModalConfig

extends

BaseAdapterConfig

{ loginMethods ? :

LoginMethodConfig ; }

interface

BaseAdapterConfig

{ label :

string ; showOnModal ? :

boolean ; showOnMobile ? :

boolean ; showOnDesktop ? :

boolean ; } loginMethods: { label: { params } }

InloginMethods , you can configure the visibility of each social login method for the openlogin adapter. The social login is corresponded by thelabel parameter. Below is the table indicating the differentparams available for customization.

Forlabels you can choose between these options:google ,facebook ,twitter ,reddit ,discord ,twitch ,apple ,line ,github ,kakao ,linkedin ,weibo ,wechat ,email_passwordless

- Table
- Type Declaration

params

Parameter Description name? Display Name. It acceptsstring as a value. description? Description for the button. If provided, it renders as a full length button. else, icon button. It acceptsstring as a value. logoHover? Logo to be shown on mouse hover. It acceptsstring as a value. logoLight? Light logo for dark background. It acceptsstring as a value. logoDark? Dark logo for light background. It acceptsstring as a value. mainOption? Show login button on the main list. It acceptsoolean as a value. Default value is false. showOnModal? Whether to show the login button on modal or not. Default value is true. showOnDesktop? Whether to show the login button on desktop. Default value is true. showOnMobile? Whether to show the login button on mobile. Default value is true. declare

type

LoginMethodConfig

=

Record < string , { /* * *Display Name. If not provided, we use the default for openlogin app*/ name :

string ; /* * *Description for button. If provided, it renders as a full length button. else, icon button*/ description ? :

string ; /* * *Logo to be shown on mouse hover. If not provided, we use the default for openlogin app*/ logoHover ? :

string ; /* * *Logo to be shown on dark background (dark theme). If not provided, we use the default for openlogin app*/ logoLight ? :

string ; /* * *Logo to be shown on light background (light theme). If not provided, we use the default for openlogin app*/ logoDark ? :

string ; /* * *Show login button on the main list*/ mainOption ? :

boolean ; /* * *Whether to show the login button on modal or not*/ showOnModal ? :

boolean ; /* * *Whether to show the login button on desktop*/ showOnDesktop ? :

boolean ; /* * *Whether to show the login button on mobile*/ showOnMobile ? :

boolean ; }

;

Exampleâ

- With Modal Configurations
- Without Modal Configurations

await web3auth . initModal ({ modalConfig :

{ [WALLET_ADAPTERS . OPENLOGIN] :

{ label :

"openlogin" , loginMethods :

{ google :

{ name :

"google login" , logoDark :

"url to your custom logo which will shown in dark mode" , } , facebook :

{ // it will hide the facebook option from the Web3Auth modal. name :

"facebook login" , showOnModal :

false , } , } , // setting it to false will hide all social login methods from modal. showOnModal :

true , } , } , }) ; await web3auth . initModal ( ) Edit this page Previous Install Next Usage