Openlogin Adapter for Social Logins

@web3auth/openlogin-adapter

â

The default adapter of Web3Auth is theopenlogin-adapter . This adapter is a wrapper around theopenlogin library from Web3Auth and enables the social login features. By default, Web3Auth has certain configurations set to enable a quick integration, however, for customizing features, likeWhitelabel ,Custom Authentication , etc. you need to configure the Openlogin Adapter. With the Openlogin Adapter package installed and instantiated, you can explore several options and customize the login experience of the user as per your needs.

Basic Detailsâ

Adapter Name:openlogin

â

Package Name:@web3auth/openlogin-adapter

â

authMode:WALLET

,DAPP â

chainNamespace:EIP155

,SOLANA â

Default:YES

â

Installationâ

- npm
- Yarn
- pnpm

npm install --save @web3auth/openlogin-adapter yarn add @web3auth/openlogin-adapter pnpm add @web3auth/openlogin-adapter

Instantiationâ

Import theOpenloginAdapter

class from@web3auth/openlogin-adapter â

import

{

OpenloginAdapter

}

from

"@web3auth/openlogin-adapter" ;

Assign theOpenloginAdapter

class to a variableâ

const openloginAdapter =

new

OpenloginAdapter (OpenloginAdapterOptions) ; This OpenloginAdapter constructor takes an object withOpenloginAdapterOptions as input.

Argumentsâ

OpenloginAdapterOptions

â

- Table
- Interface

Parameter type adapterSettings? OpenLoginOptions loginSettings? LoginSettings privateKeyProvider? PrivateKeyProvider interface

OpenloginAdapterOptions

extends

BaseAdapterSettings

{ adapterSettings ? :

MakeOptional < OpenLoginOptions ,

"clientId"

|

"network"

&

{ useCoreKitKey ? :

boolean ; } ; loginSettings ? :

LoginSettings ; privateKeyProvider ? :

PrivateKeyProvider ; }

Openlogin Adapter Settingsâ

adapterSettings

â

- Table
- Type Declaration

OpenLoginOptions

â

Variable Type Description Mandatory Default Value clientId string Client ID for web3auth. Obtain yourclientId from theWeb3Auth Developer Dashboard . Not required if you're passing the clientId in the web3auth core constructor. Yes N/A network OPENLOGIN_NETWORK_TYPE Specify the network your project is deployed on. Yes N/A buildEnv? BUILD_ENV_TYPE This parameter will be used to change the build environment of openlogin sdk. No 'production' redirectUrl? string redirectUrl is the dapp's URL where the user will be redirected after login. Register this URL on theWeb3Auth Developer Dashboard initialization will give an error. No N/A uxMode? string Two uxModes are supported: *'POPUP' * : In this uxMode, a popup will be shown to the user for login. * 'REDIRECT' * : In this uxMode, the user will be redirected to a new window tab for login. No 'POPUP' replaceUrlOnRedirect? boolean replaceUrlOnRedirect removes the params from the redirected URL after login No true loginConfig? LoginConfig loginConfig enables you to pass your own login verifiers configuration for various loginProviders.loginConfig is a key value map where each key should be a valid loginProvider and the value should be a custom configuration for that loginProvider. No N/A whiteLabel? WhiteLabelData*

Options for whitelabling default openlogin modal. No N/A storageKey? enum (session andlocal) setting to "local" will persist the social login session across browser tabs. No 'local' sessionTime? number Determine the session length in seconds. By default, the value is set at 86400 seconds, ie. 7 days. No 86400 mfaSettings? MfaSettings This parameter will be used to enable mfa factors and set priority on UI listing. List of factors available backUpShareFactor * socialFactor * passwordFactor * deviceShareFactor No false useMpc? boolean This parameter will be used to use openlogin mpc No false export

type

OpenLoginOptions

=

{ /* * *You can get your clientId/projectId by registering your * dapp on {@link "https://dashboard.web3auth.io"| developer dashboard}* / clientId :

string ; /* * *network specifies the openlogin sdk url to be used. * * -'mainnet': https://app.openlogin.com will be used which is the production version. * - 'cyan': https://cyan.openlogin.com will be used which is the production cyan version. * -'testnet': https://testing.openlogin.com will be used which is the testing version. * - 'development': http://localhost:3000 will be used for development purpose.* / network :

OPENLOGIN_NETWORK_TYPE ; /* * *This parameter will be used to change the build environment of openlogin sdk. * @defaultValue production* / buildEnv ? :

BUILD_ENV_TYPE ; /* * *redirectUrl is the dapp's url where user will be redirected after login. * * @remarks * Register this url at {@link "https://dashboard.web3auth.io"| developer dashboard} * else initialization will give error.* / redirectUrl ? :

string ; /* * *two uxModes are supported:- * -'POPUP': In this uxMode, a popup will be shown to user for login. * -'REDIRECT': In this uxMode, user will be redirected to a new window tab for login. * * @defaultValue 'POPUP' * @remarks * * Use of 'REDIRECT' mode is recommended in browsers where popups might get blocked.*/ uxMode ? :

UX_MODE_TYPE ; /* * *replaceUrlOnRedirect removes the params from the redirected url after login * * @defaultValue true* / replaceUrlOnRedirect ? :

boolean ; /* * *loginConfig enables you to pass your own login verifiers configuration for various * loginProviders. * * loginConfig is key value map where each key should be a valid loginProvider and value * should be custom configuration for that loginProvider * * @remarks * You can deploy your own verifiers from {@link "https://dashboard.web3auth.io"| developer dashboard} * to use here. **/ loginConfig ? :

LoginConfig ; /* * *options for whitelabling default openlogin modal.*/ whiteLabel ? :

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

"session"

|

"local" ; /* * *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 ; /* * *This parameter will be used to enable mfa factors and set priority on UI listing. * List of factors available * backUpShareFactor | socialFactor | passwordFactor * @defaultValue false* / mfaSettings ? :

MfaSettings ; /* * *This parameter will be used to use openlogin mpc * @defaultValue false*/ useMpc ? :

boolean ; } ;

Login Settingsâ

loginSettings

â

warning While you can pass yourchainConfig toOpenloginAdapter , it is not required forweb3auth/no-modal ie. The Web3Auth Plug and Play No Modal package, since you can directly passloginParams over to theconnectTo function.

Either way, both of these methods will work the same. Please note that theconnectTo function params will override

theOpenloginAdapter settings.

Read more about how to passloginParams toconnectTo in its respective sectionweb3auth/no-modal * Table * Type Declaration

LoginSettings

â

Variable Description loginProvider Sets the OAuth login method to be used. You can use any of the valid loginProvider from the supported list.

The list of available options:google ,facebook ,reddit ,discord ,twitch ,apple ,line ,github ,kakao ,linkedin ,twitter ,weibo ,wechat ,email_passwordless ,sms_passwordless ,jwt mfaLevel? You can set themfaLevel to customize when mfa screen should be shown to user. It currently accepts 4 values: *'default' * Setting mfa level todefault * will present mfa screen to user on every third login. * 'optional' * Setting mfa level todefault * will present mfa screen to user on every login but user can skip it. * 'mandatory' * Setting mfa level tomandatory * will make it mandatory for user to setup mfa after login. * 'none' * Setting mfa level tonone * will make the user skip the mfa setup screen extraLoginOptions? This can be used to pass standard oauth login options to loginProvider. For ex: you will have to passlogin_hint as user's email anddomain as your app domain inextraLoginOptions while usingemail_passwordless loginProvider. It acceptsExtraLoginOptions as a value. dappShare? Custom logins can get a dApp share returned to them post successful login. This is useful if the dapps want to use this share to allow users to login seamlessly.dappShare is a 24 word seed phrase. It acceptsstring as a value. curve? This curve will be used to determine the public key encoded in the jwt token which returned ingetUserInfo function after user login. You can use that public key from jwt token as a unique user identifier in your backend.* 'secp256k1' * : secp256k1 based pub key is added as a wallet public key in jwt token to use. * 'ed25519' * : ed25519 based pub key is added as a wallet public key in jwt token to use.

Note: This parameter won't change format of private key returned by openlogin. Private key returned by openlogin is alwayssecp256k1 . It acceptsSUPPORTED_KEY_CURVES_TYPE as a value. appState? Any custom state you wish to pass along. This will be returned to you post redirect. Use this to store data that you want to be available to the dApp after login. It acceptsstring as a value. redirectUrl? App's URL where user will be redirected after login. Register this URL in thedeveloper dashboard , else initialization will give error. It acceptsstring as a value. export

type

LoginSettings

=

Partial < LoginParams

&

Partial < BaseRedirectParams

;

export

type

LoginParams

=

BaseRedirectParams

&

{ /* * *loginProvider sets the oauth login method to be used. * You can use any of the valid loginProvider from the supported list.* / loginProvider :

LOGIN_PROVIDER_TYPE

|

CUSTOM_LOGIN_PROVIDER_TYPE ; /* * *You can set themfaLevel to customize when mfa screen should be shown to user. * It currently accepts 4 values:- * - 'default': Setting mfa level todefault will present mfa screen to user on every third login. * - 'optional': Setting mfa level todefault will present mfa screen to user on every login but user can skip it. * - 'mandatory':*

Setting mfa level to `mandatory` *will make it mandatory for user to setup mfa after login. * -* `'none'`*: Setting mfa level to* `none` *will make the user skip the mfa setup screen * * Defaults to* `default` *@defaultValue* `default` / mfaLevel ? :

MfaLevelType ; /* * *extraLoginOptions can be used to pass standard oauth login options to * loginProvider. * * For ex: you will have to pass* `login_hint` *as user's email and* `domain` *as your app domain in* `extraLoginOptions` *while using* `email_passwordless` *loginProvider* / extraLoginOptions ? :

ExtraLoginOptions ; /* * *Custom Logins can get a dapp share returned to them post successful login. * This is useful if the dapps want to use this share to allow users to login seamlessly * dappShare is a 24 word seed phrase* / dappShare ? :

string ; /* * *This curve will be used to determine the public key encoded in the jwt token which returned in* `getUserInfo` *function after user login. * You can use that public key from jwt token as a unique user identifier in your backend. * * -* `'secp256k1'`*: secp256k1 based pub key is added as a wallet public key in jwt token to use. * -* `'ed25519'`*: ed25519 based pub key is added as a wallet public key in jwt token to use. * * Note: This parameter won't change format of private key returned by openlogin. Private key returned * by openlogin is always* `secp256k1`*. As of now you have to convert it to* `'ed25519'` *if you want. * You can use* `@toruslabs/openlogin-ed25519` *npm package for this purpose. * * * @defaultValue secp256k1*/ curve ? :

SUPPORTED_KEY_CURVES_TYPE ; } ;

export

type

BaseRedirectParams

=

{ /* * *redirectUrl is the dapp's url where user will be redirected after login. * * @remarks * Register this url at {@link "https://dashboard.web3auth.io"| developer dashboard} * else initialization will give error.* / redirectUrl ? :

string ; /* * *Any custom state you wish to pass along. This will be returned to you post redirect. * Use this to store data that you want to be available to the dapp after login.* / appState ? :

string ; } ;

/* * *{@label loginProviderType}*/ export

type

LOGIN_PROVIDER_TYPE

=

(typeof

LOGIN_PROVIDER) [keyof

typeof

LOGIN_PROVIDER] ; export

declare

const

LOGIN_PROVIDER :

{ readonly

GOOGLE :

"google" ; readonly

FACEBOOK :

"facebook" ; readonly

REDDIT :

"reddit" ; readonly

DISCORD :

"discord" ; readonly

TWITCH :

"twitch" ; readonly

APPLE :

"apple" ; readonly

LINE :

"line" ; readonly

GITHUB :

"github" ; readonly

KAKAO :

"kakao" ; readonly

LINKEDIN :

"linkedin" ; readonly

TWITTER :

"twitter" ; readonly

WEIBO :

"weibo" ; readonly

WECHAT :

"wechat" ; readonly

EMAIL_PASSWORDLESS :

"email_passwordless" ; readonly

SMS_PASSWORDLESS :

"sms_passwordless" ; readonly

WEBAUTHN :

"webauthn" ; readonly

JWT :

"jwt" ; } ;

export

type

CUSTOM_LOGIN_PROVIDER_TYPE

=

string

&

{ toString ? :

(radix ? :

number)

=>

string ; } ;

/* * {@label MfaLevelType}/

export

type

MfaLevelType

=

(typeof

MFA_LEVELS) [keyof

typeof

MFA_LEVELS] ; export

declare

const

MFA_LEVELS :

{ readonly

DEFAULT :

"default" ; readonly

OPTIONAL :

"optional" ; readonly

MANDATORY :

"mandatory" ; readonly

NONE :

"none" ; } ;

/* * {@label SUPPORTED_KEY_CURVES_TYPE}/

export

type

SUPPORTED_KEY_CURVES_TYPE

=

(typeof

SUPPORTED_KEY_CURVES) [keyof

typeof

SUPPORTED_KEY_CURVES] ; export

declare

const

SUPPORTED_KEY_CURVES :

{ readonly

SECP256K1 :

"secp256k1" ; readonly

ED25519 :

"ed25519" ; } ;

Multi-Factor Authenticationâ

At Web3Auth, we prioritize your security by offering Multi-Factor Authentication (MFA). MFA is an extra layer of protection that verifies your identity when accessing your account. To ensure ownership, you must provide two or more different backup factors. You have the option to choose from the device, social, backup factor (seed phrase), and password factors to guarantee access to your Web3 account. Once you create a recovery factor, MFA is enabled, and your keys are divided into three shares for off-chain multi-sig, making the key self-custodial. With backup factors, you can easily recover your account if you lose access to your original device or help log into a new device.

For a dApp, we provide various options to set up Multi-Factor Authentication. You can customize the MFA screen by setting themfaLevel argument. You can enable or disable a backup factor and change their order. Currently, there are four values formfaLevel :

- default
- : presents the MFA screen every third login
- optional
- : presents the MFA screen on every login, but you can skip it
- mandatory
- : make it mandatory to set up MFA after login
- none
- : skips the MFA setup screen

We offer the following backup factors undermfaSettings :

- deviceShareFactor
- ,
- backUpShareFactor
- ,
- socialBackupFactor
- , and
- passwordFactor
- .

Exampleâ

const openloginAdapter =

new

OpenloginAdapter ({ loginSettings :

{ mfaLevel :

"optional" ,

// default, optional, mandatory, none } , adapterSettings :

{ // SCALE and above plan only feature mfaSettings :

{ deviceShareFactor :

{ enable :

true , priority :

1 , mandatory :

true , } , backUpShareFactor :

{ enable :

true , priority :

2 , mandatory :

false , } , socialBackupFactor :

{ enable :

true , priority :

3 , mandatory :

false , } , passwordFactor :

{ enable :

true , priority :

4 , mandatory :

false , } , } , } , }) ;

Whitelabelâ

Web3Auth's Social Logins and Email Login run using the OpenLogin Flow. The whole OpenLogin user experience can also be whitelabeled using OpenLogin Adapter settings. For this, you need to pass on thewhiteLabel configuration parameter to theadapterSettings property of theopenlogin-adapter .

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

; } ;

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

While instantiating the Openlogin Adapter, you can pass some configuration objects to the constructor. One of these configurations is theadapterSettings configuration which enables you to make changes in the adapter, enabling you to do things like Whitelabeling and Custom Authentication among other things.

loginConfig

â

TheloginConfig parameter ofadapterSettings inopenlogin-adapter contains the following properties:

- Table
- Type Declarations

loginConfig: { "identifier of social login": { params } }

params

Parameter Description verifier The name of the verifier that you have registered on the Web3Auth Dashboard. It's a mandatory field, and acceptsstring as a value. typeOfLogin Type of login of this verifier, this value will affect the login flow that is adapted. For example, if you choosegoogle , a Google sign-in flow will be used. If you choosejwt , you should be providing your own JWT token, no sign-in flow will be presented. It's a mandatory field, and acceptsTypeOfLogin as a value. clientId Client id provided by your login provider used for custom verifier. e.g. Google's Client ID or Web3Auth's client Id if using 'jwt' as TypeOfLogin. It's a mandatory field, and acceptsstring as a value. name? Display name for the verifier. If null, the default name is used. 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. verifierSubIdentifier? The field in JWT token which maps to verifier id. Please make sure you selected correct JWT verifier id in the developer dashboard. 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 acceptsboolean 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. showOnSocialBackupFactor? If we are using social logins as a backup factor, then this option will be used to show the type of social login on the social backup login screen. jwtParameters? Custom jwt parameters to configure the login. Useful for Auth0 configuration. It acceptsJwtParameters as a value. export

type

LoginConfig

=

Record < string , { verifier :

string ; /* * *The type of login. Refer to enum*LOGIN_TYPE / typeOfLogin :

TypeOfLogin ; /* * *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 ; /* * *Custom client_id. If not provided, we use the default for openlogin app*/ clientId ? :

string ; verifierSubIdentifier ? :

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 ; /* * *If we are using social logins as a backup factor, * then this option will be used to show the type of social login * on the social backup login screen.* / showOnSocialBackupFactor ? :

boolean ; /* * *Custom jwt parameters to configure the login. Useful for Auth0 configuration*/ jwtParameters ? :

JwtParameters ; }

;

export

type

TypeOfLogin

= |

"google" |

"facebook" |

"reddit" |

"discord" |

"twitch" |

"apple" |

"github" |

"linkedin" |

"twitter" |

"weibo" |

"line" |

"email_password" |

"passwordless" |

"jwt" |

"webauthn" ;

Custom Authentication within Web3Auth Modalâ

When we're using the@web3auth/modal , ie. the Plug and Play Modal SDK, theloginConfig should correspond to the socials mentioned in the modal. This means you can use your own authentication services for the following services:

google |facebook |twitter |reddit |discord |twitch |apple |line |github |kakao |linkedin |weibo |wechat |passwordless

info You can customize all or a few of the social logins and others will remain default. You can also remove the ones you don't want using the whitelabeling option.

Exampleâ

- 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) ;

Logging in through your Custom JWT Tokenâ

When we are using@web3auth/no-modal , ie. Web3Auth Plug and Play No Modal SDK, we have the option to use theconnectTo function, which enables you to customize the login process according to the parameters you have for your custom authentication service.

connectTo()

â

To log a user in the Web3Auth Plug and Play No Modal SDK, you need to call theconnectTo() function. This function helps you customize the login process according to your own needs, by taking the following parameters:

- Table
- Function Definition

Variable Description walletName Wallet Adapter you want to use for logging in your user. It acceptsWALLET_ADAPTER_TYPE . loginParams? Login Parameters specific to your wallet adapter. connectTo < T

(walletName :

WALLET_ADAPTER_TYPE , loginParams ? :

T) :

Promise < IProvider

|

null

; export

type

WALLET_ADAPTER_TYPE

=

(typeof

WALLET_ADAPTERS) [keyof

typeof

WALLET_ADAPTERS] ; export

declare

const

WALLET_ADAPTERS :

{ OPENLOGIN :

string ; WALLET_CONNECT_V2 :

string ; TORUS_SOLANA :

string ; PHANTOM :

string ; SOLFLARE :

string ; SLOPE :

string ; TORUS_EVM :

string ; METAMASK :

string ; COINBASE :

string ; } ; tip Know more about theconnectTo function in theUsage SDK Reference Further, to enable Custom Authentication, theloginParams parameter takes in another object calledextraLoginOptions which contains the following properties:

- Table
- Interface

ExtraLoginOptions

â

Parameter Description additionalParams? Additional params in[key: string] format for OAuth login, use id_token(JWT) to authenticate with web3auth. domain? Your custom authentication domain instring format. For example, if you are using Auth0, it can be example.au.auth0.com. client_id? Client id instring format, provided by your login provider used for custom verifier. leeway? The value used to account for clock skew in JWT expirations. The value is in the seconds, and ideally should no more than 60 seconds or 120 seconds at max. It takesstring as a value. verifierIdField? The field in JWT token which maps to verifier id. Please make sure you selected correct JWT verifier id in the developer dashboard. It takesstring as a value. isVerifierIdCaseSensitive? boolean to confirm Whether the verifier id field is case sensitive or not. display? Allows developers the configure the display of UI. It takesstring as a value. prompt? Prompt shown to the user during authentication process. It takesstring as a value. max_age? Max time allowed without reauthentication. If the last time user authenticated is greater than this value, then user must reauthenticate. It takesstring as a value. ui_locales? The space separated list of language tags, ordered by preference. For instancefr-CA fr en . id_token_hint? It denotes the previously issued ID token. It takesstring as a value. id_token? JWT (ID Token) to be passed for login. login_hint? It is used to send the user's email address during Email Passwordless login. It takesstring as a value. acr_values? acc_values scope? The default scope to be used on authentication requests. The defaultScope defined in the Auth0Client is included along with this scope. It takesstring as a value. audience? The audience, presented as the aud claim in the access token, defines the intended consumer of the token. It takesstring as a value. connection? The name of the connection configured for your application. If null, it will redirect to the Auth0 Login Page and show the Login Widget. It takesstring as a value. redirect_uri? It can be used to specify the default url, where your custom jwt verifier can redirect your browser to with the result. If you are using Auth0, it must be whitelisted in the Allowed Callback URLs in your Auth0's application. export

interface

ExtraLoginOptions

extends

BaseLoginOptions

{ /* * *Your Auth0 account domain such as'example.auth0.com', * 'example.eu.auth0.com' or , 'example.mycompany.com' * (when using custom domains)* / domain ? :

string ; /* * *The Client ID found on your Application settings page*/ client_id ? :

string ; /* * *The default URL where Auth0 will redirect your browser to with * the authentication result. It must be whitelisted in * the "Allowed Callback URLs" field in your Auth0 Application's * settings. If not provided here, it should be provided in the other * methods that provide authentication.* / redirect_uri ? :

string ; /* * *The value in seconds used to account for clock skew in JWT expirations. * Typically, this value is no more than a minute or two at maximum. * Defaults to 60s.* / leeway ? :

number ; /* * *The field in jwt token which maps to verifier id*/ verifierIdField ? :

string ; /* * *Whether the verifier id field is case sensitive * @defaultValue true*/ isVerifierIdCaseSensitive ? :

boolean ; } export

interface

BaseLoginOptions

{ /* * *If you need to send custom parameters to the Authorization Server, * make sure to use the original parameter name.*/ [key :

string] :

unknown ; /* * - *'page': displays the UI with a full page view * -'popup': displays the UI with a popup window * -'touch': displays the UI in a way that leverages a touch interface * - 'wap': displays the UI with a "feature phone" type interface*/ display ? :

"page"

|

"popup"

|

"touch"

|

"wap"

|

string ; /* * - *'none': do not prompt user for login or consent on re-authentication * -'login': prompt user for re-authentication * - 'consent': prompt user for consent before processing request * -'select_account': prompt user to select an account*/ prompt ? :

"none"

|

"login"

|

"consent"

|

"select_account"

|

string ; /* * *Maximum allowable elapsed time (in seconds) since authentication. * If the last time the user authenticated is greater than this value, * the user must be re-authenticated.* / max_age ? :

string

|

number ; /* * *The space-separated list of language tags, ordered by preference. * For example:'fr-CA fr en'.* / ui_locales ? :

string ; /* * *Previously issued ID Token.*/ id_token_hint ? :

string ; /* * *The user's email address or other identifier. When your app knows * which user is trying to authenticate, you can provide this parameter * to pre-fill the email box or select the right session for sign-in. * * This currently only affects the classic Lock experience.* / login_hint ? :

string ; id_token ? :

string ; acr_values ? :

string ; /* * *The default scope to be used on authentication requests. * The defaultScope defined in the Auth0Client is included * along with this scope* / scope ? :

string ; /* * *The default audience to be used for requesting API access.*/ audience ? :

string ; /* * *The name of the connection configured for your application. * If null, it will redirect to the Auth0 Login Page and show * the Login Widget.* / connection ? :

string ; }

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" , loginConfig :

{ jwt :

{ verifier :

"YOUR-VERIFIER-NAME-ON-WEB3AUTH-DASHBOARD" , typeOfLogin :

"jwt" , clientId :

"YOUR-CLIENTID-FROM-LOGIN-PROVIDER" , } , } , } , privateKeyProvider , }) ;

web3auth . configureAdapter (openloginAdapter) ; * Google * Auth0 * JWT * Facebook * Discord * Email Passwordless * SMS Passwordless * Twitter * Reddit * Twitch * Apple * GitHub * LinkedIn

import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ; // inside your async function with on click handler const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"google" , }) ; const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"jwt" , extraLoginOptions :

{ verifierIdField :

"sub" ,

// same as your JWT Verifier ID domain :

"https://YOUR-APPLICATION-DOMAIN" ,

// your service provider domain, e.g. Auth0 } , }) ; // Login using JWT, either obtained from Firebase, Okta, Auth0 or bring your own JWT. const web3authProvider =

```
await web3auth . connectTo ( WALLET_ADAPTERS . OPENLOGIN ,
{ loginProvider :
"jwt" , extraLoginOptions :
{ id_token :
"idToken" ,
// in JWT Format verifierIdField :
"sub" ,
// same as your JWT Verifier ID } , } ) ; import
{
WALLET_ADAPTERS
}
from
"@web3auth/base" ;
const web3authProvider =
await web3auth . connectTo ( WALLET_ADAPTERS . OPENLOGIN ,
{ loginProvider :
"facebook" , } ) ; import
{
WALLET_ADAPTERS
}
from
"@web3auth/base" ;
const web3authProvider =
await web3auth . connectTo ( WALLET_ADAPTERS . OPENLOGIN ,
{ loginProvider :
"email_passwordless" , extraLoginOptions :
{ login_hint :
"hello@web3auth.io" ,
// email to send the OTP to } , } ) ; import
{
WALLET_ADAPTERS
}
from
"@web3auth/base" ;
const web3authProvider =
await web3auth . connectTo ( WALLET_ADAPTERS . OPENLOGIN ,
{ loginProvider :
```

"sms_passwordless" , extraLoginOptions :

{ login_hint :

"+65-XXXXXXX" , } , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"discord" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"twitter" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"reddit" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"twitch" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"apple" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"github" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . OPENLOGIN ,

{ loginProvider :

"linkedin" , }) ;

Initializationâ

Finally, once all the configurations are set, you need to initialize the Openlogin Adapter.

web3auth . configureAdapter (openloginAdapter) ;

Change Adapter Settingsâ

You can change the adapter settings by calling thesetAdapterSettings() function on the adapter instance. This function takes the below-mentioned parameters as well asOpenlogin Adapter Settings .

Argumentsâ

- Table

- Interface

Parameter type clientId? string sessionTime? number chainConfig? CustomChainConfig web3AuthNetwork? OPENLOGIN_NETWORK_TYPE interface

BaseAdapterSettings

{ clientId ? :

string ; sessionTime ? :

number ; chainConfig ? :

CustomChainConfig ; web3AuthNetwork ? :

OPENLOGIN_NETWORK_TYPE ; }

Exampleâ

@web3auth/modal

â

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ uxMode :

"redirect" ,

// "redirect" | "popup" whiteLabel :

{ logoLight :

"https://web3auth.io/images/w3a-L-Favicon-1.svg" , logoDark :

"https://web3auth.io/images/w3a-D-Favicon-1.svg" , defaultLanguage :

"en" ,

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

"dark" ,

// whether to enable dark, light or auto mode. defaultValue: auto [system theme] } , // SCALE and above plan only feature mfaSettings :

{ deviceShareFactor :

{ enable :

true , priority :

1 , mandatory :

true , } , backUpShareFactor :

{ enable :

true , priority :

2 , mandatory :

false , } , socialBackupFactor :

{ enable :

true , priority :

3 , mandatory :

false , } , passwordFactor :

{ enable :

true , priority :

4 , mandatory :

false , } , } , loginConfig :

{ // Add login configs corresponding to the providers on modal // 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 } , // 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 } , // Add other login providers here } , } , loginSettings :

{ mfaLevel :

"mandatory" , } , }) ; web3auth . configureAdapter (openloginAdapter) ;

// You can change the adapter settings by calling the`setAdapterSettings()` function on the adapter instance.

openloginAdapter . setAdapterSettings ({ web3AuthNetwork :

"sapphire_mainnet" , sessionTime :

3600 ,

// 1 hour in seconds chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

// This is the public RPC we have added, please pass on your own endpoint while creating an app } , clientId , redirectUrl :

"http://localhost:3000" , }) ;

web3auth/no-modal

â

const openloginAdapter =

new

OpenloginAdapter ({ adapterSettings :

{ network :

"sapphire_mainnet" , uxMode :

"popup" , whiteLabel :

{ appName :

"W3A Heroes" , appUrl :

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

"https://web3auth.io/images/w3a-L-Favicon-1.svg" , logoDark :

"https://web3auth.io/images/w3a-D-Favicon-1.svg" , defaultLanguage :

"en" ,

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

"auto" ,

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

{ primary :

"#768729" , } , useLogoLoader :

true , } , // SCALE and above plan only feature mfaSettings :

{ deviceShareFactor :

{ enable :

true , priority :

1 , mandatory :

true , } , backUpShareFactor :

{ enable :

true , priority :

2 , mandatory :

false , } , socialBackupFactor :

{ enable :

true , priority :

3 , mandatory :

false , } , passwordFactor :

{ enable :

true , priority :

4 , mandatory :

false , } , } , loginConfig :

{ jwt :

{ verifier :

"web3auth-auth0-demo" , typeOfLogin :

"jwt" , clientId :

"294QRkchfq2YaXUbPri7D6PH7xzHgQMT" , } , } , } , }) ; web3auth . configureAdapter (openloginAdapter) ;

// You can change the adapter settings by calling the setAdapterSettings() function on the adapter instance.

openloginAdapter . setAdapterSettings ({ web3AuthNetwork :

"sapphire_mainnet" , sessionTime :

3600 ,

// 1 hour in seconds chainConfig :

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

// This is the public RPC we have added, please pass on your own endpoint while creating an app } , clientId , redirectUrl :

"http://localhost:3000" , }) ;Edit this page Previous Adapters for Web3Auth PnP Web SDKs Next Torus EVM Wallet