Auth Adapter for Social Logins

@web3auth/auth-adapter

â

The default adapter of Web3Auth is theauth-adapter . This adapter is a wrapper around theauth library from Web3Auth (previously Torus) and enables the main social login features of Web3Auth. By default, Web3Auth has certain configuration set to enable a quick integration, however, for customising features, like Whitelabel, Custom Authentication, etc. you need to customise the Auth Adapter. With the Auth Adapter package installed and instantiated, you can explore a number of options and can customise the login experience of the user as per your needs.

Basic Detailsâ

Adapter Name:auth

â

Package Name:@web3auth/auth-adapter

â

Default:YES

â

Installationâ

- npm
- Yarn
- pnpm

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

Instantiationâ

Import theAuthAdapter

class from@web3auth/auth-adapter â

import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

Assign theAuthAdapter

class to a variableâ

const authAdapter =

new

AuthAdapter (AuthAdapterOptions) ; This AuthAdapter constructor takes an object withAuthAdapterOptions as input.

Argumentsâ

AuthAdapterOptions

â

- Table
- Interface

Parameter type adapterSettings? authOptions loginSettings? LoginSettings privateKeyProvider? PrivateKeyProvider export

interface

AuthAdapterOptions

extends

BaseAdapterSettings

{ adapterSettings ? :

MakeOptional < AuthOptions ,

"clientId"

|

"network"

; loginSettings ? :

LoginSettings ; privateKeyProvider ? :

PrivateKeyProvider ; }

Auth Adapter Settingsâ

adapterSettings

â

- Table
- Type Declaration

AuthOptions

â

Variable Type Description Mandatory Default Value clientId string You can get your clientId/projectId by registering your dapp ondeveloper dashboard Yes - network WEB3AUTH_NETWORK_TYPE network specifies the web3auth network to be used. Yes - buildEnv BUILD_ENV_TYPE This parameter will be used to change the build environment of auth sdk. No production redirectUrl string redirectUrl is the dapp's url where user will be redirected after login. Register this url atdeveloper dashboard else initialization will give error. No - uxMode UX_MODE_TYPE 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. Use of'REDIRECT' mode is recommended in browsers where popups might get blocked. No 'POPUP' replaceUrlOnRedirect boolean replaceUrlOnRedirect removes the params from the redirected url after login No true originData OriginData originData is used to verify the origin of dapp by iframe. You don't have to pass originData explicitly if you have registered your dapp atdeveloper dashboard . originData contains a signature of dapp's origin url which is generated using project's secret. No - loginConfig LoginConfig 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 You can deploy your own verifiers fromdeveloper dashboard to use here. No - webauthnTransports AuthenticatorTransport[] webauthnTransport enables you to configure the transport type user can use for saving their share. This is only available for v1 users. No ["internal"] sdkUrl string sdkUrl is for internal development use only and is used to override thenetwork parameter. No - dashboardUrl string dashboardUrl is for internal development use only and is used to override thebuildEnv parameter. No - whiteLabel WhiteLabelData options for whitelabling default auth modal. No - storageServerUrl string Specify a custom storage server url No https://session.web3auth.io storageKey "session" | "local" setting to "local" will persist social login session across browser tabs. No "local" sessionTime number How long should a login session last at a minimum in seconds No 86400 seconds sessionNamespace string This option is for internal use only in torus wallet and has no effect on user's login on other dapps. No - mfaSettings MfaSettings This parameter will be used to enable mfa factors and set priority on UI listing. List of factors available backUpShareFactor | socialFactor | passwordFactor | authenticatorFactor No false useMpc boolean This parameter will be used to use auth mpc No false useCoreKitKey boolean This parameter will be used to select core kit key. No false export

type

AuthOptions

=

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

string ; /* * network specifies the web3auth network to be used.* / network :

WEB3AUTH_NETWORK_TYPE ; /* * This parameter will be used to change the build environment of auth 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 ; /* * originData is used to verify the origin of dapp by iframe. * * @internal * @remarks * You don't have to pass originData explicitly if you have registered your dapp at * {@link "https://dashboard.web3auth.io"| developer dashboard}. * * originData contains a signature of dapp's origin url which is generated using * project's secret. * / originData ? :

OriginData ; /* * 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 ; /* * webauthnTransport enables you to configure the transport type user can use * for saving their share. * * @defaultValue ["internal"] * * @remarks * This is only available for v1 users. * / webauthnTransports ? :

AuthenticatorTransport [ ] ; /* * sdkUrl is for internal development use only and is used to override the *network parameter. * @internal * / sdkUrl ? :

string ; /* * dashboardUrl is for internal development use only and is used to override the *buildEnv parameter. * @internal * / dashboardUrl ? :

string ; /* * options for whitelabling default auth modal.* / whiteLabel ? :

WhiteLabelData ; /* * Specify a custom storage server url * @defaultValue https://session.web3auth.io * @internal* / storageServerUrl ? :

string ; /* * setting to "local" will persist social login session across 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 option is for internal use only in torus wallet and has no effect * on user's login on other dapps. * @internal * / sessionNamespace ? :

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

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

boolean ; /* * This parameter will be used to select core kit key. * @defaultValue false* / useCoreKitKey ? :

boolean ; } ;

Login Settingsâ

loginSettings

â

warning While you can pass yourchainConfig toAuthAdapter , 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 theAuthAdapter settings.

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

LoginSettings

â

Variable Description loginProvider loginProvider sets the oauth login method to be used. You can use any of the valid loginProvider from the supported list. 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 Defaults tonone @defaultValuenone getWalletKey This option is for internal use only in torus wallet and has no effect on user's login on other dapps. Defaults to false @defaultValue false @internal extraLoginOptions extraLoginOptions 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 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 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 auth. Private key returned by auth is alwayssecp256k1 . dappUrl Allows the dapp to set a custom redirect url for the manage mfa flow. 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 tonone will*

*make the user skip the mfa setup screen * * Defaults to none * @defaultValue none* / mfaLevel ? :

MfaLevelType ; /* * *This option is for internal use only in torus wallet and has no effect * on user's login on other dapps. * * Defaults to false * @defaultValue false * @internal* / getWalletKey ? :

boolean ; /* * *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 auth. Private key returned * by auth is always secp256k1. * * * @defaultValue secp256k1*/ curve ? :

SUPPORTED_KEY_CURVES_TYPE ; /* * *Allows the dapp to set a custom redirect url for the manage mfa flow. */* dappUrl ? :

string ; } ;

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

FARCASTER :

"farcaster" ; 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 authAdapter =

new

AuthAdapter ({ 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 Auth Flow. The whole Auth user experience can also be whitelabeled using Auth Adapter settings. For this, you need to pass on thewhiteLabel configuration parameter to theadapterSettings property of theauth-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 display in the UI appUrl App url logoLight App logo to use in light mode logoDark App logo to use in dark mode defaultLanguage language which will be used by web3auth. app will use browser language if not specified. if language is not supported it will use "en" mode theme useLogoLoader Use logo loader theme Used to customize your theme tncLink Language specific link for terms and conditions on torus-website. See (examples/vue-app) to configure privacyPolicy Language specific link for privacy policy on torus-website. See (examples/vue-app) to configure 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 * tr: turkish * * @defaultValue en* / defaultLanguage ? :

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

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

boolean ; /* * *Used to customize your theme*/ theme ? :

WHITE_LABEL_THEME ; /* * *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

; } ;

export

declare

const

LANGUAGES :

{ readonly en :

"en" ; readonly ja :

"ja" ; readonly ko :

"ko" ; readonly de :

"de" ; readonly zh :

"zh" ; readonly es :

"es" ; readonly fr :

"fr" ; readonly pt :

"pt" ; readonly nl :

"nl" ; readonly tr :

"tr" ; } ; export

type

LANGUAGE_TYPE

=

(typeof

LANGUAGES) [keyof

typeof

LANGUAGES] ;

export

type

WHITE_LABEL_THEME

=

{ /* **primary* *color that represents your brand * Will be applied to elements such as primary button, nav tab(selected), loader, input focus, etc.* / primary ? :

string ; /* * *onPrimary color that is meant to contrast with the primary color * Applies to elements such as the text in a primary button or nav tab(selected), blocks of text on top of a primary background, etc.* / onPrimary ? :

string ; } ;

export

declare

const

THEME_MODES :

{ readonly light :

"light" ; readonly dark :

"dark" ; readonly auto :

"auto" ; } ; export

type

THEME_MODE_TYPE

=

(typeof

THEME_MODES) [keyof

typeof

THEME_MODES] ;

Exampleâ

import

{

AuthAdapter ,

WHITE_LABEL_THEME ,

WhiteLabelData

}

from

"@web3auth/auth-adapter" ; import

{ CHAIN_NAMESPACES , IProvider , UX_MODE , WALLET_ADAPTERS , WEB3AUTH_NETWORK , }

from

"@web3auth/base" ;

const authAdapter =

new

AuthAdapter ({ adapterSettings :

{ clientId ,

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

WEB3AUTH_NETWORK . SAPPHIRE_MAINNET ,

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

UX_MODE . REDIRECT , 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" , }

as

WHITE_LABEL_THEME , useLogoLoader :

true , }

as

WhiteLabelData , } , privateKeyProvider , }) ; web3auth . configureAdapter (authAdapter) ;

Custom Authenticationâ

While instantiating the Auth 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 inauth-adapter contains the following properties:

- Table
- Type Declarations

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

params

Parameter Description verifier typeOfLogin The type of login. Refer to enumLOGIN_TYPE name Display Name. If not provided, we use the default for auth app description Description for button. If provided, it renders as a full length button. else, icon button clientId Custom client_id. If not provided, we use the default for auth app 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. If not provided, we use the default for auth app logoLight Logo to be shown on dark background (dark theme). If not provided, we use the default for auth app logoDark Logo to be shown on light background (light theme). If not provided, we use the default for auth app mainOption Show login button on the main list showOnModal Whether to show the login button on modal or not showOnDesktop Whether to show the login button on desktop showOnMobile Whether to show the login button on mobile 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 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 auth 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 auth app*/ clientId ? :

string ; verifierSubIdentifier ? :

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

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

string ; /* * *Logo to be shown on light background (light theme). If not provided, we use the default for auth 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" |

"passkeys" |

"email_passwordless" |

"sms_passwordless" ;

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
- Twitter
- LinkedIn
- Github
- Apple
- Line

import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

const authAdapter =

new

AuthAdapter ({ adapterSettings :

{ 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 (authAdapter) ; import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

```
const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ 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 ( authAdapter ) ; import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ 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 ( authAdapter ) ; import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

const authAdapter =

new
```

```
AuthAdapter ( { adapterSettings :

{ 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 ( authAdapter ) ;
import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ loginConfig :

{ // Twitter login twitter :

{ verifier :

"YOUR_AUTH0_VERIFIER_NAME" ,

// Since twitter login is not supported directly, you can use Auth0 as a verifier typeOfLogin :

"twitter" , clientId :

"YOUR_AUTH0_CLIENT_ID" ,

//use your app client id from Auth0, since twitter login is not supported directly jwtParameters :

{ domain :

"YOUR_AUTH0_DOMAIN" , verifierIdField :

"YOUR_AUTH0_VERIFIER_ID_FIELD" , isVerifierIdCaseSensitive :

true ,

// only if the verifier id is case sensitive } , } , } , } , privateKeyProvider , } ) ; web3auth . configureAdapter ( authAdapter ) ;
import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;
```

```
const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ loginConfig :

{ // LinkedIn login linkedin :

{ verifier :

"YOUR_AUTH0_VERIFIER_NAME" ,

// Since linkedin login is not supported directly, you can use Auth0 as a verifier typeOfLogin :

"linkedin" , clientId :

"YOUR_AUTH0_CLIENT_ID" ,

//use your app client id from Auth0, since linkedin login is not supported directly jwtParameters :

{ domain :

"YOUR_AUTH0_DOMAIN" , verifierIdField :

"YOUR_AUTH0_VERIFIER_ID_FIELD" , isVerifierIdCaseSensitive :

true ,

// only if the verifier id is case sensitive } , } , } , } , privateKeyProvider , } ) ; web3auth . configureAdapter ( authAdapter ) ;
import

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ loginConfig :

{ // Github login github :

{ verifier :

"YOUR_AUTH0_VERIFIER_NAME" ,

// Since github login is not supported directly, you can use Auth0 as a verifier typeOfLogin :

"github" , clientId :

"YOUR_AUTH0_CLIENT_ID" ,

//use your app client id from Auth0, since github login is not supported directly jwtParameters :

{ domain :

"YOUR_AUTH0_DOMAIN" , verifierIdField :

"YOUR_AUTH0_VERIFIER_ID_FIELD" , isVerifierIdCaseSensitive :

true ,

// only if the verifier id is case sensitive } , } , } , } , privateKeyProvider , } ) ; web3auth . configureAdapter ( authAdapter ) ;
```

```
import
{
AuthAdapter
}
from
"@web3auth/auth-adapter" ;
const authAdapter =
new
AuthAdapter ( { adapterSettings :
{ loginConfig :
{ // Apple login apple :
{ verifier :
"YOUR_AUTH0_VERIFIER_NAME" ,
// Since apple login is not supported directly, you can use Auth0 as a verifier typeOfLogin :
"apple" , clientId :
"YOUR_AUTH0_CLIENT_ID" ,
//use your app client id from Auth0, since apple login is not supported directly jwtParameters :
{ domain :
"YOUR_AUTH0_DOMAIN" , verifierIdField :
"YOUR_AUTH0_VERIFIER_ID_FIELD" , isVerifierIdCaseSensitive :
true ,
// only if the verifier id is case sensitive } , } , } , } , privateKeyProvider , } ) ; web3auth . configureAdapter ( authAdapter ) ;
import
{
AuthAdapter
}
from
"@web3auth/auth-adapter" ;
const authAdapter =
new
AuthAdapter ( { adapterSettings :
{ loginConfig :
{ // Line login line :
{ verifier :
"YOUR_AUTH0_VERIFIER_NAME" ,
// Since line login is not supported directly, you can use Auth0 as a verifier typeOfLogin :
"line" , clientId :
"YOUR_AUTH0_CLIENT_ID" ,
```

//use your app client id from Auth0, since line login is not supported directly jwtParameters :

{ domain :

"YOUR_AUTH0_DOMAIN" , verifierIdField :

"YOUR_AUTH0_VERIFIER_ID_FIELD" , isVerifierIdCaseSensitive :

true ,

// only if the verifier id is case sensitive } , } , } , } , privateKeyProvider , }) ; web3auth . configureAdapter (authAdapter) ;

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. Although this is defined as a generic typeT , you can useAuthLoginParams as a reference for typical parameters. connectTo < T

(walletName :

WALLET_ADAPTER_TYPE , loginParams ? :

T) :

Promise < IProvider

|

null

; export

declare

const

WALLET_ADAPTERS :

{ AUTH :

string ; WALLET_CONNECT_V2 :

string ; SFA :

string ; TORUS_SOLANA :

string ; TORUS_EVM :

string ; COINBASE :

string ; } ; export

type

WALLET_ADAPTER_TYPE

=

(typeof

WALLET_ADAPTERS) [keyof

typeof

WALLET_ADAPTERS] ;

export

type

AuthLoginParams

=

LoginParams

&

{ login_hint ? :

string ; } ;

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 the* *mfaLevel* *to customize when mfa screen should be shown to user. * It currently accepts 4 values:- * - 'default': Setting mfa level to* *default* *will present mfa screen to user on every third login. * - 'optional': Setting mfa level to* *default* *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* *none* ** @defaultValue* *none* / mfaLevel ? :

MfaLevelType ; /* * *This option is for internal use only in torus wallet and has no effect * on user's login on other dapps. * * Defaults to false * @defaultValue false * @internal* / getWalletKey ? :

boolean ; /* * *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 auth. Private key returned * by auth is always* *secp256k1*. * * * *@defaultValue secp256k1*/ curve ? :

SUPPORTED_KEY_CURVES_TYPE ; /* * *Allows the dapp to set a custom redirect url for the manage mfa flow. */* dappUrl ? :

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 domain Your Auth0 account domain such as'example.auth0.com' ,'example.eu.auth0.com' or ,'example.mycompany.com' (when usingcustom domains) client_id The Client ID found on your Application settings page redirect_uri 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. leeway 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. verifierIdField The field in jwt token which maps to verifier id isVerifierIdCaseSensitive Whether the verifier id field is case sensitive additionalParams If you need to send custom parameters to the Authorization Server, make sure to use the original parameter name. display -'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 prompt -'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 max_age 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. ui_locales The space-separated list of language tags, ordered by preference. For example:'fr-CA fr en' . id_token_hint Previously issued ID Token. login_hint 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. acr_values scope The default scope to be used on authentication requests. The defaultScope defined in the Auth0Client is included along with this scope audience The default audience to be used for requesting API access. 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. 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 ; 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

{

AuthAdapter

}

from

"@web3auth/auth-adapter" ;

```
const authAdapter =

new

AuthAdapter ( { adapterSettings :

{ loginConfig :

{ jwt :

{ verifier :

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

"jwt" , clientId :

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

web3auth . configureAdapter ( authAdapter ) ; * Google * Auth0 * Farcaster * 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 . AUTH ,

{ loginProvider :

"google" , } ) ; const web3authProvider =

await web3auth . connectTo ( WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"jwt" , extraLoginOptions :

{ verifierIdField :

"sub" ,

// same as your JWT Verifier ID domain :

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

// your service provider domain, e.g. Auth0 } , } ) ; import

{

WALLET_ADAPTERS

}

from

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

await web3auth . connectTo ( WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"farcaster" , } ) ; // Login using JWT, either obtained from Firebase, Okta, Auth0 or bring your own JWT. const web3authProvider =

await web3auth . connectTo ( WALLET_ADAPTERS . AUTH ,
```

{ 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 . AUTH ,

{ loginProvider :

"facebook" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ 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 . AUTH ,

{ loginProvider :

"sms_passwordless" , extraLoginOptions :

{ login_hint :

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

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"discord" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"twitter" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"reddit" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"twitch" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"apple" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"github" , }) ; import

{

WALLET_ADAPTERS

}

from

"@web3auth/base" ;

const web3authProvider =

await web3auth . connectTo (WALLET_ADAPTERS . AUTH ,

{ loginProvider :

"linkedin" , }) ;

Initializationâ

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

web3auth . configureAdapter (authAdapter) ;

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 as

Auth Adapter Settings .

Argumentsâ

- Table

- Interface

AuthOptions

Parameter type description clientId string Your projectId/clientId from the dashboard network WEB3AUTH_NETWORK_TYPE Network specifies the web3auth network to be used. buildEnv? BUILD_ENV_TYPE This parameter will be used to change the build environment of auth sdk. redirectUrl? string redirectUrl is the dapp's url where user will be redirected after login. Register this url at " https://dashboard.web3auth.io " else initialization will give error. uxMode? UX_MODE_TYPE Either of the twouxMode values:popup orredirect . Use ofREDIRECT mode is recommended in browsers where popups might get blocked. replaceUrlOnRedirect? boolean replaceUrlOnRedirect removes the params from the redirected url after login originData? OriginData originData contains a signature of dapp's origin url which is generated using project's secret. loginConfig? LoginConfig loginConfig is key value map where each key should be a valid loginProvider and value should be custom configuration for that loginProvider. webauthnTransports? AuthenticatorTransport[] webauthnTransports enables you to configure the transport type user can use for saving their share. whiteLabel? WhiteLabelData whiteLabel is for whitelabling default auth modal. storageServerUrl? string storageServerUrl specifies a custom storage server url storageKey? "session" | "local" storageKey setting tolocal will persist social login session across browser tabs. sessionTime? number How long should a login session last at a minimum in seconds. Maximum value of sessionTime can be 7 * 86400 (7 days) mfaSettings? MfaSettings This parameter will be used to enable mfa factors and set priority on UI listing. useMpc? boolean This parameter will be used to use auth mpc useCoreKitKey? boolean This parameter will be used to select core kit key. BaseAdapterSettings

Parameter type clientId string sessionTime number chainConfig CustomChainConfig web3AuthNetwork WEB3AUTH_NETWORK_TYPE useCoreKitKey boolean setAdapterSettings (adapterSettings :

Partial < AuthOptions

&

BaseAdapterSettings

&

{ privateKeyProvider ? :

PrivateKeyProvider ; }) :

void ;

export

type

PrivateKeyProvider

=

IBaseProvider < string

;

export

type

AuthOptions

=

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

string ; /* * *network specifies the web3auth network to be used.*/ network :

WEB3AUTH_NETWORK_TYPE ; /* * *This parameter will be used to change the build environment of auth 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 ; /* * *originData is used to verify the origin of dapp by iframe. * * @internal * @remarks * You don't have to pass originData explicitly if you have registered your dapp at * {@link "https://dashboard.web3auth.io"| developer dashboard}. * * originData contains a signature of dapp's origin url which is generated using * project's secret.* / originData ? :

OriginData ; /* * *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 ; /* * *webauthnTransport enables you to configure the transport type user can use * for saving their share. * * @defaultValue ["internal"] * * @remarks * This is only available for v1 users.* / webauthnTransports ? :

AuthenticatorTransport [ ] ; /* * *sdkUrl is for internal development use only and is used to override the *network parameter. * @internal* / sdkUrl ? :

string ; /* * *dashboardUrl is for internal development use only and is used to override the *buildEnv parameter. * @internal*/ dashboardUrl ? :

string ; /* * *options for whitelabling default auth modal.*/ whiteLabel ? :

WhiteLabelData ; /* * *Specify a custom storage server url * @defaultValue https://session.web3auth.io * @internal*/ storageServerUrl ? :

string ; /* * *setting to "local" will persist social login session across 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 option is for internal use only in torus wallet and has no effect * on user's login on other dapps. * @internal* / sessionNamespace ? :

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

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

boolean ; /* * *This parameter will be used to select core kit key. * @defaultValue false*/ useCoreKitKey ? :

boolean ; } ;

export

interface

BaseAdapterSettings

{ clientId ? :

string ; sessionTime ? :

number ; chainConfig ? :

CustomChainConfig ; web3AuthNetwork ? :

WEB3AUTH_NETWORK_TYPE ; useCoreKitKey ? :

boolean ; }

Exampleâ

@web3auth/modal

â

const authAdapter =

new

AuthAdapter ({ 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 (authAdapter) ;

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

authAdapter . 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 authAdapter =

new

AuthAdapter ({ 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 (authAdapter) ;

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

authAdapter . 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 Default Solana Adapter Next Torus EVM Wallet