Custom Authentication with PnP Web Modal SDK

Custom Authentication is a way to authenticate users with your own custom authentication service. For example, while authenticating with Google, you have the ability to use your own Google Client ID and Dashboard to authenticate users directly.

note This is a paid feature and the minimumpricing plan to use this SDK in a production environment is theGrowth Plan . You can use this feature in the development environment for free. warning For a Custom JWT based authentication services you need to use theWeb3Auth Plug and Play No Modal SDK , since the Web3Auth Modal will only help you configure the social logins present within the Modal UI. For enabling this, you needCreate a Verifier from theCustom Auth section of theWeb3Auth Developer Dashboard with your desired configuration.

tip If you want to know more about setting up a verifier and how to use it, please refer to the Custom Authentication Documentation .

Installing Openlogin Adapterâ

Social logins in Web3Auth are enabled by theopenlogin-adapter . Natively, it is already present and preconfigured within the Plug and Play SDK, but for custom configurations, you need to install the adapter package.

- npm
- Yarn
- pnpm

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

Configuring Openlogin Adapterâ

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.

tip Checkout theopenlogin-adapter SDK Reference for more details on different configurations you can pass for customisations. Further, theloginConfig parameter of theadapterSettings configuration helps us to customise the social logins. Since we're using the@web3auth/modal , ie. the Plug and Play Modal SDK, theloginConfig should be corresponding to the socials mentioned in the modal. This means you can use your own authentication services for the following services:

google |facebook |discord |twitch |

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

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" ;

Exampleâ

Since we're using the@web3auth/modal ie. the Plug and Play Modal SDK, theloginConfig should be corresponding to the socials mentioned in the modal. Here, we're customising Google and Facebook to be custom verified, rest all other socials will be default.

- 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) ; Edit this page Previous Whitelabel Next Multi Factor Authentication