Multi Factor Authentication with PnP Web Modal SDK

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 helps login 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

Note If you are using default verifiers, your users may have set up MFA on other dApps that also use default Web3Auth verifiers. In this case, the MFA screen will continue to appear if the user has enabled MFA on other dApps. This is because MFA cannot be turned off once it is enabled. Minimum Growth plan required 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.

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 theloginSettings configuration which enables you to make changes in the adapter, enabling add more settings to your login function using web3auth.

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

Login Settingsâ

loginSettings

â

- Table
- Type Declaration

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 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 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

declare

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

declare

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

declare

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
declare
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" ; } ; Usage import
{
OpenloginAdapter
}
```

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ loginSettings :

{ mfaLevel :

"mandatory" ,

// default, optional, mandatory, none } , }) ;

Adapter Settingsâ

adapterSettings

â

- Table
- Type Declaration

Variable Description clientId? Client ID for web3auth. Obtain yourclientId from theWeb3Auth Developer Dashboard . Not required if you're passing the clientId in the web3auth core constructor. It acceptsstring as a value. network Network specifies the openlogin iframe url 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://beta.openlogin.com * will be used which is the beta version.

. It acceptsOPENLOGIN_NETWORK_TYPE as a value. redirectUrl? redirectUrl is the dapp's url where user will be redirected after login. Register this url on theWeb3Auth Developer Dashboard else initialization will give error. It acceptsstring as a value. uxMode? 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.

. Default value is "POPUP". replaceUrlOnRedirect? replaceUrlOnRedirect removes the params from the redirected url after login. Default value istrue . 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. whiteLabel? Options for whitelabling default openlogin modal. It acceptsWhiteLabelData as a value. useCoreKitKey? UseuseCoreKitKey astrue , if you're using this SDK with the Single Factor Auth Web SDK. Default value isfalse . mfaSettings? MFA Settings for the user.SCALE and above plan only feature . It acceptsMfaSettings as a value. adapterSettings ? :

MakeOptional < OpenLoginOptions ,

"clientId"

|

"network"

&

{ useCoreKitKey ? :

boolean ; } ;

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 ; /* * *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 ; } ;

export

type

LoginConfig

=

Record < string ,

{ verifier :

string ; /* * *The type of login. Refer to enumLOGIN_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

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
                ; } ;
export
type
MfaSettings
=
Partial < Record < MFA_FACTOR_TYPE ,
MFA_SETTINGS
                ; export
declare
const
MFA_FACTOR :
{ readonly
DEVICE :
"deviceShareFactor" ; readonly
BACKUP_SHARE :
"backUpShareFactor" ; readonly
SOCIAL_BACKUP :
"socialBackupFactor" ; readonly
PASSWORD :
"passwordFactor" ; } ; export
type
MFA_FACTOR_TYPE
=
( typeof
MFA_FACTOR ) [ keyof
typeof
MFA_FACTOR ] ; export
type
MFA_SETTINGS
=
{ enable :
boolean ; priority ? :
number ; mandatory ? :
boolean ; } ;
```

mfaSettings

â

Minimum SCALE plan required This is a paid feature and the minimumpricing plan to use this SDK in a production environment is theSCALE Plan . You can use this feature in the development environment for free. Usage import

{

OpenloginAdapter

}

from

"@web3auth/openlogin-adapter" ;

const openloginAdapter =

new

OpenloginAdapter ({ loginSettings :

{ mfaLevel :

"mandatory" ,

// 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 , } , } , } , }) Edit this page Previous Custom Authentication Next Top-up Wallet