Authentication in MPC Core Kit Web SDK

There are two ways to login your users, depending on the type of authentication method you've chosen. If you are looking for an Implicit Authentication Flow in your application (like Auth0 SPA Flow), you can use theloginWithOAuth() function. If you are looking to pass a JWT based Id Token to the SDK from in your application (like Auth0 RWA Flow), you can use theloginWithJWT() function.

As a prerequisite, before triggering the login function, you need to create a verifier for your login method on the Web3Auth Dashboard .

Creating a Verifierâ

Since this a Core Kit SDK, it does not provide any default authentication methods. You need to create a custom verifier in order to use this SDK. This means that you need to authenticate users with your own custom authentication service. For example, while authenticating with Google, you have to use your own Google Client ID and Dashboard to authenticate users directly, or use aggregate services like Auth0, Firebase, AWS Cognito etc. Additionally, you can make your own JWT token authentication system and pass over the ID Token to the Web3Auth to generate a private key for them.

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 theCustom Authentication Documentation . warning Core Kit SDK only supports Sapphire Mainnet and Devnet networks. The other networks don't support MPC functionalities.

Log In with OAuth (Implicit Flow)â

loginWithOauth(loginParams: OauthLoginParams): Promise;

â

Variable Type Description Mandatory loginParams object Login Parameters Yes

OauthLoginParams

â

- Table
- Type

Parameter Type Description Mandatory subVerifierDetails SubVerifierDetails Details of the verifier the app needs to connect to. This is named a "sub verifier" to accommodate for aggregate verifiers as well. Yes aggregateVerifierIdentifier? string Identifier/ Name of the aggregate verifier you've made on the Web3Auth Dashboard. Not needed if you're connecting to a single verifier. No subVerifierDetailsArray? SubVerifierDetails[] Array of the details of verifiers the app needs to connect to. No export type OauthLoginParams

=

SubVerifierDetailsParams

|

AggregateVerifierLoginParams ;

export

interface

SubVerifierDetailsParams

extends

BaseLoginParams

{ subVerifierDetails :

SubVerifierDetails ; }

export

interface

AggregateVerifierLoginParams

extends

BaseLoginParams

{ aggregateVerifierIdentifier ? : string ; aggregateVerifierType ? :

AGGREGATE_VERIFIER_TYPE ; subVerifierDetailsArray ? :

SubVerifierDetails [ ] ; }

export

interface

BaseLoginParams

{ // offset in seconds serverTimeOffset ? : number ; }

export type AGGREGATE_VERIFIER_TYPE

=

(typeof

AGGREGATE_VERIFIER) [keyof typeof

AGGREGATE_VERIFIER] ;

export declare const

AGGREGATE_VERIFIER :

{ readonly SINGLE_VERIFIER_ID :

"single_id_verifier" ; } ;

SubVerifierDetails

â

Contains the details of verifier the app needs to connect to. Use this in case of a single verifier.

- Table
- Interface

Parameter Type Description Mandatory typeOfLogin 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. Yes verifier string Name/ Identifier of the verifier/ sub verifier in case of aggregate verifiers, you'd like your app to connect to. Yes clientId string Client Id given by the auth provider. Pass a random string in case you're connecting to a JWT based setup. Yes export

interface

SubVerifierDetails

{ typeOfLogin :

LOGIN_TYPE ; verifier : string ; clientId : string ; jwtParams ? :

Auth0ClientOptions ; hash ? : string ; queryParameters ? :

TorusGenericObject ; customState ? :

TorusGenericObject ; }

export type LOGIN_TYPE

=

(typeof

LOGIN) [keyof typeof

LOGIN] ;

export declare const

LOGIN :

{ readonly GOOGLE :

"google" ; readonly FACEBOOK :

"facebook" ; readonly REDDIT :

"reddit" ; readonly DISCORD :

"discord" ; readonly TWITCH :

"twitch" ; readonly APPLE :

"apple" ; readonly GITHUB :

"github" ; readonly LINKEDIN :

"linkedin" ; readonly TWITTER :

"twitter" ; readonly WEIBO :

"weibo" ; readonly LINE :

"line" ; readonly EMAIL_PASSWORD :

"email_password" ; readonly PASSWORDLESS :

"passwordless" ; readonly JWT :

"jwt" ; readonly WEBAUTHN :

"webauthn" ; } ;

AggregateVerifierLoginParams

â

Contains the details of an aggregate verifier the app needs to connect to. Use this in case of an aggregate verifier.

- Table
- Interface

Parameter Type Description Mandatory aggregateVerifierIdentifier string The name of your aggregate verifier Yes subVerifierDetailsArray SubVerifierDetails[] An array containing the details of your sub verifiers. Yes aggregateVerifierType? AGGREGATE_VERIFIER_TYPE What kind of aggregation is needed for your aggregate verifier. Use"single_id_verifier" by default in most cases No export

interface

AggregateVerifierLoginParams

extends

BaseLoginParams

{ aggregateVerifierIdentifier : string ; subVerifierDetailsArray :

SubVerifierDetails [ ] ; aggregateVerifierType ? :

AGGREGATE_VERIFIER_TYPE ; } export type AGGREGATE_VERIFIER_TYPE

=

(typeof

AGGREGATE_VERIFIER) [keyof typeof

AGGREGATE_VERIFIER] ;

Usageâ

General Verifierâ

import

{

Web3AuthMPCCoreKit ,

SubVerifierDetailsParams

}

from

"@web3auth/mpc-core-kit" ;

const verifierConfig =

{ subVerifierDetails :

{ typeOfLogin :

"google" , verifier :

"w3a-google-demo" , clientId :

"519228911939-cri01h55lsjbsia1k7ll6qpalrus75ps.apps.googleusercontent.com" , } , }

as

SubVerifierDetailsParams ;

await coreKitInstance . loginWithOauth (verifierConfig) ;

Aggregate Verifierâ

import

{

Web3AuthMPCCoreKit ,

AggregateVerifierLoginParams

}

from

"@web3auth/mpc-core-kit" ;

const verifierConfig =

{ aggregateVerifierIdentifier :

"aggregate-sapphire" , subVerifierDetailsArray :

[{ typeOfLogin :

"google" , verifier :

"w3a-google" , clientId :

"774338308167-q463s7kpvja16l4l0kko3nb925ikds2p.apps.googleusercontent.com" , } ,] , }

as

AggregateVerifierLoginParams ;

await coreKitInstance . loginWithOauth (verifierConfig) ;

Popup Modeâ

If you're using the popup mode (default) in your application, while logging in, you need to have a service worker running, which essential catches the login redirect and sends it back to the main DOM with the parameters. These parameters are then used to login the user with Web3Auth.

Service Workerâ

A service worker is a script that is run by the browser. It does not have any direct relationship with the DOM and provides many out of the box network-related features. Web3Auth Core Kit tKey SDK needs a service worker relative tobaseUrl to capture the auth redirect atredirectPathName path.

For example, while using service worker ifbaseUrl ishttp://localhost:3000/serviceworker then user will be redirected tohttp://localhost:3000/serviceworker/redirect page after login where service worker will capture the results and send it back to original window where login was initiated.

tip * Using service worker is required only in the popup flow. * Service worker is needed if you are usingpopup * uxMode within your MPC Core Kit Configuration. * For browsers where service workers are not supported, or in the case you wish to not use service workers, create and serve redirect page (i.eredirect.html * file).

Service Worker Setupâ

- If you're using React, to setup service worker, you need to create asw.js
- file in your public folder and register it in yourindex.html
- file.
- You can find more information about it inthis blog
- .
- For Angular,this guide
- will be helpful in setting up the service worker.
- For Vue,this guide
- is a great way to get
- started with service workers.

Service Worker Codeâ

You can directly copy the service worker file code from here and paste in your respective folder. You can also find the code in ourMPC Core Kit Example (Popup Flow) .

Service Worker Code /public/serviceworker/sw.js / *eslint-disable* / function

getScope ( )

{ return self . registration . scope ; }

self . addEventListener ("message" ,

function

(event)

{ if

(event . data

&& event . data . type

===

"SKIP_WAITING")

{ self . skipWaiting ( ) ; } }) ;

self . addEventListener ("fetch" ,

function

(event)

{ try

{ const url =

new

URL (event . request . url) ; if

(url . pathname . includes ("redirect")

&& url . href . includes (getScope ( )))

{ event . respondWith (new

Response (new

Blob ([`

{ "" } `

] , { type :

"text/html"

}))) ; } }

catch

(error)

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

Redirect Modeâ

If you are using the redirect mode, don't have to use service worker orredirect.html file. You can get login result by callinginit() function on redirected page mount.

For example, ifbaseUrl ishttp://localhost:3000 andredirectPathName isauth then user will be redirected tohttp://localhost:3000/auth page after login where you can get login result by callinginit() function on redirected page mount.

Log In with JWT (IdToken Flow)â

loginWithJWT(idTokenLoginParams: IdTokenLoginParams): Promise;

â

Variable Type Description Mandatory idTokenLoginParams object Login Parameters Yes

IdTokenLoginParams

â

- Table
- Type

Parameter Type Mandatory Description verifier string Yes Name of the verifier created on Web3Auth Dashboard. In case of Aggregate Verifier, the name of the top level aggregate verifier. verifierId string Yes Unique Identifier for the User. The verifier identifier field set for the verifier/ sub verifier. E.g. "sub" field in your on jwt id token. idToken string Yes The idToken received from the Auth Provider. subVerifier? string No Name of the sub verifier in case of aggregate verifier setup. This field is mandatory in case of an aggregate verifier. extraVerifierParams? WebAuthnExtraParams No Extra verifier params in case of a WebAuthn verifier type. additionalParams? ExtraParams No Any additional parameter (key value pair) you'd like to pass to the login function. export

interface

IdTokenLoginParams

{ /* * *Name of the verifier created on Web3Auth Dashboard. In case of Aggregate Verifier, the name of the top level aggregate verifier.* / verifier : string ;

/* * *Unique Identifier for the User. The verifier identifier field set for the verifier/ sub verifier. E.g. "sub" field in your on jwt id*

token. / verifierId : string ;

/* * *The idToken received from the Auth Provider.*/ idToken : string ;

/* * *Name of the sub verifier in case of aggregate verifier setup. This field should only be provided in case of an aggregate verifier.* / subVerifier ? : string ;

/* * *Extra verifier params in case of a WebAuthn verifier type*./ extraVerifierParams ? :

WebAuthnExtraParams ;

/* * *Any additional parameter (key value pair) you'd like to pass to the login function*/ additionalParams ? :

ExtraParams ; }

export

interface

ExtraParams

{ [key : string] : unknown ; }

export type WebAuthnExtraParams

=

{ signature ? : string , clientDataJSON ? : string , authenticatorData ? : string , publicKey ? : string , challenge ? : string , rpOrigin ? : string , credId ? : string , transports ? :

AuthenticatorTransport [ ] , } ;

type AuthenticatorTransport

=

"ble"

|

"hybrid"

|

"internal"

|

"nfc"

|

"usb" ;

Usageâ

General Verifierâ

import

{

Web3AuthMPCCoreKit , idTokenLoginParams }

from

"@web3auth/mpc-core-kit" ;

const loginRes =

await

signInWithGoogle ( ) ; const idToken =

await loginRes . user . getIdToken (true) ; const parsedToken =

parseToken (idToken) ;

const idTokenLoginParams =

{ verifier :

"w3a-firebase-demo" , verifierId : parsedToken . email , idToken , }

as

IdTokenLoginParams ;

await coreKitInstance . loginWithJWT (idTokenLoginParams) ;

Aggregate Verifierâ

import

{

Web3AuthMPCCoreKit , idTokenLoginParams }

from

"@web3auth/mpc-core-kit" ;

const loginRes =

await

signInWithGoogle ( ) ; const idToken =

await loginRes . user . getIdToken (true) ; const parsedToken =

parseToken (idToken) ;

const idTokenLoginParams =

{ verifier :

"aggregate-sapphire" subVerifier :

"w3a-google" , verifierId : parsedToken . email , idToken , }

as

IdTokenLoginParams ;

await coreKitInstance . loginWithJWT (idTokenLoginParams) ;

Logging out the Userâ

logout(): Promise;

â

Disconnect the user's connected wallet/ provider and log them out of the Web3Auth MPC Core Kit SDK.

Usageâ

await coreKitInstance . logout ( ) ;Edit this page Previous Initialize Next Usage