Usage of Core Kit SFA Web SDK

Once you've installed and successfully initialized Web3Auth, you can use it to authenticate your users. Further, you can use the native provider given by Web3Auth to connect the users to the respective blockchain network.

Natively, the instance of Web3Auth (referred to as web3auth in our examples) returns the following functions:

- connect()
-
  - Logs in the User using theverifier
- ,verifierId
- &idToken
- mandatorily whilesubVerifierInfoArray
- andserverTimeOffset
- are
- optional.
- provider()
-
  - Returns the native provider that can be used to make different blockchain transactions.
- sessionId()
-
  - Returns the sessionId of the user as a string.
- authenticateUser()
-
  - Returns a promise ofUserAuthInfo
- object containing theidToken
- of the user.
- addChain()
-
  - Add chain configuration to the web3auth instance.
- switchChain()
-
  - Switch the chainId to one of the added chainIds.
- logout()
-
  - Logs out the user from the web3auth instance.

Logging in your Userâ

connect(loginParams: LoginParams)

To login a user in the Web3Auth SFA Web SDK, you need to callconnect() function.

Variable Type Description Mandatory loginParams object Login Parameters Yes

Returnsâ

connect (loginParams :

LoginParams) :

Promise < IProvider

|

null

LoginParams

â

connect(loginParams: LoginParams)

On successful login, theconnect() function returns aIProvider instance. This instance contains the respective provider corresponding to your selected blockchain. You can use this provider to connect your user to the blockchain and make transactions.

On unsuccessful login, this function will return anull value.

LoginParams

â

- Table
- Type Declarations

Parameter Type Description Mandatory verifier string Details of the verifier (verifier type, ie.torus ,metamask ,openlogin etc.) Yes verifierId string Verifier ID's value,sub oremail value present in the idToken. Yes idToken string A newly createdJWT Token that has not already been sent to Web3Auth or aDuplicate Token error will be thrown. Yes subVerifierInfoArray? TorusSubVerifierInfo[] Sub verifier info No serverTimeOffset? number Server time offset No export

type

LoginParams

=

{ verifier :

string ; verifierId :

string ; idToken :

string ; subVerifierInfoArray ? : TorusSubVerifierInfo [ ] ; serverTimeOffset ? :

number ; } ;

TorusSubVerifierInfo

â

- Table
- Interface

Parameter Type Description verifier string Details of the verifier (verifier type, ie.torus ,metamask ,openlogin etc.) idToken string A newly createdJWT Token that has not already been sent to Web3Auth or aDuplicate Token error will be thrown. export

interface

TorusSubVerifierInfo

{ verifier :

string ; idToken :

string ; }

Usageâ

await web3auth . connect ({ verifier :

"verifier-name" ,

// e.g. web3auth-sfa-verifier replace with your verifier name, and it has to be on the same network passed in init(). verifierId :

"verifier-id-value" ,

// e.g. Yux1873xnibdui or name@email.com replace with your verifier id(sub or email)'s value. idToken :

"JWT Token" ,

// replace with your newly created unused JWT Token. }) ;

Get a native providerâ

provider()

Returns the native provider that can be used to make different blockchain transactions.

Returnsâ

get

provider ( ) :

IProvider

|

null ;

Get sessionIdâ

sessionId()

Returns the sessionId of the user as a string.

Returnsâ

get

sessionId ( ) : string ;

Authenticate the userâ

authenticateUser()

Returns a promise ofUserAuthInfo object containing theidToken of the user.

Returnsâ

authenticateUser ( ) :

Promise < UserAuthInfo

;

UserAuthInfo

â

export

type

UserAuthInfo

=

{ idToken :

string ; } ;

Add a new chainâ

addChain(chainConfig: CustomChainConfig)

Add chain configuration to the web3auth instance.

Variable Type Description Mandatory chainConfig CustomChainConfig Chain specific configuration Yes

CustomChainConfig

â

- Table
- Type Declarations

Parameter Type Description Optional chainNamespace * string * - (Choose betweeneip155 * ,solana * &other *) * or * ChainNamespaceType * - (Choose betweenCHAIN_NAMESPACES.EIP155 * ,CHAIN_NAMESPACES.SOLANA * &CHAIN_NAMESPACES.OTHER *) The namespace of your preferred chain. CheckoutProviders SDK Reference for understanding RPC Calls. No chainId string The chain id of the selected blockchain inhex No rpcTarget string * RPC Target URL for the selectedchainNamespace * &chainId * . * We provide a default RPC Target for certain blockchains, but due to congestion it might be slow hence it is recommended to provide your own RPC Target URL. No wsTarget string Web socket target URL for the chain Yes displayName string Display Name for the chain No blockExplorer string Blockchain's explorer URL. (eg:https://etherscan.io) No ticker string Default currency ticker of the network (e.g:ETH) No tickerName string Name for currency ticker (e.g:Ethereum) No decimals number Number of decimals for the currency ticker (e.g:18) Yes declare

const

CHAIN_NAMESPACES :

{ readonly

EIP155 :

"eip155" ; readonly

SOLANA :

"solana" ; readonly

OTHER :

"other" ; } ;

declare

type

ChainNamespaceType

=

(typeof

CHAIN_NAMESPACES) [keyof

typeof

CHAIN_NAMESPACES] ; declare

type

CustomChainConfig

=

{ chainNamespace : ChainNamespaceType ; /* * *The chain id of the chain*/ chainId :

string ; /* * *RPC target Url for the chain*/ rpcTarget :

string ; /* * *web socket target Url for the chain*/ wsTarget ? :

string ; /* * *Display Name for the chain*/ displayName :

string ; /* * *Url of the block explorer*/ blockExplorer :

string ; /* * *Default currency ticker of the network (e.g: ETH)*/ ticker :

string ; /* * *Name for currency ticker (e.g:Ethereum)* / tickerName :

string ; /* * *Number of decimals for the currency ticker (e.g: 18)*/ decimals ? :

number ; } ;

Switch the chainâ

switchChain(params: {chainId: string;})

Switch to one of the added chains

Variable Type Description Mandatory chainId string id of the chain to switch to yes

Logging out the userâ

logout()

Logs out the user and clears the session.

Note @web3auth/single-factor-auth SDK only works for users who havenot enabled MFA . MFA enabled users For MFA enabled users, you'll seeError("User has already enabled mfa, please use the @webauth/webauth-web sdk for login with mfa");

Exampleâ

Custom JWT Exampleâ

Custom JWT Example import

Web3Auth

from

"@web3auth/single-factor-auth" ; import

{

EthereumPrivateKeyProvider

}

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ; import

jwt

from

"jsonwebtoken" ; import

fs

from

"fs" ;

// Use "openssl genrsa -out privateKey.pem 2048" to generate a private key // Also, use this private key to generate a public key using "openssl rsa -in privateKey.pem -pubout -out publicKey.pem" // Convert PEM to JWKS and expose it on a public URL, and make a web3auth verifier using that. // Check out https://web3auth.io/docs/auth-provider-setup/byo-jwt-provider for more details. var privateKey = fs . readFileSync ("privateKey.pem") ;

// Define the chain config according to the chain you want to connect to const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

"https://etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , } ;

// Instantiate Web3Auth SFA SDK const web3authSfa =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

// Get your Client ID from the Web3Auth Dashboard web3AuthNetwork :

"sapphire_mainnet" ,

// Replace with your network name usePnPKey :

false ,

// Setting this to true returns the same key as PnP Web SDK, By default, this SDK returns CoreKitKey. }) ;

// Instantiate Ethereum Provider const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig }

}) ;

// Initialize Web3Auth SFA SDK web3authSfa . init (privateKeyProvider) ;

const user =

{ id :

"faj2720i2fdG7NsqznOKrthDvq43" ,

// must be unique to each user name :

"Mohammad Shahbaz Alam" , email :

"shahbaz@web3auth.io" , profileImage :

"https://avatars.githubusercontent.com/u/46641595?v=4" , } ;

// Login the user const web3authSfaprovider =

await web3auth . connect ({ verifier :

"web3auth-sfa-verifier" ,

// e.g. web3auth-sfa-verifier replace with your verifier name, and it has to be on the same network passed in init(). verifierId : user . id ,

// e.g. Yux1873xnibdui or name@email.com replace with your verifier id(sub or email)'s value. idToken : jwt . sign ({ sub : user . id ,

// must be unique to each user name : user . name , email : user . email , picture : user . profileImage , aud :

"urn:my-resource-server" ,

// -> to be used in Custom Authentication as JWT Field iss :

"https://my-authz-server" ,

// -> to be used in Custom Authentication as JWT Field iat :

Math . floor (Date . now ( )

/

1000) , exp :

Math . floor (Date . now ( )

/

1000)

+

60

*

60 , } , privateKey ,

// <-- privateKey to sign the JWT Token { algorithm :

"RS256" , keyid :

"1bb9605c36e69386830202b2d"

} ,) ,

// or replace it with your newly created unused JWT Token. }) ;

// Get the private key, The private key returned here is the CoreKit Key, since the usePnPKey is set to false. const ethPrivateKey =

await web3authSfaprovider . request ({ method :

"eth_private_key"

}) ;

console . log ("ETH Private Key" , ethPrivateKey) ;

Firebase JWT Exampleâ

Firebase JWT Example import

Web3Auth

from

"@web3auth/single-factor-auth" ; import

{

EthereumPrivateKeyProvider

}

from

"@web3auth/ethereum-provider" ; import

{

CHAIN_NAMESPACES

}

from

"@web3auth/base" ; import

{

GoogleAuthProvider , getAuth , signInWithPopup , UserCredential

}

from

"firebase/auth" ;

// Define the chain config according to the chain you want to connect to const chainConfig =

{ chainNamespace :

CHAIN_NAMESPACES . EIP155 , chainId :

"0x1" , rpcTarget :

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

"Ethereum Mainnet" , blockExplorer :

"https://etherscan.io" , ticker :

"ETH" , tickerName :

"Ethereum" , } ;

// Your web app's Firebase configuration const firebaseConfig =

{ apiKey :

"AIzaSyB0nd9YsPLu-tpdCrsXn8wgsWVAiYEpQ_E" , authDomain :

"web3auth-oauth-logins.firebaseapp.com" , projectId :

"web3auth-oauth-logins" , storageBucket :

"web3auth-oauth-logins.appspot.com" , messagingSenderId :

"461819774167" , appId :

"1:461819774167:web:e74addfb6cc88f3b5b9c92" , } ;

// Instantiate Web3Auth SFA SDK const web3authSfa =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

// Get your Client ID from the Web3Auth Dashboard web3AuthNetwork :

"sapphire_mainnet" ,

// Replace with your network name usePnPKey :

false ,

// Setting this to true returns the same key as PnP Web SDK, By default, this SDK returns CoreKitKey. }) ;

// Instantiate Ethereum Provider const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig }

}) ;

// Initialize Web3Auth SFA SDK web3authSfa . init (privateKeyProvider) ;

// Initialize Firebase and get idToken from Firebase const app =

initializeApp (firebaseConfig) ; const auth =

getAuth (app) ; const googleProvider =

new

GoogleAuthProvider ( ) ; const res =

await

signInWithPopup (auth , googleProvider) ; const idToken =

await res . user . getIdToken (true) ; // get sub value from firebase id token const

{ sub }

=

parseToken (token) ;

// parseToken is a function that parses the token and returns the payload.

// Login the user const web3authSfaprovider =

await web3auth . connect ({ verifier :

"web3auth-sfa-verifier" ,

// e.g. web3auth-sfa-verifier replace with your verifier name, and it has to be on the same network passed in init(). verifierId : sub ,

// e.g. Yux1873xnibdui or name@email.com replace with your verifier id(sub or email)'s value. idToken : idToken ,

// or replace it with your newly created unused JWT Token. }) ;

// Get the private key, The private key returned here is the CoreKit Key, since the usePnPKey is set to false. const ethPrivateKey =

await web3authSfaprovider . request ({ method :

"eth_private_key"

}) ;

console . log ("ETH Private Key" , ethPrivateKey) ;Edit this page Previous Authentication Next Overview