Using Core Kit SFA Node 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
- .

Logging in your Userâ

connect()

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

- Table
- Function Definition

Variable Type Description Mandatory loginParams object Login Parameters Yes async connect(loginParams: LoginParams): Promise { if (!this.torusUtils || !this.nodeDetailManager || !this.privKeyProvider) throw new Error("Please call init first"); const { verifier, verifierId, idToken, subVerifierInfoArray } = loginParams; const verifierDetails = { verifier, verifierId };

const { torusNodeEndpoints, torusIndexes, torusNodePub } = await this.nodeDetailManager.getNodeDetails(verifierDetails);

// does the key assign const pubDetails = await this.torusUtils.getUserTypeAndAddress(torusNodeEndpoints, torusNodePub, verifierDetails, true);

if (pubDetails.typeOfUser === "v1" || pubDetails.upgraded) { throw new Error("User has already enabled mfa, please use the @web3auth/web3auth-web sdk for login with mfa"); }

let finalIdToken = idToken; let finalVerifierParams = { verifier_id: verifierId }; if (subVerifierInfoArray && subVerifierInfoArray?.length > 0) { const aggregateVerifierParams: AggregateVerifierParams = { verify_params: [], sub_verifier_ids: [], verifier_id: "" }; const aggregateIdTokenSeeds = []; for (let index = 0; index < subVerifierInfoArray.length; index += 1) { const userInfo = subVerifierInfoArray[index]; aggregateVerifierParams.verify_params.push({ verifier_id: verifierId, idtoken: userInfo.idToken }); aggregateVerifierParams.sub_verifier_ids.push(userInfo.verifier); aggregateIdTokenSeeds.push(userInfo.idToken); } aggregateIdTokenSeeds.sort(); finalIdToken = keccak256(aggregateIdTokenSeeds.join(String.fromCharCode(29))).slice(2); aggregateVerifierParams.verifier_id = verifierId; finalVerifierParams = aggregateVerifierParams; }

const retrieveSharesResponse = await this.torusUtils.retrieveShares(torusNodeEndpoints, torusIndexes, verifier, finalVerifierParams, finalIdToken); const finalKey = subkey(retrieveSharesResponse.privKey.padStart(64, "0"), Buffer.from(this.options.clientId, "base64")); await this.privKeyProvider.setupProvider(finalKey.padStart(64, "0")); return this.privKeyProvider.provider; }

Returnsâ

connect (loginParams :

LoginParams) :

Promise < SafeEventEmitterProvider

|

null

> Note One needs to re-initialize the SDK for each new user. On successful login, theconnect() function returns aSafeEventEmitterProvider 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.

connect(loginParams: LoginParams)

â

LoginParams

- Table
- Type

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 andnot the stringsub oremail . 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 export type LoginParams

=

{ verifier : string , verifierId : string , idToken : string , subVerifierInfoArray ? :

TorusSubVerifierInfo [ ] , } ; Usage 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. }) ; 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 const

{

Web3Auth

}

=

require ("@web3auth/node-sdk") ; const

{

EthereumPrivateKeyProvider

}

=

require ("@web3auth/ethereum-provider") ; const jwt =

require ("jsonwebtoken") ; const fs =

require ("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") ;

const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig :

{ chainId :

"0xaa36a7" , rpcTarget :

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

"Ethereum Sepolia" , blockExplorer :

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

"ETH" , tickerName :

"Ethereum" , } , } , }) ;

// Instantiate Web3Auth Node.js SDK const web3auth =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

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

web3auth . init ({ provider : 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" , } ;

const

connect

=

async

( )

=>

{ const web3authNodeprovider =

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 , { algorithm :

"RS256" , keyid :

"1bb9605c36e69386830202b2d"

} ,) ,

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

await web3authNodeprovider . request ({ method :

"eth_private_key"

}) ; // The private key returned here is the CoreKitKey console . log ("ETH Private Key" , ethPrivateKey) ; } ; connect ( ) ;
Note One needs to re-initialize the SDK for each new user.

Provider JWT Exampleâ

JWT Example const

{

Web3Auth

}

=

require ("@web3auth/node-sdk") ; const

{

EthereumPrivateKeyProvider

}

=

require ("@web3auth/ethereum-provider") ;

const privateKeyProvider =

new

EthereumPrivateKeyProvider ({ config :

{ chainConfig :

{ chainId :

"0xaa36a7" , rpcTarget :

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

"Ethereum Sepolia" , blockExplorer :

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

"ETH" , tickerName :

"Ethereum" , } , } , }) ;

// Instantiate Web3Auth Node.js SDK const web3auth =

new

Web3Auth ({ clientId :

"WEB3AUTH_CLIENT_ID" ,

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

"sapphire_mainnet" , usePnPKey :

false ,

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

web3auth . init ({ provider : privateKeyProvider }) ;

const

connect

=

async

( )

=>

{ const web3authNodeprovider =

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 :

"verifier-id-value" ,

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

"JWT Token" ,

// or replace it with your newly created unused JWT Token you get from your auth provider. }) ; const ethPrivateKey =

await web3authNodeprovider . request ({ method :

"eth_private_key"

}) ; // The private key returned here is the CoreKitKey console . log ("ETH Private Key" , ethPrivateKey) ; } ; connect ( ) ;
Note One needs to re-initialize the SDK for each new user. Edit this page Previous Authentication Next Overview