Usage of Core Kit SFA React Native 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.
- connected()
-
    - Returnstrue
- orfalse
- depending on whether the web3auth instance is available or not.
- 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 log 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.

- 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 connected statusâ

connected()

Returnstrue orfalse depending on whether the web3auth instance is available or not.

Returnsâ

get

connected ( ) : boolean ;

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-react-native 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

- Expo app with Auth0 authentication
- Bare app with Firebase authentication

Note Please use these examples in conjunction with our troubleshooting guide for React Native Bundler polyfill issueshere .
App.js import

{

Auth0Provider , useAuth0 }

from

"react-native-auth0" ; import

*

as

SecureStore

from

"expo-secure-store" ; import

{ EthereumPrivateKeyProvider }

from

'@web3auth/ethereum-provider' ; import

Web3Auth

from

'@web3auth/single-factor-auth-react-native' ;

// Instantiate Web3Auth SFA SDK const web3AuthSfa =

```
new
Web3Auth ( SecureStore ,
{ clientId :
"WEB3AUTH_CLIENT_ID" ,
// Get your Client ID from the Web3Auth Dashboard web3AuthNetwork :
'sapphire_mainnet' , usePnPKey :
false ,
// By default, this sdk returns CoreKitKey } ) ;
// Instantiate Ethereum Provider const privateKeyProvider =
new
EthereumPrivateKeyProvider ( { config :
{ / pass the chain config that you want to connect with all chainConfig fields are required/ chainConfig :
{ chainId :
'0x1' , rpcTarget :
'https://rpc.ankr.com/eth' , displayName :
'mainnet' , blockExplorer :
'https://etherscan.io/' , ticker :
'ETH' , tickerName :
'Ethereum' , } , } , } ) ;
// Initialize Web3Auth SFA SDK await web3AuthSfa . init ( privateKeyProvider ) ;
// Get idToken from auth0 const
signInWithAuth0
=
async
( )
=>
{ try
{ await
authorize ( {
scope :
"openid profile email"
} ,
{
customScheme :
"auth0.com.web3authsfaauth0"
} ,
{
```

responseType :

"token id_token"

}) ; const credentials =

await

getCredentials ( ) ;

return credentials . idToken ; }

catch

(error)

{ console . error (error) ; } } ;

const idToken =

await

signInWithAuth0 ( ) ;

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

parseToken (idToken) ;

const verifier =

"web3auth-auth0-demo" ; const verifierId = parsedToken . sub ;

// Login the user const web3authProvider =

await web3auth ! . connect ({ verifier ,

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

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

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

await web3authProvider ! . request ({ method :

'eth_private_key' , }) ;

console . log ("ETH Private key" , finalPrivateKey) ; Note Please use these examples in conjunction with our troubleshooting guide for React Native Bundler polyfill issueshere . Note Please set up Firebase configuration in your React Native bare app using the guidehere . App.tsx import

auth

from

"@react-native-firebase/auth" ; import

EncryptedStorage

from

"react-native-encrypted-storage" ; import

{

EthereumPrivateKeyProvider

}

from

"@web3auth/ethereum-provider" ; import

Web3Auth

```
from
"@web3auth/single-factor-auth-react-native" ;
// Get idToken from Firebase async
function
signInWithEmailPassword ( )
{ try
{ const res =
await
auth ( ) . signInWithEmailAndPassword ( "custom+jwt@firebase.login" ,
"Testing@123" ) ; return res ; }
catch
( error )
{ console . error ( error ) ; } }
// Instantiate Web3Auth SFA SDK const web3AuthSfa =
new
Web3Auth ( EncryptedStorage ,
{ clientId :
"WEB3AUTH_CLIENT_ID" ,
// Get your Client ID from the Web3Auth Dashboard web3AuthNetwork :
"sapphire_mainnet" ,
// Get your network from the Web3Auth Dashboard usePnPKey :
false ,
// By default, this sdk returns CoreKitKey } ) ;
// Instantiate Ethereum Provider const privateKeyProvider =
new
EthereumPrivateKeyProvider ( { config :
{ / pass the chain config that you want to connect with all chainConfig fields are required/ chainConfig :
{ chainId :
"0x1" , rpcTarget :
"https://rpc.ankr.com/eth" , displayName :
"mainnet" , blockExplorer :
"https://etherscan.io/" , ticker :
"ETH" , tickerName :
"Ethereum" , } , } , } ) ;
// Initialize Web3Auth SFA SDK await web3AuthSfa . init ( privateKeyProvider ) ;
const loginRes =
await
```

signInWithEmailPassword ( ) ; const idToken =

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

parseToken (idToken) ;

const verifier =

"web3auth-firebase-examples" ; // get sub value from firebase id token const verifierId = parsedToken . sub ;

// Login the user const web3authProvider =

await web3auth ! . connect ({ verifier ,

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

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

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

await web3authProvider ! . request ({ method :

"eth_private_key" , }) ;Edit this page Previous Authentication Next Web3Auth Core Kit Single Factor Auth Flutter SDK