Initializing PnP React Native SDK

After Installation, the next step to use Web3Auth is to Initialize the SDK. The Initialization is a two-step process,

- Importing Required Packages
- Instantiating Web3Auth

Please note that these are the most critical steps where you need to pass on different parameters according to the preference of your project. Additionally, Whitelabeling and Custom Authentication have to be configured within this step, if you wish to customize your Web3Auth Instance.

Importing Required packagesâ

ImportingWeb3Auth

â

You may also import additional types from the SDK to help in the development process.

import Web3Auth ,

{

LOGIN_PROVIDER ,

OPENLOGIN_NETWORK

}

from

"@web3auth/react-native-sdk" ;

Importing aWebBrowser

implementationâ

- Expo Managed Workflow
- Bare React Native Workflow

import

*

as WebBrowser from

"expo-web-browser" ; import

*

as WebBrowser from

"@toruslabs/react-native-web-browser" ;

Importing aStorage

implementationâ

- Expo Managed Workflow
- Bare React Native Workflow

import

*

as SecureStore from

"expo-secure-store" ; import EncryptedStorage from

"react-native-encrypted-storage" ;

Instantiating Web3Authâ

NOTE Now withreact-native-sdk v4SecureStore orEncryptedStorage gets passed with the Web3Auth constructor, depending on the workflow you are using. This is due to the addition of session management without storing the private key in the device. * Expo Managed Workflow * Bare React Native Workflow

const web3auth =

new

Web3Auth (WebBrowser , SecureStore , SdkInitParams) ; const web3auth =

new

Web3Auth (WebBrowser , EncryptedStorage , SdkInitParams) ;

SdkInitParams

objectâ

TheWeb3Auth constructor in the React Native SDK takes anSdkInitParams object respectively as an argument. The fields of such objects are listed below.

- Table
- Interface

Parameter Description clientId Your Web3Auth Client ID. You can get it from Web3Auth Dashboard under project details. It's a mandatory field of typestring network Defines the Web3Auth network. It's a mandatory field of typeOPENLOGIN_NETWORK_TYPE . redirectUrl URL that Web3Auth will redirect API responses upon successful authentication from browser. It's a mandatory field of typestring . whiteLabel? WhiteLabel options for web3auth. It helps you define custom UI, branding, and translations for your brand app. It takesWhiteLabelData as a value. loginConfig? Login config for the custom verifiers. It takesLoginConfig as a value. useCoreKitKey? Use CoreKit Key to get core kit key. It's an optional field with default value asfalse . mfaSettings? Allows developers to configure the Mfa settings for authentication. It takesMfaSettings as a value. sessionTime? It allows developers to configure the session management time. Session Time is in seconds, default is 86400 seconds which is 1 day.sessionTime can be max 7 days storageServerUrl Specify a custom storage url. Default value ishttps://broadcast-server.tor.us . storageKey Specify the storage key. Setting to "local" will persist social login session accross browser tabs. Available options are "local" and "session". useMpc Whether to use openlogin MPC or not. Default value isfalse . export

type

SdkInitParams

=

{ /* * *You can get your clientId/projectId by registering your * dapp on {@link "https://dashboard.web3auth.io"| developer dashboard}* / clientId :

string ; network :

OPENLOGIN_NETWORK_TYPE ; /* * *This parameter will be used to change the build environment of openlogin sdk. * @defaultValue production* / buildEnv ? :

BUILD_ENV_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 ; /* * *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 ; /* * *Specify a custom storage server url * @defaultValue https://broadcast-server.tor.us * @internal* storageServerUrl ? :

string ; /* * *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 ; /* * *This parameter will be used to use openlogin mpc * @defaultValue false*/ useMpc ? :

boolean ; enableLogging ? :

boolean ; useCoreKitKey ? :

boolean ; } ;

Exampleâ

- Expo General
- Bare General
- Expo Custom Auth
- Bare Custom Auth

import

*

as

WebBrowser

from

"expo-web-browser" ; import

*

as

SecureStore

from

"expo-secure-store" ; import

Web3Auth ,

{

LOGIN_PROVIDER ,

OPENLOGIN_NETWORK

}

from

"@web3auth/react-native-sdk" ;

const resolvedRedirectUrl = Constants . appOwnership

==

AppOwnership . Expo

||

Constants . appOwnership

==

AppOwnership . Guest ?

```
Linking . createURL ( "web3auth" ,
{ } ) :
Linking . createURL ( "web3auth" ,
{ scheme : scheme } ) ;
const clientId =
"YOUR WEB3AUTH CLIENT ID" ;
const web3auth =
new
Web3Auth ( WebBrowser ,
SecureStore ,
{ clientId , network :
OPENLOGIN_NETWORK . SAPPHIRE_MAINNET ,
// or other networks } ) ; import
*
as
WebBrowser
from
"@toruslabs/react-native-web-browser" ; import
EncryptedStorage
from
"react-native-encrypted-storage" ; import
Web3Auth ,
{
LOGIN_PROVIDER ,
OPENLOGIN_NETWORK
}
from
"@web3auth/react-native-sdk" ;
const scheme =
"web3authrnbareauth0example" ;
// Or your desired app redirection scheme const resolvedRedirectUrl =
{ scheme } ://openlogin ;
const clientId =
"YOUR WEB3AUTH CLIENT ID" ;
const web3auth =
new
Web3Auth ( WebBrowser ,
```

```
EncryptedStorage ,
{ clientId , network :
OPENLOGIN_NETWORK . SAPPHIRE_MAINNET ,
// or other networks } ) ; import
*
as
WebBrowser
from
"expo-web-browser" ; import
*
as
SecureStore
from
"expo-secure-store" ; import
Web3Auth ,
{
LOGIN_PROVIDER ,
OPENLOGIN_NETWORK
}
from
"@web3auth/react-native-sdk" ;
const resolvedRedirectUrl = Constants . appOwnership
==
AppOwnership . Expo
||
Constants . appOwnership
==
AppOwnership . Guest ?
Linking . createURL ( "web3auth" ,
{ } ) :
Linking . createURL ( "web3auth" ,
{ scheme : scheme } ) ;
const clientId =
"YOUR WEB3AUTH CLIENT ID" ;
const web3auth =
new
Web3Auth ( WebBrowser ,
```

```
SecureStore ,
{ clientId , network :
OPENLOGIN_NETWORK . SAPPHIRE_MAINNET ,
// or other networks loginConfig :
{ jwt :
{ name :
"Web3Auth-Auth0-JWT" , verifier :
"web3auth-auth0-example" ,
// Verifier's name from Web3Auth Dashboard typeOfLogin :
"jwt" , clientId :
"294QRkchfq2YaXUbPri7D6PH7xzHgQMT" ,
// Auth0 Client ID } , } , } ) ; import
*
as
WebBrowser
from
"@toruslabs/react-native-web-browser" ; import
EncryptedStorage
from
"react-native-encrypted-storage" ; import
Web3Auth ,
{
LOGIN_PROVIDER ,
OPENLOGIN_NETWORK
}
from
"@web3auth/react-native-sdk" ;
const scheme =
"web3authrnbareauth0example" ;
// Or your desired app redirection scheme const resolvedRedirectUrl =
{ scheme } ://openlogin ;
const clientId =
"YOUR WEB3AUTH CLIENT ID" ;
const web3auth =
new
Web3Auth ( WebBrowser ,
EncryptedStorage ,
```

{ clientId , network :

OPENLOGIN_NETWORK . SAPPHIRE_MAINNET ,

// or other networks loginConfig :

{ jwt :

{ name :

"Web3Auth-Auth0-JWT" , verifier :

"web3auth-auth0-example" ,

// Verifier's name from Web3Auth Dashboard typeOfLogin :

"jwt" , clientId :

"294QRkchfq2YaXUbPri7D6PH7xzHgQMT" ,

// Auth0 Client ID } , } , }) ;Edit this page Previous Install Next Usage