Usage of PnP Flutter SDK

Logging in a Userâ

login(LoginParams)

â

Trigger login flow will navigate the user to a browser model allowing the user to login into the service. You can pass in the supported providers to the login method for specific social logins such as GOOGLE, APPLE, FACEBOOK, etc., and do whitelabel login. Thelogin function takes inLoginParams as a required input.

Argumentsâ

LoginParams

- Table
- Class

Parameter Description loginProvider It sets the oAuth login method to be used. You can use any of the supported values areGOOGLE ,FACEBOOK ,REDDIT ,DISCORD ,TWITCH ,APPLE ,LINE ,GITHUB ,KAKAO ,LINKEDIN ,TWITTER ,WEIBO ,WECHAT ,EMAIL_PASSWORDLESS . extraLoginOptions? It can be used to set the OAuth login options for correspondingloginProvider . For instance, you'll need to pass user's email address as. Default value for the field isnull , and it acceptsExtraLoginOptions as a value. redirectUrl? Url where user will be redirected after successfull login. By default user will be redirected to same page where login will be initiated. Default value for the field isnull appState? It can be used to keep track of the app state when user will be redirected to app after login. Default isnull , and acceptsString as a value. mfaLevel? Customize the MFA screen shown to the user during OAuth authentication. Default value for field isMFALevel.DEFAULT , which shows MFA screen every 3rd login. It acceptsMFALevel as a value. dappShare? Custom verifier logins can get a dapp share returned to them post successful login. This is useful if the dapps want to use this share to allow users to login seamlessly. It acceptsString as a value. curve? It will be used to determine the public key encoded in the jwt token which returned ingetUserInfo function after user login. This parameter won't change format of private key returned by We3Auth. Private key returned bygetPrivKey is always secp256k1. To get the ed25519 key you can usegetEd25519PrivKey method. The default value isCurve.secp256k1 . class

LoginParams

{ final

Provider loginProvider ; final

String ? dappShare ; final

Curve ? curve ; final

ExtraLoginOptions ? extraLoginOptions ; final

Uri ? redirectUrl ; final

String ? appState ; final

MFALevel ? mfaLevel ;

LoginParams ({ required this . loginProvider , this . dappShare , this . curve , this . extraLoginOptions , this . redirectUrl , this . appState , this . mfaLevel , }) ;

Map < String ,

dynamic

toJson ( )

=

{ "loginProvider" : loginProvider . name , "dappShare" : dappShare , "curve" : curve ? . name , "extraLoginOptions" : extraLoginOptions ? . toJson ( ) , "redirectUrl" : redirectUrl ? . toString ( ) , "appState" : appState , "mfaLevel" : mfaLevel ? . type , } ; }

getPrivkey()

â

Use getPrivkey() to get the private key of the user. The method returns an EVM compatible private key which can be used to sign transactions on EVM compatible chains.

getEd25519PrivKey()

â

Use getEd25519PrivKey() to get the Ed25519 private key of the user. This private key can be used to sign transactions on Solana.

getUserInfo()

â

Use getUserInfo() to get the user info of the user.

UserInfo Responseâ

{ "userInfo": { "email": "w3a-heroes@web3auth.com", "name": "Web3Auth Heroes", "profileImage": "https://lh3.googleusercontent.com/a/Ajjjsdsmdjmnm...", "verifier": "torus", "verifierId": "w3a-heroes@web3auth.com", "typeOfLogin": "google", "aggregateVerifier": "w3a-google-sapphire", "dappShare": "", // 24 words of seed phrase will be sent only incase of custom verifiers "idToken": "", "oAuthIdToken": "", // will be sent only incase of custom verifiers "oAuthAccessToken": "", // will be sent only incase of custom verifiers "isMfaEnabled": false // Returns whether the user has enabled MFA or not } } * Google * Facebook * Discord * Twitch * Email Passwordless * JWT

Usage Future < void

initWeb3Auth ( )

async

{ Uri redirectUrl ;

if

(Platform . isAndroid)

{ redirectUrl =

Uri . parse ('{SCHEME}://{HOST}/auth') ; // w3a://com.example.w3aflutter/auth }

else

if

(Platform . isIOS)

{ redirectUrl =

Uri . parse ('{bundleId}://auth') ; // com.example.w3aflutter://openlogin }

else

{ throw

UnKnownException ('Unknown platform') ; }

await

Web3AuthFlutter . init (Web3AuthOptions (clientId :

"WEB3AUTH_CLIENT_ID" , network :

Network . sapphire_mainnet , redirectUrl : redirectUrl ,)) ;

await

Web3AuthFlutter . initialize ( ) ; }

// Login final

```
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . google ) ) ; Usage Future < void
initWeb3Auth ( )
async
{
Uri redirectUrl ; if
( Platform . isAndroid )
{ redirectUrl =
Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }
else
if
( Platform . isIOS )
{ redirectUrl =
Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }
else
{ throw
UnKnownException ( 'Unknown platform' ) ; }
await
Web3AuthFlutter . init ( Web3AuthOptions ( clientId :
"WEB3AUTH_CLIENT_ID" , network :
Network . sapphire_mainnet , redirectUrl : redirectUrl , ) ) ;
await
Web3AuthFlutter . initialize ( ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . facebook ) ) ; Usage Future < void
initWeb3Auth ( )
async
{
Uri redirectUrl ; if
( Platform . isAndroid )
{ redirectUrl =
Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }
else
if
( Platform . isIOS )
{ redirectUrl =
Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }
await
Web3AuthFlutter . init ( Web3AuthOptions ( clientId :
"WEB3AUTH_CLIENT_ID" , network :
Network . sapphire_mainnet , redirectUrl : redirectUrl , ) ) ;
await
Web3AuthFlutter . initialize ( ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . discord ) ) ; Usage Future < void
initWeb3Auth ( )
async
{
Uri redirectUrl ; if
( Platform . isAndroid )
{ redirectUrl =
Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }
else
if
( Platform . isIOS )
{ redirectUrl =
Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }
else
{ throw
UnKnownException ( 'Unknown platform' ) ; }
await
Web3AuthFlutter . init ( Web3AuthOptions ( clientId :
"WEB3AUTH_CLIENT_ID" , network :
Network . sapphire_mainnet , redirectUrl : redirectUrl , ) ) ;
await
Web3AuthFlutter . initialize ( ) ; }
```

```
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . twitch ) ) ; Usage Future < void
initWeb3Auth ( )
async
{ final additionalParams =
HashMap < String ,
String
      ( ) ;
// Default is 'code' additionalParams [ 'flow_type' ]
=
"link" ;
Uri redirectUrl ; if
( Platform . isAndroid )
{ redirectUrl =
Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }
else
if
( Platform . isIOS )
{ redirectUrl =
Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }
else
{ throw
UnKnownException ( 'Unknown platform' ) ; }
await
Web3AuthFlutter . init ( Web3AuthOptions ( clientId :
"WEB3AUTH_CLIENT_ID" , network :
Network . testnet , redirectUrl : redirectUrl , ) ) ;
await
Web3AuthFlutter . initialize ( ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . email_passwordless , extraLoginOptions :
```

```
ExtraLoginOptions ( login_hint :
"hello@web3auth.io" , additionalParams : additionalParams ) ) , ) ; Usage Future < void
initPlatformState ( )
async
{
final loginConfig =
new
HashMap < String ,
LoginConfigItem
        ( ) ; loginConfig [ 'jwt' ]
=
LoginConfigItem ( // get it from web3auth dashboard verifier :
"verifier-name" , typeOfLogin :
TypeOfLogin . jwt , // Auth0, Google, Facebook, Twitch, Discord or Web3Auth Client ID client_id :
"CLIENT_ID" , ) ;
Uri redirectUrl ; if
( Platform . isAndroid )
{ redirectUrl =
Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }
else
if
( Platform . isIOS )
{ redirectUrl =
Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }
else
{ throw
UnKnownException ( 'Unknown platform' ) ; }
await
Web3AuthFlutter . init ( Web3AuthOptions ( clientId :
"WEB3AUTH_CLIENT_ID" , network :
Network . testnet , redirectUrl : redirectUrl , , loginConfig : loginConfig ) ) ;
await
Web3AuthFlutter . initialize ( ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
```

Provider . jwt , extraLoginOptions :

ExtraLoginOptions (id_token :

"YOUR_JWT_TOKEN" ,))) ;

Selecting Curveâ

TheLoginParams class has acurve parameter. This parameter can be used to select the elliptic curve to use for the signature.

final

Web3AuthResponse response =

await

Web3AuthFlutter . login (LoginParams (loginProvider :

Provider . google , curve :

Curve . secp256k1 // Can be ed25519, secp256k1)) ; * SECP256K1 * ED25519

Usage Future < void

initWeb3Auth ( )

async

{

Uri redirectUrl ; if

(Platform . isAndroid)

{ redirectUrl =

Uri . parse ('{SCHEME}://{HOST}/auth') ; // w3a://com.example.w3aflutter/auth }

else

if

(Platform . isIOS)

{ redirectUrl =

Uri . parse ('{bundleId}://auth') ; // com.example.w3aflutter://openlogin }

else

{ throw

UnKnownException ('Unknown platform') ; }

await

Web3AuthFlutter . init (Web3AuthOptions (clientId :

"WEB3AUTH_CLIENT_ID" , network :

Network . sapphire_mainnet , redirectUrl : redirectUrl ,)) ;

await

Web3AuthFlutter . initialize ( ) ; }

final

Web3AuthResponse response =

await

```
Web3AuthFlutter . login ( LoginParams ( loginProvider :

Provider . google , curve :

Curve . secp256k1 // Can be ed25519, secp256k1 ) ) ; Usage Future < void

initWeb3Auth ( )

async

{

Uri redirectUrl ; if

( Platform . isAndroid )

{ redirectUrl =

Uri . parse ( '{SCHEME}://{HOST}/auth' ) ; // w3a://com.example.w3aflutter/auth }

else

if

( Platform . isIOS )

{ redirectUrl =

Uri . parse ( '{bundleId}://auth' ) ; // com.example.w3aflutter://openlogin }

else

{ throw

UnKnownException ( 'Unknown platform' ) ; }

await

Web3AuthFlutter . init ( Web3AuthOptions ( clientId :

"WEB3AUTH_CLIENT_ID" , network :

Network . testnet , redirectUrl : redirectUrl ) ) ;

await

Web3AuthFlutter . initialize ( ) ; }

final

Web3AuthResponse response =

await

Web3AuthFlutter . login ( LoginParams ( loginProvider :

Provider . google , curve :

Curve . ed25519 // Can be ed25519, secp256k1 ) ) ;
```

Logging out a userâ

logout()

â

This method will logout the user and remove the session id from the device. The user will need to login again to use the dApp next time the dApp is opened.

```
await

Web3AuthFlutter . logout ( ) ;
```
sample-app Get started with a sample app found here .

Triggering Login exceptionsâ

setResultUrl()

â

This method will trigger login exceptions for Android. For iOS, you don't need this method to trigger the login exceptions. The Android SDK uses the custom tabs and from current implementation of chrome custom tab, it's not possible to add a listener directly to chrome custom tab close button and trigger login exceptions.

Hence, it's necessary to usersetResultUrl method in your login screen to trigger exceptions.

```
class

LoginScreen

extends

State < T

with

WidgetsBindingObserver

{

@override void

initState ( )

{ super . initState ( ) ; WidgetsBinding . instance . addObserver ( this ) ; }

@override void

dispose ( )

{ super . dispose ( ) ; WidgetsBinding . instance . removeObserver ( this ) ; }

@override void

didChangeAppLifecycleState ( final

AppLifecycleState state )

{ // This is important to trigger the user cancellation on Android. if

( state ==

AppLifecycleState . resumed )

{ Web3AuthFlutter . setResultUrl ( ) ; } }

@override Widget

build ( BuildContext context )

{ // Your UI code }

Future < void

_login ( )

async

{ try

{ await

Web3AuthFlutter . login ( LoginParams ( loginProvider :

Provider . google ) ) ; }

on
```

UserCancelledException

{ log ("User cancelled.") ; }

on

UnKnownException

{ log ("Unknown exception occurred") ; }

catch

(e)

{ log (e . toString ( )) ; } }Edit this page Previous Initialize Next Whitelabel