Using Custom Authentication in PnP Flutter SDK

To use custom authentication (Using Social providers or Login providers like Auth0, AWS Cognito, Firebase etc. or even your own custom JWT login) you can add the configuration to theloginConfig parameter of theLoginParams object during the initialization.

TheloginConfig parameter is a key value map. The key should be one of theTypeOfLogin in its string form, and the value should be aLoginConfigItem struct.

To use custom authentication, first you'll need to configure your own verifier in the Web3Auth Dashboard in Custom Authentication section.

Create Custom Verifier Check out how to create aCustom Verifier on the Web3Auth Dashboard. using dapp share * dApp Share is only returned for the Custom verifiers. * Also, 2FA should be enabled for the account using it. UsemfaLevel = MFALevel.MANDATORY * in theLoginParams * during login. SeeMFA * for more details. note This is a paid feature and the minimumpricing plan to use this SDK in a production environment is theGrowth Plan . You can use this feature in the development environment for free. Then, specify the details of your verifier in theLoginConfigItem struct as follows:

LoginConfigItem

â

Argumentsâ

LoginConfigItem

- Table
- Class

Parameter Description verifier The name of the verifier that you have registered on the Web3Auth Dashboard. It's a mandatory field, and acceptsString as a value. typeOfLogin Type of login of this verifier, this value will affect the login flow that is adapted. For example, if you choosegoogle , a Google sign-in flow will be used. If you choosejwt , you should be providing your own JWT token, no sign-in flow will be presented. It's a mandatory field, and acceptsTypeOfLogin as a value. clientId Client id provided by your login provider used for custom verifier. e.g. Google's Client ID or Web3Auth's client Id if using 'jwt' as TypeOfLogin. It's a mandatory field, and acceptsString as a value. name? Display name for the verifier. If null, the default name is used. It acceptsString as a value. description? Description for the button. If provided, it renders as a full length button. else, icon button. It acceptsString as a value. verifierSubIdentifier? The field in JWT token which maps to verifier id. Please make sure you selected correct JWT verifier id in the developer dashboard. It acceptsString as a value. logoHover? Logo to be shown on mouse hover. It acceptsString as a value. logoLight? Light logo for dark background. It acceptsString as a value. logoDark? Dark logo for light background. It acceptsString as a value. mainOption? Show login button on the main list. It acceptsbool as a value. showOnModal? Whether to show the login button on modal or not. showOnDesktop? Whether to show the login button on desktop. showOnMobile? Whether to show the login button on mobile. class

LoginConfigItem

{ final

String verifier ; final

TypeOfLogin typeOfLogin ; final

String clientId ; final

String ? name ; final

String ? description ; final

String ? verifierSubIdentifier ; final

String ? logoHover ; final

String ? logoLight ; final

String ? logoDark ; final bool ? mainOption ; final bool ? showOnModal ; final bool ? showOnDesktop ; final bool ? showOnMobile ;

LoginConfigItem ({ required this . verifier , required this . typeOfLogin , required this . clientId , this . name , this . description , this . verifierSubIdentifier , this . logoHover , this . logoLight , this . logoDark , this . mainOption , this . showOnModal , this .

showOnDesktop , this . showOnMobile , }) ;

Map < String ,

dynamic

toJson ( )

{ return

{ 'verifier' : verifier , 'typeOfLogin' : typeOfLogin . name , 'clientId' : clientId , 'name' : name , 'description' : description , 'verifierSubIdentifier' : verifierSubIdentifier , 'logoHover' : logoHover , 'logoLight' : logoLight , 'logoDark' : logoDark , 'mainOption' : mainOption , 'showOnModal' : showOnModal , 'showOnDesktop' : showOnDesktop , 'showOnMobile' : showOnMobile } ; } }

typeOfLogin

â

enum

TypeOfLogin

{ google , facebook , reddit , discord , twitch , github , apple , kakao , linkedin , twitter , weibo , wechat , line , email_passwordless , email_password , jwt } * Google * Facebook * JWT

Usage Future < void

initWeb3Auth ( )

async

{ final themeMap =

HashMap < String ,

String

        ( ) ; themeMap ['primary']

=

"#229954" ;

final loginConfig =

new

HashMap < String ,

LoginConfigItem

        ( ) ; loginConfig ['google']

=

LoginConfigItem (verifier :

"verifier-name" ,

// get it from web3auth dashboard typeOfLogin :

TypeOfLogin . google , clientId :

"google_client_id"

// google's 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 . sapphire_mainnet , redirectUrl : redirectUrl , loginConfig : loginConfig ) , ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . google ) ) ; Usage Future < void
initWeb3Auth ( )
async
{ final themeMap =
HashMap < String ,
String
     ( ) ; themeMap [ 'primary' ]
=
"#229954" ;
final loginConfig =
new
HashMap < String ,
LoginConfigItem
     ( ) ; loginConfig [ 'facebook' ]
=
LoginConfigItem ( verifier :
"verifier-name" ,
// get it from web3auth dashboard typeOfLogin :
TypeOfLogin . facebook , clientId :
"facebook_client_id"
```

```
// facebook's 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 ) , ) ; }
// Login final
Web3AuthResponse response =
await
Web3AuthFlutter . login ( LoginParams ( loginProvider :
Provider . facebook ) ) ; Usage Future < void
initWeb3Auth ( )
async
{ final themeMap =
HashMap < String ,
String
        ( ) ; themeMap [ 'primary' ]
=
"#229954" ;
final loginConfig =
new
HashMap < String ,
LoginConfigItem
        ( ) ; loginConfig [ 'jwt' ]
=
LoginConfigItem ( verifier :
```

"verifier-name" ,

// get it from web3auth dashboard typeOfLogin :

TypeOfLogin . jwt , clientId :

"web3auth_client_id"

// web3auth's plug and play 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) ,) ; }

// Login final

Web3AuthResponse response =

await

Web3AuthFlutter . login (LoginParams (loginProvider :

Provider . jwt , extraLoginOptions :

ExtraLoginOptions (id_token :

"YOUR_JWT_TOKEN"))) ;

ExtraLoginOptions

for special login methodsâ

Additional to theLoginConfig you can pass extra options to thelogin function to configure the login flow for cases requiring additional info for enabling login. TheExtraLoginOptions accepts the following parameters:

- Table
- Interface

Parameter Description additionalParams? Additional params inMap format for OAuth login, use id_token(JWT) to authenticate with web3auth. domain? Your custom authentication domain inString format. For example, if you are using Auth0, it can be example.au.auth0.com. client_id? Client id inString format, provided by your login provider used for custom verifier. leeway? The value used to account for clock skew in JWT expirations. The value is in the seconds, and ideally should no more than 60 seconds or 120 seconds at max. It takesString as a value. verifierIdField? The field in JWT token which maps to verifier id. Please make sure you selected correct JWT verifier id in the developer dashboard. It takesString as a value. isVerifierIdCaseSensitive? Boolean to confirm Whether the verifier id field is case sensitive or not. display?

Allows developers the configure the display of UI. It takesDisplay as a value. prompt? Prompt shown to the user during authentication process. It takesPrompt as a value. max_age? Max time allowed without reauthentication. If the last time user authenticated is greater than this value, then user must reauthenticate. It takesString as a value. ui_locales? The space separated list of language tags, ordered by preference. For instancefr-CA fr en . id_token_hint? It denotes the previously issued ID token. It takesString as a value. id_token? JWT (ID Token) to be passed for login. login_hint? It is used to send the user's email address during Email Passwordless login. It takesString as a value. acr_values? acc_values scope? The default scope to be used on authentication requests. The defaultScope defined in the Auth0Client is included along with this scope. It takesString as a value. audience? The audience, presented as the aud claim in the access token, defines the intended consumer of the token. It takesString as a value. connection? The name of the connection configured for your application. If null, it will redirect to the Auth0 Login Page and show the Login Widget. It takesString as a value. state? state response_type? Defines which grant to execute for the authorization server. It takesString as a value. nonce? nonce redirect_uri? It can be used to specify the default url, where your custom jwt verifier can redirect your browser to with the result. If you are using Auth0, it must be whitelisted in the Allowed Callback URLs in your Auth0's application. class

ExtraLoginOptions

{ final

Map ? additionalParams ; final

String ? domain ; final

String ? client_id ; final

String ? leeway ; final

String ? verifierIdField ; final bool ? isVerifierIdCaseSensitive ; final

Display ? display ; final

Prompt ? prompt ; final

String ? max_age ; final

String ? ui_locales ; final

String ? id_token_hint ; final

String ? id_token ; final

String ? login_hint ; final

String ? acr_values ; final

String ? scope ; final

String ? audience ; final

String ? connection ; final

String ? state ; final

String ? response_type ; final

String ? nonce ; final

String ? redirect_uri ;

ExtraLoginOptions ({ this . additionalParams =

const

{ } , this . domain , this . client_id , this . leeway , this . verifierIdField , this . isVerifierIdCaseSensitive , this . display , this . prompt , this . max_age , this . ui_locales , this . id_token_hint , this . id_token , this . login_hint , this . acr_values , this . scope , this . audience , this . connection , this . state , this . response_type , this . nonce , this . redirect_uri , }) ;

Map < String ,

dynamic

toJson ( )

=

{ "additionalParams" : additionalParams , "domain" : domain , "client_id" : client_id , "leeway" : leeway , "verifierIdField" : verifierIdField , "isVerifierIdCaseSensitive" : isVerifierIdCaseSensitive , "display" : display ? . name , "prompt" : prompt ? . name , "max_age" : max_age , "ui_locales" : ui_locales , "id_token_hint" : id_token_hint , "id_token" : id_token , "login_hint" : login_hint , "acr_values" : acr_values , "scope" : scope , "audience" : audience , "connection" : connection , "state" : state , "response_type" : response_type , "nonce" : nonce , "redirect_uri" : redirect_uri , } ; }

Using Auth0 Loginâ

Auth0 has a special login flow, called the SPA flow. This flow requires aclient_id anddomain to be passed, and Web3Auth will get the JWTid_token from Auth0 directly. You can pass these configurations in theExtraLoginOptions object in thelogin function.

```
Future < void

initWeb3Auth ( )

async

{ final themeMap =

HashMap < String ,

String

        ( ) ; themeMap [ 'primary' ]

=

"#229954" ;

final loginConfig =

new

HashMap < String ,

LoginConfigItem

        ( ) ; loginConfig [ 'jwt' ]

=

LoginConfigItem ( verifier :

"verifier-name" ,

// get it from web3auth dashboard for auth0 configuration typeOfLogin :

TypeOfLogin . jwt , clientId :

"auth0_client_id"

// get it from auth0 dashboard ) ;

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 , loginConfig : loginConfig ,) ,) ; }

// Login final

Web3AuthResponse response =

await

Web3AuthFlutter . login (LoginParams (loginProvider :

Provider . jwt , extraLoginOptions :

ExtraLoginOptions (domain :

"https://tenant-name.us.auth0.com" ,

// Domain of your auth0 app verifierIdField :

"sub" ,

// The field in jwt token which maps to verifier id.))) ;

Custom JWT Loginâ

If you're using any other provider like Firebase, AWS Cognito or deploying your own Custom JWT server, you need to put the jwt token into theid_token parameter of theExtraLoginOptions .

Future < void

initWeb3Auth ( )

async

{ final themeMap =

HashMap < String ,

String

    ( ) ; themeMap ['primary']

=

"#229954" ;

final loginConfig =

new

HashMap < String ,

LoginConfigItem

    ( ) ; loginConfig ['jwt']

=

LoginConfigItem (verifier :

"verifier-name" ,

// get it from web3auth dashboard typeOfLogin :

TypeOfLogin . jwt ,) ;

```
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 , loginConfig : loginConfig , ) , ) ; }

// Login final

Web3AuthResponse response =

await

Web3AuthFlutter . login ( LoginParams ( loginProvider :

Provider . jwt , extraLoginOptions :

ExtraLoginOptions ( id_token :

"YOUR_ID_TOKEN" , ) ) ) ;
```

Email Passwordlessâ

To use theEMAIL_PASSWORDLESS login, you need to put the email into thelogin_hint parameter of theExtraLoginOptions . By default, the login flow will becode flow, if you want to use thelink flow, you need to putflow_type into theadditionalParams parameter of theExtraLoginOptions .

```
Future < void

initWeb3Auth ( )

async

{ final themeMap =

HashMap < String ,

String

     ( ) ; themeMap [ 'primary' ]

=

"#229954" ;

final additionalParams =

HashMap < String ,
String

    ( ) ; additionalParams [ 'flow_type' ]

=

"link" ;

// default is 'code'

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 , ) , ) ; }

// Login final

Web3AuthResponse response =

await

Web3AuthFlutter . login ( LoginParams ( loginProvider :

Provider . email_passwordless , extraLoginOptions :

ExtraLoginOptions ( login_hint :

"hello@web3auth.io" , additionalParams : additionalParams ) , ) , ) ;
```
Edit this page Previous Whitelabel Next Multi Factor Authentication