

Using of PnP Unity SDK

Once you've installed and successfully initialized Web3Auth, you can use it to authenticate your users.

Logging in a User

web3Auth.login(LoginParams)

[â](#)

This function helps your user to trigger the login process. The login flow will be triggered based on the selected provider. You can view the list of selected providers in the [provider](#) section.

tip Additionally, you can associate a function to be triggered on successful login. This function will receive the user's profile as a parameter.

void

Start () { web3Auth =

GetComponent < Web3Auth

() ; web3Auth . setOptions (new

Web3AuthOptions () {

}) ; web3Auth . onLogin += onLogin ; } private

void

onLogin (Web3AuthResponse response) { // Functions to be called after logging in your user }

Arguments

web3Auth.login() requires LoginParams as a required input.

LoginParams

[â](#)

- Table
- Interface

Parameter Description loginProvider It sets the OAuth login method to be used. You can use any of the supported values are GOOGLE , 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 corresponding loginProvider . For instance, you'll need to pass user's email address as. Default value for the field is null , and it accepts ExtraLoginOptions as a value. redirectUrl? Url where user will be redirected after successful login. By default user will be redirected to same page where login will be initiated. Default value for the field is null , and accepts Uri as a value. appState? It can be used to keep track of the app state when user will be redirected to app after login. Default is null , and accepts string as a value. mfaLevel? Customize the MFA screen shown to the user during OAuth authentication. Default value for field is MFALevel.DEFAULT , which shows MFA screen every 3rd login. It accepts MFALevel 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 accepts string as a value. curve? It will be used to determine the public key encoded in the jwt token which returned in getUserInfo function after user login. This parameter won't change format of private key returned by Web3Auth. Private key returned by getPrivKey is always secp256k1. The default value is Curve.SECP256K1 . sessionTime? It allows developers to configure the session management time. Session Time is in seconds, and takes int as a value. public

class

LoginParams { public

Provider loginProvider {

get ;

set ;

```

} public

string dappShare {

get ;

set ;

} public

ExtraLoginOptions extraLoginOptions {

get ;

set ;

} public

Uri redirectUrl {

get ;

set ;

} public

string appState {

get ;

set ;

} public

MFALevel mfaLevel {

get ;

set ;

} public

int sessionTime {

get ;

set ;

} public

Curve curve {

get ;

set ;

} }

```

Provider

[â](#)

```

public

enum

Provider { [ EnumMember ( Value =

"google" ) ] GOOGLE , [ EnumMember ( Value =

"facebook" ) ] FACEBOOK , [ EnumMember ( Value =

"reddit" ) ] REDDIT , [ EnumMember ( Value =

```

```

"discord" ) ] DISCORD , [ EnumMember ( Value =
"twitch" ) ] TWITCH , [ EnumMember ( Value =
"apple" ) ] APPLE , [ EnumMember ( Value =
"line" ) ] LINE , [ EnumMember ( Value =
"github" ) ] GITHUB , [ EnumMember ( Value =
"kakao" ) ] KAKAO , [ EnumMember ( Value =
"linkedin" ) ] LINKEDIN , [ EnumMember ( Value =
"twitter" ) ] TWITTER , [ EnumMember ( Value =
"weibo" ) ] WEIBO , [ EnumMember ( Value =
"wechat" ) ] WECHAT , [ EnumMember ( Value =
"email_passwordless" ) ] EMAIL_PASSWORDLESS , [ EnumMember ( Value =
"email_password" ) ] EMAIL_PASSWORD , [ EnumMember ( Value =
"jwt" ) ] JWT , [ EnumMember ( Value =
"CUSTOM_VERIFIER" ) ] CUSTOM_VERIFIER }

```

ExtraLoginOptions

[â](#)

TheLoginParams class accepts theExtraLoginOptions as an optional input, containing advanced options for custom authentication, email passwordless login etc. This parameter can be considered as advanced login options needed for specific cases.

```

LoginParams ( )

```

```

{ selectedLoginProvider , extraLoginOptions =

```

```

ExtraLoginOptions ( )

```

```

} * Table * Interface

```

Parameter Description additionalParams? Additional params inDictionary 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? bool 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. public

```

class

```

```

ExtraLoginOptions

```

```

{ public

```

```

Dictionary < string ,

```

```
string
    additionalParams {
get ;
set ;
} public
string domain {
get ;
set ;
} public
string client_id {
get ;
set ;
} public
string leeway {
get ;
set ;
} public
string verifierIdField {
get ;
set ;
} public
bool isVerifierIdCaseSensitive {
get ;
set ;
} public
Display display {
get ;
set ;
} public
Prompt prompt {
get ;
set ;
} public
string max_age {
get ;
set ;
} public
```

```
string ui_locales {  
    get ;  
    set ;  
} public  
string id_token_hint {  
    get ;  
    set ;  
} public  
string login_hint {  
    get ;  
    set ;  
} public  
string id_token {  
    get ;  
    set ;  
} public  
string acr_values {  
    get ;  
    set ;  
} public  
string scope {  
    get ;  
    set ;  
} public  
string audience {  
    get ;  
    set ;  
} public  
string connection {  
    get ;  
    set ;  
} public  
string state {  
    get ;  
    set ;  
} public  
string response_type {
```

```

get ;

set ;

} public

string nonce {

get ;

set ;

} public

string redirect_uri {

get ;

set ;

} }

```

Curve

[â](#)

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

```
LoginParams ( )
```

```
{ selectedLoginProvider , curve = Curve . SECP256K1 } * SECP256K1 * ED25519
```

```
public
```

```
void
```

```
login ( ) { var selectedProvider = Provider . GOOGLE ; var options =
```

```
new
```

```
LoginParams ( ) { loginProvider = selectedProvider , curve = Curve . SECP256K1 } ; web3Auth . login ( options ) ; } public
```

```
void
```

```
login ( ) { var selectedProvider = Provider . GOOGLE ; var options =
```

```
new
```

```
LoginParams ( ) { loginProvider = selectedProvider , curve = Curve . ED25519 } ; web3Auth . login ( options ) ; }
```

Sample Response[â](#)

```
{ "ed25519PrivKey": "666523652352635....", "privKey": "0ajjsdsd....", "coreKitKey": "0ajjsdsd....", "coreKitEd25519PrivKey":
"666523652352635....", "sessionId": "....", "error": "....", "userInfo": { "aggregateVerifier": "w3a-google", "email":
"john@gmail.com", "name": "John Dash", "profileImage": "https://lh3.googleusercontent.com/a/Ajjjsdsmdjmm...",
"typeOfLogin": "google", "verifier": "torus", "verifierId": "john@gmail.com", "dappShare": "<24 words 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": true // returns true if user has enabled
mfa } }
```

Example[â](#)

- Google
- Facebook
- Discord
- Twitch
- Email Passwordless
- JWT

```
public
```

```

void

login ( ) { var selectedProvider = Provider . GOOGLE ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider } ; web3Auth . login ( options ) ; } public

void

login ( ) { var selectedProvider = Provider . FACEBOOK ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider } ; web3Auth . login ( options ) ; } public

void

login ( ) { var selectedProvider = Provider . DISCORD ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider } ; web3Auth . login ( options ) ; } public

void

login ( ) { var selectedProvider = Provider . TWITCH ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider } ; web3Auth . login ( options ) ; } public

void

login ( ) { var selectedProvider = Provider . EMAIL_PASSWORDLESS ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider , extraLoginOptions =

new

ExtraLoginOptions ( ) { login_hint =

"hello@web3auth.io" } } ; web3Auth . login ( options ) ; } public

void

login ( ) { var selectedProvider = Provider . JWT ; var options =

new

LoginParams ( ) { loginProvider = selectedProvider , extraLoginOptions =

new

ExtraLoginOptions ( ) { id_token =

"your_jwt_token" } } ; web3Auth . login ( options ) ; }

```

Logging out a user^â

web3Auth.logout()

^â

Trigger logout flow. This function doesn't take any parameters. A completable future containing void object is returned on successfull logout otherwise an error response is returned.

tip Additionally you can associate a function to be triggered on successful logout.

```
void
```

```

Start ( ) { web3Auth =
GetComponent < Web3Auth
    ( ) ; web3Auth . setOptions ( new
Web3AuthOptions ( ) {
} ) ; web3Auth . onLogout += onLogout ; } private
void
onLogout ( ) { // Functions to be called after user logs out }

```

Example

```

public
void
logout ( ) { web3Auth . logout ( ) ; Debug . Log ( "Logged out!" ) ; }

```

Working Example

```

/Assets/Web3Auth.cs using
System ; using
System . Linq ; using
System . Collections . Generic ; using
UnityEngine ; using
UnityEngine . UI ; using
Newtonsoft . Json ;
public
class
Web3custom
:
MonoBehaviour { Web3Auth web3Auth ;
// Start is called before the first frame update void
Start ( ) { web3Auth =
GetComponent < Web3Auth
    ( ) ; web3Auth . setOptions ( new
Web3AuthOptions ( ) { redirectUrl =
new
Uri ( "torusapp://com.torus.Web3AuthUnity/auth" ) , clientId =
"BAwFgL-r7wzQKmtcdiz2uHJKNZdK7gzEf2q-m55xfzSZOw8jLOyli4AVvvzaEQO5nv2dFLEmf9LBkF8kaq3aErg" , network =
Web3Auth . Network . TESTNET , } ) ; web3Auth . onLogin += onLogin ; web3Auth . onLogout += onLogout ; }
public
void
login ( ) { var selectedProvider = Provider . GOOGLE ; var options =
new

```



```
LoginParams ( ) { loginProvider = selectedProvider } ; web3Auth . login ( options ) ; }
```

```
private
```

```
void
```

```
onLogin ( Web3AuthResponse response ) { var userInfo = JsonConvert . SerializeObject ( response . userInfo , Formatting . Indented ) ; Debug . Log ( userInfo ) ; }
```

```
public
```

```
void
```

```
logout ( ) { web3Auth . logout ( ) ; }
```

```
private
```

```
void
```

```
onLogout ( ) { Debug . Log ( "Logged out!" ) ; } Edit this page Previous Initialize Next Whitelabel
```