Usage of PnP Android SDK

Web3Auth's Android SDK is a set of libraries and tools that allow you to easily integrate Web3 with Android applications.

Logging in a Userâ

web3Auth.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 , and acceptsUri 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 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 . data

class

LoginParams (val loginProvider : Provider , var dappShare : String ?

=

null , val extraLoginOptions : ExtraLoginOptions ?

=

null , @Transient

var redirectUrl : Uri ?

=

null , val appState : String ?

=

null , val mfaLevel : MFALevel ?

=

null , val curve : Curve ?

= Curve . SECP256K1)

web3Auth.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.

web3Auth.getEd25519PrivKey()

â

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

web3Auth.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 import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{ val selectedLoginProvider = Provider . GOOGLE // Can be REDDIT, TWITTER, GITHUB etc. val loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login (LoginParams (selectedLoginProvider))

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

(error ==

null)

{ // render logged in UI println ("PrivKey: "

+ web3Auth . getPrivkey ( )) println ("ed25519PrivKey: "

+ web3Auth . getEd25519PrivKey ( )) }

else

{ // render error UI }

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{ val selectedLoginProvider = Provider . FACEBOOK // Can be APPLE, LINE, KAKAO, LINKEDIN etc. val loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login (LoginParams (selectedLoginProvider))

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

(error ==

null)

{ // render logged in UI println ("PrivKey: "

+ web3Auth . getPrivkey ( )) println ("ed25519PrivKey: "

+ web3Auth . getEd25519PrivKey ( )) }

else

{ // render error UI }

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

```
var web3Auth : Web3Auth

private

fun

signIn ( )

{ val selectedLoginProvider = Provider . DISCORD // Can be WECHAT, WEIBO, LINE etc. val loginCompletableFuture :
CompletableFuture < Web3AuthResponse

= web3Auth . login ( LoginParams ( selectedLoginProvider ) )

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

( error ==

null )

{ // render logged in UI println ( "PrivKey: "

+ web3Auth . getPrivkey ( ) ) println ( "ed25519PrivKey: "

+ web3Auth . getEd25519PrivKey ( ) ) }

else

{ // render error UI }

} }

override

fun

onCreate ( savedInstanceState : Bundle ? )

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

        ( R . id . signInButton ) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{ val selectedLoginProvider = Provider . TWITCH // Can be GOOGLE, FACEBOOK, TWITCH etc. val
loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login ( LoginParams ( selectedLoginProvider ) )

loginCompletableFuture . whenComplete

{ loginResponse , error -> if
```

```
( error ==
null )
{ // render logged in UI println ( "PrivKey: "
+ web3Auth . getPrivkey ( ) ) println ( "ed25519PrivKey: "
+ web3Auth . getEd25519PrivKey ( ) ) }
else
{ // render error UI }
} }
override
fun
onCreate ( savedInstanceState : Bundle ? )
{ .. . // Setup UI and event handlers val signInButton = findViewById < Button
        ( R . id . signInButton ) signInButton . setOnClickListener
{
signIn ( )
} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle
class MainActivity :
AppCompatActivity ( )
{ private
lateinit
var web3Auth : Web3Auth
private
fun
signIn ( )
{ val selectedLoginProvider = Provider . EMAIL_PASSWORDLESS // extraLoginOptions = ExtraLoginOptions(login_hint =
"email address") val loginCompletableFuture : CompletableFuture < Web3AuthResponse
= web3Auth . login ( LoginParams ( selectedLoginProvider , extraLoginOptions =
ExtraLoginOptions ( login_hint =
"hello@web3auth.io" ) ) )
loginCompletableFuture . whenComplete
{ loginResponse , error -> if
( error ==
null )
{ // render logged in UI println ( "PrivKey: "
+ web3Auth . getPrivkey ( ) ) println ( "ed25519PrivKey: "
+ web3Auth . getEd25519PrivKey ( ) ) }
else
{ // render error UI }
```

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{ val selectedLoginProvider = Provider . JWT // extraLoginOptions = ExtraLoginOptions(id_token = "jwt_token") val loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login (LoginParams (selectedLoginProvider , extraLoginOptions =

ExtraLoginOptions (id_token =

"your_jwt_token"))) // Use this for login with your generated JWT token, // please refer to https://web3auth.io/docs/auth-provider-setup/byo-jwt-providers to learn more

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

(error ==

null)

{ // render logged in UI println ("PrivKey: "

+ web3Auth . getPrivkey ( )) println ("ed25519PrivKey: "

+ web3Auth . getEd25519PrivKey ( )) }

else

{ // render error UI }

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . }

Selecting Curveâ

Theweb3Auth.login() method accepts acurve parameter. This parameter can be used to select the elliptic curve to use for the signature.

web3Auth . login (LoginParams (selectedLoginProvider , curve = Curve . SECP256K1))

// Can be ED25519, SECP256K1 * SECP256K1 * ED25519

Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{

val selectedLoginProvider = Provider . GOOGLE // Can be GOOGLE, FACEBOOK, TWITCH etc. val loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login (LoginParams (selectedLoginProvider , curve = Curve . SECP256K1))

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

(error ==

null)

{ // render logged in UI }

else

{ // render error UI }

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. . } Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signIn ( )

{

val selectedLoginProvider = Provider . GOOGLE // Can be GOOGLE, FACEBOOK, TWITCH etc. val loginCompletableFuture : CompletableFuture < Web3AuthResponse

= web3Auth . login (LoginParams (selectedLoginProvider , curve = Curve . ED25519))

loginCompletableFuture . whenComplete

{ loginResponse , error -> if

(error ==

null)

{ // render logged in UI }

else

{ // render error UI }

} }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signInButton = findViewById < Button

(R . id . signInButton) signInButton . setOnClickListener

{

signIn ( )

} .. . } .. .

}

Logging out a userâ

web3Auth.logout()

â

Thelogout() method is used to log out a user from Web3Auth. This method will clear the session data and the user will be logged out from Web3Auth.

Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

signOut ( )

{ val logoutCompletableFuture = web3Auth . logout ( ) logoutCompletableFuture . whenComplete

{ _ , error -> if

(error ==

null)

{ // render logout UI }

else

{ // render error UI } } }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val signOutButton = findViewById < Button

(R . id . signOutButton) signOutButton . setOnClickListener

{

signOut ( )

} .. . } .. . }

Enable MFA for a userâ

TheenableMFA method is used to trigger MFA setup flow for users. The method takesLoginParams which will used during custom verifiers. If you are using default login providers, you don't need to passLoginParams . If you are using custom jwt verifiers, you need to pass the JWT token inloginParams as well.

Usage import android . widget . Button import com . web3auth . core . Web3Auth import android . os . Bundle

class MainActivity :

AppCompatActivity ( )

{ private

lateinit

var web3Auth : Web3Auth

private

fun

enableMFA ( )

{ val completableFuture = web3Auth . enableMFA ( )

completableFuture . whenComplete { _ , error -> if

(error ==

null)

{ Log . d ("MainActivity_Web3Auth" ,

"Launched successfully") // Add your logic }

else

{ // Add your logic on error Log . d ("MainActivity_Web3Auth" , error . message ?:

"Something went wrong") } } }

override

fun

onCreate (savedInstanceState : Bundle ?)

{ .. . // Setup UI and event handlers val enableMFAButton = findViewById < Button

(R . id . enableMFAButton) enableMFAButton . setOnClickListener

{

enableMFA ( )

} .. . } .. . }

Launch Wallet Servicesâ

ThelaunchWalletServices method launches a WebView which allows you to use the templated wallet UI services. The method takesLoginParams andChainConfig as the required input.Wallet Services is currently only available for EVM chains.

Argumentsâ

ChainConfig

- Table
- Class

Parameter Description chainNamespace Custom configuration for your preferred blockchain. As of now only EVM supported. Default value isChainNamespace.EIP155 . decimals Number of decimals for the currency ticker. Default value is 18, and acceptsInt as value. blockExplorerUrl? Blockchain's explorer URL. (eg:https://etherscan.io) chainId The chain id of the selected blockchain in hexString . displayName? Display Name for the chain. logo? Logo for the selectedchainNamespace &chainId . rpcTarget RPC Target URL for the selectedchainNamespace &chainId . ticker Default currency ticker of the network (e.g:ETH) tickerName? Name for currency ticker (e.g:Ethereum) data

class

ChainConfig (val chainNamespace : ChainNamespace = ChainNamespace . EIP155 , val decimals : Int =

18 , val blockExplorerUrl : String ?

=

null , val chainId : String , val displayName : String ?

=

null , val logo : String ?

=

null , val rpcTarget : String , val ticker : String , val tickerName : String ?

=

null ,) Usage val loginParams =

LoginParams (Provider . GOOGLE)

val chainConfig =

ChainConfig (chainId =

"0x1" , rpcTarget =

"https://rpc.ankr.com/eth" , ticker =

"ETH" , chainNamespace = ChainNamespace . EIP155)

val completableFuture = web3Auth . launchWalletServices (loginParams , chainConfig)

completableFuture . whenComplete { _ , error -> if (error ==

null)

{ // Add your logic Log . d ("MainActivity_Web3Auth" ,

"Wallet services launched successfully") }

else

{ // Add your logic for error Log . d ("MainActivity_Web3Auth" , error . message ?:

"Something went wrong") } } Minimum Scale plan required Access to Wallet Services is gated. You can use this feature in the development environment for free. The minimumpricing plan to use this feature in a production environment is theScale Plan . Edit this page Previous Initialize Next Whitelabel