Enabling Implicit Authentication Flow in tKey

If you're looking for using an implicit authentication flow in your application, you can use the Torus Service Provider SDK to do so. The Torus Service Provider is built on top of theCustomAuth SDK from Web3Auth, which is an internal library used in the PnP SDKs to provide a common interface for authentication. Here, you need to do some additional setup to enable the SDK to be able to catch the authentication redirect.

This guide will help you in setting up the Torus Service Provider SDK in your application. It is a 3 step process in general:

- Instantiating the Torus Service Provider
- Initiating the Service Provider Instance within tKey
- Handling the OAuth Redirection
- Triggering Login

Instantiationâ

Service Provider intKey is used for generating aShare A , i.e. the private key share managed by a wallet service provider via their authentication flows. This share in ourwallet infrastructure refers to the social login aspect, where we associate a private key share with the user's social login, enabling the seamless login experience.

@tkey/service-provider-torus

â

- npm
- Yarn
- pnpm

npm install --save @tkey/service-provider-torus yarn add @tkey/service-provider-torus pnpm add @tkey/service-provider-torus

Parametersâ

declare

class

TorusServiceProvider

extends

ServiceProviderBase

{ directWeb :

CustomAuth ; singleLoginKey :

BN ; customAuthArgs :

CustomAuthArgs ; constructor ({ enableLogging , postboxKey , customAuthArgs } :

TorusServiceProviderArgs) ; static

fromJSON (value :

StringifiedType) :

TorusServiceProvider ; init (params :

InitParams) :

Promise < void

; triggerLogin (params :

SubVerifierDetails) :

Promise < TorusLoginResponse

; triggerAggregateLogin (params :

AggregateLoginParams) :

Promise < TorusAggregateLoginResponse

; triggerHybridAggregateLogin (params :

HybridAggregateLoginParams) :

Promise < TorusHybridAggregateLoginResponse

; toJSON ( ) :

StringifiedType ; } export

interface

TorusServiceProviderArgs

extends

ServiceProviderArgs

{ customAuthArgs :

CustomAuthArgs ; } ThisTorusServiceProvider constructor takes an object withcustomAuthArgs as input. It contains the following parameters:

customAuthArgs

â

ThecustomAuthArgs object is the mandatory object needed to be passed within theTorusServiceProvider constructor. It contains the following parameters:

- Table
- Interface

Parameter Type Description Default Value Mandatory baseUrl string Redirect Uri for OAuth isbaseUrl +redirectPathName which means that you must specifybaseUrl +redirectPathName as redirect_uri at verifier's interface.

Yes metadataUrl? string Specify a custom metadata host. https://metadata.tor.us No network TORUS_NETWORK_TYPE Auth Network to target options: mainnet, testnet, aqua & cyan. mainnet No networkUrl string Network Url to read blockchain data from (eg: infura url)

No enableLogging? boolean This option is used to specify whether to enable logging. false No enableOneKey? boolean Use one key feature that allows users to have the same account in tKey. Note: This flag shouldn't be changed once set for an account; changing it will lead to a different account. false No redirectToOpener? boolean For chrome extensions, the general methods for capturing auth redirects don't work. So, we redirect to the window which opens the auth window. false No redirectPathName? string This option is used to specify the url path where user will be redirected after login. Redirect Uri for OAuth is baseUrl/redirectPathName. At verifier's interface, please use baseUrl/redirectPathName as the redirect_uri. redirect No apiKey? string API Key for Web3Auth to enable higher access limits

No uxMode? UX_MODE_TYPE Two uxModes are supported:* 'popup' * : In this uxMode, a popup will be shown to user for login. * 'redirect' * : In this uxMode, user will be redirected to a new window tab for login.

Use of'REDIRECT' mode is recommended in browsers where popups might get blocked. popup No locationReplaceOnRedirect? boolean Whether to replace the url hash/query params from OAuth at the end of the redirect flow false No popupFeatures? string Features of popup window. Please checkhttps://developer.mozilla.org/en-US/docs/Web/API/Window/open#window_features for further documentation.

No storageServerUrl? string Specify a custom storage server url https://broadcast-server.tor.us No customAuthArgs: CustomAuthArgs;

export

interface

CustomAuthArgs

{ /* * *baseUrl , along with redirectPathName is used to construct the uri of page * where user will be redirected after login*. baseUrl :

string ; /* * *Specify a custom metadata host * @defaultValue https://metadata.tor.us*/ metadataUrl ? :

string ; /* * *Auth Network to target options: mainnet | testnet | cyan | aqua * @defaultValue mainnet*/ network ? :

TORUS_NETWORK_TYPE ; /* * *Network Url to read blockchain data from (eg: infura url)*/ networkUrl ? :

string ; /* * *This option is used to specify whether to enable logging * * @defaultValue false*/ enableLogging ? :

boolean ; /* * *Use one key features * * @defaultValue false*/ enableOneKey ? :

boolean ; /* * *For chrome extensions, the general methods for capturing auth redirects don't work. * So, we redirect to the window which opens the auth window. * * @defaultValue false* / redirectToOpener ? :

boolean ; /* * *This option is used to specify the url path where user will be * redirected after login. Redirect Uri for OAuth is baseUrl/redirectPathName. * * * @defaultValue redirect * * /* redirectPathName ? :

string ; /* * *API Key for Web3Auth to enable higher access limits */* apiKey ? :

string ; /* * *Two uxModes are supported:- * -'popup': In this uxMode, a popup will be shown to user for login. * 'redirect': In this uxMode, user will be redirected to a new window tab for login. * * @defaultValue 'popup' * @remarks * * Use of'REDIRECT' mode is recommended in browsers where popups might get blocked.* / uxMode ? :

UX_MODE_TYPE ; /* * *Whether to replace the url hash/query params from OAuth at the end of the redirect flow * * @defaultValue false* / locationReplaceOnRedirect ? :

boolean ; /* * *Features of popup window. Please check https://developer.mozilla.org/en-US/docs/Web/API/Window/open#window_features * for further documentation.* / popupFeatures ? :

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

string ; }

Usageâ

const web3AuthClientId =

"YOUR_WEB3AUTH_CLIENT_ID" ;

// get from https://dashboard.web3auth.io

// Configuration of Service Provider const customAuthArgs =

{ web3AuthClientId , baseUrl :

{ window . location . origin } /serviceworker , network :

"sapphire_mainnet" ,

// based on the verifier network. uxMode :

"popup" ,

// or redirect } ;

const serviceProvider =

new

TorusServiceProvider ({ enableLogging :

false , customAuthArgs : customAuthArgs as

any , }) ;

Initializing Service Providerâ

You need to initialize your Service Provider within your constructor function to use it while logging your user in through the social accounts. This is done by calling theinit() function within thetKey instance'sserviceProvider property.

tKey.serviceProvider.init(initParams);

Parametersâ

- Table
- Interface

Parameter Type Description Default Value skipSw? boolean Skips the installation / check for service worker false skipInit? boolean Skips the init function false skipPrefetch? boolean Skips the prefetching of redirect url false interface

InitParams

{ /* * *skips the installation / check for service worker * @defaultValue false*/ skipSw ? :

boolean ; /* * *skips the init function * @defaultValue false*/ skipInit ? :

boolean ; /* * *skips the prefetching of redirect url * @defaultValue false */* skipPrefetch ? :

boolean ; }

Usageâ

useEffect (( )

=>

{ const

init

=

async

( )

=>

{ // Initialization of Service Provider try

{ await

(tKey . serviceProvider

as

any) . init ( ) ; }

catch

(error)

{ console . error (error) ; } } ; init ( ) ; } ,

[ ]) ;

Handling Redirectionâ

In the Implicit Flow, you need to set up a redirect page to capture the authentication redirect. This is needed since the authentication redirect contains the authentication information of the user, which is needed to generate the OAuthKey. The redirect page is used to capture this information and pass it over to the Torus Service Provider.

TheuxMode parameter in thecustomAuthArgs object is used to determine the type of redirect page to be used. There are two types of redirect

- Popup Mode (default):
- We deploy a service worker to capture the redirect.
- Redirect Mode:
- You need to create a redirect page to capture the redirect.

Redirect Pageâ

You can get login result by callinggetRedirectResult on redirected page mount. For example, if baseUrl ishttp://localhost:3000 andredirectPathName isauth then user will be redirected tohttp://localhost:3000/auth page after login where you can get login result by callinggetRedirectResult on redirected page mount.

Usageâ

```
useEffect ( ( )

=>

{ const

init

=

async

( )

=>

{ // Initialization of Service Provider try

{ // Init is required for Redirect Flow but skip fetching sw.js and redirect.html ) ( tKey . serviceProvider

as

any ) . init ( { skipInit :

true

} ) ; if

( window . location . pathname

===

"/auth"

&&

window . location . hash . includes ( "#state" ) )

{ let result =

await

( tKey . serviceProvider

as

any ) . directWeb . getRedirectResult ( ) ; tKey . serviceProvider . postboxKey

=

new

BN ( ( result . result

as

any ) . privateKey ! ,

"hex" ) ; await tKey . initialize ( ) ; } }

catch

( error )

{ console . error ( error ) ; } } ; init ( ) ; } ,

[ ] ) ;
```

Service Workerâ

A service worker is a scripts that is run by the browser. It does not have any direct relationship with the DOM and provides many out of the box network-related features. Web3Auth Core Kit tKey SDK needs a service worker relative tobaseUrl to capture the auth redirect atredirectPathName path.

For example, while using service worker ifbaseUrl ishttp://localhost:3000/serviceworker then user will be redirected tohttp://localhost:3000/serviceworker/redirect page after login where service worker will capture the results and send it back to original window where login was initiated.

tip * Using service worker is optional, but highly recommended. You can skip it by passingskipSw * param whileinitializing tKey * . * Service worker is needed if you are usingpopup * uxMode within yourService Provider configuration * . * For browsers where service workers are not supported, or in the case you wish to not use service workers, create and serve redirect page (i.eredirect.html * file).

Service Worker Setupâ

- If you're using React, to setup service worker, you need to create asw.js
- file in your public folder and register it in yourindex.html
- file.
- You can find more information about it inthis blog
- .
- For Angular,this guide
- will be helpful in setting up the service worker.
- For Vue,this guide
- is a great way to get
- started with service workers.

Service Worker Codeâ

You can directly copy the service worker file code from here and paste in your respective folder. You can also find the code in ourtKey JS Examples (Popup Flow) .

Service Worker Code /public/serviceworker/sw.js / *eslint-disable* / function

getScope ( )

{ return self . registration . scope ; }

self . addEventListener ("message" ,

function

(event)

{ if

(event . data

&& event . data . type

===

"SKIP_WAITING")

{ self . skipWaiting ( ) ; } }) ;

self . addEventListener ("fetch" ,

function

(event)

{ try

{ const url =

new

URL (event . request . url) ; if

(url . pathname . includes ("redirect")

&& url . href . includes (getScope ( )))

{ event . respondWith (new

Response (new

Blob ([`

{ "" } `

] , { type :

"text/html"

}))) ; } }

catch

(error)

{ console . log ("Hello") ; console . error (error) ; } }) ;

Log Inâ

The login with the tKey SDK is a two step process. First, you need to trigger the login process by calling thetriggerLogin() function of the Service Provider. Following which using the returned information, use theinitialize() function of the tKey to generate the Threshold Key corresponding to the user.

However, before starting this process, you need to set up Custom Authentication on your Web3Auth Dashboard. For this, you need toCreate a Verifier from theCustom Auth section of theWeb3Auth Developer Dashboard with your desired configuration.

tip If you want to know more about setting up a verifier and how to use it, please refer to theCustom Authentication Documentation .

Triggering Loginâ

tKey.serviceProvider.triggerLogin(SubVerifierDetails)

â

This is a needed step since this will generate a private key which will be needed by the tKey to generate it's share. This is done by calling thetriggerLogin() function within thetKey instance'sserviceProvider .

SubVerifierDetails

â

ThetriggerLogin function inTorusServiceProvider accepts the following parameters:

- Table
- Interface

Parameter Type Description Mandatory typeOfLogin LOGIN_TYPE Type of your login verifier Yes verifier string Verifier Name fromWeb3Auth Dashboard Yes clientId string Client ID from your login service provider Yes jwtParams? Auth0ClientOptions Additional JWT Params No hash? string Your JWT in hash No queryParameters? TorusGenericObject Additional Query Params No customState? TorusGenericObject Additional Custom State Params No interface

SubVerifierDetails

{ typeOfLogin :

LOGIN_TYPE ; verifier :

string ; clientId :

string ; jwtParams ? :

Auth0ClientOptions ; hash ? :

string ; queryParameters ? :

TorusGenericObject ; customState ? :

TorusGenericObject ; }

export

declare

type

TorusGenericObject

=

{ [key :

string] :

string ; } ;

jwtParams

â

- Table
- Interface

Parameter Type Description Mandatory domain string Domain of your Auth0 App such as'example.auth0.com'. Please use https://` as a prefix Yes client_id? string The Client ID found on your Auth0 Application settings page Yes redirect_uri? string The default URL where Auth0 will redirect your browser to with the authentication result. It must be whitelisted in the "Allowed Callback URLs" field in your Auth0 Application's settings. If not provided here, it should be provided in the other methods that provide authentication. No leeway? number The value in seconds used to account for clock skew in JWT expirations. Typically, this value is no more than a minute or two at maximum. Defaults to 60s. No verifierIdField? string The field in jwt token which maps to verifier id No isVerifierIdCaseSensitive? boolean Whether the verifier id field is case sensitive. @defaultValue true No id_token? string Pass on theid_token directly here. Useful in case of RWA No access_token? string Pass on theaccess_token directly here. No user_info_route? string The route for user info endpoint. This will be padded to domain. @defaultValue userinfo No interface

Auth0ClientOptions

extends

BaseLoginOptions

{ /* * *Your Auth0 account domain such as'example.auth0.com', * 'example.eu.auth0.com' or , 'example.mycompany.com' * (when using [custom domains](#))* / domain :

string ; /* * *The Client ID found on your Application settings page*/ client_id ? :

string ; /* * *The default URL where Auth0 will redirect your browser to with * the authentication result. It must be whitelisted in * the "Allowed Callback URLs" field in your Auth0 Application's * settings. If not provided here, it should be provided in the other * methods that provide authentication.* / redirect_uri ? :

string ; /* * *The value in seconds used to account for clock skew in JWT expirations. * Typically, this value is no more than a minute or two at maximum. * Defaults to 60s.* / leeway ? :

number ; /* * *The field in jwt token which maps to verifier id*/ verifierIdField ? :

string ; /* * *Whether the verifier id field is case sensitive * @defaultValue true*/ isVerifierIdCaseSensitive ? :

boolean ; id_token ? :

string ; access_token ? :

string ; /* * *The route for user info endpoint. This will be padded to domain * @defaultValue userinfo */* user_info_route ? :

string ; }

typeOfLogin

```
â

export

type

LOGIN_TYPE

=

typeof

LOGIN [ keyof

typeof

LOGIN ] ;

export

declare

const

LOGIN :

{ readonly

GOOGLE :

"google" ; readonly

FACEBOOK :

"facebook" ; readonly

REDDIT :

"reddit" ; readonly

DISCORD :

"discord" ; readonly

TWITCH :

"twitch" ; readonly

APPLE :

"apple" ; readonly

GITHUB :

"github" ; readonly

LINKEDIN :

"linkedin" ; readonly

TWITTER :

"twitter" ; readonly

WEIBO :

"weibo" ; readonly

LINE :

"line" ; readonly

EMAIL_PASSWORD :
```

"email_password" ; readonly

PASSWORDLESS :

"passwordless" ; readonly

JWT :

"jwt" ; readonly

WEBAUTHN :

"webauthn" ; } ;

Usageâ

const loginResponse =

await (tKey . serviceProvider

as

any) . triggerLogin ({ typeOfLogin :

"google" ,

// "google","facebook","reddit","discord","twitch", etc. verifier :

"google-tkey-w3a" , clientId :

"774338308167-q463s7kpvja16l4l0kko3nb925ikds2p.apps.googleusercontent.com" , }) ; const user = loginResponse . userInfo ; console . log ("User Details: " , user) ;

Get User Informationâ

triggerLogin (params :

SubVerifierDetails) :

Promise < TorusLoginResponse

> ; The tKey Service Provider returns aTorusLoginResponse object which contains the user's information and details about the login. You can access theuserInfo property within it to get the user details from the login provider.

TorusLoginResponse

â

- Table
- Interface

Parameter Type Description accessToken string User Access Token idToken? string Userid_token ref? string User refs extraParams? string Any extra parameters email string User Email ID name string User Name profileImage string User Profile Image aggregateVerifier? string Aggregate Verifier Details verifier string Verifier Details verifierId string Verifier ID typeOfLogin LOGIN_TYPE Type of Social Login export

declare

type

TorusLoginResponse

=

TorusSingleVerifierResponse

&

TorusKey ; export

interface

TorusSingleVerifierResponse

{ userInfo :

TorusVerifierResponse

&

LoginWindowResponse ; } export

interface

LoginWindowResponse

{ accessToken :

string ; idToken ? :

string ; ref ? :

string ; extraParams ? :

string ; extraParamsPassed ? :

string ; state :

TorusGenericObject ; } export

interface

TorusVerifierResponse

{ email :

string ; name :

string ; profileImage :

string ; aggregateVerifier ? :

string ; verifier :

string ; verifierId :

string ; typeOfLogin :

LOGIN_TYPE ; ref ? :

string ; registerOnly ? :

boolean ; }

TorusKey

â

- Table
- Interface

Parameter Type Description publicAddress string User Public Address privateKey string User Private Key metadataNonce string Metadata Nonce related to the user typeOfUser enum("v1", "v2") User ifv1 orv2 pub_key_X string X coordinate of Public Key pub_key_Y string Y coordinate of Public Key export

interface

TorusKey

extends

TorusKeyPub

{ publicAddress :

string ; privateKey :

string ; metadataNonce :

string ; typeOfUser :

"v1"

|

"v2" ; } export

interface

TorusKeyPub

{ pubKey ? :

{ pub_key_X :

string ; pub_key_Y :

string ; } ; }

Trigger Login using Aggregate Verifierâ

triggerAggregateLogin()

â

await

(tKey . serviceProvider

as

TorusServiceProvider) . triggerAggregateLogin (AggregateLoginParams) Takes in the aggregate verifier details asAggregateLoginParams .

interface

AggregateLoginParams

{ aggregateVerifierType :

AGGREGATE_VERIFIER_TYPE ; verifierIdentifier :

string ; subVerifierDetailsArray :

SubVerifierDetails [ ] ; }Edit this page Previous Private Key Next Web3Auth CoreKit tKey iOS SDK