OIDC Explainer

The following diagram outlines the general authentication flow for an integrating app:

Registration

Before OIDC authentication can take place, developers must register their applications with Worldcoin. This is a one-time action. You can create a new application on theDeveloper Portal .

During registration, you will need to provide the following values:

- Redirect URIs:Required
- , list of approved websites the user can be redirected to after successful authentication.
- App name (also called client name):Optional
- , the name of the application that's displayed to users

All redirect URIs must be over HTTPS, and contain no port numbers or URL fragments. For example,https://app.example.com/login is valid, buthttps://app.example.com:3000/login is not. Similarly,https://app.example.com/login#foo is not valid, buthttps://app.example.com/login?foo=bar is.

After registration is complete, you will have a validapp_id that will be needed for every other step in the authentication process. This value is equivalent toclient_id from theOIDC specification

Flows

World ID supports theauthorization code ,implicit , andhybrid flows from the OIDC spec. Applications can use any one of these flows to authenticate users.

Generally , applications should implement the authorization code flow, as it is more secure than the implicit flow. Applications without backend servers (that may be running purely client-side) are more suited to implicit authentication.

Authentication

Authentication begins with a request to the/authorize endpoint.

When using the native Sign in with World ID page, most of the OIDC process is handled for you. You can begin the authentication cycle by redirecting your users to:

https://id.worldcoin.org/authorize?client_id={app_id}&response_type={code|token|id_token}&redirect_uri={encoded_redirect_url}&scope={scope}&state={state_value}&nonce={nonce_value} Copy Copied!

Example values could be:

- client_id
- : obtained from the Developer Portal (example:app_IshSNnaJfdt6Sohu6YAA
-).
- response_type
- : response type as specified in OIDC spec, remember to URL encode (example:code%20id_token
-).
- redirect_uri
- : where the user is redirected upon successful authentication. Must be on the approved redirect URI list which can always be updated in the Developer Portal (example:https%3A%2F%2Fapp.example.com%2Flogin
-).
- scope
- : space-separated list of scopes to request. Defaults toopenid
- if not provided. (example:openid%20profile%20email
-).
- state
- : unique value used to track a user's session (example:session_102030405060708090
-).
- nonce
- : random value to prevent replay attacks (example:z-dkEmoy_ujfk7B8uTiQpp
-). Required when using theid_token
- response type.

The user will then authenticate with their World ID via the World app. Once successfully authorized, the user is redirected back to your application. The redirect URL will contain a number of values, depending on the flow you are using.

Redirect Responses

If using the default authorization code flow, the redirect URL will contain the following params:

- code
- : An authorization code that can be exchanged for an ID token
- state
- : The optional state value passed to the/authorize
- endpoint

If you received a response containing anauthorization code , you must exchange it for an ID token on the/token endpoint. Request details can beseen here .

If using implicit flow, the redirect URL will contain the following params:

- id_token
- : A signed JWT identifying the user, and any requested scope information

ID tokens must always be verified, and should not be blindly accepted! Additionally, you must validate that thenonce within the ID token matches the nonce you provided.

To verify an ID token, fetch the public key from the/jwks endpoint. You can read more about this process at theAuth0 blog orJWT.io , but one example method could be:

import

*

as jose from

'jose'

const

verifyJwt

= (token :

string) => { const

JWKS

=

jose .createRemoteJWKSet (new

URL ('https://id.worldcoin.org/jwks.json'))

const { payload ,

header } =

await

jose .jwtVerify (token ,

JWKS , { issuer :

'https://id.worldcoin.org' , aud :

'app_IshSNnaJfdt6Sohu6YAA' , })

return payload }

verifyJwt ('eyJhbGciOiJSUzI1NiIsInR5cCI6Ikp.eyAs.XVCJ9...') Copy Copied!