Protocol Internals

Documentation of certain protocol elements. Intended forinternal use only.

Universal Link

The universal link is used so that the World App can recognize a World ID request and handle it correctly. As more clients are built, the universal link will be extended to other wallets. The structure of the universal link is documented here. The source of truth function for this can be foundon GitHub .

- The universal link base is:https://worldcoin.org/verify
- .* If the query stringt
-
  - is equal towld
-
  - , the request is using the Wallet Bridge, and the following requirements apply:* i
-
  -
    - is arequired
-
  -
    - query string, which contains therequestId
-
  -
    - , a UUIDv4 used to identify the request
-
  -
    - k
-
  -
    - is arequired
-
  -
    - query string, which contains the URL Base64 encoded AES-GCM key
-
  -
    - b
-
  -
    - is anoptional
-
  -
    - query string, which contains the URL of the Wallet Bridge. This defaults tohttps://bridge.worldcoin.org/
-
  -
    - .
-
  - Coming Soon:
-
  - If the query stringt
-
  - is equal towmobile
-
  - , the request is passed directly from a native mobile app to World App, and the following requirements apply:* app_id
-
  -
    - is arequired
-
  -
    - query string. This is theapp_id
-
  -
    - from the Developer Portal, or is equal toself_hosted
-
  -
    - .

-
  -
    - action
-
  -
    - is arequired
-
  -
    - query string, which contains the action ID
-
  -
    - credential_types
-
  -
    - is anoptional
-
  -
    - query string, which is a comma-separated string of accepted credentials.
-
  -
    - signal
-
  -
    - is anoptional
-
  -
    - query string, which is the signal to be used during proof generation.
-
  -
    - Eitherreturn_to
-
  -
    - orcallback_url
-
  -
    - is arequired
-
  -
    - query string, which is the URL to return to after the proof is generated.* return_to
-
  -
    -
      - is the URL to return to after the proof is generated. This is used for web-based applications.
-
  -
    -
      - callback_url
-
  -
    -
      - is the URL to which the proof will be submitted directly. This is used for native mobile applications.

Wallet Bridge

The Wallet Bridge acts as the intermediary between IDKit and the user's World ID wallet (e.g. World App). It is responsible for relaying a verification request to the user's identity wallet (e.g. World App), and returning the proof to IDKit. A functional diagram is shown here, with a step-by-step explanation to follow:

All requests to the Wallet Bridge must include aContent-Type: application/json header, aUser-Agent header, and a valid JSON body.

1. IDKit initiates the proof request session on the Wallet Bridge.* An app configures and opens IDKit with the required parametersapp_id
2.
   - andaction
3.
   - , and the optional parameterssignal
4.

- ,credential_types
- , andaction_description
- .* If an application is self-hosted, theapp_id
  - must be equal toself_hosted
  - . See more details about self-hosted applications here
  - .
- IDKit generates an AES-GCMkey
- andiv
- , used to encrypt the verification request. IDKit temporarily stores thekey
- in memory.
- IDKit encrypts the above parameters with thekey
- andiv
- , forming the encrypted body of the request to the Wallet Bridge. Theiv
- is included unencrypted in the request body.

Bridge POST Request

- Unencrypted JSON
- Encrypted Request Body
- {
- "app_id"
- :
- "app_2cde98081f4673c86082d418e6a59f58"
- ,
- // starts with "app_", or is equal to "self_hosted"
- "action"
- :
- "verify-account"
- ,
- "signal"
- :
- "@username"
- ,
- // optional, defaults to ""

36.
    - "credential_types"
37.
    - :
38.
    - [
39.
    - "orb"
40.
    - ,
41.
    - "device"
42.
    -]
43.
    - ,
44.
    - // optional, defaults to ["orb"]
45.
    - "action_description"
46.
    - :
47.
    - "Verify your account on an example social media website!"
48.
    - ,
49.
    - // optional
50.
    - "return_to"
51.
    - :
52.
    - "https://your-application.com/world-id"
53.
    - // coming soon -- only used when deeplinking to mobile application. must be a valid https URL
54.
    - }
55.
    - Copy
56.
    - Copied!
57.
    - IDKit sends a POST request to the Bridge:https://bridge.worldcoin.org/request
58.
    - , with the encrypted JSON body described above, thus initiating the session and sending the payload required to generate a proof. The Bridge returns a UUIDv4requestId
59.
    - to IDKit, which is used to identify the request.
60.
    - The universal link is formed as described in the above section, with therequestId
61.
    - and AES-GCMkey
62.
    - included in the query string.
63. World App or the Worldcoin Simulator (referred to going forward as the client) receives the payload from the bridge, generates the zero-knowledge proof, and returns the proof as a payload in a proof response session on the Bridge.* The client receives the universal link via QR code, mobile app deeplink, or pasting from the clipboard (only for the Worldcoin Simulator).
64.
    - The client parses the query string to extract therequestId
65.
    - andkey
66.
    - . Thekey
67.
    - is used to decrypt the payload, and therequestId
68.

- is used to identify the request.

69.
    - The client sends a GET request to the Wallet Bridge to fetch the payload:https://bridge.worldcoin.org/request{requestId}

70.
    - . The payload is returned to the client and removed from the Bridge, thus ending the proof request session.* The client can optionally send a HEAD request to check if a payload is available for the givenrequestId

71.
    -
        - . A200

72.
    -
        - response indicates a payload is available, while a404

73.
    -
        - response indicates no payload is available. The payload is not returned, nor removed from the bridge if it is present.

74.
    - The client parses the payload to get the AES-GCMiv

75.
    - and encrypted body. Thekey

76.
    - from the universal link andiv

77.
    - are used to decrypt the body, which contains theapp_id

78.
    - andaction

79.
    - for the request, as well as the optionalsignal

80.
    - ,credential_types

81.
    - , andaction_description

82.
    - parameters.

83.
    - The client calculates theexternalNullifier

84.
    - from theapp_id

85.
    - andaction

86.
    - , and generates the zero-knowledge proof using the local Semaphore identity after receiving the user's consent.

87.
    - The client forms the body, which either contains the successful proof or an error response.

88.
    - The client encrypts the body with thekey

89.
    - and a newly-generatediv

90.
    - , and sends a PUT request to the Wallet Bridge:https://bridge.worldcoin.org/response{requestId}

91.
    - , with the new iv alongside the encrypted proof or error as the body.

92.
    - The client must delete any information received from the Bridge, as this could be used to disclose which actions a user has performed.

93. IDKit polls the Bridge for the response from the client, receives the encrypted response, decrypts it, and calls the callback with the proof or handles the returned error.The response will only be returned once, and will be deleted from the Bridge after being returned.

94.
    - IDKit sends a GET request tohttps://bridge.worldcoin.org/response{requestId}

95.
    - to check if the proof is available. The JSON returned will include astatus

96.
    - field with one of the following values:* initialized

97.
    -
        - : The verification request has been received by the Bridge, but not yet received by the client.

- retrieved
- : The verification request has been received by the client, but the proof has not yet been returned.
- completed
- : The proof (or an error) has been returned by the client, and will be included in this response.

Bridge Response

Completed
Initialized
Retrieved
{
"status"
:
"completed"
,
"response"
:
{
"iv"
:
"a2xhanNsbnJ2b2VzanJ2YTtvanNpZWZubGFp"
,
// Base64 encoded
"payload"
:
"YWVvbmJybGFpc2VudWJybGFvbTtvdmNubGRrZmE..."
// Base64 encoded encrypted body
}
}
Copy
Copied!
IDKit decrypts the response using thekey
it generated earlier and theiv

- included in the response, and either handles the error or calls the configured callback function(s) with the proof.

- **Decrypted Bridge Response**

- Proof
- Error
- {
- "proof"
- :
- "0x..."
- ,
- "merkle_root"
- :
- "0x..."
- ,
- "nullifier_hash"
- :
- "0x..."
- ,
- "credential_type"
- :
- "orb"
- ,
- // <-- or "device"
- }
- Copy
- Copied!
- View theErrors Reference
- for more information on potential error codes.

External Nullifier

WithinSemaphore , thezero knowledge protocol that World ID is based on, the external nullifier is a public 32-byte value that scopes the uniqueness of verifications. It is one of two inputs in the ZK circuit, the other being the secret identity nullifier. These two values are hashed to produce the nullifier hash, which identifies the user.

World ID actions (whether for Sign In or Incognito Actions) are identified by their external nullifiers. This value is derived from theapp_id and stringified action passed by IDKit to the World app. Our implementation can befound here .

Generally you won't need to generate the external nullifier yourself. IDKit will handle this automatically, but for custom integrations it's helpful to keep this in mind. When performing a request to the/precheck endpoint, you must pass the valid

external nullifier for the givenapp_id andaction . You can accomplish this with thegenerateExternalNullifier function from IDKit:

import { IDKitInternal } from

'@worldcoin/idkit'

const

getExternalNullifier

= (app_id :

string , action :

string) => { return

IDKitInternal .generateExternalNullifier (app_id , action).digest }

getExternalNullifier ('app_staging_7550e829082fc558e112e0620c1c7a59' ,

'test action') Copy Copied!

OpenID Connect Claims

Within the ID token returned by the World ID provider, a minimal number ofOIDC claims are set. This is due to the privacy-focused nature of the protocol. The set claims are:

- iss
- : The issuer of the token, always 'https://id.worldcoin.org
- "
- sub
- : The unique user identifier for the specific application (this is thenullifier_hash
- from the World ID ZKP)
- aud
- : The identifier of the requesting application. This is always theapp_id
- from the Developer Portal or/register
- endpoint
- jti
- : A unique identifier for this ID token, only used once
- iat
- : The timestamp of the token issuance
- exp
- : The timestamp of the token's expiration
- alg
- : The algorithm used to sign the ID token, only RS256 is supported
- scope
- : The scopes requested by the application. Must always containopenid
- . Theprofile
- andemail
- scopes are also supported for compatibility, but use should be avoided.
- https://worldcoin.org/beta
- : Describes claims specific to World ID.Deprecated and replaced byhttps://id.worldcoin.org/v1
- .* likely_human
  - : "strong" or "weak", corresponding to whether the user has strong sybil protection or likelihood of being human. Biometrically verified users have astrong
  - value.
  - credential_type
  - :orb
  - ordevice
  - , represents the credential the user used to authenticate. In general, for Sign in with World ID, the highest credential available to the user will be used.
- https://worldcoin.org/v1
- : Describes claims specific to World ID.* verification_level

-
  - :orb
-
  - ordevice
-
  - , represents the verification level of the user. In general, for Sign in with World ID, the highest verification level available to the user will be used.

Signup Sequencer

World ID utilizes a sequencer to insert identity commitments on-chain, generate inclusion proofs, and track the state of the contract Merkle trees. This is done to simplify the amount of state applications need to handle to work with World ID.

The sequencer utilizes a batching process to reduce gas costs and improve performance. When verifying a proof from the World app, the Developer Portal will query the sequencer to determine if the proof is valid, since the user could be a part of the current batch (which is not yet on-chain). To interact with the sequencer, you can use these endpoints:

Staging

- Orb credential:https://signup-batching.stage-crypto.worldcoin.org
- Device credential:https://phone-signup.stage-crypto.worldcoin.org

Production

- Orb credential:https://signup.crypto.worldcoin.org
- Device credential:https://phone-signup.crypto.worldcoin.org

Endpoints

The sequencer exposes the following endpoints:

- /inclusionProof
- : Fetches the inclusion proof for a given identity commitment
- /insertIdentity
- : Inserts an identity commitment into the current sequencer batch
- /verifySemaphoreProof
- : Verifies the given ZK proof (from the World app) is valid, even if it is not yet on-chain

More details about these endpoints can be found in ou[Swagger documentation.]