IDKit

IDKit is open source and accepts contributions! Head over toGitHub and submit a pull request.

There are three packages available in the IDKit Monorepo:@worldcoin/idkit ,@worldcoin/idkit-standalone , and@worldcoin/idkit-core . The@worldcoin/idkit package is the main package that should be used with the React JS framework. The@worldcoin/idkit-standalone package is a standalone package that can be used in vanilla JavaScript applications. The@worldcoin/idkit-core package is a core functionality package that is used by the other two packages, and should be used only when creating a new IDKit package.

IDKit (React)

The@worldcoin/idkit package is the main package that should be used with theReact framework or any other framework that supports React components, such asNext.JS .

Components

IDKitWidget

TheIDKitWidget component is the main component that renders the World ID widget. It should be mounted in your React app and passed the relevant parameters. Accepts a function as a child that receives anopen function to open the widget.

import { IDKitWidget } from

'@worldcoin/idkit'

< IDKitWidget app_id = "app_GBkZ1KlVUdFTjeMXKlVUdFT"

// obtained from the Developer Portal action = "vote_1"

// this is your action name from the Developer Portal signal = "user_value"

// any arbitrary value the user is committing to, e.g. a vote onSuccess = {onSuccess} verification_level = "device"

// minimum verification level accepted, defaults to "orb"

{({ open }) => < button

onClick

{open}>Verify with World ID</ button

} </ IDKitWidget

Copy Copied!

Parameters

The following parameters can be passed as props to theIDKitWidget component:

- Name
- app_id
- Type
- string
- Required
- REQUIRED
- Description
- Unique identifier for the app verifying the action. This should be the App ID obtained from theDeveloper Portal
- .
- Name
- action
- Type
- string
- Required
- REQUIRED
- Description
- Identifier for the action the user is performing. This should be the action name set in the Developer Portal.

- Name
- onSuccess
- Type
- function(ISuccessResult)
- Required
- REQUIRED
- Description
- Function to trigger when verification is successful and the modal is closed. Should receive a single parameter of typeISuccessResult
- which containsthe proof details.
- Name
- handleVerify
- Type
- function(ISuccessResult)
- Description
- Called after the proof is returned from the user's identity wallet (e.g. World App), but before showing the success screen. Should receive a single parameter of typeISuccessResult
- which containsthe proof details.
- Throwing an error in this screen will show the user a custom error.
- handleVerify
- should be used for API proof verifications to create the best user experience. This will show a pending state while the proof is verified and present any errors thrown in a user-readable fashion.
- Name
- onError
- Type
- function(IErrorState)
- Description
- Called when IDKit is closed after an error. Should receive a single parameter of typeIErrorState
- which containsthe error details.
- Name
- verification_level
- Type
- string:orb
- Description
- The minimum verification level accepted. Can beorb
- ordevice
- . Defaults
- toorb
- .TypeScript apps can use theVerificationLevel
- enum.
- Name
- signal
- Type
- string:""
- Description
- The signal to be included in the zero-knowledge proof. Typically used for on-chain actions, read more inthe On-chain section
- .
- Name
- bridge_url
- Type
- string:"https://bridge.worldcoin.org"
- Description
- The URL of theWallet Bridge
- to use for establishing a connection between IDKit and the user's World ID Wallet. Defaults to the bridge service hosted by Worldcoin.Only change this if you are running your own bridge service.
- Read more inProtocol Internals
- .World App will temporarily prevent users from connecting to a Wallet Bridge that is not hosted by Worldcoin or Tools for Humanity while security reviews are ongoing, so werecommend using the default value
- by leaving thebridge_url
- parameter undefined.
- Name
- action_description
- Type
- string
- Description
- The description of the specific action (shown to users in World App).Only used for Dynamic Actions.
- Name

- autoClose
- Type
- boolean:true
- Description
- Whether to automatically close the widget after completion. Defaults totrue
- .
- Name
- advanced
- Type
- JSON
- Description
- A JSON object containing advanced configuration options that may be unstable or subject to change. SeeAdvanced Configuration
- for more details.

Hooks

useIDKit

TheuseIDKit hook allows you to programmatically open the IDKit widget without mounting any buttons on screen. Note that you still need to mount the component for this to work.

import { IDKitWidget , useIDKit } from

'@worldcoin/idkit'

const { open ,

setOpen } =

useIDKit ()

useEffect (() => { setOpen (true) } , [])

return (< div

< IDKitWidget

app_id

"..."

action

"..." /> </ div

) Copy Copied!

Types

ISuccessResult

- Name
- merkle_root
- Type
- string
- Description
- This is the hash pointer to the root of the Merkle tree that proves membership of the user's identity in the
- list of identities verified by the Orb. ABI encoded.
- Name
- nullifier_hash
- Type
- string
- Description
- Essentially the user's unique identifier for your app (and specific action if using Incognito Actions). ABI
- encoded.
- Name

- proof
- Type
- string
- Description
- The Zero-knowledge proof of the verification. ABI encoded.
- Name
- verification_level
- Type
- "orb" | "device"
- Description
- Eitherorb
- ordevice
- . Returns the verification level used to generate the proof.
- Name
- credential_type
- Type
- "orb" | "device"
- Deprecated
- DEPRECATED
- Description
- Eitherorb
- ordevice
- . Will always return the strongest credential with which a user has been verified.
- This property is deprecated and will be removed in a future release. Useverification_level
- instead.

ISuccessResult

{ "merkle_root" :

"0x1f38b57f3bdf96f05ea62fa68814871bf0ca8ce4dbe073d8497d5a6b0a53e5e0" , "nullifier_hash" :

"0x0339861e70a9bdb6b01a88c7534a3332db915d3d06511b79a5724221a6958fbe" , "proof" :

"0x063942fd7ea1616f17787d2e3374c1826ebcd2d41d2394..." , "verification_level" :

"orb" } Copy Copied!

IErrorState

- Name
- code
- Type
- string
- Description
- The error code.
- Name
- detail
- Type
- string
- Description
- A human-readable description of the error.

IErrorState

{ "code" :

"already_signed" , "detail" :

"User has previously signed and submitted proof for this action." } Copy Copied!

Error Handling

An error in IDKit will generally be returned as the input to theonError callback. IDKit will display an error to the user and call theonError callback with anIErrorState object when the modal is closed.

View theErrors Reference for assistance when troubleshooting.

IDKit Standalone

The@worldcoin/idkit-standalone package is intended for vanilla JS applications. It is a standalone package that acts as a wrapper around the@worldcoin/idkit package.

Methods

The.init() and.update() methods take the same parameters as the React package's IDKitWidget component. Seeabove for more details.

.init()

The.init() method is the main initialization method used for vanilla JS apps. It should be called to start up IDKit and configure the widget.

import { IDKit } from

'@worldcoin/idkit-standalone'

const

onSuccess

= (result) => { // handle success }

IDKit .init ({ app_id :

'app_lshSNnaJfdt6Sohu6YAA' , action :

'my_action' , onSuccess : onSuccess , }) Copy Copied!

.update()

The.update() method reinitializes the widget with new parameters. It can only be called after the.init() method.

IDKit .update ({ app_id :

'app_lshSNnaJfdt6Sohu6YAA' , action :

'my_new_action' , onSuccess : onSuccess , }) Copy Copied!

.open()

The.open() method is used to open the widget. It can only be called after the.init() method, typically in response to a button click.

This method returns a Promise object that will resolve when theonSuccess callback is called, or reject when theonError callback is called.

IDKit .open () Copy Copied!

Advanced configuration

This section outlines advanced configuration options that may be unstable or subject to change. These options are passed as a JSON object to theadvanced prop of theIDKitWidget component.

Self-Hosted Applications

Self-hosted applications bypass the Worldcoin Developer Portal entirely. The proof returnedcan not be verified by the Developer Portal API. Instead, you must verify the proof on-chain or with a custom prover service.

When using self-hosted mode, noapp_id is required, and any value passed to IDKit will be ignored. Theaction you set must have sufficient uniqueness to avoid collisions with other applications. We recommend using a prefix that includes your application name, e.g.your_app_name_vote_1 .

import { IDKitWidget } from

'@worldcoin/idkit'

< IDKitWidget // no app_id is set for self-hosted applications action = "your_app_name_vote_1"

// this is your action, set to whatever you'd like signal = "user_value" onSuccess = {onSuccess} verification_level = "orb"

// only orb verifications are supported for self-hosted applications advanced = { self_hosted ,

// enable the self-hosted mode }

{({ open }) => < button

onClick

{open}>Verify with World ID</ button

} </ IDKitWidget

Copy Copied!