Bridging ETH to OP Mainnet With the Optimism SDK

This tutorial explains how you can use theOptimism SDK(opens in a new tab) to bridge ETH from L1 (Ethereum or Sepolia) to L2 (OP Mainnet or OP Sepolia). The Optimism SDK is an easy way to add bridging functionality to your javascript-based application. It also provides some safety rails to prevent common mistakes that could cause ETH or ERC-20 tokens to be made inaccessible.

Behind the scenes, the Optimism SDK uses theStandard Bridge contracts to transfer ETH and ERC-20 tokens. Make sure to check out theStandard Bridge guide if you want to learn more about how the bridge works under the hood.

Supported Networks

The Optimism SDK supports any of theSuperchain networks .Some Superchain networks(opens in a new tab) are already included in the SDK by default. If you want to use a network that isn't included by default you can simplyinstantiate the SDK with the appropriate contract addresses .

Dependencies

- node(opens in a new tab)
- pnpm(opens in a new tab)

Create a Demo Project

You're going to use the Optimism SDK for this tutorial. Since the Optimism SDK is a Node.js(opens in a new tab) library, you'll need to create a Node.js project to use it.

Make a Project Folder

mkdir

op-sample-project cd

op-sample-project

Initialize the Project

pnpm

init

Install the Optimism SDK

pnpm

add

@eth-optimism/sdk

Install ethers.js

pnpm

add

ethers@^5 Want to create a new wallet for this tutorial? If you havecast (opens in a new tab) installed you can runcast wallet new in your terminal to create a new wallet and get the private key.

Get ETH on Sepolia

This tutorial explains how to bridge ETH from Sepolia to OP Sepolia. You will need to get some ETH on Sepolia to follow along.

You can usethis faucet(opens in a new tab) to get ETH on Sepolia.

Add a Private Key to Your Environment

You need a private key in order to sign transactions. Set your private key as an environment variable with theexport command. Make sure this private key corresponds to an address that has ETH on Sepolia.

export TUTORIAL_PRIVATE_KEY = 0 x...

Start the Node REPL

You're going to use the Node REPL to interact with the Optimism SDK. To start the Node REPL run the following command in your terminal:

node This will bring up a Node REPL prompt that allows you to run javascript code.

Import Dependencies

You need to import some dependencies into your Node REPL session.

Import the Optimism SDK

const

optimism

=

require ("@eth-optimism/sdk")

Import ethers.js

const

ethers

=

require ("ethers")

Set Session Variables

You'll need a few variables throughout this tutorial. Let's set those up now.

Load your private key

const

privateKey

=

process . env . TUTORIAL_PRIVATE_KEY

Create the RPC providers and wallets

const

l1Provider

=

new

ethers . providers .StaticJsonRpcProvider ("https://rpc.ankr.com/eth_sepolia") const

l2Provider

=

new

ethers . providers .StaticJsonRpcProvider ("https://sepolia.optimism.io") const

l1Wallet

=

new

ethers .Wallet (privateKey , l1Provider) const

l2Wallet

=

new

ethers .Wallet (privateKey , l2Provider)

Get ETH

You're going to need some ETH on L1 that you can bridge to L2. You can get some Sepolia ETH from this faucet(opens in a new tab) .

Deposit ETH

Now that you have some ETH on L1 you can deposit that ETH into theL1StandardBridge contract. You'll then receive the same number of ETH on L2 in return.

Check your wallet balance on L1

See how much ETH you have on L1 so you can confirm that the deposit worked later on.

console .log ((await

l1Wallet .getBalance ()) .toString ())

Create a CrossChainMessenger instance

The Optimism SDK exports aCrossChainMessenger class that makes it easy to interact with theL1StandardBridge contract.

Create an instance of theCrossChainMessenger class:

const

messenger

=

new

optimism .CrossChainMessenger ({ l1ChainId :

11155111 ,

// 11155111 for Sepolia, 1 for Ethereum l2ChainId :

11155420 ,

// 11155420 for OP Sepolia, 10 for OP Mainnet l1SignerOrProvider : l1Wallet , l2SignerOrProvider : l2Wallet , })

Deposit your ETH

Now you can deposit your ETH into the Standard Bridge contract. You'll deposit a small amount of ETH just to demonstrate the process.

tx

await

messenger .depositETH (ethers . utils .parseEther ('0.006942')) await

tx .wait () Using a smart contract wallet? As a safety measure,depositETH will fail if you try to deposit ETH from a smart contract wallet without specifying arecipient . Add therecipient option to thedepositETH call to fix this. Check out theOptimism SDK docs(opens in a new tab) for more info on the options you can pass todepositETH .

Wait for the deposit to be relayed

You can use thewaitForMessageStatus function to wait for the deposit to be relayed to L2.

await

messenger .waitForMessageStatus (tx .hash ,

optimism . MessageStatus . RELAYED)

Check your wallet balance on L1

You should now have less ETH on L1.

console .log ((await

l1Wallet .getBalance ()) .toString ())

Check your wallet balance on L2

You should now have more ETH on L2.

console .log ((await

l2Wallet .getBalance ()) .toString ())

Withdraw ETH

You just bridged some ETH from L1 to L2. Nice! Now you're going to repeat the process in reverse to bridge some ETH from L2 to L1.

Start your withdrawal on L2

The first step to withdrawing ETH from L2 to L1 is to start the withdrawal on L2.

const

withdrawal

=

await

messenger .withdrawETH (ethers . utils .parseEther ('0.004269')) await

withdrawal .wait ()

Check your wallet balance on L2

You should now have less ETH on L2, but your wallet balance on L1 will not have changed yet.

console .log ((await

l2Wallet .getBalance ()) .toString ())

Wait until the withdrawal is ready to prove

The second step to withdrawing ETH from L2 to L1 is to prove to the bridge on L1 that the withdrawal happened on L2. You first need to wait until the withdrawal is ready to prove.

await

messenger .waitForMessageStatus (withdrawal .hash ,

optimism . MessageStatus . READY_TO_PROVE) This step can take a few minutes. Feel free to take a quick break while you wait.

Prove the withdrawal on L1

Once the withdrawal is ready to be proven, you'll send an L1 transaction to prove that the withdrawal happened on L2.

await

messenger .proveMessage (withdrawal .hash)

Wait until the withdrawal is ready for relay

The final step to withdrawing ETH from L2 to L1 is to relay the withdrawal on L1. This can only happen after the fault proof period has elapsed. On OP Sepolia, this is only a few seconds. On OP Mainnet, this takes 7 days.

await

messenger .waitForMessageStatus (withdrawal .hash ,

optimism . MessageStatus . READY_FOR_RELAY)

Relay the withdrawal on L1

Once the withdrawal is ready to be relayed you can finally complete the withdrawal process.

await

messenger .finalizeMessage (withdrawal .hash)

Wait until the withdrawal is relayed

Now you simply wait until the message is relayed.

await

messenger .waitForMessageStatus (withdrawal .hash ,

optimism . MessageStatus . RELAYED)

Check your wallet balance on L1

You should now have more ETH on L1.

console .log ((await

l1Wallet .getBalance ()) .toString ())

Next Steps

Congrats! You've just deposited and withdrawn ETH using the Optimism SDK. You should now be able to write applications that use the Optimism SDK to transfer ETH between L1 and L2. Although this tutorial used Sepolia and OP Sepolia, the same process works for Ethereum and OP Mainnet.

Communicating Between Chains in Solidity Bridging ERC-20 Tokens With the Optimism SDK