Triggering OP Mainnet Transactions from Ethereum

OP Mainnet currently uses a single-Sequencer block production model. This means that there is only one Sequencer active on the network at any given time. Single-Sequencer models are simpler than their highly decentralized counterparts but they are also more vulnerable to potential downtime.

Sequencer downtime must not be able to prevent users from transacting on the network. As a result, OP Mainnet includes a mechanism for "forcing" transactions to be included in the blockchain. This mechanism involves triggering a transaction on OP Mainnet by sending a transaction on Ethereum.

In this tutorial you'll learn how to trigger a transaction on OP Mainnet from Ethereum. You'll use the OP Sepolia testnet, but the same logic will apply to OP Mainnet.

Dependencies

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

Create a Demo Project

You're going to use the@eth-optimism/contracts-ts package for this tutorial. Since the@eth-optimism/contracts-ts package is aNode.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 Contracts Package

```
pnpm

add

@eth-optimism/contracts-ts
```

Install the Utils Package

```
pnpm

add

@eth-optimism/core-utils
```

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 and OP Sepolia

This tutorial explains how to bridge tokens from Sepolia to OP Sepolia. You will need to get some ETH on both of these testnets.

You can usethis faucet(opens in a new tab) to get ETH on Sepolia. You can use theSuperchain Faucet(opens in a new tab) to get ETH on OP 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 both Sepolia and OP 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 Contracts Package

const

contracts

=

require ("@eth-optimism/contracts-ts")

Import the Utils Package

const

utils

=

require ("@eth-optimism/core-utils")

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)
```

Check Your Initial Balance

You'll be sending a small amount of ETH as part of this tutorial. Quickly check your balance on OP Sepolia so that you know how much you had at the start of the tutorial.

```
const

initialBalance

=

await

l2Wallet .getBalance () console .log ( ethers . utils .formatEther (initialBalance))
```

Trigger the Transaction

Now you'll use theOptimismPortal (opens in a new tab) contract to trigger a transaction on OP Sepolia by sending a transaction on Sepolia.

Create the OptimismPortal object

```
const

OptimismPortal

=

new

ethers .Contract ( '0x16Fc5058F25648194471939df75CF27A2fdC48BC' , contracts .optimismPortalABI , l1Wallet , )
```

Estimate the required gas

When sending transactions via theOptimismPortal contract it's important to always include a gas buffer. This is because theOptimismPortal charges a variable amount of gas depending on the current demand for L2 transactions triggered via L1. If you do not include a gas buffer, your transactions may fail.

```
const
```

```
gas

=

await

OptimismPortal . estimateGas .depositTransaction ( '0x1000000000000000000000000000000000000000' ,

// _to ethers . utils .parseEther ( '0.000069420' ) ,

// _value 1e6 ,

// _gasLimit false ,

// _isCreation '0x' ,

// _data )
```

Send the transaction

Now you'll send the transaction. Note that you are including a buffer of 20% on top of the gas estimate.

```
const

tx

=

await

OptimismPortal .depositTransaction ( '0x1000000000000000000000000000000000000000' ,

// _to ethers . utils .parseEther ( '0.000069420' ) ,

// _value 1e6 ,

// _gasLimit false ,

// _isCreation '0x' ,

// _data { gasLimit :

gas .mul ( 120 ) .div ( 100 ) // Add 20% buffer } )
```

Wait for the L1 transaction

First you'll need to wait for the L1 transaction to be mined.

```
const

receipt

=

await

tx .wait ()
```

Wait for the L2 transaction

Now you'll need to wait for the corresponding L2 transaction to be included in a block. This transaction is automatically created as a result of your L1 transaction. Here you'll determine the hash of the L2 transaction using the@eth-optimism/core-utils library and then wait for that transaction to be included in the L2 blockchain.

```
const

deposit

=

utils . DepositTx .fromL1Receipt (receipt ,

0 ) await
```

l2Provider .waitForTransaction (deposit .hash ())

Check Your Updated Balance

You should have a little less ETH on OP Sepolia now. Check your balance to confirm.

const

finalBalance

=

await

l2Wallet .getBalance () console .log (ethers . utils .formatEther (finalBalance)) Make sure that the difference is equal to the amount you were expecting to send.

const

difference

=

initialBalance .sub (finalBalance) console .log (ethers . utils .formatEther (difference))

Next Steps

You've successfully triggered a transaction on OP Sepolia by sending a transaction on Sepolia. Although this tutorial demonstrated the simple example of sending a basic ETH transfer from your L2 address via the OptimismPortal contract, you can use this same technique to trigger any transaction you want. You can trigger smart contracts, send ERC-20 tokens, and more.

Estimating Transaction Costs Solidity Compatibility