Using the Optimism SDK

This tutorial will walk you through the process of using the Optimism SDK(opens in a new tab) to interact with your OP Stack chain. The Optimism SDKnatively supports various OP Chains including OP Mainnet and Base. To check whether your OP Chain is already supported, see the Optimism SDK docs(opens in a new tab).

You will need to already have created your own OP Stack chain to follow this tutorial. Check out the tutorial on Creating Your Own L2 Rollup Testnet if you haven't done so already.

Dependencies

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

Find Contract Addresses

You will need to find the addresses for a number of smart contracts that are part of your OP Stack chain in order to use the Optimism SDK. If you followed the instructions in the Creating Your Own L2 Rollup Testnet tutorial, you can find the addresses by looking at the JSON artifacts within the directoryoptimism/packages/contracts-bedrock/deployments/getting-started .

Simply run the following command from inside the directoryoptimism/packages/contracts-bedrock to print the list of addresses you'll need:

./scripts/print-addresses.sh

getting-started

--sdk Make sure you have jq (opens in a new tab) installed when running this command. You should see output similar to the following:

AddressManager: 0x... L1CrossDomainMessengerProxy: 0x... L1StandardBridgeProxy: 0x... L2OutputOracleProxy: 0x... OptimismPortalProxy: 0x... Save these addresses somewhere so you can use them in the next section.

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-sdk-sample-project cd

op-sdk-sample-project

Initialize the Project

pnpm

init

Install the Optimism SDK

pnpm

add

@eth-optimism/sdk

Install ethers.js

pnpm

add

ethers@^5

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.

Create the RPC providers

const

l1Provider

=

new

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

l2Provider

=

new

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

Set the contract addresses

Using the addresses you accessed earlier, set the contract addresses in the following variables:

const

AddressManager

=

'0x...' const

L1CrossDomainMessenger

=

'0x...' const

L1StandardBridge

=

'0x...' const

OptimismPortal

=

'0x...' const

L2OutputOracle

=

'0x...'

Set the chain IDs

const

l1ChainId

=

await

l1Provider .getNetwork () .then (network =>

network .chainId) const

l2ChainId

=

await

l2Provider .getNetwork () .then (network =>

network .chainId)

Initialize the Optimism SDK

You can now create an instance of theCrossChainMessenger object from the Optimism SDK. This will allow you to easily handle cross-domain messages between L1 and L2.

Simply create the object:

const

messenger

=

new

optimism .CrossChainMessenger ({ l1SignerOrProvider : l1Provider , l2SignerOrProvider : l2Provider , l1ChainId , l2ChainId ,

// This is the only part that differs from natively included chains. contracts : { l1 : { AddressManager , L1CrossDomainMessenger , L1StandardBridge , OptimismPortal , L2OutputOracle ,

// Need to be set to zero for this version of the SDK. StateCommitmentChain :

ethers . constants .AddressZero , CanonicalTransactionChain :

ethers . constants .AddressZero , BondManager :

ethers . constants .AddressZero , } } }) Note that you've passed in the RPC providers you created earlier, the addresses of the smart contracts you deployed, and the chain ID of your OP Stack chain.

Next Steps

You can now use the SDK to interact with your OP Stack chain just like you would with other chains like OP Mainnet. See existing tutorials, like the tutorial on Bridging ETH With the Optimism SDK or Bridging ERC-20 Tokens With the Optimism SDK , for examples of how to use the Optimism SDK.

Creating Your Own L2 Rollup Testnet Adding Attributes to the Derivation Function