How to Configure Challenger For Your Chain

This guide provides a walkthrough of setting up the configuration and monitoring options forop-challenger . See theOP-Challenger Explainer for a general overview of this fault proofs feature.

Build the Executable

- Clone the monorepo

git

clone

https://github.com/ethereum-optimism/optimism.git * Check out thelatest release ofop-challenger (opens in a new tab)* and use the commit to deploy. Alternatively, chain operators can use the prebuiltchallenger docker images(opens in a new tab)* . * If a Docker image is used, it already comes withop-program * server and an executable for Cannon embedded, so the Cannon bin doesn't need to be specified. * git * checkout * op-challenger/vX.Y.Z

⚠ Chain operators need to specify the arguments andop-program server ifop-challenger is running outside of Docker, but there's a Cannon server option which points toop-program 's executable. * Build challenger

cd

optimism pnpm

install make

op-challenger

Configure Challenger

- Configure challenger with the required flags. Tip: Use theop-challenger --help
- to view all subcommands, command line, and environment variable options.
- The example config file below shows the flags to configure in this step:

challenger: user: "1000" image: us-docker.pkg.dev/oplabs-tools-artifacts/images/op-challenger:v0.2.11 command: - "op-challenger" - "--l1-eth-rpc=http://sepolia-el-1:8545" - "--l1-beacon=http://sepolia-cl-1:5051" - "--l2-eth-rpc=http://op-sepolia-el-1:8545" - "--rollup-rpc=http://op-sepolia-cl-1:5051" - "--selective-claim-resolution" - "--private-key=...." - "--network=..." - "--datadir=/data" - "--cannon-prestates-url=..." volumes: - "./challenger-data:/data"

--l1-eth-rpc

- This is the HTTP provider URL for a standard L1 node, can be a full node.op-challenger
- will be sending many requests, so chain operators need a node that is trusted and can easily handle many transactions.
- Note: Challenger has a lot of money, and it will spend it if it needs to interact with games. That might risk not defending games or challenging games correctly, so chain operators should really trust the nodes being pointed at Challenger.

--l1-beacon

- This is needed just to get blobs from.
- In some instances, chain operators might need a blob archiver or L1 consensus node configured not to prune blobs:* If the chain is proposing regularly, a blob archiver isn't needed. There's only a small window in the blob retention period that games can be played.
-
    - If the chain doesn't post a valid output root in 18 days, then a blob archiver running a challenge game is needed. If the actor gets pushed to the bottom of the game, it could lose if it's the only one protecting the chain.

--l2-eth-rpc

- This needs to beop-geth
- archive node, withdebug
- enabled.
- Technically doesn't need to go to bedrock, but needs to have access to the start of any game that is still in progress.

--rollup-rpc

- This needs to be anop-node
- archive node because challenger needs access to output roots from back when the games start.

--private-key

- Chain operators must specify a private key or use something else (likeop-signer
-).
- This uses the same transaction manager arguments asop-node
- , batcher, and proposer, so chain operators can choose one of the following options:* a mnemonic
-
    - a private key
-
    - op-signer
-
    - endpoints

--network

- This identifies the L2 networkop-challenger
- is running for, e.g.,op-sepolia
- orop-mainnet
- .
- When using the--network
- flag, the--game-factory-address
- will be automatically pulled from thesuperchain-registry (opens in a new tab)
- .
- When cannon is executed, challenger needs the roll-up config and the L2 Genesis, which is op-geth's Genesis file. Both files are automatically loaded when Cannon Network is used, but custom networks will need to specify both Cannon L2 Genesis and Cannon rollup config.
- For custom networks not in thesuperchain-registry (opens in a new tab)
- , the--game-factory-address
- and rollup must be specified, as follows:
- --cannon-rollup-config rollup.json \
- --cannon-l2-genesis genesis-l2.json \

- # use this if running challenger outside of the docker image

- --cannon-server ./op-program/bin/op-program \

- # json or url, version of op-program deployed on chain

- # if you use the wrong one, you will lose the game

- # if you deploy your own contracts, you specify the hash, the root of the json file

- # op mainnet are tagged versions of op-program

- # make reproducable prestate

- # challenger verifies that onchain

- --cannon-prestate ./op-program/bin/prestate.json \

- # load the game factory address from system config or superchain registry

- # point the game factory address at the dispute game

factory proxy

- --game-factory-address

⚠ These options vary based on which--network is specified. Chain operators always need to specify a way to load prestates and must also specify the cannon-server whenever the docker image isn't being used.

--datadir

- This is a directory thatop-challenger
- can write to and store whatever data it needs. It will manage this directory to add or remove data as needed under that directory.
- If running in docker, it should point to a docker volume or mountpoint, so the data isn't lost on every restart. The data can be recreated if needed but particularly if challenger has executed cannon as part of responding to a game it may mean a lot of extra processing.

--cannon-prestates-url

The pre-state is effectively the version ofop-program that is deployed on chain. And chain operators must use the right version.op-challenger will refuse to interact with games that have a different absolute prestate hash to avoid making invalid claims. If deploying your own contracts, chain operators must specify an absolute prestate hash taken from themake reproducible-prestate command during contract deployment, which will also build the required prestate json file.

All governance approved releases use a tagged version ofop-program . These can be rebuilt by checking out the version tag and runningmake reproducible-prestate .

- There are two ways to specify the prestate to use:* --cannon-prestate
-
    - : specifies a path to a single Cannon pre-state Json file
-
    - --cannon-prestates-url
-
    - : specifies a URL to load pre-states from. This enables participating in games that use different prestates, for example due to a network upgrade. The prestates are stored in this directory named by their hash.
- Example final Url for a prestate:* https://example.com/prestates/0x031e3b504740d0b1264e8cf72b6dde0d497184cfb3f98e451c6be8b33bd3f808.json(opens in a new tab)
-
    - This file contains the cannon memory state.

⚠ Challenger will refuse to interact with any games if it doesn't have the matching prestate.

Execute Challenger

The final step is to execute challenger with the required flags. It will look something like this (but with required flags added):

./op-challenger/bin/op-challenger \ --trace-type

cannon \ --l1-eth-rpc

http://localhost:8545 \ --rollup-rpc

http://localhost:9546 \ --game-factory-address DISPUTE_GAME_FACTORY \ --datadir

temp/challenger-data \ --cannon-rollup-config

.devnet/rollup.json \ --cannon-l2-genesis

.devnet/genesis-l2.json \ --cannon-bin

./cannon/bin/cannon \ --cannon-server

./op-program/bin/op-program \ --cannon-prestate

./op-program/bin/prestate.json \ --l2-eth-rpc

http://localhost:9545 \ --mnemonic

"test test test test test test test test test test test junk" \ --hd-path

"m/44'/60'/0'/0/8" \

Test and Debug Challenger (Optional)

This is an optional step to useop-challenger subcommands, which allow chain operators to interact with the fault proof system onchain for testing and debugging purposes. For example, it is possible to test and explore the system in the following ways:

- create games yourself, and it doesn't matter if the games are valid or invalid.
- perform moves in games and then claim and resolve things.

Here's the list of op-challenger subcommands:

subcommand description list-games List the games created by a dispute game factory list-claims List the claims in a dispute game list-credits List the credits in a dispute game create-game Creates a dispute game via the factory move Creates and sends a move transaction to the dispute game resolve Resolves the specified dispute game if possible resolve-claim Resolves the specified claim if possible Additionally, chain operators should consider runningop-dispute-mon . It's an incredibly useful securities monitoring service to keep an eye on games, basically giving chain operators visibility into all the status of the games for the last 28 days. Chain operators can easily create their grafana dashboard for Dispute Monitor using the following json file:Download the Dispute Monitor JSON .

Next Steps

- Additional questions? See the FAQ section in theOP Challenger Explainer
- .
- For more detailed info onop-challenger
- , see thespecs(opens in a new tab)
- .
- If you experience any problems, please reach out todeveloper support(opens in a new tab)
- .

Integrating a New DA Layer Block Explorer