IBC Relayer

The following information is based off ofthis guide created byOsmosis Team .

Minimum Requirements

- 8-core (4 physical core), x86_64 architecture processor
- 32 GB RAM (or equivalent swap file set up)
- 1TB+ nVME drives

If running many nodes on a single VM,ensure your open files limit is increased

Prerequisites

In this guide, we will be relaying between Neutron and Cosmos Hub. When setting up your Cosmos Hub and Neutron full nodes, be sure to offset the ports being used in both the app.toml and config.toml files of the respective chains (we will show how to do this below).

Neutron Node Settings

Here we will leavegrpc server on port 9090 in theapp.toml directory:

nano HOME/.neutrond/config/app.toml [grpc]

Enable defines if the gRPC server should be enabled.

enable

=

true

Address defines the gRPC server address to bind to.

address

=

"0.0.0.0:9090" Here we will leave thepprof_laddr set to port 6060,rpc laddr to port 26657, andp2p laddr to 26656 in theconfig.toml directory:

nano HOME/.neutrond/config/config.toml

pprof listen address (https://golang.org/pkg/net/http/pprof)

pprof_laddr

=

"localhost:6060" [rpc]

TCP or UNIX socket address for the RPC server to listen on

laddr

=

"tcp://127.0.0.1:26657" [p2p]

Address to listen for incoming connections

laddr

=

"tcp://0.0.0.0:26656"

Target Chain Node Settings

Here we will setgrpc server to port 9092 in theapp.toml directory:

nano HOME/.gaiad/config/app.toml [grpc]

Enable defines if the gRPC server should be enabled.

enable

=

true

Address defines the gRPC server address to bind to.

address

=

"0.0.0.0:9092" Here I will set thepprof_laddr to port 6062,rpc laddr to port 26757, andp2p laddr to 26756 in theconfig.toml directory:

nano HOME/.gaiad/config/config.toml

pprof listen address (https://golang.org/pkg/net/http/pprof)

pprof_laddr

=

"localhost:6062" [rpc]

TCP or UNIX socket address for the RPC server to listen on

laddr

=

"tcp://127.0.0.1:26757" [p2p]

Address to listen for incoming connections

laddr

=

"tcp://0.0.0.0:26756"

Install Rust Dependencies

Install the following rust dependencies:

curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh source HOME/.cargo/env sudo apt-get install pkg-config libssl-dev sudo apt install librust-openssl-dev build-essential git

Build & setup Hermes

Use rust version1.71.0 to install, otherwise it might not compile! This will install hermes intoHOME/.cargo/bin/ directory:

cargo install ibc-relayer-cli --bin hermes --locked --version 1.6.0 Make hermes config & keys directory, copy config-template to config directory:

mkdir -p HOME/.hermes mkdir -p HOME/.hermes/keys cp config.toml HOME/.hermes Check hermes version & config dir setup

hermes version

2023-08-07T15:27:49.236821Z INFO ThreadId(01) running Hermes v1.6.0 hermes 1.6.0 Edit hermes config (use ports according to the port configuration we set above, add only chains you want to relay)

nano HOME/.hermes/config.toml Neutron introduces smart-contract level callbacks for IBC packets. From an IBC relayer's perspective, this means that custom application logic can be executed when a packet is submitted to Neutron, which can potentially drain the relayer's funds. This naturally brings us to a situation in which protocols would prefer to set up their own relayers and restrict the channels they are willing to relay for. For example, you can do this by adding achains.packet_filter config:

[[chains]] id

=

'cosmoshub-4' rpc_addr

=

'http://127.0.0.1:26757' grpc_addr

=

'http://127.0.0.1:9092' event_source

=

{

mode

=

'push' ,

url

=

'ws://127.0.0.1:26757/websocket' ,

batch_delay

=

'200ms'

} rpc_timeout

=

'10s' account_prefix

=

'cosmos' key_name

=

'cosmos' address_type

=

{

derivation

=

'cosmos'

} store_prefix

=

'ibc' default_gas

=

2000000 max_gas

=

10000000 gas_price

=

{

price

=

0.005 ,

denom

=

'uatom'

} gas_adjustment

=

0.1 max_msg_num

=

25 max_tx_size

=

180000 clock_drift

=

'10s' max_block_time

=

'10s' trusting_period

=

'14days' memo_prefix

=

'' trust_threshold

=

{

numerator

=

'1' ,

denominator

=

'3'

} [chains.packet_filter] policy

=

'allow' list

=

[

allow relaying only for chanels created by a certain contract

['icacontroller-neutron14hj2tavq8fpesdwxxcu44rty3hh90vhujrvcmstl4zr3txmfvw9s5c2epq*' ,

'*'] ,]

[[chains]] id

=

'neutron-1'

Whether or not this is a CCV consumer chain. Default: false

Only specify true for CCV consumer chain (Neutron), but NOT for sovereign chains.

ccv_consumer_chain

=

true rpc_addr

=

'http://127.0.0.1:26657' grpc_addr

=

'http://127.0.0.1:9090' event_source

=

{

mode

=

'push' ,

url

=

'ws://127.0.0.1:26657/websocket' ,

batch_delay

=

'200ms'

} rpc_timeout

=

'10s' account_prefix

=

'neutron' key_name

=

'neutron' address_type

=

{

derivation

=

'cosmos'

} store_prefix

=

'ibc' default_gas

=

5000000 max_gas

=

15000000 gas_price

=

{

price

=

0.0026 ,

denom

=

'untrn'

} gas_adjustment

=

0.1 max_msg_num

=

20 max_tx_size

=

209715 clock_drift

=

'20s' max_block_time

=

'10s' trusting_period

=

'10days' memo_prefix

=

'Relayed by Czar' trust_threshold

=

{

numerator

=

'1' ,

denominator

=

'3'

} [chains.packet_filter] policy

=

'allow' list

=

[

allow relaying only for chanels created by a certain contract

['icacontroller-neutron14hj2tavq8fpesdwxxcu44rty3hh90vhujrvcmstl4zr3txmfvw9s5c2epq*' ,

'*'] ,] Add your relayer wallet to Hermes' keyring (located in HOME/.hermes/keys)

Best practice is to use the same mnemonic over all networks. Do not use your relaying-addresses for anything else because it will lead to account sequence errors.

hermes keys restore cosmoshub-4 -m "24-word mnemonic seed" hermes keys restore neutron-1 -m "24-word mnemonic seed" Ensure this wallet has funds in both NTRN and ATOM in order to pay the fees required to relay.

Final Checks

You can validate your hermes configuration file:

hermes config validate INFO ThreadId(01) using default configuration from '/home/relay/.hermes/config.toml' Success: "validation passed successfully" Perform health check to see if all connected nodes are up and synced:

hermes health-check INFO ThreadId(01) using default configuration from '/home/relay/.hermes/config.toml' INFO ThreadId(01) telemetry service running, exposing metrics at http://0.0.0.0:3001/metrics INFO ThreadId(01) starting REST

API server listening at http://127.0.0.1:3000 INFO ThreadId(01) [cosmoshub-4] chain is healthy INFO ThreadId(01) [neutron-1] chain is healthy When your nodes are fully synced you can start the hermes daemon:

hermes start Watch hermes' output for successfully relayed packets or any errors. It will try & clear any unreceived packets after startup has completed. Previous Overview Next ICQ Relayer