Relaying on Namada

This document describes how to operate a relayer for the Inter-Blockchain Communication (IBC) protocol with Namada. This documentation covers being able to create connections through IBC as well as setting up local chains of Namada for testing purposes.

This document covers essential steps for using IBC with Namada:

The below is intended for those that wish to relay IBC message transfers between two Namada chains. There is of course the capability to do this between any two IBC compatible chains (such as a Cosmos chain). In this case, it is necessary to have a node running on both the destination and the source chain in order to make any package transfers. Below, we discuss first how to enable this connection between two pre-existing chains by Hermes, and second, setting up two Namada local chains for this purpose.

Configure Hermes

Hermes is an IBC relayer to relay IBC packets between chains. Namada uses a fork of Hermes supporting Namada chains(opens in a new tab) .

Make Hermes config file

One essential piece of the puzzle is to create aconfig.toml file that describes what connections will be set up that the relayer will be responsible for.

export HERMES_CONFIG = "/config.toml" touch HERMES_CONFIG If you don't specify the file path,~/.hermes/config.toml is read as default.

You can find an example of the config file below. Essentially, you change only the chain IDs, the RPC addresses, and the key names in the config file for Namada. If you don't have nodes, please set up nodes manually or through ourscripts .

Example: config.toml [global] log_level =

'info'

[mode]

[mode.clients] enabled =

true refresh =

true misbehaviour =

true

[mode.connections] enabled =

false

[mode.channels] enabled =

false

[mode.packets] enabled =

true clear_interval =

10 clear_on_start =

false tx_confirmation =

true

[telemetry] enabled =

false host =

'127.0.0.1' port =

3001

[[chains]] id =

'namada-test.0a4c6786dbda39f786'

set your chain ID

type

'Namada' rpc_addr =

'http://127.0.0.1:27657'

set the IP and the port of the chain

grpc_addr

'http://127.0.0.1:9090'

not used for now

event_source

{ mode

'push' , url =

'ws://127.0.0.1:27657/websocket' , batch_delay =

'500ms' }

set the IP and the port of the chain

account_prefix

''

not used

key_name

'relayer'

The key is an account name you made

store_prefix

'ibc' gas_price = { price =

0.001 , denom =

'atest1v4ehgw36x9ry2dphx5mrvdjxgez5gdengeq5gs2pg3znx32yg9p5yv2zg3pnjvf4g9q5x329epndn0' }

the price isn't used for now, the denom should be a raw token address

[[chains]] id =

'namada-test.647287156defa8728c' type =

'Namada' rpc_addr =

'http://127.0.0.1:28657' grpc_addr =

'http://127.0.0.1:9090' event_source = { mode =

'push' , url =

'ws://127.0.0.1:28657/websocket' , batch_delay =

'500ms' } account_prefix =

'' key_name =

'relayer' store_prefix =

'ibc' gas_price = { price =

0.001 , denom =

'atest1v4ehgw36xazry3pn89q5xv3ngyungs3hxc65vwp3xez5zdp3xppy2v6yxc65zsfegyc5yd2ptuf9d8' } The path to the config file, which is saved in the variableHERMES_CONFIG will be useful later.

ð§© Interpreting the toml

Each chain configuration is specified under the[[chains]] object. These are the pieces of this puzzle you want to keep your ð on:

- chains.id
- is the name of the chain
- chains.rpc_address
- specifies the port that the channel is communicating through, and will be the argument for theledger_address
- of Namada when interacting with the ledger (will become clearer later)* Make sure to change the IP address to the IP address of your local machine that is running this node!
- chains.key_name
- specifies the key of the signer who signs a transaction from the relayer. The key should be generated before starting the relayer.
- chains.event_source
- specifies the URL of the chain's websocket. This must be the same as therpc_address
- for Hermes to work properly.
- chains.gas_price.denom
- specifies the token that the relayer pays for IBC transactions.chains.gas_price.price
- isn't used for now.

You can see more details of configuration inthe official document(opens in a new tab).

Export environment variables

The relaying user will need to save certain environment variables. These are:

export CHAIN_A_ID = "" export CHAIN_B_ID = "" export HERMES_CONFIG = ""

Install Hermes

Before conducting any IBC operations, one must download Heliax's fork Hermes binary or build it from source.

From binaries

One can download the latest binary release from ourreleases page(opens in a new tab)by choosing the appropriate architecture.

E.g.

export TAG = "v1.7.4-namada-beta7" export ARCH = "x86_64-unknown-linux-gnu"

or "aarch64-apple-darwin"

curl

-Lo

/tmp/hermes.tar.gz

https://github.com/heliaxdev/hermes/releases/download/ {TAG} /hermes- {TAG} - {ARCH} .tar.gz tar

-xvzf

/tmp/hermes.tar.gz

-C

/usr/local/bin For some systems,/usr/local/bin is a protected directory. In this case, you may need to run the above command withsudo . I.e

sudo

tar

-xvzf

/tmp/hermes.tar.gz

-C

/usr/local/bin This is also true for the commandcp ./target/release/hermes /usr/local/bin/ below (see the comment).

From source

export TAG = "v1.7.4-namada-beta7"

git

clone

https://github.com/heliaxdev/hermes.git git

checkout TAG cd

hermes cargo

build

--release

--bin

hermes export HERMES = (pwd)

if needed

Check the binary:

./target/release/hermes

--version

or sudo cp ./target/release/hermes /usr/local/bin/

It is recommended to now add hermes toPATH such that it is callable without any pre-fixes. For ubuntu users, this can be achieved by

cp

./target/release/hermes

/usr/local/bin/

Setting up the relayer

Create the relayer account

On each chain, there must be arelayer account. The alias should be the same aschains.key_name in the config. On a namada chain, this can be done by running

namadaw

gen

--alias

relayer This will generate a key for the relayer account. The key will be stored in thewallet.toml that is found in thbase directory of the node, inside thechain-id folder. For example, if thechain-id isnamada-test.0a4c6786dbda39f786 , thewallet.toml will be found inHOME/.local/share/namada/namada-test.0a4c6786dbda39f786/wallet.toml (on a ubuntu machine wherebase-dir has not been set up properly).

The relayer account should have some balance to pay the fee of transactions. Before creating an IBC channel or relaying an IBC packet, you need to transfer the fee token to the relayer account.

Add the relayer key to Hermes

To sign each transaction, the relayer's key should be added to Hermes withkeys add command in advance. It requires thewallet.toml which should have the key ofchains.key_name . Once the key has been added, Hermes doesn't need the wallet anymore.

hermes

--config HERMES_CONFIG keys

add

--chain CHAIN_ID --key-file WALLET_PATH Hermes will store the key in~/.hermes/keys{CHAIN_ID} as default. You can specify the directory by settingchains.key_store_folder in the config file.

If you want to use an encrypted key with a password, you have to set an environment variableNAMADA_WALLET_PASSWORD_FILE for the password file orNAMADA_WALLET_PASSWORD to avoid entering the password for each transaction submission. It is now possible to set up the client.

Create IBC channel

The "create channel" command (below) creates not only the IBC channel but also the necessary IBC client connection.

hermes

--config HERMES_CONFIG \ create

channel \ --a-chain CHAIN_A_ID \ --b-chain CHAIN_B_ID \ --a-port

transfer \ --b-port

transfer \ --new-client-connection

--yes Note that the aboveCHAIN_IDs will depend on your own setup, so do check this for yourself! When the creation has been completed, you can see the channel IDs. For example, the following text shows that a channel with ID7 has been created on Chain Anamada-test.0a4c6786dbda39f786 , and a channel with ID12 has been created on Chain Bnamada-test.647287156defa8728c . You will need the channel IDs for a transfer over IBC. It means that you have to specifychannel-7 as a channel ID (The prefixchannel- is always required) for a transfer from Chain A to Chain B. Also, you have to specifychannel-12 as a channel ID for a transfer from Chain B to Chain A.

SUCCESS Channel { ordering: Unordered, a_side: ChannelSide { chain: BaseChainHandle { chain_id: ChainId { id:

"namada-test.0a4c6786dbda39f786", version: 0, }, runtime_sender: Sender { .. }, }, client_id: ClientId("07-tendermint-0",), connection_id: ConnectionId("connection-3",), port_id: PortId("transfer",), channel_id: Some(ChannelId("channel-7",),), version: None, }, b_side: ChannelSide { chain: BaseChainHandle { chain_id: ChainId { id: "namada-test.647287156defa8728c", version: 0, }, runtime_sender: Sender { .. }, }, client_id: ClientId("07-tendermint-1",), connection_id: ConnectionId("connection-2",), port_id: PortId("transfer",), channel_id: Some(ChannelId("channel-12",),), version: None, }, connection_delay: 0ns, } Expand

Start the relayer

Once you run Hermes, it monitors chains via the nodes and relays packets according to monitored events.

hermes

--config HERMES_CONFIG start You can see more details of Hermes at the official document(opens in a new tab).

After the sync, you can create the channel and start Hermes as explained above .

create a channel

hermes

--config HERMES_CONFIG \ create

channel \ --a-chain CHAIN_A_ID \ --b-chain CHAIN_B_ID \ --a-port

transfer \ --b-port

transfer \ --new-client-connection

--yes

Transferring assets over IBC

It is now possible to transfer assets between the two chains.

Set up local Namada chains using the Hermes script

The scriptsetup-namada will set up two chains with one validator node, copy necessary files for Hermes, and make an account for Hermes on each ledger. Also, it will make a Hermes' config fileconfig_for_namada.toml in thehermes directory.

First, you will need to export some environment variables:

export NAMADA_DIR = "" export TAG = "v1.7.4-namada-beta7" git

clone

https://github.com/heliaxdev/hermes.git git

checkout TAG

The branch is the same as our Hermes

cd

hermes ./scripts/setup-namada NAMADA_DIR In this case, the user doesn't have to wait for sync. If the relayer account on each instance has enough balance, the user can create a channel and start Hermes immediately as explained above . The user finds these chain IDs of the chains in the config fileconfig_for_namada.toml . One can rungrep "id" {HERMES_CONFIG} .

create a channel

hermes

--config HERMES_CONFIG \ create

channel \ --a-chain CHAIN_A_ID \ --b-chain CHAIN_B_ID \ --a-port transfer \ --b-port

transfer \ --new-client-connection

--yes

Run Hermes

hermes

--config HERMES_CONFIG start Each node data and configuration files are inhermes/data/namada-*/.namada .

In order to close any ledgers setup by the script, one can run

killall

namadan