IBC relaying guide

Celestia usesIBC (Inter-Blockchain Communication protocol) to enable cross-chain transfer of tokens. To support this capability it relies on relayers, processes that can be run by anyone which constantly scan for outbound packets on one chain and submits these packets alongside corresponding proofs on the destination chain. This section describes how one can setup a relayer and create new connections between chains. There are two standard implementations:

- Hermes
- built in Rust
- Go Relayer
- built in Go

The following guide explains how to establish IBC connections and relay packets betweenMocha testnet andCosmos hub testnet networks by using the Hermes relayer.

Check thelatest celestia-app release'sgo.mod for the version of ibc-go that is currently used.

Hermes

Hermes is an open-source Rust implementation of an IBC relayer released as part of theibc-relayer-cli crate. It includes a CLI for relaying packets between Cosmos SDK chains, as well as Prometheus metrics and a REST API.

Please follow the steps atHermes Quick Start to install Hermes. Before proceeding, verify that Hermes is installed correctly by runninghermes version .

TIP

Hermes currently doesn't support configuring the TendermintCompatMode in chain config (seehermes#3623). Until that issue is resolved, please use Hermesv1.7.0 + because it falls back to TendermintCompatMode v0.34 (seehermes#3663) which is compatible with Celestia.

Configuration

After you have successfully installed Hermes and created the necessary folders, you now have to editconfig.toml and add the appropriate configurations for the chains you want to relay between.

For this tutorial, we will be using the following chains:

- Celestia'smocha-4
- testnet
- Cosmos Hub'stheta-testnet-001
- testnet

Edit the Hermes configuration.

bash vim HOME /.hermes/config.toml vim HOME /.hermes/config.toml toml [global] log_level = "info"

[mode . clients] enabled = true refresh = true misbehaviour = true

[mode . connections] enabled = false

[mode . channels] enabled = false

[mode . packets] enabled = true clear_interval = 100 clear_on_start = true tx_confirmation = false auto_register_counterparty_payee = false

[rest] enabled = false host = "127.0.0.1" port = 3000

[telemetry] enabled = false host = "127.0.0.1" port = 3001

[telemetry . buckets . latency_submitted] start = 500 end = 20000 buckets = 10

[telemetry . buckets . latency_confirmed] start = 1000 end = 30000 buckets = 10

[[chains]] id = "theta-testnet-001" type = "CosmosSdk" rpc_addr = "https://rpc.sentry-01.theta-testnet.polypore.xyz" grpc_addr = "https://grpc.sentry-01.theta-testnet.polypore.xyz" rpc_timeout = "10s" trusted_node = false account_prefix = "cosmos" key_name = "key-cosmos" key_store_type = "Test" store_prefix = "ibc" default_gas = 100000 max_gas = 400000 gas_multiplier = 1.5 max_msg_num = 30 max_tx_size = 180000 max_grpc_decoding_size = 33554432 clock_drift = "5s" max_block_time = "30s" ccv_consumer_chain = false memo_prefix = "" sequential_batch_tx = false

[chains . event_source] mode = "push" url = "ws://rpc.sentry-01.theta-testnet.polypore.xyz:26657/websocket" batch_delay = "500ms"

[chains . trust_threshold] numerator = "1" denominator = "3"

[chains . gas_price] price = 0.025 denom = "uatom"

[chains . packet_filter] policy = "allow" list = [["transfer" , "channel-3108"]]

[chains . packet_filter . min_fees]

[chains . address_type] derivation = "cosmos"

[[chains]] id = "mocha-4" type = "CosmosSdk" rpc_addr = "https://rpc-celestia-mocha.architectnodes.com" grpc_addr = "https://grpc.celestia-mocha.com:443" rpc_timeout = "10s" trusted_node = false account_prefix = "celestia" key_name = "celestia-key" key_store_type = "Test" store_prefix = "ibc" default_gas = 100000 max_gas = 400000 gas_multiplier = 1.5 max_msg_num = 30 max_tx_size = 180000 max_grpc_decoding_size = 33554432 clock_drift = "5s" max_block_time = "30s" ccv_consumer_chain = false memo_prefix = "" sequential_batch_tx = false

[chains . event_source] mode = "push" url = "ws://rpc-mocha.pops.one:26657/websocket" batch_delay = "500ms"

[chains . trust_threshold] numerator = "1" denominator = "3"

[chains . gas_price] price = 0.1 denom = "utia"

[chains . packet_filter] policy = "allow" list = [["transfer" , "channel-0"]]

[chains . packet_filter . min_fees]

[chains . address_type] derivation = "cosmos" [global] log_level = "info"

[mode . clients] enabled = true refresh = true misbehaviour = true

[mode . connections] enabled = false

[mode . channels] enabled = false

[mode . packets] enabled = true clear_interval = 100 clear_on_start = true tx_confirmation = false auto_register_counterparty_payee = false

[rest] enabled = false host = "127.0.0.1" port = 3000

[telemetry] enabled = false host = "127.0.0.1" port = 3001

[telemetry . buckets . latency_submitted] start = 500 end = 20000 buckets = 10

[telemetry . buckets . latency_confirmed] start = 1000 end = 30000 buckets = 10

[[chains]] id = "theta-testnet-001" type = "CosmosSdk" rpc_addr = "https://rpc.sentry-01.theta-testnet.polypore.xyz" grpc_addr = "https://grpc.sentry-01.theta-testnet.polypore.xyz" rpc_timeout = "10s" trusted_node = false account_prefix = "cosmos" key_name = "key-cosmos" key_store_type = "Test" store_prefix = "ibc" default_gas = 100000 max_gas = 400000 gas_multiplier = 1.5 max_msg_num = 30 max_tx_size = 180000 max_grpc_decoding_size = 33554432 clock_drift = "5s" max_block_time = "30s" ccv_consumer_chain = false memo_prefix = "" sequential_batch_tx = false

[chains . event_source] mode = "push" url = "ws://rpc.sentry-01.theta-testnet.polypore.xyz:26657/websocket" batch_delay = "500ms"

[chains . trust_threshold] numerator = "1" denominator = "3"

[chains . gas_price] price = 0.025 denom = "uatom"

[chains . packet_filter] policy = "allow" list = [["transfer" , "channel-3108"]]

[chains . packet_filter . min_fees]

[chains . address_type] derivation = "cosmos"

[[chains]] id = "mocha-4" type = "CosmosSdk" rpc_addr = "https://rpc-celestia-mocha.architectnodes.com" grpc_addr = "https://grpc.celestia-mocha.com:443" rpc_timeout = "10s" trusted_node = false account_prefix = "celestia" key_name = "celestia-key" key_store_type = "Test" store_prefix = "ibc" default_gas = 100000 max_gas = 400000 gas_multiplier = 1.5 max_msg_num = 30 max_tx_size = 180000 max_grpc_decoding_size = 33554432 clock_drift = "5s" max_block_time = "30s" ccv_consumer_chain = false memo_prefix = "" sequential_batch_tx = false

[chains . event_source] mode = "push" url = "ws://rpc-mocha.pops.one:26657/websocket" batch_delay = "500ms"

[chains . trust_threshold] numerator = "1" denominator = "3"

[chains . gas_price] price = 0.1 denom = "utia"

[chains . packet_filter] policy = "allow" list = [["transfer" , "channel-0"]]

[chains . packet_filter . min_fees]

[chains . address_type] derivation = "cosmos"

Add relayer wallets

Now that we have successfully configured our relaying chains, we need to import the wallets that will be used for relaying. Please note that both wallets need to be funded with the native tokens of each chain.

You can get testnet tokens from faucets for bot testnets via Discord:

- Celestia:https://discord.gg/celestiacommunity
- Cosmos Hub:https://discord.gg/cosmosnetwork

Add your seed phrase to a file and upload it to the server. Do not use wallets for anything else but relaying to avoid running into account sequence errors.

Follow the steps atadding-keys-to-hermes to add keys for each chain

bash hermes

keys

add

--chain

mocha-4

--mnemonic-file

< seed-fil e

hermes

keys

add

--chain

theta-testnet-001

--mnemonic-file

< seed-fil e

hermes

keys

add

--chain

mocha-4

--mnemonic-file

< seed-fil e

hermes

keys

add

--chain

theta-testnet-001

--mnemonic-file

< seed-fil e

Verify configuration files

After editingconfig.toml and adding wallet keys, it’s time to test the configurations and ensure the system is healthy. Run the following:

bash hermes

health-check hermes

config

validate hermes

health-check hermes

config

validate If everything was set up correctly, you should see output like:

bash SUCCESS

performed

health

check

for

all

chains

in

the

config SUCCESS

"configuration is valid" SUCCESS

performed

health

check

for

all

chains

in

the

config SUCCESS

"configuration is valid"

Create a connection between 2 chains

If you’re attempting to create new connections, verify that the chains in question don’t already have connections and clients

in place and use the existing ones if they do. In that case you canskip this step and go toConfigure channels in Hermes section.

In this example, we are creating new clients and a new connection betweenmocha-4 andtheta-testnet-001 networks.

Create clients

To learn if a client already exists, you can use the following command:

bash hermes

query

clients

--host-chain

mocha-4

--reference-chain

theta-testnet-001 hermes

query

clients

--host-chain

mocha-4

--reference-chain

theta-testnet-001 To create a new client, use thecreate-client command :

bash hermes

create

client

--host-chain

mocha-4

--reference-chain

theta-testnet-001 hermes

create

client

--host-chain

mocha-4

--reference-chain

theta-testnet-001 Create a second client:

bash hermes

create

client

--host-chain

theta-testnet-001

--reference-chain

mocha-4 hermes

create

client

--host-chain

theta-testnet-001

--reference-chain

mocha-4

Open connection over new clients

To create a new connection over clients, use the following command:

bash hermes

create

connection

--a-chain

mocha-4

--b-chain

theta-testnet-001 hermes

create

connection

--a-chain

mocha-4

--b-chain

theta-testnet-001 You should be seeing a similar output to this:

bash SUCCESS

Connection

{ delay_period:

0 ns, a_side:

ConnectionSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"theta-testnet-001", version:

0 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-2382" ,) , connection_id:

Some (ConnectionId("connection-2727" ,) ,), }, b_side:

ConnectionSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"mocha-4", version:

4 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-0" ,) , connection_id:

Some (ConnectionId("connection-0" ,) ,), }, } SUCCESS

Connection

{ delay_period:

0 ns, a_side:

ConnectionSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"theta-testnet-001", version:

0 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-2382" ,) , connection_id:

Some (ConnectionId("connection-2727" ,) ,), }, b_side:

ConnectionSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"mocha-4", version:

4 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-0" ,) , connection_id:

Some (ConnectionId("connection-0" ,) ,), }, } Now that the connection has been established over the clients, we need to create a new channel, by leveraging an existing connection:

bash hermes

create

channel

--a-chain

theta-testnet-001

--a-connection

connection-2727

--a-port

transfer

--b-port

transfer hermes

create

channel

--a-chain

theta-testnet-001

--a-connection

connection-2727

--a-port

transfer

--b-port

transfer You should be seeing a similar output to this:

bash SUCCESS

Channel

{ ordering:

Unordered, a_side:

ChannelSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"theta-testnet-001", version:

0 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-2382" ,) , connection_id:

ConnectionId ("connection-2727" ,) , port_id:

PortId ("transfer" ,) , channel_id:

Some (ChannelId("channel-3152" ,) ,), version:

None, }, b_side:

ChannelSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"mocha-4", version:

4 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-0" ,) , connection_id:

ConnectionId ("connection-0" ,) , port_id:

PortId ("transfer" ,) , channel_id:

Some (ChannelId("channel-0" ,) ,), version:

None, }, connection_delay:

0 ns, } SUCCESS

Channel

{ ordering:

Unordered, a_side:

ChannelSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"theta-testnet-001", version:

0 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-2382" ,) , connection_id:

ConnectionId ("connection-2727" ,) , port_id:

PortId ("transfer" ,) , channel_id:

Some (ChannelId("channel-3152" ,) ,), version:

None, }, b_side:

ChannelSide

{ chain:

BaseChainHandle

{ chain_id:

ChainId

{ id:

"mocha-4", version:

4 , }, runtime_sender:

Sender

{

..

}, }, client_id:

ClientId ("07-tendermint-0" ,) , connection_id:

ConnectionId ("connection-0" ,) , port_id:

PortId ("transfer" ,) , channel_id:

Some (ChannelId("channel-0" ,) ,), version:

None, }, connection_delay:

0 ns, } Congratulations!

You have successfully created a new IBC connection between two networks.

Configure channels in Hermes

Now that we have created new connections and opened channels, we need to editconfig.toml again and add the newly created channels, or use the already existing ones.

Formocha-4 add:

bash [chains.packet_filter] policy

=

'allow' list

= [['transfer' , 'channel-0'] ,

theta-testnet-001

] [chains.packet_filter] policy

=

'allow' list

= [['transfer' , 'channel-0'] ,

theta-testnet-001

] Fortheta-testnet-001 add:

bash [chains.packet_filter] policy

=

'allow' list

= [['transfer' , 'channel-3108'] ,

mocha-4

] [chains.packet_filter] policy

=

'allow' list

= [['transfer' , 'channel-3108'] ,

mocha-4

]

Start the relayer

Start the relayer viahermes start

Transfer

The Celestia state machine is built with the IBC transfer module, allowing for the native Celestia token to be transferred to any other IBC enabled chain. Transfer can be initialized through thecelestia-appd CLI. Information can be found via the help label as follows:

bash celestia-appd

tx

ibc-transfer

transfer

--help celestia-appd

tx

ibc-transfer

transfer

--help

Token filter

The transfer module uses a token filter middleware which serves to prevent non-native Celestia tokens from being on Celestia. If a user is to try to send a token from another chain across, it will be simply rejected and the token returned back to the user. [][Edit this page on GitHub] Last updated: Previous page Consensus Next page Metrics []