Getting started

This guide serves as a brief guide on how to utilize the pricefeed module in your Cosmos SDK app.

Prerequisites

Be sure you have met the prerequisites before you follow this guide.

Operating systems

- Ubuntu 22.04

Go (for ignite and consumer chain)

- 1.19.5 or higher

Ignite CLI

- 0.27.2 or higher

Rust (for Hermes Relayer)

- 1.65.0 or higher

Installation

Installing build-essential package in Ubuntu

sudo apt update && sudo apt install build-essential

Creating a new blockchain

To create a new blockchain project with Ignite, you will need to run the following command:

ignite scaffold chain example The ignite scaffold chain command will create a new blockchain in a new directory example.

Step 1: Replace the genesis state

Config proposal voting period by Ignite

To expedite the testing of the pricefeed module, modify the default voting period to 40 seconds using Ignite feature to replace the genesis state by incorporating this code inconfig.yml .

... genesis : app_state : gov : params : voting_period :

"40s"

Initiate source channel and symbol requests by Ignite

To utilize the Ignite feature to replace the genesis state without openupdate-symbol-requests proposal, insert the code shown below into theconfig.yml file.

... genesis : app_state : ... pricefeed : params : source_channel :

"channel-0" symbol_requests :

[{ "symbol" :

"BAND" ,

"oracle_script_id" :

396 ,

"block_interval" :

40 }]

Step 2: Import pricefeed module to your cosmos app

Edit Cosmos SDK and IBC-go version

To ensure compatibility with the pricefeed module, kindly update the Cosmos SDK version tov0.47.5 .

require (... github . com / cosmos / Cosmos SDK v0 . 47.5) Additionally, ensure that ibc-go version is v7.2.1

require (... github . com / cosmos / ibc - go / v7 v7 . 2.1) Then rungo mod tidy to update all module packages.

Add pricefeed keeper inapp/app.go

Add pricefeed proposal

import

(... pricefeedclient "github.com/bandprotocol/oracle-consumer/x/pricefeed/client")

func

getGovProposalHandlers ( )

[ ] govclient . ProposalHandler { var govProposalHandlers [ ] govclient . ProposalHandler

govProposalHandlers

append (... pricefeedclient . ProposalHandler ,)

return govProposalHandlers }

Add pricefeed module basic

import

(... pricefeed "github.com/bandprotocol/oracle-consumer/x/pricefeed")

ModuleBasics

module . NewBasicManager (... pricefeed . AppModuleBasic { } ,)

Add pricefeed keeper type

import

(... pricefeedkeeper "github.com/bandprotocol/oracle-consumer/x/pricefeed/keeper")

type App struct

{ ... PricefeedKeeper pricefeedkeeper . Keeper

... ScopedPricefeedKeeper capabilitykeeper . ScopedKeeper }

Add pricefeed store key

import

(... pricefeedtypes "github.com/bandprotocol/oracle-consumer/x/pricefeed/types")

keys := sdk . NewKVStoreKeys (... pricefeedtypes . StoreKey ,)

Create a new pricefeed keeper

scopedPricefeedKeeper := app . CapabilityKeeper . ScopeToModule (pricefeedtypes . ModuleName) app .

ScopedPricefeedKeeper = scopedPricefeedKeeper app . PricefeedKeeper = pricefeedkeeper . NewKeeper (appCodec , keys [pricefeedtypes . StoreKey] , app . GetSubspace (pricefeedtypes . ModuleName) , app . IBCKeeper . ChannelKeeper , app . IBCKeeper . ChannelKeeper , & app . IBCKeeper . PortKeeper , scopedPricefeedKeeper ,)

Create a new pricefeed module

import

(... pricefeedmodule "github.com/bandprotocol/oracle-consumer/x/pricefeed")

pricefeedModule := pricefeedmodule . NewAppModule (appCodec , app . PricefeedKeeper) pricefeedIBCModule := pricefeedmodule . NewIBCModule (app . PricefeedKeeper)

Add pricefeed module in IBC router

ibcRouter . AddRoute (...) . AddRoute (pricefeedtypes . ModuleName , pricefeedIBCModule)

Add pricefeed module in governance Handler router

govRouter . AddRoute (...) . AddRoute (pricefeedtypes . RouterKey , pricefeedmodule . NewUpdateSymbolRequestProposalHandler (app . PricefeedKeeper))

Add pricefeed module in the module manager

app . mm = module . NewManager (... , pricefeedModule ,)

Set pricefeed order in begin block, end block and init genesis

app . mm . SetOrderBeginBlockers (... , pricefeedtypes . ModuleName ,)

app . mm . SetOrderEndBlockers (... , pricefeedtypes . ModuleName ,)

app . mm . SetOrderInitGenesis (pricefeedtypes . ModuleName ,)

Set pricefeed order for deterministic simulations

app . sm = module . NewSimulationManager (... pricefeedModule ,)

Add pricefeed subspace in params Keeper

func

initParamsKeeper (...) paramskeeper . Keeper { paramsKeeper . Subspace (...) paramsKeeper . Subspace (pricefeedmoduletypes . ModuleName) } Once you have completed the addition of the pricefeed module in the app.go file, execute the commandgo mod tidy to import and update the necessary modules.

Now have completed importing the pricefeed module and can now execute the chain by running this command

ignite chain serve -v

Step 3: Setup a relayer

The second step is to set up a relayer to listen and relay IBC packets between your chain and BandChain.

Here are the simple guides for setting up a relayer.

- Setup Hermes relayer
- (Recommend)
- Setup Go relayer

Step 4 (optional): Open proposal for change params and update symbol requests

Since you have already configured the symbol requests and source channel in theconfig.yml file during thestep 1 , you may skip this particular step.

Step 4.1 Open source channel param change proposal and vote

The current default value for the source channel is[not_set] . If you wish to obtain BandChain data through IBC, you will need to open the proposal to change the source channel param to your own source channel. An example of how to open a parameter change proposal is provided below.

create param-change-proposal.json

{ "messages" :

[{ "@type" :

"/pricefeed.MsgUpdateParams" , "authority" : , "params" :

{ "ask_count" :

"16" , "min_count" :

"10" , "min_ds_count" :

"3" , "prepare_gas_base" :

"3000" , "prepare_gas_each" :

"600" , "execute_gas_base" :

"70000" , "execute_gas_each" :

"7500" , "source_channel" :

"channel-1" , "fee_limit" :

[{ "amount" :

"1000000" , "denom" :

"uband" }] } }] , "metadata" :

"ipfs://CID" , "deposit" :

"10000000stake" , "title" :

"Param change for SourceChannel" , "summary" :

"Proposal for change SourceChannel param in pricefeed module" } Note that you have to put your gov module address in authority field, you can get it by running this commandoracle-consumerd query auth module-account gov

Submit proposal

exampled tx gov submit-proposal param-change-proposal.json --from alice

Vote the proposal

exampled tx gov vote 1 yes --from alice exampled tx gov vote 1 yes --from bob

Step 4.2: Open update symbol request proposal and vote

The purpose of this proposal is to request price data from BandChain atblock_interval specified in the proposal. If the proposal is approved, the pricefeed module will retrieve the data and store the response on the consumer chain.

create update-symbol-requests-proposal.json

{ "title" :

"Update Symbol requests" , "description" :

"Update symbol that request price from BandChain" , "symbol_requests" :

[{ "symbol" :

"BTC" , "oracle_script_id" :

"396" , "block_interval" :

"40" } , { "symbol" :

"ETH" , "oracle_script_id" :

"396" , "block_interval" :

"40" }] , "deposit" :

"10000000stake" } Note: You can also delete symbol request by set"block_interval": "0" on this proposal.

Submit proposal

exampled tx gov submit-legacy-proposal update-symbol-request update-symbol-requests-proposal.json --from alice

Vote the proposal

exampled tx gov vote 2 yes --from alice exampled tx gov vote 2 yes --from bob

Check proposal status

exampled query gov proposals

Query the latest price that got from BandChain

Once the proposal has been approved, the pricefeed module will query BTC and ETH from BandChain every 40 blocks on your chain, and you can view the latest price by executing this command.

exampled query pricefeed price [symbol] Previous Introduction Next Example Use Cases