Epochs

Suggest Edits

Epochs

Abstract

While using the SDK, we often want to run certain code periodically. The epochs module allows other modules to be configured such that they are signaled once every period. So another module can specify it wants to execute code once a week, starting at UTC-time = x. epochs creates a generalized epoch interface to other modules so that they can easily be signalled upon such events.

Contents

Concepts

Epochs are on-chain timers that have timer ticks at specific time intervals, triggering the execution of certain logic that is constrained by a specific epoch. The purpose of the epochs module is to provide a generalized epoch interface to other modules so that they can easily implement epochs without keeping their own code for epochs.

Every epoch has a unique identifier. Every epoch will have a start time, and an end time, where end_time = start_time + duration . When an epoch triggers the execution of code, that code is executed at the first block whose blocktime is greater than end_time . It follows that the start_time of the following epoch will be the end_time of the previous epoch.

Stride uses three epoch identifiers as found in x/epochs/genesis.go

1. DAY_EPOCH
2. : this identifies an epoch that lasts 24 hours.
3. STRIDE_EPOCH
4. : this identifies an epoch that lasts 5 minutes on local mode testnet (although this may be changed) and longer on public testnet and mainnet, and is used in the
5. x/stakeibc/
6. module as a time interval in accordance with which the Stride app chain performs certain functions, such as autocompound staking rewards.

State

The epochs module keeps EpochInfo objects and modifies the information as epoch info changes. Epochs are initialized as part of genesis initialization, and modified on begin blockers or end blockers.

Epoch information type

protobuf message EpochInfo { string identifier = 1; google.protobuf.Timestamp start_time = 2 [(gogoproto.stdtime) = true, (gogoproto.nullable) = false, (gogoproto.moretags) = "yaml:\"start_time\""]; google.protobuf.Duration duration = 3 [(gogoproto.nullable) = false, (gogoproto.stdduration) = true, (gogoproto.jsontag) = "duration,omitempty", (gogoproto.moretags) = "yaml:\"duration\""]; int64 current_epoch = 4; google.protobuf.Timestamp current_epoch_start_time = 5 [(gogoproto.stdtime) = true, (gogoproto.nullable) = false, (gogoproto.moretags) = "yaml:\"current_epoch_start_time\""]; bool epoch_counting_started = 6; reserved 7; int64 current_epoch_start_height = 8; } EpochInfo keeps identifier , start_time , duration , current_epoch , current_epoch_start_time , epoch_counting_started , current_epoch_start_height .

1. identifier
2. keeps epoch identification string.
3. start_time
4. keeps epoch counting start time, if block time passes
5. start_time
6. ,

7. epoch_counting_started
8. is set.
9. duration
10. keeps target epoch duration.
11. current_epoch
12. keeps current active epoch number.
13. current_epoch_start_time
14. keeps the start time of current epoch.
15. epoch_number
16. is counted only when
17. epoch_counting_started
18. flag is set.
19. current_epoch_start_height
20. keeps the start block height of current epoch.

Events

The epochs module emits the following events:

BeginBlocker

Type Attribute Key Attribute Value epoch_start epoch_number {epoch_number} epoch_start start_time {start_time}

EndBlocker

Type Attribute Key Attribute Value epoch_end epoch_number {epoch_number}

Keeper

Keeper Functions

epochs/keeper/ module provides utility functions to manage epochs.

Go // Keeper is the interface for lockup module keeper type Keeper interface {// GetEpochInfo returns epoch info by identifier GetEpochInfo (ctx sdk . Context ,identifier string)types . EpochInfo // SetEpochInfo set epoch info SetEpochInfo (ctx sdk . Context ,epoch types . EpochInfo)// DeleteEpochInfo delete epoch info DeleteEpochInfo (ctx sdk . Context ,identifier string)// IterateEpochInfo iterate through epochs IterateEpochInfo (ctx sdk . Context ,fn func (index int64 ,epochInfo types . EpochInfo) (stop bool))// Get all epoch infos AllEpochInfos (ctx sdk . Context) []types . EpochInfo }

Hooks

Go // the first block whose timestamp is after the duration is counted as the end of the epoch AfterEpochEnd (ctx sdk . Context ,epochIdentifier string ,epochNumber int64)// new epoch is next block of epoch end block BeforeEpochStart (ctx sdk . Context ,epochIdentifier string ,epochNumber int64) The BeforeEpochStart hook does different things depending on the identifier.

If in a day identifier it:

1. begins unbondings
2. sweeps unbonded tokens to the redemption account
3. cleans up old records
4. creates empty epoch unbonding records for the next day

If in a stride_epoch identifier it: 5. creates and deposits records on each host zone 6. sets withdrawal addresses 7. updates redemption rates (if the epoch coincides with the correct interval) 8. processes TRANSFER_QUEUE deposit records to the delegation Interchain Account (if the epoch coincides with the correct interval) 9. processes DELEGATION_QUEUE deposit records to the delegation Interchain Account (if the epoch coincides with the correct interval) 10. Query the rewards account using interchain queries, with the transfer callback to a delegation account as a staked record (if at proper interval)

How modules receive hooks

On the hook receiver functions of other modules, they need to filter epochIdentifier and execute for only a specific epochIdentifier . Filtering epochIdentifier could be in Params of other modules so that they can be modified by governance. Governance can change an epoch from week to day as needed.

Queries

epochs module provides the below queries to check the module's state

protobuf service Query { // EpochInfos provide running epochInfos rpc EpochInfos(QueryEpochsInfoRequest) returns (QueryEpochsInfoResponse) {} // CurrentEpoch provide current epoch of specified identifier rpc CurrentEpoch(QueryCurrentEpochRequest) returns (QueryCurrentEpochResponse) {} }

Future Improvements

Lack point using this module

In current design each epoch should be at least 2 blocks as start block should be different from endblock. Because of this, each epoch time will be max(blocks_time x 2, epoch_duration) . If epoch_duration is set to 1s , and block_time is 5s , actual epoch time should be 10s . We definitely recommend configure epoch_duration as more than 2x block_time, to use this module correctly. If you enforce to set it to 1s, it's same as 10s - could make module logic invalid.

TODO for postlaunch: We should see if we can architect things such that the receiver doesn't have to do this filtering, and the epochs module would pre-filter for them.

Block-time drifts problem

This implementation has block time drift based on block time. For instance, we have an epoch of 100 units that ends at t=100, if we have a block at t=97 and a block at t=104 and t=110, this epoch ends at t=104. And new epoch start at t=110. There are time drifts here, for around 1-2 blocks time. It will slow down epochs.

It's going to slow down epoch by 10-20s per week when epoch duration is 1 week. This should be resolved after launch. Updated3 months ago

Interchainquery Mint Did this page help you?Yes No *Table of Contents * *Epochs * * *Abstract * * *Contents * * *Concepts * * *State * * *Events * * *Keeper * * *Hooks * * *Queries * * *Future Improvements