Overview

Abstract

This document specifies the Cron module for the Neutron network.

Cron module implement a mechanism to add cron schedules through governance proposals to execute arbitrary cosmwasm messages with given period.

Concepts

High level Mechanism

- add schedule using governance proposals[Permissioned - Main DAO];
- remove schedule using governance proposals[Permissioned - Main DAO or Security DAO];
- every given block period execute cosmwasm msgs for added schedules.

General Mechanics

The module allows to receiveAddSchedule andRemoveSchedule custom neutron messages from cosmwasm contracts.

It also contains permissions:

- AddSchedule can only be executed as main dao governance proposal
- RemoveSchedule can only be executed as main dao governance proposal OR security subdao proposal

In EndBlocker module searches for all schedules (with limit byParams.Limit) that are ready to be executed, usinglast_execute_height .

That way after the schedule was added it will be executed everyperiod of blocks (or more thanperiod if too many schedules ready to execute).

The formats are as follows:

// AddSchedule adds new schedule to the cron module type AddSchedule struct

{ // Name of the schedule Name string

json:"name" // Period of the schedule in blocks Period uint64

json:"period" // Msgs that will be executed every period Msgs [ ] MsgExecuteContract json:"msgs" }

// MsgExecuteContract defined separate from wasmtypes since we can get away with just passing the string into bindings type MsgExecuteContract struct

{ // Contract is the address of the smart contract Contract string

json:"contract,omitempty" // Msg json encoded message to be passed to the contract Msg string

json:"msg,omitempty" } After collecting all schedules ready for execution, we execute them in order.

For each schedule, every stored msg is complemented with more necessary fields to form wasmtypes.MsgExecuteContract:

// wasmtypes.MsgExecuteContract msg :=

type MsgExecuteContract struct

{ Sender string

// Cron module account Contract string

// Passed with AddSchedule.Msgs Msg // Passed with AddSchedule.Msgs Funds sdk . Coins // Empty Coins } Then it's executed using wasmd WasmMsgServer implementation.

For state to be modified, all messages in a given schedule should return successful result. If any cosmwasm msg fails to execute for any reason, all messages in a given schedule will be rolled back.

Example

Adding schedule

To add schedule we need to send governance proposal using dao contracts.

Construct a message in a following format:

{ "propose" :

{ "msg" :

{ "propose" :

{ "title" :

"Proposal title" , "description" :

"Proposal description" , "msgs" :

[{ "custom" :

{ "add_schedule" :

{ "name" :

"simple" ,

// schedule name "period" :

5 ,

// period in blocks [{ "contract" :

"neutron123412341234" ,

// contract address to be called "msg" :

"{\"send\": {\"to\": " neutron123 ", \"amount\": 100}}" ,

// message to be executed } ,] , } , } } ,] , } , } , } , } Submit the proposal to the Main DAO using prePropose contract address.

If it will be accepted, schedule will be added with the given params.

Removing schedule

To remove schedule we need to send governance proposal using dao contracts.

Construct a message in a following format:

{ "propose" :

{ "msg" :

{ "propose" :

{ "title" :

"Proposal title" , "description" :

"Proposal description" , "msgs" :

[{ "custom" :

{ "remove_schedule" :

{ "name" :

"simple" ,

// schedule name } , } } ,] , } , } , } , } Submit the proposal to the Main DAO using prePropose contract address.

If it will be accepted, schedule will be added with the given params.

TODO: how to propose RemoveSchedule message using Security DAO?Previous State Next Client