Overview

This document describes the IBC Transfer module for the Neutron network.

The Neutron's IBC Transfer module is a wrapper over the original IBC Transfer Module with a couple of additional features. For the modules are very similar, this documentation describes only the differences. Learn more about the original IBC Transfer module (and therefore about the Neutron's one) by the link to the ibc-go documentation site:https://ibc.cosmos.network/main/apps/transfer/overview.html .

What makes this module different from the original one:

- IBC transfer results passed through the module to the sender contract to make them processable by senders;
- an informative response on Transfer messages broadcast instead of an empty one to help contracts inSudo
- messages handling.

Read about these features below to make a better understanding.

IBC transfer results handover

Since the Neutron's core audience is smart contracts, it's crucial to make interchain transfers handy and responsive for their usage. The original IBC Transfer module informs about acknowledgement and timeout packets by events emission. Smart contracts, however, are not able to subscribe to events and handle them in realtime, but what they are capable of is handlingSudo calls. Thus, in order to let a contract process its IBC transfer result, Neutron's IBC Transfer module in addition to events emission as the original module does also performs aSudo call delivering the corresponding IBC packet straight to the contract. The module does so for all possible transfer outcomes:AcknowledgementResult ,AcknowledgementError , andTimeout .

Enhanced Transfer response

IBC transfers take some time, therefore contracts don't get the IBC transfer response immediately after executing transfers. And here comes a problem of distinguishing between different IBC transfers inside a single contract's scope: when received an IBC response, how can a contract match it with the corresponding request? To ease this task, each Transfer message provides the caller with the channel'ssequence_id and the sending's chainchannel name. Contracts should preserve these values because later on correspondingSudo call the provided message will contain the IBC packet information with these values inside. By mapping these values with the preserved ones, a contract can tell its transfer results apart.

Sudo errors handling

Transfer module configured the following way, all the errors from a sudo handler are being suppressed by[contract manager middleware](#) , sudo handler is limited with[LIMIT](#) amount of gas [Previous Client](#) [Next State](#)